Adaptive and formal memetic frameworks.

Nguyen, Quang Huy.

2010

Nguyen, Q. H. (2010). Adaptive and formal memetic frameworks. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/47468>

<https://doi.org/10.32657/10356/47468>

Nanyang Technological University

Downloaded on 26 Apr 2023 18:40:19 SGT

91
2X
D/Blue

ADAPTIVE AND FORMAL MEMETIC FRAMEWORKS

NGUYEN QUANG HUY

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2010

Abstract

Over the recent years, there has been increasing research activities made on improving the efficacy of Memetic Algorithm (MA) for solving complex optimization problems. Particularly, these efforts have revealed the success of MA on a wide range of real world problems. MAs not only converge to high quality solutions, but also search more efficiently than their conventional counterparts. Despite the success and surge in interests on MAs, only restricted theoretical knowledge is available in the field of MA and limited progress has been made on formal memetic frameworks.

The key design issue of MA lies in the successful promotion of competition and cooperation between the forces of evolution and individual learning through appropriate configuration of the algorithmic parameters. This can be achieved by 1) applying different parameter configurations to find the most suitable setting or 2) adapting the algorithmic parameters of MA while the search progresses. While the former is straightforward, it could only produce reasonable configurations that are suitable for solving a given problem at hand. Further, such an option may not be feasible due to the high computational cost involved. The latter, on the other hand, may be applied to a wide range of problems more efficiently, but would require suitable online adaptation schemes and frameworks.

This thesis provides some insights into the success of MAs and summarizes investigation on the study of effective memetic frameworks. It addresses several important design issues of MAs in the context of continuous optimization problems and proposes two adaptive approaches to efficiently and effectively configure the MA design parameters during the search process. Subsequently, a formal probabilistic memetic framework is presented to highlight the inter-relationship among the MA design issues. Finally, the thesis presents a complexity analysis on the numerical methods as individual learning in MAs.

Acknowledgments

I would like to express my deepest gratitude to my academic supervisor in Nanyang Technological University (NTU), **Associate Professor Ong Yew Soon**.

His sincere encouragement during my indecisive moments and inspirational insights into the many aspects of my research direction have faithfully guided me throughout my research work. Without Professor Ong's guidance, the fruitful achievement of this project will never be possible.

I would also like to thank Associate Professor Lim Meng Hiot for his beneficial comments and support.

I would also like to thank my friends for their encouragements and the staffs in Emerging Research Lab (ER Lab) and Center for Computational Intelligence (C2I) for their technical supports.

Last but not least, special thanks to my parents for their constant support, understanding and love.

Nguyen Quang Huy

Contents

Abstract	i
Acknowledgments	ii
List of Abbreviations	viii
List of Figures	x
List of Tables	xiv
1 Introduction	1
1.1 Research Objective	3
1.2 Core Contributions	3
1.3 Organization of the Thesis	4
2 Background	6
2.1 Evolutionary Algorithm	6
2.1.1 Genetic Algorithm	7
2.1.2 Differential Evolution	12
2.1.3 Evolutionary Strategy	13
2.2 Individual learning methods	14
2.2.1 Second order method	14
2.2.2 First order method	15
2.2.3 Zeroth order method	15
3 Memetic Algorithms	18
3.1 Classification of Memetic Algorithms	21
3.1.1 First generation	21
3.1.2 Second generation	22

3.1.3	Third generation	24
3.2	Popular Instances of Memetic Algorithm	24
3.3	Research interest	25
4	A Study on the Design Issues of Memetic Algorithms	28
4.1	Experimental Procedure	29
4.2	Empirical Results	31
4.2.1	Impact of different Population-Based and Individual Learning proce- dures on MA	32
4.2.2	Impact of Individual Learning frequency and intensity on MA	38
4.2.3	Impact of individual subset selection scheme, Ω_{il} , on MA	41
4.3	Conclusions	44
5	Cellular Memetic Algorithms	46
5.1	Cellular Genetic Algorithm	48
5.2	Cellular Memetic Algorithm	50
5.2.1	Canonical Cellular Memetic Algorithm	50
5.2.2	Adaptive Cellular Memetic Algorithm	51
5.3	Empirical study	56
5.3.1	Experimental Study	57
5.3.2	Results and Analysis	58
5.3.3	Cellular MAs with Comparison to Other Evolutionary and Memetic Approaches	71
5.4	Conclusion	73
6	Diffusion Memetic Algorithms	74
6.1	Diffusion Memetic Algorithm	76
6.2	Experimental Study	80
6.2.1	Meme Diffusion in MA, DMA	82
6.2.2	First generation Cellular Memetic Algorithms	83
6.2.3	Benefits of Meme Diffusion	83
6.2.4	The neighborhood size in cellular structure	87
6.3	Conclusions	90

7	A Probabilistic Memetic Framework	92
7.1	A Probabilistic Model for Memetic Algorithm	93
7.1.1	Memetic Algorithm	93
7.1.2	A Theoretical Upper Bound for t_{il} in MA	96
7.2	A Theoretical Analysis of the Probabilistic Memetic Framework	97
7.2.1	Unimodal Sphere function	98
7.2.2	Multi-Modal Step function	99
7.2.3	Approximate Probabilistic Memetic Framework	100
7.3	Empirical study	105
7.3.1	Search quality and efficiency	106
7.3.2	Computational complexity	116
7.3.3	Simplicity & Robustness	116
7.4	APrMF with Comparison to Other Evolutionary and Memetic Approaches . . .	117
7.5	Conclusion	125
8	Complexity Analysis of Individual Learning Methods in Memetic Algorithm	127
8.1	2^{nd} order method: Newton's method in optimization	128
8.2	1^{st} order methods - gradient descent	130
8.3	Zeroth order methods - direct search	131
8.3.1	Deterministic direct search methods	131
8.3.2	Stochastic direct search methods	132
8.4	Empirical study	136
8.4.1	Numerical study on noiseless Sphere model	136
8.4.2	Empirical study on noisy Sphere model	137
8.5	Conclusions	141
9	Conclusions & Future Work	142
9.1	Conclusions	142
9.2	Future Work	143
9.2.1	Greater exploration on the probabilistic memetic framework	143
9.2.2	Probabilistic memetic framework for discrete optimization	144
9.2.3	Applications to real-world problems	144

9.2.4	Towards “memetic-driven” MA	144
9.2.5	Memetic algorithm frameworks for problems in uncertain environment	145
References		146
Appendix		
Appendix A Benchmark Test Problems		161
A.1	Sphere function	161
A.2	Step function	162
A.3	Schwefel’s problem 1.2	162
A.4	Elliptic function	162
A.5	Schwefel’s problem 1.2 with Noise in Fitness	162
A.6	Schwefel’s problem 2.6 with Global Optimum on Bounds	163
A.7	Rosenbrock Test Function	163
A.8	Ackley Test Function	163
A.9	Griewank Test Function	163
A.10	Rastrigin Test Function	164
A.11	Weierstrass Test Function	164
A.12	Schwefel’s problem 2.13	164
A.13	Expanded Griewank’s plus Rosenbrock’s function	165
A.14	Expanded Scaffer’s F6 function	165
A.15	Composition functions	165
A.16	Frequency Modulation Sounds Parameter Identification Problem [144]	165
Appendix B Conventional numerical search methods		166
B.1	Newton method	166
B.2	Quasi-Newton strategies: Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods	167
B.2.1	DFP method	167
B.2.2	BFGS method	168
B.3	Simplex method	168
B.3.1	The Basic Simplex Method	168

- B.3.2 The Modified Simplex Method [103] 170
- B.4 Davies, Swann, and Campey method with Gram-Schmidt (DSCG) or Palmer
(DSCP) orthogonalization 172
- B.4.1 Pseudo-code 172
- B.4.2 Flow chart 174

List of Abbreviations

APrMF	Approximate Probabilistic Memetic Framework
ACMA	Adaptive Cellular Memetic Algorithm
BFGS	Broyden-Fletcher-Goldfarb-Shanno method
BLX-GL50	Hybrid Real-Coded Genetic Algorithms
BLX-MA	Real-coded memetic algorithm with adaptive local search probability and local search length
CGA	Cellular Genetic Algorithm
CMA	Cellular Memetic Algorithm
DE	Differential Evolution
DEshcSPX	Differential evolution with crossover-based local search
DFP	Davidon-Fletcher-Powell's quasi-Newton strategy
DMA	Diffusion Memetic Algorithm
DMS-L-PSO	Dynamic Multi-Swarm Particle Swarm Optimizer with Local search
DSCG	Davies, Swann, and Campey with Gram-Schmidt orthogonalization
DSCP	Davies, Swann, and Campey with Palmer orthogonalization
EA	Evolutionary Algorithm
EDA	Estimation of Distribution Algorithm
EP	Evolutionary Programming
ES	Evolution Strategy
ES-IL	Evolution Strategy based Individual Learning method
FMS	Frequency Modulation Sounds parameter identification problem
FEC	Function Evaluation Call
GA	Genetic Algorithm
GP	Genetic Programming

HTGA	Hybrid Taguchi-Genetic Algorithm
OGA/Q	Orthogonal Genetic Algorithm
MA	Memetic Algorithm
MS-IL	Multi Start search with Individual Learning
MS-DSCG	Multi Start search with DSCG individual learning method
PrMF	Probabilistic Memetic Framework
PSO	Particle Swam Optimization

List of Figures

2.1	Stochastic universal selection (SUS).	8
2.2	One point crossover.	9
2.3	Two point crossover.	10
2.4	Uniform crossover.	10
3.1	First generation of MA	22
3.2	Second generation of MA	23
3.3	Third generation of MA	24
4.1	Search performance of different population based-individual learning or global-local MAs on 30D Sphere function	34
4.2	Search performance of different population based-individual learning or global-local MAs on 30D Ackley function	35
4.3	Search performance of different population based-individual learning or global-local MAs on 30D Weierstrass function	35
4.4	Search performance of MA with different configurations of (f_{il}, t_{il}) on 30D Sphere function	40
4.5	Search performance of MA with different configurations of (f_{il}, t_{il}) on 30D Ackley function	40
4.6	Search performance of MA with different configurations of (f_{il}, t_{il}) on 30D Weierstrass function	41
4.7	Search performance of MA for diverse individual subset selection schemes and t_{il} on 30D Sphere function	43
4.8	Search performance of MA for diverse individual subset selection schemes and t_{il} on 30D Ackley function	43

4.9	Search performance of MA for diverse individual subset selection schemes and t_{il} on 30D Weierstrass function	44
5.1	Neighborhood structure in Cellular Genetic Algorithm	49
5.2	Fitness distribution at generation 60 (minimizing 10D Rastrigin)	55
5.3	Fitness distribution at generation 122 (minimizing 10D Rastrigin)	55
5.4	Fitness distribution at generation 240 (minimizing 10D Rastrigin)	55
5.5	Fitness distribution at generation 266 (minimizing 10D Rastrigin)	55
5.6	Search traces (average of 50 runs) for minimizing 10D Sphere function. (Square box indicates that the Global Optimum is found)	60
5.7	Search traces (average of 50 runs) for minimizing 30D Elliptic function. (Square box indicates that the Global Optimum is found)	60
5.8	Search traces (average of 50 runs) for minimizing 30D Schwefel function.	61
5.9	Search traces (average of 50 runs) for minimizing 30D Ackley function. (Square box indicates that the Global Optimum is found)	63
5.10	Search traces (average of 50 runs) for minimizing 30D Rastrigin function. (Square box indicates that the Global Optimum is found)	64
5.11	Search traces (average of 50 runs) for minimizing 30D Griewank function. (Square box indicates that the Global Optimum is found)	64
5.12	Search traces (average of 50 runs) for minimizing 30D Rosenbrock function.	66
5.13	Search traces (average of 50 runs) for minimizing 30D Weierstrass function.	68
5.14	Search traces (average of 50 runs) for minimizing 30D Scaffer function.	69
5.15	Search traces (average of 50 runs) for minimizing FMS function.	70
5.16	Number of function evaluations used by different algorithms in solving the 30D benchmark functions.	72
6.1	Neighborhood structure in Cellular Genetic Algorithm	77
6.2	Memetic map across different search generations of DMA (DSCG + DFP) on 30D Griewank	81
6.3	Search performance of CGA and CMAs on 30D Griewank	84
6.4	Search performance of CGA and CMAs on 30D Rastrigin	84
6.5	Search performance of CGA and DMAs on 30D Griewank	85

6.6	Search performance of CGA and DMAs on 30D Rastrigin	85
6.7	Meme distribution during the search process of DMA (DSCG + Simplex + DFP) on 30D Griewank	86
6.8	Meme distribution during the search process of DMA (DSCG + Simplex + DFP) on 30D Rastrigin	87
6.9	Neighborhood structure in meme selection algorithm (a) Distance = 1 (b) Distance = 1 and 2	88
6.10	Meme distribution of DSCG during the search process of DMA (DSCG + DFP) on 30D Griewank	89
7.1	Illustrations of type I and type II points	94
7.2	Unimodal Sphere function	99
7.3	Multi-Modal Step function	100
7.4	Illustration of algorithm (a) Narrow, deep basin (b) Wide, shallow basin	104
7.5	Search trends of Canonical MA (with a fixed individual learning intensity, $t_{il} = 100$) and APrMF (GA-DFP) on the Unimodal Sphere function	108
7.6	Search trends of Canonical MA (with a fixed individual learning intensity, $t_{il} = 100$) and APrMF (GA-DFP) on the Step function	109
7.7	Legend for Figures 7.8-7.10	112
7.8	APrMF with Upward Trends in Individual learning/global search Ratio and t_{il} .	113
7.9	APrMF Adapting the Individual learning/global search Ratio to its Optimal Configuration	114
7.10	APrMF with Downward Trends in Individual learning/global search Ratio and t_{il}	115
7.11	Number of function evaluations used by different algorithms in solving the 30D benchmark functions	124
8.1	Performance of numerical methods on Sphere function	137
8.2	Performance of numerical methods on increasing problem dimension	138
8.3	Performance of numerical methods on increasing noise levels	140
B.1	Geometric interpretation of lower dimensional simplices	169

B.2 An example of a typical optimization sequence with the basic simplex method.
Change in the levels for two control variables with the response marked as
contours. 169

B.3 Basic Simplex method flowchart. 170

B.4 Different simplex moves from the rejected trial condition (W). R = reflection,
E = expansion, C+ = positive contraction and C- = negative contraction. . . . 171

B.5 Modified Simplex method flowchart. 171

B.6 The strategy of Davies, Swann, and Campey with Gram-Schmidt orthogonal-
ization (DSCG). 174

List of Tables

- 4.1 Benchmark functions used in the study 30
- 4.2 MA parameters setting 31
- 4.3 Optimization results of multistart individual learning on the benchmark problems. For Sphere function, the average number of evaluations incurred by the algorithms in locating the global optimum is reported instead. 32
- 4.4 Friedman (non-parametric) test for Sphere function. (Null hypothesis: the relative performances among individual learning methods are independent with the choice of stochastic search strategies) 36
- 4.5 Friedman (non-parametric) test for Sphere function. (Null hypothesis: the relative performances among individual learning methods are independent with the choice of stochastic search strategies) 37
- 4.6 Friedman (non-parametric) test for Sphere function. (Null hypothesis: the relative performances among individual learning methods are independent with the choice of stochastic search strategies) 37
- 4.7 Chromosome selection schemes 42

- 5.1 Benchmark functions used in the study (epi*: epistasis, mul*: multimodality) . 58
- 5.2 Fitness error values and Success Rate (30 Dimensional Sphere)(note that the best performing algorithm is highlighted in bold typeface) 61
- 5.3 Fitness error values and success rate (30 Dimensional Elliptic)(note that the best performing algorithm is highlighted in bold typeface) 62
- 5.4 Fitness error values and success rate (30 Dimensional Schwefel 1.02)(note that the best performing algorithm is highlighted in bold typeface) 62
- 5.5 Fitness error values and success rate (30 Dimensional Ackley)(note that the best performing algorithm is highlighted in bold typeface) 63

5.6	Fitness error values and success rate (30 Dimensional Rastrigin)(note that the best performing algorithm is highlighted in bold typeface)	65
5.7	Fitness error values and success rate (30 Dimensional Griewank)(note that the best performing algorithm is highlighted in bold typeface)	65
5.8	Fitness error values and success rate (30 Dimensional Rosenbrock)(note that the best performing algorithm is highlighted in bold typeface)	67
5.9	Fitness error values and success rate (30 Dimensional Weierstrass)(note that the best performing algorithm is highlighted in bold typeface)	68
5.10	Fitness error values and success rate (30 Dimensional Scaffer)(note that the best performing algorithm is highlighted in bold typeface)	69
5.11	Fitness error values and success rate (FMS)(note that the best performing algorithm is highlighted in bold typeface)	70
5.12	Statistic test on comparing ACMA5 vs. OGA/Q (Null hypothesis: ACMA5 and OGA/Q incur equal number of evaluations vs. alternative hypothesis: ACMA5 and OGA/Q incur different number of evaluations to reach to the global optimum)	72
5.13	Statistic test on comparing ACMA5 vs. HTGA (Null hypothesis: ACMA5 and HTGA incur equal number of evaluations vs. alternative hypothesis: ACMA5 and HTGA incur different number of evaluations to reach to the global optimum)	72
6.1	Benchmark functions used in the study	80
6.2	Search performance of DMA on 30D Griewank	88
7.1	Benchmark functions used in the study.	105
7.2	Parameter setting of APrMF.	110
7.3	Parameter Configuration of APrMF based on GA-DSCG.	118
7.4	Optimization result using APrMF (GA-DSCG) on 10D benchmark functions. FEC represents Function Evaluation Calls incurred to converge at the global optimum solution of each respective function.	119
7.5	Optimization result using APrMF (GA-DSCG) on 30D benchmark functions. FEC represents Function Evaluation Calls incurred to converge at the global optimum solution of each respective function.	120

7.6	Memetic Algorithms or Hybrid EA-Local search used in Comparison.	121
7.7	Success measure of the algorithms in solving the 10D benchmark functions. For instance, 0.6 (25) on F_{Sphere} implies that APrMF incurred an average of (0.6*1000) function evaluation calls on 25 successful independent runs. A '-' entry in the table implies that the respective algorithm fails to converge to the global optimum successfully. Bold italic also highlights the best search per- formance among the approaches considered (based on pair-wise t-test between each respective algorithm pairs).	122
7.8	Result of t-test with 95% confidence level comparing statistical values for APrMF and those of the other algorithms in solving the 10D benchmark func- tions (s+, s-, and $\approx$ indicate that APrMF is significantly better, significantly worse, and indifferent, respectively).	122
7.9	Success measure of the algorithms in solving the 30D benchmark functions. For instance, 87.0 (25) on $F_{Sch1.2}$ implies that APrMF incurred an average of (87*1000) function evaluation calls on 25 successful independent runs. A '-' entry in the table implies that the respective algorithm fails to converge to the global optimum successfully. Bold italic also highlights the best search performance among the approaches considered.	123
7.10	Result of t-test with 95% confidence level comparing statistical values for APrMF and those of the other algorithms in solving the 30D benchmark func- tions (s+, s-, and $\approx$ indicate that APrMF is significantly better, significantly worse, and indifferent, respectively).	123
7.11	Statistic test on comparing ACMA5 vs. APrMF (Null hypothesis: ACMA5 and APrMF incur equal number of evaluations vs. alternative hypothesis: ACMA5 and APrMF incur different number of evaluations to reach to the global optimum)	125
8.1	Computational and space complexity of numerical optimization methods	135

Chapter 1

Introduction

Optimization is the study of the mathematical properties of minimization or maximization problems and the analysis of algorithms for their solutions. It deals with the problem of minimizing or maximizing a set of given objective functions such as cost, fuel consumption, etc., subject to a set of constraints. This research considers the general non-linear programming problem of the form:

- A set of target objective functions $f_i(\mathbf{x})$, $i = 1, 2, \dots, n$ to be minimized or maximized, where n is the number of objective functions.
- A set of variables, $\mathbf{x}$, $\mathbf{x}_{\text{low}} \leq \mathbf{x} \leq \mathbf{x}_{\text{up}}$, which affect the value of the objective functions, where $\mathbf{x}_{\text{low}}$ and $\mathbf{x}_{\text{up}}$ are the lower and upper bounds, respectively.
- A set of equality/inequality constraints $g_w(\mathbf{x})$ that allow the unknowns to take on certain values but exclude others. For example, the constraints may take the form of $g_w(\mathbf{x}) \leq 0$, for $w = 1, 2, \dots, b$, where b is the number of constraints.

Over the last half century, a plethora of optimization techniques have been studied extensively by many researchers. Optimization methods have since evolved considerably and many algorithms and implementations are now available and used in the engineering optimization community. They can generally be classified into three broad categories: conventional numerical optimization methods (mostly gradient based), stochastic optimization methods and hybrid

CHAPTER 1. INTRODUCTION

methods. Typical conventional numerical methods are steepest-descent methods, conjugate-gradient, quadratic programming, direct search methods and linear approximation methods. These methods have the known advantage of their efficiency; however, they are very sensitive to starting point selection and are more likely to stop at non-global optima than modern stochastic algorithms. Several research efforts on these conventional numerical optimization methods have been applied with much success to complex engineering problems including aircraft design [33].

Modern stochastic algorithms such as evolutionary algorithms (EA) draw inspiration from biological evolution. EAs, unlike conventional numerical optimization methods, produce new search points that do not use information about the local slope of the objective function and are thus not prone to stalling at local optima. Instead they involve a search from a “population” of solutions; making use of competitive selection, recombination and mutation operators to generate new solutions which are biased towards better regions of the search space. Further, they have shown considerable potentials for solving optimization problems that are characterized by non-convex, disjoint or noisy solution spaces. Modern stochastic optimizers, which have attracted a great deal of attention in recent years, include simulated annealing, tabu search, genetic algorithms, evolutionary programming, evolutionary strategies, differential evolution and many others [11, 35, 68, 98]. These stochastic methods have been successfully applied to many real world optimization problems.

The third category of optimization methods are hybrid methods, which are formed by combining different optimization methods or sometimes with the use of artificial intelligence. InterGEN [141] is a hybrid method that combined a Genetic Algorithm with a conventional numerical optimizer, and uses a rule based expert system to decide when to switch between the two. Gage [41] also combined Genetic Algorithm with Sequential Quadratic Programming for the design of aircraft wings. In addition, there exists a breed of hybrid Evolutionary Algorithms - Local Searches that implicitly involve the use of learning procedures within the population. Such hybrid methods are also commonly referred to as Memetic Algorithms [100].

1.1 Research Objective

Memetic algorithms (MAs) represent one of the recent growing areas in evolutionary algorithm research and have been demonstrated to converge to high quality solutions more efficiently than their conventional counterparts on a wide range of real world problems. Despite the success and surge in interests on MAs, many of the successful MAs reported have been crafted to suit problems in very specific domains and are mostly designed based on heuristics that comes with little theoretical motivation. As a result, there is a need for the development of novel methodologies and formal theoretical frameworks that efficiently facilitate the application of memetic framework in complex optimization problems.

This thesis aims to address and handle different design issues arising from the synergy between global exploration and local exploitation search in memetic algorithms. In particular, this research introduces methodologies and architectures that efficiently promote the incorporation of suitable individual learning mechanisms for enhancing the evolutionary search efficiency. Subsequently, a formal probabilistic memetic framework is proposed and employed as a foundation for creating novel memetic algorithms.

1.2 Core Contributions

Our contribution in the thesis begins with providing a comprehensive survey on existing state-of-the-art Memetic Algorithm (MA) optimization frameworks. From the survey, a taxonomy for MA is derived. Further, the effect of different design issues of MA on the search performance is discussed and evaluated empirically.

In the research topic on MA design, two novel Memetic Algorithms are proposed for solving continuous optimization problems. Each of these algorithms takes into account the different design issues of MA and presents heuristics for adjusting the parameters while the search progresses. In particular, Diffusion MA takes its inspiration from the mechanism of social virus

infection in the choice of meme. On the other hand, Adaptive Cellular MA defines the degree of local learning based on the diversity of the population.

Given the restricted theoretical knowledge available in the field of MA and the limited progress made on formal MA frameworks, a novel Probabilistic Memetic Framework is presented, in which MAs are modeled as a process involving the decision of embracing the separate independent actions of evolution or individual learning and analyze the probability of each process in locating the global optimum. Further, the framework balances evolution and individual learning by governing the learning intensity of each individual according to the theoretical upper bound derived while the search progresses. Theoretical and empirical studies on benchmark problems commonly used in the literature are presented to demonstrate the characteristics and efficacies of the probabilistic memetic framework.

Last but not least, the thesis provides an analysis on computational complexity and memory usage of several commonly used numerical individual learning strategies. The analysis highlights the differences among zeroth, first, and second order numerical search methods including the popular stochastic evolutionary strategy (ES).

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows:

- In Chapter 2, the fundamental notions of optimization and other relevant topics to the thesis, namely, evolutionary algorithms and numerical methods are briefly described. In this chapter, readers will grasp the background knowledge required for better understanding of the advanced topics in chapters 3-8.
- Chapter 3 presents a survey on memetic algorithm. From the survey, a taxonomy of MA is proposed. In the rest of this chapter, a discussion on various variants of MAs is also presented.

CHAPTER 1. INTRODUCTION

- Chapters 4-8 contain the core contributions of this thesis. Particularly, Chapter 4 presents an extensive experimental study on the impact of each individual design issue and their relative influences on memetic search performances by means of three commonly used synthetic problems. Chapters 5 and 6 introduce two instances of adaptive MAs. Cellular memetic algorithm selects the individuals undergoing learning based on the population diversity index while the diffusion memetic algorithm promotes the spread of memes based on the idea of virus spreading. Chapter 7 presents a mathematical framework for MA and derives a theoretical upper bound of the individual learning intensity in the proposed probabilistic memetic framework. Subsequently, Chapter 8 presents a theoretical analysis on the performance and complexity of some typical deterministic and stochastic individual learning methods commonly used in continuous optimization.
- Finally, Chapter 9 concludes this thesis and outlines possible future research directions in the area of memetic optimization.

Chapter 2

Background

In this chapter, an overview of evolutionary algorithm (EA) and individual learning approaches is presented.

2.1 Evolutionary Algorithm

Evolutionary algorithms (EAs) such as Genetic Algorithms (GA), Evolutionary Programming (EP) and Evolution Strategies (ES) are stochastic search methods that have been applied successfully for solving complex engineering optimization problems where classical deterministic methods are known to have performed poorly [64, 79, 70]. These include turbine blade design [112], multi-disciplinary rotor blade design [50], large flexible space structures design [102], Open-shop Scheduling [69] and others.

EA, inspired by the Neo-Darwinian theory of natural evolution, proceeds in an iterative generation of new populations, $Pop^{(k)}$, ($k = 0, 1, 2, \dots$), based on the previous. Every individual in the population is the encoded (binary, real, integer, etc.) version of a tentative solution. An evaluation of the objective function associates a fitness value to each individual indicating its suitability to the problem. The canonical EA involves stochastic operators such as selection, crossover, and/or mutation on an initial population of randomly generated individuals to produce the subsequent generation of new individuals. The nature of the operators used generally

CHAPTER 2. BACKGROUND

defines the type of EA considered. The halting condition may be configured as reaching a pre-programmed fixed number of iterations of the algorithm, convergence to solutions that satisfy a certain degree of error to some optimum, or otherwise lack of search improvement for some time.

2.1.1 Genetic Algorithm

Among the many breeds of EA, Genetic algorithm (GA) [46, 56] which is often considered as the core approach, is used as the major framework in the present study. Inspired by the mechanisms of natural genetics, GA operates on a population of artificial creatures, which compete in a struggle for life in which the fittest survive and are allowed to reproduce. New creatures are created using bits and pieces of members of the previous generation, and occasionally, creatures are changed randomly.

The first issue to consider while using GA is encoding representation. Binary encodings, introduced by Holland [56], is among the most commonly used encodings of genetic algorithms. There have been many extensions to the basic binary encoding schema, such as Gray coding, real coding and Hillis’s diploid binary encoding scheme. In this research, the two encoding schemes used are binary and real coding.

Upon deciding on an encoding representation, the nature of GA operators will be considered. Algorithm 1 outlines the basic steps of a genetic algorithm, followed by the detailed description of each GA operator.

Algorithm 1 Genetic Algorithm

- 1: *Generate* an initial population;
 - 2: **while** Stopping conditions are not satisfied **do**
 - 3: *Evaluate* all individuals in the population.
 - 4: *Select* parent chromosomes.
 - 5: *Apply* genetic operators (crossover, mutation) to generate offsprings.
 - 6: *Replace* a proportion or the entire population with the offsprings.
 - 7: **end while**
-

CHAPTER 2. BACKGROUND

2.1.1.1 Selection operator

The selection operator defines the individuals in a population that survive to the next generation. Generally, individuals with better fitness values have higher probability of surviving. Let $P(x)$ be the probability that an individual x is selected from the current population to survive to the next generation. Several popular selection schemes exist as follow:

Figure 2.1: Stochastic universal selection (SUS).

- **Fitness Proportionate Selection (FPS):** selection probabilities assigned to the individual, $P(x)$, are proportional to their fitness value, i.e., $P(x) \sim f(x)$, where $f(x)$ is the fitness value of x . Biased roulette wheel (RW) [10] and stochastic universal selection (SUS) [13] are two popular examples. In RW, the individuals are selected, one after another, by picking a random point on the roulette wheel. In this scheme, one dominant individuals can be selected repeatedly, thus can lead to premature convergence. SUS avoids the limitation by simultaneously selecting the individuals from the roulette wheel (Figure 2.1). It is also worth noting that commonly, points are uniformly sampled from the wheel, though other probability distributions can be employed.
- **Ranking [12]:** If P_{max} and P_{min} are the probabilities assigned to the individual with best and worst fitness values, respectively and n is the population size, the probability of

CHAPTER 2. BACKGROUND

selecting the i^{th} -rank individual of the population, $P(x_i)$, is defined as:

$$P(x_i) = P_{max} - \frac{i-1}{n-1}(P_{max} - P_{min})$$

- **Tournament [10]:** Each time, two or more individuals are randomly picked from the population. The fitter/fittest individual is then selected to survive across the next generation, while the rest is then returned to the population for the subsequent selection.

2.1.1.2 Crossover operator

Crossover or recombination operator imitates the complex process of two chromosomes in the cell division phase of living beings. This operator randomly chooses one or more locations (loci) of breakage and exchanges the subsequences before and after the loci between two individuals to create two new offspring. Clearly, the mechanisms of the crossover operator would depend on the encoding scheme chosen. For binary or real encoded chromosomes, three typical forms of crossover operator are described here.

- **One-point crossover:** Single point crossover [56] represents the simplest form of the three. A single crossover position is chosen at random and portions of the parents starting from the crossover loci position are exchanged to create two new offspring. In other words, a point k is chosen randomly along the string length L . Two new strings are generated by swapping all bits between positions $k+1$ and L (see Figure 2.2).

Figure 2.2: One point crossover.

CHAPTER 2. BACKGROUND

- **Two-point crossover:** Two cut points are chosen at random and the portion of the parent chromosomes between these points is exchanged (see Figure 2.3). Similarly, in multi-point crossover operator, multiple locus positions are chosen at random and the segments between them exchanged.

Figure 2.3: Two point crossover.

- **Uniform crossover:** For each bit position 1 to L , randomly pick each bit from either of the two parent strings (see Figure 2.4). This means that each bit is inherited independently from any other bit. As a result, uniform crossover is unbiased with respect to length L . Uniform crossover is generally more disruptive than one-point crossover. However, its disruptive character might be helpful for migrating the issue of convergence to local optima.

Figure 2.4: Uniform crossover.

In real coded chromosomes, arithmetical crossover is also commonly used. Consider parent chromosomes $\mathbf{x}_1$ and $\mathbf{x}_2$, the offsprings are then defined as $\mathbf{x}_1^* = \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2$ and $\mathbf{x}_2^* = \lambda_1 \mathbf{x}_2 + \lambda_2 \mathbf{x}_1$. The values of λ_1 and λ_2 thus define the variants of crossover mechanism: convex, affine or linear.

CHAPTER 2. BACKGROUND

2.1.1.3 Mutation operator

The mutation procedure randomly modifies the genes of an individual, subjected to a small mutation probability, p_m . This introduces some randomness in the population. Like crossover operators, mutation operators are also encoding scheme dependent.

In binary-coded GA, the bit flip operator is among the most commonly used. For each bit in the solution string, a random number r_m is generated in the range $[0, 1]$. If $r_m \leq p_m$, the bit is flipped, i.e., $0 \rightarrow 1$ and $1 \rightarrow 0$.

In real-coded GA, more variants of mutation operator have emerged. Some of the typical mutation schemes include the following:

- Uniform mutation: genes at random positions are replaced by some random values.
- Gaussian mutation: a random Gaussian noise is added to the current gene: $\mathbf{x}^* = \mathbf{x} + N(0, \sigma)$.
- Arithmetical mutation: $\mathbf{x}^* = \mathbf{x} + r(\mathbf{x}_{\text{up}} - \mathbf{x})(1 - (t/T)^b)$, where $\mathbf{x}_{\text{up}}$ denotes the upper bound of $\mathbf{x}$, t is the current generation counter index in a total number of T generations, b may be used to adjust the convergence speed and r is a random number in the range $[0, 1]$.

2.1.1.4 Replacement scheme

Replacement scheme specifies how many offspring to generate and re-insert into the population. Popular replacement schemes include the following:

- *Simple GA* replaces entire population by the offsprings.
- *GA with elitist strategy* copies the fittest parent(s) to the next generation while the rest of the population is replaced with new offspring.

CHAPTER 2. BACKGROUND

- *Steady-state GA* replaces just a few weak parents with new offsprings.
- *Incremental GA* replaces just one or two of the parents.

2.1.1.5 Building block hypothesis

Despite being relatively simple to implement, GAs achieve great success at generating solutions of high fitness when applied to practical problems. Several hypotheses have been proposed to explain the key to success of GAs. Notable ones include the building block hypothesis [46] which proposes that the construction of genetic algorithm implicitly performs an adaptation heuristic that is capable of identifying and recombining good “building blocks” (i.e., low order, low defining-length schemata with above average fitness). In particular, instead of building high-performance solution strings by trying every conceivable combination, GA constructs better and better strings from the best partial solutions of past samplings, referred to as building blocks. By working with these building blocks, the complexity of the optimization problems is greatly reduced. The name “building block” is also explained as “Because highly fit schemata of low defining length and low order play such an important role in the action of genetic algorithms, we have already given them a special name: building blocks. Just as a child creates magnificent fortresses through the arrangement of simple blocks of wood, so does a genetic algorithm seek near optimal performance through the juxtaposition of short, low-order, high-performance schemata, or building blocks.” [46]

2.1.2 Differential Evolution

Differential Evolution (DE) can be considered as a variant of the real-coded GA. The core difference between the canonical DE and other variants of real-coded GA lies mainly in the mutation operator used. Differential Evolution obtains its name from the idea of using differences in solution values to perturb the solution vector population.

CHAPTER 2. BACKGROUND

The mutated individual $\mathbf{u}$ of an individual $\mathbf{x}$ in DE is defined as:

$$\mathbf{u} = \mathbf{x}_1 + \xi_1(\mathbf{x}_2 - \mathbf{x}_3) + \xi_2(\mathbf{x}_4 - \mathbf{x}_5)$$

where $\mathbf{x}_i, (i = 1, \dots, 5)$ may either be the best individual found so far in the search, the current individual $\mathbf{x}$ or an arbitrary individual in the population. The two constants ξ_i is referred to as learning rates selected between 0 and 2.

Subsequently, the mutated individual $\mathbf{u}$ and $\mathbf{x}$ undergo crossover operation to produce an offspring $\mathbf{v}$. Tournament selection then proceeds with $\mathbf{x}$ and $\mathbf{v}$ to decide on the individuals that will survive to the next generation.

2.1.3 Evolutionary Strategy

Similar to Genetic Algorithm and Differential Evolution, Evolutionary Strategy [11] takes its inspiration from nature evolution. The main difference between ES and other EAs lies in the employment of mutation rate adaptation. As a rule of thumb, the mutation rate is usually configured with a large value in the beginning so as to facilitate greater exploration of the search space. As the search progresses, the mutation rate is decreased to promote search exploitation. In practice, Evolutionary Strategy incorporates a self-adaptation mechanism for mutation rate adjustment. There exists several variants of ES that differ in term of the replacement strategies, choice of starting point and schemes for generating the subsequent generations.

Most commonly, Evolution Strategies are classified according to their reproduction strategies. More popular ES schemes include the $(1+1)$ -ES which makes use of a single-chromosome population. At each iteration, a single offspring is generated and undergo tournament selection with the parent chromosome for choosing the individual to form the next generation. On the other hand, $(\mu + \lambda)$ -ES maintains a population of size μ and generates λ offspring in each search generation. The entire set of $\mu + \lambda$ individuals then undergoes ranking selection, and the fittest μ individuals will survive to the next generation. Last but not least, the (μ, λ) -ES is

also among the commonly used ES schemes where ranking selection is conducted only on the set of λ offspring and the fittest μ individuals then form the next population.

2.2 Individual learning methods

Individual learning [23] involves the process of searching in the neighborhood region of a solution and adopting a better solution if it exists. The pseudo codes of a typical individual learning method is outlined in Algorithm 2. The search begins with a choice in the direction of movement, subsequently a line search, trust region approach, or otherwise is performed to determine an appropriate step. The process is repeated at the new point and the algorithm continues until a local optimum is found.

Algorithm 2 Individual learning procedure

```

Start from some given point  $x^{(1)}$ .
Assign iteration counter  $k = 1$ .
while not converged do
    Calculate a search direction  $D^{(k)}$ , Determine an appropriate step length  $\lambda^{(k)}$ .
    Next search point  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} D^{(k)}$ 
     $k = k + 1$ 
end while

```

Different individual learning methods can be distinguished by their choice of search directions. In addition, according to how the algorithm makes use of derivative information, continuous parametric individual learning methods can be categorized as zeroth, first or second order method.

2.2.1 Second order method

This category of method make use of the functional values, its first (partial) derivative vector and the second derivative matrix - the Hessian to conduct the search. Newton's method [66] is an excellent example of this kind. The method is powerful, simple to implement and guarantees convergence to a stable point from any sufficiently close starting value. However, the need to

CHAPTER 2. BACKGROUND

compute the Hessian matrix may be regarded as a drawback since this process can be highly computational intensive. For a detailed description of the Newton's method, the readers is referred to appendix B.1.

2.2.2 First order method

In practice, the Hessian matrix may not be positive definite or may be difficult to calculate, thus making the second order methods infeasible for use. Various techniques and variants of Newton method have been proposed for handling such cases.

When the Hessian matrix is not positive definite, a typical trick is to perturb the diagonal of the matrix with some noise so that the final matrix becomes positive definite. Such approach is commonly referred to as line search. On the other hand, when the second derivative of the objective function is unavailable, it can either be approximated using finite differencing or via some iterative techniques. Methods where the Hessian matrix is updated in an iterative manner are generally established as Quasi-Newton approach. The strategies of Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) are examples of Quasi-Newton methods where the Hessian matrix is approximated and updated in an iterative manner. For a detailed description of these two Quasi-Newton methods, the readers are referred to Appendix B.2

2.2.3 Zeroth order method

Also known as direct search techniques, zeroth order method is often very useful when the derivative information, either the Hessian or gradients are unavailable, unreliable or expensive to compute [65]. The main feature of zeroth order method is that they only need the object function values, or even only the relative rank of objective values. This information is sufficient to determine the local behavior of the function. Simplex search, pattern search, and methods with adaptive sets of search directions are the three main categories of direct search.

CHAPTER 2. BACKGROUND

- Simplex: The search points are spread out efficiently in the form of a simplex. In particular, the shapes of the simplex in a one, two and three variable search space, are a line, triangle and tetrahedron, respectively. The Basic Simplex method was later modified by Nelder and Mead [103] and the details are presented in Appendix B.3.
- Pattern search: The method begins with point $\mathbf{x}^{(0)}$ by building a lattice $\mathcal{L}$ defined by $\mathcal{L}(\mathbf{x}^{(0)}, \lambda^{(0)}) = \{\mathbf{x} : \mathbf{x} = \mathbf{x}^{(0)} + \lambda^{(0)}\delta, \delta \in \Delta\}$, where $\lambda^{(0)}$ denotes the initial resolution. Δ is the lattice of integral point of $\mathbb{N}^n$. As the name suggested, the purpose of this method is to sample the whole neighborhood of $\mathbf{x}^{(0)}$ uniformly. In the k^{th} iteration, $\mathbf{x}^{(k)}$ is chosen as the best point found so far and $\lambda^{(k)} = \lambda^{(0)}/\tau^k$ with some integer $\tau > 1$ and the process repeats. The convergence analysis of the pattern search method was first established in [85].

Several variants of pattern search solver exist to date, which also make use of interval division such as Golden section or Fibonacci division and differ mainly in the way that the lattice size is reduced upon boxing the minimum. The Lagrange interpolation method may also be regarded as a form of pattern search category.

- Method with adaptive sets of search direction: This category of strategies emerges from Rosenbrock's idea of rotating coordinates. Starting from point $\mathbf{x}_0^{(0)}$ (the superscript is the iteration count), Rosenbrock uses the unit vector set $\{e_i\}$ in the Euclidean coordinate system as the initial set of search directions $\{\mathbf{v}_0^{(0)}\}$. Line search in each independent search direction is applied alternatively and repeatedly until at least one success and one failure is found in each directions. Subsequently, the coordinate system is rotated using the Gram-Schmidt orthogonalization procedure.

Extending from Rosenbrock's basic idea, Davis, Swann and Campey [131] proposed a revision to reduce the computational cost of orthogonalization. In their method, starting with the same unit vector direction set, line search will be conducted in each direction

CHAPTER 2. BACKGROUND

only once. Note that this differs from Rosenbrock's original approach where each vector direction is searched iteratively.

$$\mathbf{x}_0^{(k)} \xrightarrow{\text{linesearch}(\mathbf{v}_1^{(k)})} \mathbf{x}_1^{(k)} \xrightarrow{\text{linesearch}(\mathbf{v}_2^{(k)})} \mathbf{x}_2^{(k)} \dots \xrightarrow{\text{linesearch}(\mathbf{v}_n^{(k)})} \mathbf{x}_n^{(k)}$$

Subsequently, one extra line search is considered in the direction of $\mathbf{x}_n^{(k)} - \mathbf{x}_0^{(k)}$ by starting from $\mathbf{x}_n^{(k)}$.

Note that when $\|\mathbf{x}_{n+1}^{(k)} - \mathbf{x}_0^{(k)}\| \leq s^{(k)}$, where $s^{(k)}$ denotes the step length of the line search at iteration k , the step length is reduced and the coordinate does not undergo any rotation. Moreover, the improvement in each search direction are sorted and only direction vectors that have gained sufficient improvement in the previous iteration are employed to perform the subsequent iterations.

For detailed description of Davis, Swann and Campey's strategy, readers are referred to Appendix B.4

Chapter 3

Memetic Algorithms

Memetic Computation represents one of the successful computational intelligence methodologies in current use today. Stemming from the fundamentals of both Darwinian principles of natural evolution and Dawkins' notion of a meme, the term "Memetic Algorithm" (MA) was introduced in Moscato's Caltech technical report [100] in 1989 where MA is generally viewed as being close to a form of population-based hybrid evolutionary algorithm coupled with a learning procedure capable of performing refinements. The metaphorical parallels, on the one hand, to Darwinian evolution and, on the other hand, between memes and domain specific heuristics are captured within memetic algorithms thus rendering a methodology that balances well between generality and problem specificity. In a more diverse context, memetic algorithm has been used under various names including Hybrid Evolutionary Algorithm, Baldwinian Evolutionary Algorithm, Lamarckian Evolutionary Algorithms, Cultural Algorithm or Genetic Local Search.

The last decades have revealed increasing research trends in MA as demonstrated by the significantly increasing research publications in the IEEE TEC [118, 59, 9, 15, 16, 77, 80, 113, 111, 62] and other top quality journals [90, 1, 134, 95, 140, 142, 30, 40, 26, 27, 24, 96, 81, 25]. To be precise, an exponential increase in the number of research publications have been revealed in the ISI Web of Science database with the top three most well cited papers published in the IEEE Transaction on Evolutionary Computation. Special sessions on memetic algorithms

CHAPTER 3. MEMETIC ALGORITHMS

have consistently been organized in the annual IEEE Conference on Evolutionary Computation (CEC) and bi-annual IEEE World Congress on Computational Intelligence by members of Task Force on Memetic Computing in the IEEE Computational Intelligence Society Emergent Technology Technical Committee since its inauguration in 2006. In CEC'09, two special sessions relating to the field of MA have been organized and similar plans are also made for WCCI'10. The surge on MA research is also evident through the recent edited volumes on memetic algorithm [54] and multi-objective memetic algorithm [43], a handbook on memetic algorithm in progress and a recent journal dedicated to memetic computing [153].

It is worth noting one of the key driving forces of MA in search is the "no free lunch" (NFL) theorems [149]. Wolpert and Macready state that the average performance of an algorithm over all classes of problems remains constant and performance advantage for a particular class of problems can only be achieved by accepting degraded performance on others. Hence, while it is possible to use EAs for general "black-box" optimization, i.e., without incorporating domain knowledge, it comes with a significant disadvantage in not performing as well as problem-specific approaches. Knowledge about the underlying problem, if available, can be used to improve the performance of the algorithm. The importance of domain knowledge was further acknowledged by Radcliffe [126], while an overview of different domain knowledge incorporation techniques in EAs was also reported by Bonissone et al. [20] and Y. Jin [63].

On the algorithmic front, many researchers have crafted dedicated MAs that incorporated problem-specific knowledge through specialized individual encodings, constraint representation, heuristics, operators, and/or control of EA parameters, into their algorithms so as to outperform all others on instances belonging to a particular class of problem. These early activities of research has powered the field of memetic computation research for more than a decade, where remarkable success on significant instantiations of MAs across a wide range of application domains have been reported, ranging from NP-hard problems such as Quadratic Assignment [140, 96], Traveling Salesman [16, 24, 88, 104, 156], set-covering [62], Scheduling & Routing [59, 90, 40, 26, 55, 94], Multidimensional Knapsack [58], software engineering

[7], to non-linear programming problems including highly multi-modal/constrained/highly-dimensional continuous optimization problems [113, 111, 145, 105], data mining/machine learning/artificial neural networks [138, 45], optimization problems in dynamic, uncertain and computational expensive environments [148, 147, 121] and multi-objective memetic optimization [59, 1, 43, 58, 92, 44, 29, 130]. In the MA literature, two basic forms of learning schemes are often discussed, namely, Lamarckian and Baldwinian learning. Lamarckian learning forces the genotype to reflect the result of improvement through learning by placing the improved individual back into the population to compete for reproduction. Baldwinian learning, on the other hand, only alters the fitness of the individuals and the improved genotype is not encoded back into the population. Preliminary yet important studies of the theoretical and empirical analysis to reveal the search behaviors of MAs (to enhance the understandings on the search mechanism of MA) have also been made [54, 109, 53]. Note that the summary of applications is by no means exhaustive but serves to highlight the wide coverage of MA research fields.

Despite the significant research on memetic algorithms, research on and with memetic computation has reached an impressive state, and there remains many open problems. New application areas are continually emerging as intriguing challenges for the algorithm. To date it remains debatable in the MA community on whether hybrid EAs represent the true realizations of memetic computation. Hybrid instantiations of MA although encompass characteristics of cultural evolution (in the form of local refinement) in the search cycle, may not qualify fully as a true evolving system according to Universal Darwinism, since the core principles of memetic transmission, variation and selection are generally missing [107]. Within this growing trend, which relies heavily on available a priori domain knowledge of a given problem instance to optimize, some researchers have opted for algorithms that learn and re-configure itself to adapt to the problem in hand while the search progresses. Some researchers in the recent years have focused on studying the design issues of MA that maintain good balance between exploration (population-level) and exploitation (individual learning) throughout

the search [80, 53, 106, 76, 108] on general or particular classes of problems. Others have tried to seek for adaptive memetic methodologies that satisfy part or all three principles of a basic evolving system. Examples are the multi-meme MA [75], hyper-heuristic [67, 120], meta-Lamarckian MA [113, 115], co-evolution & self-generation MAs [134, 76], multi-agent memetic computing [82, 49, 145], through reward assignment and/or capturing potential regular repeated features or latent patterns of the solution space. The importance of second-order evolution in memetic algorithm and the role that it could play in future evolutionary theory, complementary to the recent studies on the evolution of mutation rates, or more generally the evolution of evolvability [38] has also been explained and discussed in the context of dynamically changing environments by Paenke et al. [121]. Efforts to capture the latent patterns of the problem space across problem instances so as to establish inter-problem instance knowledge have also emerged recently. In the process of solving a repertoire of problem instances, memes culminate in the form of recurrence patterns or structures. From basic patterns or structures, more complex higher level structures, memeplexes, and/or higher order memes or meta-memes can arise [97].

3.1 Classification of Memetic Algorithms

In this section, Memetic Algorithms are summarized and categorized into different generations, based on the meme transmission mechanism of the algorithm.

3.1.1 First generation

The first generation of MA (Figure 3.1) refers to hybrid algorithms, a marriage between a population-based global search (often in the form of an evolutionary algorithm) coupled with a cultural evolutionary stage. This first generation of MA although encompasses characteristics of cultural evolution (in the form of local refinement) in the search cycle, it may not qualify as a true evolving system according to Universal Darwinism, since all the core principles of

CHAPTER 3. MEMETIC ALGORITHMS

Figure 3.1: First generation of MA

inheritance/memetic *transmission*, *variation* and *selection* are missing. This suggests why the term MA stirred up criticisms and controversies among researchers when first introduced in [36].

Algorithm 3 Memetic Algorithm (1st generation)

- 1: Generate an initial population
- 2: **while** Stopping conditions are not satisfied **do**
- 3: { **Evolutionary phase** }
- 4: *Evaluate* all individuals in the population
- 5: *Evolve* a new population using stochastic search operators
- 6: { **Individual learning phase** }
- 7: *Perform* individual learning
- 8: *Proceed* with Lamarckian or Baldwinian learning
- 9: **end while**

3.1.2 Second generation

Multi-meme [74], Hyper-heuristic [67] and Meta-Lamarckian MA [113] are referred to as second generation MA exhibiting the principles of memetic *transmission* and *selection* in their design. In Multi-meme MA, the memetic material is encoded as part of the genotype. Subsequently, the decoded meme of each respective individual / chromosome is then used to perform a local refinement. The memetic material is then transmitted through a simple

CHAPTER 3. MEMETIC ALGORITHMS

Figure 3.2: Second generation of MA

inheritance mechanism from parent to offspring. On the other hand, in hyper-heuristic and meta-Lamarckian MA, the pool of candidate memes considered will compete, based on their past merits in generating local improvements through a reward mechanism, deciding on which meme to be selected to proceed for future local refinements. Meme having higher rewards will have greater chances of being replicated or copied subsequently. For a review on second generation MA, i.e., MA considering multiple individual learning methods within an evolutionary system, the reader is referred to [115].

Algorithm 4 Memetic Algorithm (2^{nd} generation)

- 1: Generate an initial population
- 2: Initialize the meme pool
- 3: **while** Stopping conditions are not satisfied **do**
- 4: *Evaluate* all individuals in the population
- 5: *Evolve* a new population using stochastic search operators
- 6: *Select* the subset of individuals, Ω_{il} , that should undergo the individual improvement procedure
- 7: **for** each individual in Ω_{il} **do**
- 8: *Select* a meme from meme pool
- 9: *Perform* individual learning using the selected meme
- 10: *Proceed* with Lamarckian or Baldwinian learning
- 11: **end for**
- 12: **end while**

3.1.3 Third generation

Figure 3.3: Third generation of MA

Co-evolution and self-generation MAs introduced in [132] and [75] may be regarded as 3rd generation MA where all three principles satisfying the definitions of a basic evolving system has been considered. In contrast to 2nd generation MA which assumes the pool of memes to be used being known *a priori*, a rule-based representation of local search is co-adapted alongside candidate solutions within the evolutionary system, thus capturing regular repeated features or patterns in the problem space.

3.2 Popular Instances of Memetic Algorithm

Since the original idea of Memetic Algorithm proposed by Moscato in [100], many variants of memetic approaches have been proposed over the years. In what follows, several more prominent forms of Memetic Algorithms are briefly described.

- Canonical (single objective) MA: the original form of MA. In Canonical MA, all the individuals in the population will undergo individual learning.

- **Multi-objective MA:** the field of multi-objective started with some pioneer work of Rosenberg in the late 60s. Multiple objectives are to be optimized at the same time and solution will be in the form of pareto-front. In [31], Coello made a comprehensive survey of evolutionary-based multi-objective optimization techniques while Knowles and Corne [71, 72] have spent some efforts analyzing and comparing different Memetic Algorithms for multi-objective optimization.
- **Constrained MA:** to solve the constrained optimization problems. Constrained MA is the marriage between constrained EAs and individual learning methods which are specially designed to locate feasible solutions efficiently [51].
- **Parallel MA:** inspired by the explosion and availability of parallel, multiprocessing computers, Parallel MA is an ingenious extension on basic MA. In parallel MA, multiple populations are evolved at the same time. This has been shown to provide better performance than single population-based algorithms, even when the parallel MA is simulated sequentially. Research in this area has generally concentrated on studying the migration of information under different parallel structures and topologies. In [128], Resende presented a good survey of parallel meta-heuristic methods for combinatorial optimization.

3.3 Research interest

This thesis concentrates on the second generation Memetic Algorithm. It is worth emphasizing that the research interest lies in the mechanism that governs the synergies between global search and individual learning strategies and less on the search mechanisms of particular global search or local learning techniques. In particular, the following important design issues of the second generation of MA are investigated:

- (i) Individual learning frequency, hereby denoted as f_{il} : defines how often should local learning be applied. f_{il} can be represented as a percentage of the population, i.e., the

CHAPTER 3. MEMETIC ALGORITHMS

percentage of individuals in the population that undergoes local learning, or the ratio of evolutionary to individual learning, i.e., in how many generations of global search should local learning be conducted. Alternatively, f_{il} can be replaced with the individual learning probability, P_{il} , which defines the probability at which each individual in the population should undergo local learning.

- (ii) Individual learning intensity, t_{il} : defines how much computational budget should be allocated to single local learning process. Note that t_{il} may be represented in terms of number of the objective function evaluations or time budget.
- (iii) Subset of solution undergoing individual learning, Ω_{il} : represents the subset of the solution population that undergoes local learning?
- (iv) Individual learning method: which among a given set of available local learning strategies should be employed on a given problem at hand.

On the issue of individual learning frequency, Hart [53] investigated various configurations of the individual learning frequency at different stages of the MA search and showed that applying individual learning to just a small subset of population will be more efficient. In contrast, it was also shown in [80] that it may be worthwhile to apply local learning on every individual if the computational complexity of the local search is relatively low. This highlights that adaptation mechanism to adjust the individual learning frequency would offer great benefit to the search performance. On the issue of selecting appropriate individuals among the EA population that should undergo local search, fitness-based and distribution-based strategies were studied for adapting the probability of applying local search on the population of chromosomes in continuous parametric search problems with Land [83] extending the work to combinatorial optimization problems, while Bambha et al. [15] introduced a simulated heating technique for systematically integrating parameterized local search into evolutionary algorithms to achieve

CHAPTER 3. MEMETIC ALGORITHMS

maximum solution quality. Goldberg and Voessner on the other hand attempted to optimize the performance of global-local search hybrid in the context of a fixed local search time that leads to solution with acceptable targets on simplistic synthetic functions [46]. The performance of MA search is also greatly affected by the choice of neighborhood structures. The effect of neighborhood structures on evolutionary search performances was studied by Yao in the context of simulated annealing [150]. Krasnogor [78], [77] investigated how to change the size and the type of neighborhood structures dynamically in the framework of multi-meme memetic algorithms where each meme generates biases of unique neighborhood structure, acceptance rule and local search intensity. Various schemes for choosing among multiple local learning procedures or memes during the memetic algorithm search in the spirit of Lamarckian learning, otherwise, known as meta-Lamarckian learning was also addressed in Ong et. al. [113]. Based on the terminology provided in [115], several new adaptive and self-adaptive memetic algorithms were proposed by Smith [133], Liu et. al. [90] and Caponio et. al. [27] and reported in the recent special issue on MA [114].

Chapter 4

A Study on the Design Issues of Memetic Algorithms

In this chapter, we conduct extensive experimental studies on the impact of each individual design issue and their relative impacts on memetic search performances by means of three commonly used synthetic problems (Appendix A.1, A.8, A.11). From the empirical studies obtained, we attempt to reveal the behaviors of several MA variants to enhance our understandings on the field.

For the sake of readability, we repeat some of the definitions and notations presented in the previous chapters that are used in the present chapter:

Definition 1: Individual learning frequency, f_{il} , defines the proportion of an EA population that undergoes individual learning. For instance, if po is the EA or MA population size, the number of individuals in the population that undergoes individual improvement is then $f_{il} \times po$.

Definition 2: Individual learning intensity, t_{il} , is the amount of computational budget allocated to an iteration of individual learning, i.e., the maximum computational budget allowable for individual learning to expend on improving a single solution.

Definition 3: The subset of population that should undergo the individual learning procedure is denoted by Ω_{il} , where cardinality $|\Omega_{il}| = f_{il} \times po$.

The frequency and intensity of individual learning directly define the degree of evolution (exploration) against individual learning (exploitation) in the MA search, given a fixed limited

CHAPTER 4. A STUDY ON THE DESIGN ISSUES OF MEMETIC ALGORITHMS

computational budget. Clearly, a more intense individual learning provides greater chance of convergence to local optima but limits the amount of evolution that may be expended, which may decrease the chance of locating global optimum. On the other hand, inadequate individual learning would suffer from common draw back of most stochastic search methods, i.e., results attained are lack of accuracy. Therefore, care should be taken when setting these two parameters to balance the computational budget allocated to the two objectives: exploration versus exploitation. If not all individuals of the population undergo individual learning, it becomes necessary to also consider which subset of individuals to improve so as to maximize the utility of MA search. Last but not least, the individual learning procedure used also favors a different neighborhood structure, hence the need to decide which meme or memes to use for a given optimization problem at hand.

We first describe the experimental procedures used in this study, followed by a presentation and analysis of the experimental results obtained from studies on the mechanism of each design issue as well as their relative impacts on memetic search performances.

4.1 Experimental Procedure

In this section, the experimental procedure used to study the various design issues of MAs is described. To begin, the impacts of choice of population-based and individual learning procedures used in creating an MA is investigated. Among the candidate population-based approaches considered here are i) Simple Genetic algorithm (GA) [35], and ii) Differential Evolution (DE) [137] and iii) Evolutionary Strategy (ES) [11]. Individual learning procedures or memes considered are the i) procedure of Davies, Swann, and Campey with Gram-Schmidt orthogonalization (DSCG) [131], ii) Broyden-Fletcher-Goldfarb-Shanno method (BFGS) [155] and iii) Lagrangian interpolation strategy [131], which are representatives of first and zeroth order exact individual learning methods commonly found in the literature. Note that there are a total of nine potential hybrid global-local search combinations or MA variants.

CHAPTER 4. A STUDY ON THE DESIGN ISSUES OF MEMETIC ALGORITHMS

Table 4.1: Benchmark functions used in the study

Function	Range	Characteristics	
		Epistasis	Multi-modality
F_{Sphere}	$[-100, 100]^{30}$	none	none
F_{Ackley}	$[-32, 32]^{30}$	weak	moderate
$F_{Weierstrass}$	$[-0.5, 0.5]^{30}$	weak	high

In addition, experiments for various configurations of t_{il} and f_{il} are conducted, where $f_{il} \in [0, 1]$ and t_{il} is some integer value. Here, t_{il} is limited to the set $\{100, 200, 300, 400, 500\}$ in the present study. For each setting of t_{il} , experiments are also conducted for different values of f_{il} at 0.1, 0.3, 0.5, 0.7, 0.9 and 1.0. Further, the effects of three commonly used schemes for selecting the subset of individuals, Ω_{il} , that will undergo individual learning is examined.

In our study, three commonly used continuous parametric benchmark test problems already extensively discussed in the literature are considered here to study the effects of the diverse MA design issues and their configurations. They are Sphere(A.1), Ackley(A.8) and Weierstrass(A.11) functions. These benchmark problems used represent classes of Unimodal & Multimodal, Epistatic & Non-Epistatic test functions.

For more details on the benchmark functions, the readers are referred to Appendix A provided at the end of this report. Table 4.1 summarizes these functions with their notable characteristics. Note that in this study, 30-dimensional instances of the benchmark functions are considered.

Table 4.2: MA parameters setting

General parameters	
Population-based methods	GA, DE and ES
Individual learning procedures	DSCG, Lagrange and BFGS
Stopping criteria	300,000 evaluations or convergence to global optimum
Population size	50
Genetic Algorithm parameters	
Encoding scheme	Real number
Selection scheme	Roulette wheel
Crossover operator	Two point crossover $p_c = 0.7$
Mutation operator	Gaussian mutation $p_m = 0.03$
Differential Evolution parameters	
Crossover probability	$p_c = 0.9$
Evolutionary Strategy parameters	
Selection method	$\mu + \lambda, \mu = 50, \lambda = 100$
Mutation operator	Gaussian mutation
Individual Learning parameters	
Individual learning intensity t_{il}	300

4.2 Empirical Results

This section presents the experimental results of various MAs in optimizing the three benchmark functions. For the sake of readability in the discussion, the numerical results obtained are grouped as three subsections. In the first subsection, the effects of different population based and individual learning procedures are investigated. The second subsection illustrates the effects of individual learning frequency, f_{il} and individual learning intensity t_{il} . Finally, the last subsection considers the effects of individual subset selection schemes used for selecting subset Ω_{il} .

All results presented are the average of 25 independent runs. Each run continues until the global optimum was found (i.e., the fitness function error $< 10^{-8}$) or a maximum of 300,000 function evaluations was reached. For each run, the initial population is sampled randomly within the search range. Table 4.2 summarizes the parameter settings of the evolutionary algorithms and individual learning procedures used in this study.

Table 4.3: Optimization results of multistart individual learning on the benchmark problems. For Sphere function, the average number of evaluations incurred by the algorithms in locating the global optimum is reported instead.

Function	MS-BFGS	MS-DSCG	MS-Lagrange
F_{Sphere}	(95)	(1300)	(1147)
F_{Ackley}	19.3993	1.37624	2.2336
$F_{Weierstrass}$	49.9721	13.6501	20.3064

4.2.1 Impact of different Population-Based and Individual Learning procedures on MA

First, the results of diverse MA variants with different population based and individual learning procedures are presented. From a survey of the literature, it is worth noting that among the three population based search candidates considered, Genetic Algorithm remains to be most extensively used for forming MAs [113]. Hence it becomes common for the term hybrid-GA to be used interchangeably with Memetic Algorithm in many existing works. Recently, [89] also considered the use of Differential Evolution as one of the population based search method in MAs. Nevertheless, to the best of our knowledge, a smaller number of papers have dealt with the hybridization of evolutionary strategies and local search.

The individual learning candidates used are representatives of first and zeroth order conventional numerical methods commonly found in the literature. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is a Quasi-Newton method, which determines the search directions based on the first order derivation of the fitness functions. Lagrange strategy, on the other hand, uses a second-order interpolation to generate the search points, and generally works better if the fitness surface is quadratic or closed to quadratic. Last but not least, procedure of Davies, Swann, and Campey with Gram-Schmidt orthogonalization (DSCG) executes a line search in each dimension independently. This individual learning procedure has been shown to work extremely well on search problems having low epistasis.

Table 4.3 presents the optimization results of different multistart individual learning procedures on the benchmark functions. These are used as base-line results with which other MAs may be compared. The average number of evaluation calls incurred by the different variants of MAs to converge at the global optimum of the benchmark functions or the best solution quality obtained are summarized in Figures 4.1-4.3. First, the results in Figure 4.1 on the quadratic Sphere function are examined. On this function, it is observed that the BFGS procedure outperforms the other two individual learning counterparts, regardless of the population based method used to form the MAs. The Sphere function is a convex, continuous and unimodal function, hence the gradient vector at any decision point would direct the search to the global minimum. Since BFGS makes use of the gradient information in guiding its search, it converges rapidly to the global optimum of the Sphere function by simply moving in the negative gradient direction. Hence its search performance is unaffected by the choice of global method used and is capable of converging to the global optimum regardless of the starting point used. This also explains why the results obtained by a stochastic multistart individual learning is competitive to those obtained by the hybrid GA form of MA.

Consider next the search convergence performances of the MAs on the Ackley function summarized in Figure 4.2. Since Ackley is a multimodal function, the use of gradient information in BFGS causes it to getting stuck at some local optima hence the results obtained shows that all the MAs employing BFGS as the individual learning procedure fails to locate the global optimum successfully within the allocated computational budget. On the other hand, the functional form of the Ackley function can be easily approximated using a second-order model. As a result, it can be observed in Figure 4.2 that by means of quadratic approximation, the Lagrange method generally outperforms the other counterparts on Ackley, regardless of the type of stochastic population based method used.

On the Weierstrass function, none of the MAs managed to converge to the global optimum within the maximum computational budget allowable. The best mean fitness values attained

by each MA across 25 runs are summarized in Figure 4.3. The results also indicate that both the BFGS and Lagrange interpolation individual learning procedures do not fare well on this problem. This is likely due to a mismatch between the neighborhood search structures of BFGS and Lagrange interpolation to the fitness landscape of Weierstrass. DSCG, on the other hand, fares best on this problem.

Further, from the results obtained, neither the population based nor individual learning procedure nor a particular synergy of MA displayed superiority on all the three problems considered. However, it is worth noting that the choice of individual learning procedure appears to be the key of success of the MA search. This greatly highlights the importance of selecting suitable meme for the given optimization problem at hand as discussed extensively in [115], [113]. Overall, DSCG works generally better on all three problems considered; a conclusion also obtained in [115] and [113].

Figure 4.1: Search performance of different population based-individual learning or global-local MAs on 30D Sphere function

CHAPTER 4. A STUDY ON THE DESIGN ISSUES OF MEMETIC ALGORITHMS

Figure 4.2: Search performance of different population based-individual learning or global-local MAs on 30D Ackley function

Figure 4.3: Search performance of different population based-individual learning or global-local MAs on 30D Weierstrass function

CHAPTER 4. A STUDY ON THE DESIGN ISSUES OF MEMETIC ALGORITHMS

Finally, we perform two-way ANOVA tests on the relative ranks of different individual learning methods in combining with various stochastic strategies (including random search). For example, for Sphere function, the ranks of BFGS, DSCG and Lagrange, when combining with random search (see Table 4.3) is 1, 3, and 2, respectively. It is also worth noting that rank values are obtained based on pair-wise t-test between each pair of results. Tables 4.4 - 4.6 present the test results for Sphere, Ackley and Weierstrass functions, respectively. At 95% confidence interval, it is concluded that the choice of population based method in MA have little significance on the relative performance among three individual learning methods considered, across all the three benchmark problems.

Table 4.4: Friedman (non-parametric) test for Sphere function. (Null hypothesis: the relative performances among individual learning methods are independent with the choice of stochastic search strategies)

	BFGS	DSCG	Lagrange
MS	1	3	2
DE	1	2	3
ES	1	3	2
GA	1	3	2
Results	$W = 0.81, Q = 6.50, p \approx 0.0388$		

CHAPTER 4. A STUDY ON THE DESIGN ISSUES OF MEMETIC ALGORITHMS

Table 4.5: Friedman (non-parametric) test for Sphere function. (Null hypothesis: the relative performances among individual learning methods are independent with the choice of stochastic search strategies)

	BFGS	DSCG	Lagrange
MS	3	1	2
DE	3	2	1
ES	3	2	1
GA	3	2	1
Results	$W = 0.81, Q = 6.50, p \approx 0.0388$		

Table 4.6: Friedman (non-parametric) test for Sphere function. (Null hypothesis: the relative performances among individual learning methods are independent with the choice of stochastic search strategies)

	BFGS	DSCG	Lagrange
MS	3	1	2
DE	3	1	3
ES	3	1	2
GA	3	1	2
Results	$W = 1.00, Q = 8.00, p \approx 0.0183$		

4.2.2 Impact of Individual Learning frequency and intensity on MA

Based on the results presented in the previous subsection, the MA based on a synergy of ES and DSCG or otherwise labeled in this chapter as ES-DSCG is observed to obtain highest average ranking on the three benchmark functions. As a result, the ES-DSCG is employed for studying the other design issues in the rest of this study.

In this subsection, the effects of individual learning intensity t_{il} and individual learning frequency f_{il} on MA search performance is studied. t_{il} , defines the maximum computational budget allowable to each individual learning procedure. f_{il} , on the other hand, defines the proportion of individuals in each population that will undergo individual learning. Hence for a f_{il} configuration of 0.5, only half of the MA population undergoes individual learning.

A larger value of t_{il} gives more computational budget or greater emphasis on improving each individual chromosomes, thus leading to higher level of precision or accuracy in the solution quality. Similarly, with large f_{il} , i.e. $f_{il} \rightarrow 1$, more individuals in the current population will have the opportunity to undergo individual improvement, giving a higher chance of reaching the local or global optimum. In practice, however, the maximum computational budget allowable for an MA search is often limited. Hence even though more intense individual learning, i.e., a large t_{il} or f_{il} , provides greater chance of convergence to high precisions at the local or global optimum, the amount of evolution that may be expended without incurring excessive computational resources becomes limited. As a result of the scarce computational resources available in practice, the MA could fail to hit the region or basin where the global optimum lies before the potential of individual learning could start to bite.

Figures 4.4 - 4.6 present the search performance of ES-DSCG on the Sphere, Ackley and Weierstrass problems, for different combinations of t_{il} and f_{il} configurations, i.e., $t_{il} \in \{100, 200, 300, 400, 500\}$ and $f_{il} \in \{0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$. Note that in the experiments conducted, only the elite ($f_{il} \times po$) chromosomes of the current population will undergo individual learning or individual learning improvement (po is the MA population size). Figures 4.4 and

4.5 indicate that all MAs converged to the global optimum of the Sphere and Ackley problems, hence only the number of evaluations incurred have been presented. On the Weierstrass problem, none of the MA converged to the global optimum, thus the average best fitness values obtained across 25 independent runs are reported instead. From these Figures, one can also observe that all the MA variants investigated performs poorly on small values of individual learning intensity, i.e., $t_{il} = 100$ or 200 . Clearly, this is the result of lack of sufficient computational budget provided for individual learning to generate any positive impact on search.

On Sphere function, t_{il} at 300 evaluations looks sufficient for generating the superior search performance among the MAs. Any further increase in t_{il} actually results in detrimental effects on the MA search performances, see Figure 4.4. Further, it makes good sense that MA performs better with a smaller f_{il} since any starting point in the search space will lead the MA to the global optimum, especially if sufficient computational budget, t_{il} , is provided.

Next, consider the MA performance on Ackley function with Figure 4.5 reflecting the trade-off between t_{il} and f_{il} . When f_{il} is large, i.e., approaches 1.0, MAs with $t_{il} = 300$ or 400 perform more efficiently over 500 evaluations. In contrast, MAs at $t_{il} = 500$ fares better than those counterparts with $t_{il} = 300$ or 400 and low f_{il} configurations. Clearly, this agrees with our earlier hypothesis that under some given fixed computational budget, a good balance between t_{il} and f_{il} is necessary to ensure superior search performance in the MA.

On the Weierstrass function, the results summarized in Figure 4.6 indicate the trends of improving solution quality with increasing individual learning intensity. On such a problem, the results indicate that one should provide a larger individual learning intensity for maximum solution quality and search efficiency. It is also worth noting that the $f_{il} = 0.5$ works best over the range of different t_{il} which implies that for complex problems, it may be appropriate to undergo individual learning on half of the MA population.

Figure 4.4: Search performance of MA with different configurations of (f_{il}, t_{il}) on 30D Sphere function

Figure 4.5: Search performance of MA with different configurations of (f_{il}, t_{il}) on 30D Ackley function

Figure 4.6: Search performance of MA with different configurations of (f_{il}, t_{il}) on 30D Weierstrass function

4.2.3 Impact of individual subset selection scheme, Ω_{il} , on MA

In this section, three schemes for selecting the subset of chromosomes in the EA or MA that would undergo individual learning are examined. The “random-walk” scheme provides all chromosomes in each population equal chances of undergoing individual learning. In contrast, the “best” scheme assumes that fitter chromosomes have better chances of converging to the global optimum. Last but not least, the “stratified” scheme is a compromise between the two where individual learning is only performed on individuals that are regarded as unique. The uniqueness of individuals in a MA population may be measured by various means and may be categorized into those for genotype, phenotype or fitness level. Here uniqueness or the diversity based on fitness values is considered. Table 4.7 provides a summary of these schemes.

In studying the different selection schemes, the individual learning frequency is fixed at 0.5 in the experiments, i.e., half of the MA population undergoes individual learning. On the other hand, further experiments for $t_{il} = 200, 300$ or 400 are conducted to study the relative impact

Table 4.7: Chromosome selection schemes

Scheme	Description
Random-Walk(rand)	Chromosomes undergoing individual improvement are randomly chosen from the current population using uniform sampling without replacement
Stratified(strat)	The population is sorted according to their fitness values. Subsequently, the individuals are divided into n equal range where n is the number of individuals to be chosen to undergo individual learning. Next, from each range, one individual will be selected.
Best(best)	The best n individuals will undergo individual learning.

of the selection scheme and t_{il} on MA search performance.

Figures 4.7 - 4.9 summarize the search performance of the MAs for various selection schemes on optimizing the benchmark problems. Figure 4.7 indicates that the stratified subset selection scheme works best on the Sphere function. The result appears to contradict our initial hypothesis that the “best” scheme would perform superior over the other two counterpart schemes since on the unimodal Sphere function, every candidate solution should converge precisely to global optimum if sufficient individual learning intensity or computational budget is provided. Upon greater analysis, it is realized that this is due to the search structure of DSCG that does not exploit gradient information in its search. Hence a starting point that is nearer to the global optimum in a quadratic sense does not translate to faster convergence in an MA based on ES-DSCG. On the Weierstrass function, the results in Figure 4.9 display little difference in performances for different selection schemes. On the other hand, the “best” scheme significantly outperforms all others on the Ackley function, see Figure 4.8. Across the three benchmark problems considered, the choice of selection scheme in the MA appears to have greatest impact on Ackley function. The results in the figures also indicate that the selection scheme used has little impact on the relative performances of the MA for different individual learning intensities. Hence, the configuration of individual learning intensity is generally unaffected by the choice of selection scheme in the MA.

Figure 4.7: Search performance of MA for diverse individual subset selection schemes and t_{il} on 30D Sphere function

Figure 4.8: Search performance of MA for diverse individual subset selection schemes and t_{il} on 30D Ackley function

Figure 4.9: Search performance of MA for diverse individual subset selection schemes and t_{il} on 30D Weierstrass function

4.3 Conclusions

This chapter discussed several important design issues of Memetic Algorithms and presented the results obtained from a systematic study on both the impact of each individual design issue as well as their relative impact on memetic search performances using three commonly used benchmark functions (Appendix A.1, A.8, A.11). From the empirical results, the behaviors of MAs are analyzed and discussed to explain why some synergies of stochastic population-based and individual learning optimizers led to successful Memetic Algorithms while some did not. It appeared that the choice of suitable meme for the given optimization problem at hand was more important compared to the choice of population based search when designing MA. The MA search performance is also greatly affected by the configurations of individual learning intensity and frequency. Further, a good balance between t_{il} and f_{il} must be maintained to ensure good efficiency in the MA search algorithm, under some fixed computational budget. In addition, it remains inconclusive which individual selection scheme for individual learning

CHAPTER 4. A STUDY ON THE DESIGN ISSUES OF MEMETIC ALGORITHMS

works best for a problem at hand. Nevertheless, both the “stratified” and “best” scheme have generated good and robust MA search performances. Last but not least, the configurations of individual learning intensity is generally unaffected by the choice of individual selection scheme.

It is clear that each of these parameters induces great influences on the impact of other parameters on the MA performance. On the other hand, configurations that render reasonable search performance differ from one problem to another and it is computationally inefficient to conduct exhaustive experiments on the algorithmic parameter domain. With the restricted amount of theory currently available for choosing the appropriate configuration of MA that best matches a black box problem, it becomes a popular practice to opt for algorithms that learn and redesign itself to adapt to the problem at hand while the search progresses. In the next chapter, an Adaptive Cellular Memetic Algorithm will be presented, where the set of individuals that will undergo individual learning is adapted online while the search progresses.

Chapter 5

Cellular Memetic Algorithms

An evolutionary algorithm (EA) without any structure is usually referred to as a panmictic EA and the majority of memetic algorithm have been implemented in this fashion. On the other hand, a popular way of structuring the population is by recurring to a lattice-like topology in which individual interact only with their nearest neighbors. A cellular GA (CGA) is a prime exemplar of this technique [91, 123]. Structured GAs, not only favor a parallel implementation due to the decentralized management of the populations, but also they are generally known to provide better sampling of the search space and therefore perform well in many cases [21, 6, 139]. Earlier works [101, 14] have demonstrated the efficacy of CGA for complex optimization tasks. The small overlap of neighborhoods and spatial structure of CGA induces a slow diffusion of information throughout the whole population. This has the advantage of preventing, to a large extend, premature convergence. The drawback of CGAs, however, is that significantly greater effort is often required before converging to near global optimum. In this respect, CGA has great potential for performance enhancement by equipping it with better exploitation capabilities. Thus the synergy between CGA and local search procedures was investigated by Folino et al. in [39]. By combining CGA with a random walk local search, the authors achieved better convergence rate on the satisfiability problems. Later, Alba et. al. [3] formalized this class of algorithms under a framework named cellular memetic algorithm (CMA). The work of Alba, [3], among many others such as [53, 47, 77, 115], have highlighted

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

some important design issues that must be taken into consideration when designing hybrid evolutionary-local search algorithms, including CMA.

In this chapter, the main contribution is a diversity-based adaptive mechanism that controls the number of individuals that undergo individual learning at each generation, thus helping to maintain good diversity of the population. The chapter begins with a study on the canonical Cellular MA or CMA in short. To facilitate a healthy balance in exploration (global search) and exploitation (individual learning) under a fixed computational budget, a Cellular MA with adaptive capability is proposed. In contrast to the canonical MAs and CGAs, the Adaptive Cellular MA (ACMA) contains autonomic mechanisms for controlling the individual learning frequency, i.e., how often should individual learning be applied in each search generation, and defining which individuals in the population to undergo individual learning throughout the CMA evolutionary search process. In ACMA, adaptation is carried out based on the distribution or diversity measure of a population. In particular, fitness value is used to estimate the level of similarity between individuals or chromosomes and as a basis for applying individual learning on different groups of individuals selectively, thus offering ease of implementation. Empirical studies on both CMA and ACMA are conducted using a series of commonly used benchmark test functions. Results obtained show that ACMA converges to competitive solutions at significantly lower computational cost compared to the canonical CMA and MA. Furthermore, ACMA is shown to display better robustness in terms of solution quality on the problems considered. This chapter is organized in the following manner. Section 5.1 presents a brief overview of the Cellular GA. The canonical Cellular MA and the proposed Adaptive CMA are presented in Section 5.2. Section 5.3 summarizes our empirical studies on continuous parametric benchmark functions and provides a comprehensive quantitative/statistical comparison of CGA, multi-start individual learning, CMA and ACMA. The search performances of the various algorithms in terms of solution quality, computational effort, and solution precision are also reported in the section followed by analysis of the results obtained. Finally, Section 5.4 concludes the chapter.

5.1 Cellular Genetic Algorithm

Cellular GA has a decentralized structure where each chromosome can only interact with other chromosomes within a particular neighborhood (see Figure 5.1). The overlapped small neighborhoods of CGA help facilitate search exploration and maintain good solution diversity through a slow diffusion of information across the entire population. Exploitation then takes place inside each neighborhood through the genetic operators. Consider here the two-dimensional grid, which represents one of the most commonly used topology in CGAs. As illustrated in Figure 5.1, the CGA has a neighborhood structure whereby each neighborhood is made up of five individuals, i.e., the individual considered at grid position (x, y) , together with its immediate neighbors in north, east, west, and south orientations. This is commonly known as the Von-Neumann neighborhood. The pseudo-code of the CGA is outlined in Algorithm 5. In the CGA, each individual undergoes the standard genetic operators including crossover and mutation. In contrast to the canonical GA, an individual can only mate with partner(s) in the Von-Neumann neighborhood during reproduction. Subsequently, the fitness value of the child individual is computed. The offspring then replaces the original individual at its grid location if it results in a more favorable fitness value. Even though each individual can only interact with its immediate neighbors, there is no complete isolation of individuals since the neighborhoods of the grid structure overlap one another. This linked structure allows for gradual diffusion of information across the CGA population.

In Algorithm 5, *WIDTH* and *HEIGHT* denote the width and height of the population grid. One main advantage of CGA is the internal spatial structure that allows for fitness and genotype diversity to be sustained for a larger number of generations [136]. The lower selection pressure of the CGA makes it converge at a rate slower than a panmictic GA. This property of the CGA has been shown to help reduce the risk of premature convergence in the evolutionary search on some problems. Existing applications of parallel CGAs (fine-grained) such as flow shop scheduling problems have also shown that they generally converge slower than the

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Figure 5.1: Neighborhood structure in Cellular Genetic Algorithm

Algorithm 5 Cellular Genetic Algorithm

```

1: procedure CELLULARGA
2:    $pop = \text{Create-Grid}((WIDTH * HEIGHT))$ 
3:   for  $x = 1$  to  $WIDTH$  do
4:     for  $y = 1$  to  $HEIGHT$  do
5:       Initialize  $pop(x, y)$ 
6:        $pop(x, y).fitness = \text{Evaluate}(pop(x, y))$ 
7:     end for
8:   end for
9:   while (termination condition is not satisfied) do
10:     $oldpop = pop$ 
11:    for  $x = 1$  to  $WIDTH$  do
12:      for  $y = 1$  to  $HEIGHT$  do
13:         $parent1 = oldpop(x, y)$ 
14:         $parent2 = \text{Select}(\text{Neighbors}(x, y))$ 
15:         $child = \text{Crossover}(parent1, parent2)$ 
16:        Mutate( $child$ )
17:         $child.fitness = \text{Evaluate}(child)$ 
18:        if ( $child.fitness < oldpop(x, y).fitness$ ) then
19:           $pop(x, y) = child$ 
20:        end if
21:      end for
22:    end for
23:  end while
24: end procedure

```

parallel island model, but often generate better solution quality [73]. Alba [4] also investigated various population update policies in CGA and the effect of such policies on search exploration/exploitation tradeoffs. The effect of grid structure on the performance of CGA was also studied in [2]. Nevertheless, the ability to reduce the effect of premature convergence brings about the issue of slow convergence rate in the search. This is identified as a challenge for CGA especially when used in solving optimization problems that are computationally demanding.

5.2 Cellular Memetic Algorithm

This section begins with the description of a canonical CMA. An efficient adaptive CMA is then proposed for effective selection of suitable candidates in the CMA population to undergo individual learning improvement in each search generation.

5.2.1 Canonical Cellular Memetic Algorithm

The pseudo code of a canonical CMA is outlined in Algorithm 6. For the sake of readability, the canonical CMA is presented as two separate phases. Phase I involves the CGA and is hence similar to that of Algorithm 5. After phase I, the individual then undergoes individual learning procedure in phase II. To avoid redundant individual learning which is often considered to be computationally intensive, a tabu list of black-listed past individuals that did not benefit from individual learning procedure is maintained during the search. To avoid confusion, it is worth noting that two concepts of locality (neighborhood) exist in the CMA. Neighborhoods can be defined in the context of population grid structure or that of the solution space. The individual learning procedure considers searching in the neighborhood of an individual's solution search space while selection of neighbor(s) involves the Von-Neumann neighborhood, which depends on the population grid or topology. Since they refer to a totally different neighborhood space, individuals that are structured near to each other in the topology do not translate to neighbors in the solution search space, i.e., they may be far apart in the solution search space.

5.2.2 Adaptive Cellular Memetic Algorithm

The canonical CMA employs individual learning on all individuals of the population. While the canonical model fully exploits all individuals in the population, this is a very computationally intensive and inefficient search process. At the same time, exhaustive individual learning may lead to ineffective search due to premature fall in diversity during the MA search. Recent study on the effect of individual learning frequency or probability of individual learning, i.e., how many chromosomes to undergo individual learning in an EA generation, highlights the need for adaptation of this parameter in order to attain good MA search performance [53]. In the following subsections, two schemes for selecting the individuals undergoing individual learning are studied. Based on the taxonomy in [115], both algorithms represent forms of static adaptation approach.

- Cellular Memetic Algorithm with random selection scheme (CMAR)

The most basic scheme for selecting the subset of chromosomes that will undergo the improvement procedure is random selection. In the individual learning phase, chromosomes are randomly selected to undergo individual learning - this does not adapt but has the advantage of giving all the chromosomes in the CMA population equal chance to undergo improvement.

- Stratified-Adaptive Cellular Memetic Algorithm (ACMA)

An alternative scheme for selecting suitable individuals to undergo individual learning in the CMA during run time. In particular, adaptation is carried out based on the distribution or diversity measure of a population. The diversity of a GA population may be measured by various means and may be categorized into those meant for genotype, phenotype or fitness level.

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Algorithm 6 Cellular Memetic Algorithm

```

1: procedure CELLULARMA
2:   Initialize-Meme-Pool;
3:    $pop = \text{Create-Grid}((WIDTH * HEIGHT))$ 
4:   for  $x = 1$  to  $WIDTH$  do
5:     for  $y = 1$  to  $HEIGHT$  do
6:       Initialize  $pop(x, y)$ 
7:        $pop(x, y).fitness = \text{Evaluate}(pop(x, y))$ 
8:     end for
9:   end for
10:  while (termination condition is not satisfied) do
11:     $generation = generation + 1$ 
12:     $oldpop = pop$ 
13:    for  $x = 1$  to  $WIDTH$  do
14:      for  $y = 1$  to  $HEIGHT$  do
15:        /* Phase I: CGA */
16:         $parent1 = oldpop(x, y)$ 
17:         $parent2 = \text{Select}(\text{Neighbors}(x, y))$ 
18:         $child = \text{Crossover}(parent1, parent2)$ 
19:         $child = \text{Mutate}(child)$ 
20:        /* Phase II: Individual Learning */
21:        if ( $generation \bmod \theta = 1$ ) then
22:           $child = \text{Local-Improvement}(child)$ 
23:        end if
24:         $child.fitness = \text{Evaluate}(child)$ 
25:        if ( $child.fitness < oldpop(x, y).fitness$ ) then
26:           $pop(x, y) = child$ 
27:        end if
28:      end for
29:    end for
30:  end while
31: end procedure

```

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Our approach maintains appropriate degree of individual learning and solution diversity in the evolutionary search based on a fitness uniform selection scheme. The adaptive individual learning scheme is outlined in Algorithm 7. In ACMA, fitness values are used to estimate the level of similarity between individuals or chromosomes and as a basis for applying individual learning on different groups of individuals selectively. For every search generation, the number of groups is calculated by $\alpha * n$, where α is a user-predefined percentage value and n denotes the population size, i.e., $n = WIDTH * HEIGHT$. Using the fitness bound of a population, i.e., the maximum and minimum fitness values of the current CGA population (hereby denoted as *maxfitness* and *minfitness*, respectively), the fitness space is divided into $\alpha * n$ number of uniform intervals. This is referred to as *fitnessrange* in Algorithm 7.

Algorithm 7 Adaptive individual learning

```

1: if currentgeneration %  $\theta = 0$  then
2:   /* Initialization will create  $\alpha * n$  empty groups */
3:   fitnessrange = (maxfitness – minfitness) / ( $\alpha * n$ )
4:   if fitnessrange >  $\varepsilon$  then
5:     for each individual indv in the current population pop do
6:       groupindex = (indv.fitness – minfitness) / fitnessrange
7:       assign indv to a group based on the groupindex
8:     end for
9:     for each non-empty groups g do
10:      i = random number from 1 to capacity of g
11:      if member i of group g is not in tabu-list then
12:        childls = Individual-learning(member i of group g)
13:        if childls.fitness is better than fitness of member i then
14:          replace member i with childls
15:        else
16:          add member i to tabu-list
17:        end if
18:      end if
19:    end for
20:  else
21:    replace the entire population with random search points except the elite individual
22:  end if
23: end if

```

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Subsequently, each individual in the population is then assigned to one of the $\alpha * n$ groups based on its fitness value. Henceforth, only one individual from each group undergoes individual learning. This individual may be selected either randomly or through some other selection schemes. In this manner, the likelihood of applying individual learning on similar individuals can be reduced. Here we also consider performing individual learning on the elite individual, since it is most likely to be near the best optimum solution. To minimize the tendency for redundant individual learning, a black-listing mechanism is also incorporated using a tabu-list.

In summary, the number of individual learning made in each search generation is constrained to a maximum of $\alpha * n + 1$, i.e., the number of fitness groups plus the elite individual. Thus, the actual number of individual learning may differ for every generation, as it is possible that none of the individual has a fitness value that falls under a particular group range. When this happens, no individual learning is performed for this fitness group. In this way, the ACMA performs exploration through its neighborhood structure governed by the population grid used. The degree of exploitation is defined by the amount of individual learning made which is adapted by means of individual grouping in the fitness space. In the design of memetic algorithms, it is common knowledge that a good balance between exploitation and exploration is the key to achieving good search performance. Hence, to maintain good diversity and avoid premature convergence, random individuals are introduced into the population whenever the search exhibits signs of premature convergence. Here, greater exploration is favored by introducing randomly generated individuals into the population when premature convergence is detected as $fitnessrange < \varepsilon$, where ε is a very small value. The condition implies that when the fitness range of values between successive groups is sufficiently small, individuals have become very similar in the fitness or solution space. Therefore, new search points should be introduced into the next generation to improve the diversity while the best individual is retained.

Further, the selective individual learning mechanism in the proposed ACMA is illustrated using the Rastrigin benchmark test function. The results compiled are shown in Figures 5.2-5.4.

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Figure 5.2: Fitness distribution at generation 60 (minimizing 10D Rastrigin)

Figure 5.3: Fitness distribution at generation 122 (minimizing 10D Rastrigin)

Figure 5.4: Fitness distribution at generation 240 (minimizing 10D Rastrigin)

Figure 5.5: Fitness distribution at generation 266 (minimizing 10D Rastrigin)

In particular, Figure 5.2 shows a plot of the fitness distribution at 60 generations when using the proposed ACMA to optimize the 10-D Rastrigin function. With α configured as 5% and a population size n of 100, the number of fitness groups is thus $\alpha * n = 5$. The *fitnessrange* of each fitness group is then approximately 2.2995. From each group, one individual is picked randomly to undergo individual learning. This results in a total of six individual learning made at generation 60, i.e., one individual learning for each fitness group and another on the elite individual.

At generation 122, it is possible to observe in Figure 5.3 that some form of convergence is emerging, a sign of possible over-exploitation. The population of individuals at generation 122 are similar in solution or fitness space, particularly there are many individuals having

similar fitness values in groups 1 and 3. The *fitnessrange* is now thus smaller than before, approximately 0.442926. Note that no individuals are assigned to the group in *fitnessrange* of $[2.440796, 2.883722]$ or group 4, since none of the individuals in the population has a fitness value that falls within this range. The total number of individual learning performed is thus at most five (one on each non-empty fitness grouping and another on the elite individual). At generation 240, the population diversity can be observed to have dropped significantly in Figure 5.4 with most individuals converging at local optima with fitness levels 0.994959 and 1.98992. There are now two empty groups and the number of individual learning has decreased to at most four. To summarize, Figures 5.2 - 5.5 illustrate how ACMA algorithm adapts the amount of exploitation by adapting the number of individual learning throughout its search process. As the search continues, the *fitnessrange* shrinks and individuals in the population tend to be very similar to each other in the fitness space. Here, ε is configured to a very small value of 10^{-5} in our illustration. When the condition for $\text{fitnessrange} < \varepsilon$ is satisfied, new random individuals are introduced into the population to replace all except the elite individual in an effort to maintain good diversity and solution quality in the search.

Figure 5.5 shows the fitness distribution when new random search points are introduced at generation 266, but with the elite individual of fitness 0.99496 preserved in the population. Note that the number of individual learning has been restored back to at most six and the process repeats until the termination condition is satisfied.

5.3 Empirical study

This section presents a numerical study on CGA, canonical CMA and adaptive CMAs in optimizing a series of commonly used benchmark functions used in the literature. The search performance results of a stochastic multi-start individual learning (label here as MS-DSCG in short) is also reported as the baseline which other stochastic algorithms may be compared. Here, the strategy of Davies, Swann, and Campey with Gram-Schmidt orthogonalization

(DSCG) is used as the individual learning procedure in the multi-start strategy, canonical CMA, CMAR and the ACMA, since this individual learning strategy has been shown in several previous work [113] to generate good solution qualities on continuous benchmark problems when used in the design of MAs. In particular all the meta-heuristics implemented here, namely, the multi-start local search, canonical CMA, CMAR and the ACMA will have DSCG as their core local searcher. In the next subsection, the utility of the adaptive individual learning mechanism introduced in the CGA is studied and the search performance results obtained by the algorithms on the benchmark problem are analyzed.

5.3.1 Experimental Study

Twenty five independent search runs is performed for Multi-start individual learning (MS-DSCG), Cellular GA (CGA), canonical Cellular MA (CMA), Cellular MA with random selective individual learning $\alpha = 5\%$ (CMAR5), Adaptive Cellular MA with $\alpha = 5\%$ (ACMA5), and Adaptive Cellular MA with $\alpha = 10\%$ (ACMA10). The algorithms are tested on the thirty-dimensional problem sets together with a real-world problem, the Frequency Modulation Sound (FMS) function [144] (see Table 5.1). The main purpose is to compare the performance of ACMA with CGA and CMA. Results for ACMA with $\alpha = 5\%$ and 10% is presented to observe the effect of different exploitation level on search performance.

In the present study, the evolutionary parameters for all the algorithms are configured consistently with real-encoded population of 50 individuals, Gaussian mutation rate of 0.03, two-point crossover rate of 0.7 and. In all the MA variants considered, the individual learning frequency, labeled as θ in Algorithm 6, is set to 10 generations, i.e., individual learning phase is applied for every 10 generations. In each individual learning phase, the initial step size, $s^{(0)}$, is configured to 1.0, while a maximum computational budget of 300 evaluations for a single individual refinement process is considered.

Table 5.1: Benchmark functions used in the study (epi*: epistasis, mul*: multimodality)

Function	Range	Characteristics	
		Epi*	Mul*
F_{Sphere} (Appendix A.1)	$[-100, 100]^{30}$	none	none
$F_{Elliptic}$ (Appendix A.4)	$[-100, 100]^{30}$	none	none
$F_{Schwefel1.2}$ (Appendix A.3)	$[-100, 100]^{30}$	none	none
$F_{Rosenbrock}$ (Appendix A.7)	$[-100, 100]^{30}$	high	weak
F_{Ackley} (Appendix A.8)	$[-32, 32]^{30}$	weak	normal
$F_{Griewank}$ (Appendix A.9)	$[-600, 600]^{30}$	weak	high
$F_{Rastrigin}$ (Appendix A.10)	$[-5.12, 5.12]^{30}$	none	high
$F_{Weierstrass}$ (Appendix A.11)	$[-0.5, 0.5]^{30}$	weak	high
$F_{Scaffer}$ (Appendix A.14)	$[-100, 100]^{30}$	none	none
F_{FMS} (Appendix A.16)	$[-6.4, 6.35]^6$	high	high

5.3.2 Results and Analysis

Figures 5.6-5.15 present the average convergence search trends of CGA, MS-DSCG, CMA, CMAR5, ACMA5, and ACMA10 across 50 independent simulation runs on each of the problems considered, respectively. Each simulation run continued until the global optimum is found or a maximum of $10000 * D$ (300,000 function evaluations in the present study) is reached, where D is the problem dimensionality. The statistics of the search runs, i.e., the final errors with respect to the global optimum upon search termination, the number of evaluations, etc., are presented in Tables 5.2-5.11 to allow more detailed comparison on the solution quality and efficiency of the algorithms. In particular, if the algorithm reaches the global optimum (fitness error $< 10^{-8}$) before the maximum computational budget of 300,000 evaluations elapsed, the number of evaluations incurred in the search process will be reported. For instance, the best result displayed by CMA in Table 5.2 is reported as 0.0 (451). This implies the best run of CMA (across the 50 runs) is one that successfully converged to a fitness error of $< 10^{-8}$ from the global optimum at 451 evaluations.

For the sake of brevity, we present the discussion and analysis of our results divided into two main categories. The first category covers the unimodal problems, namely, Sphere, Elliptic and Schwefel 1.02 functions. The second category discussed the results obtained for the

multimodal problem sets, in this case, Ackley, Rastrigin, Griewank, Rosenbrock, Weierstrass, Scaffer and FMS problems.

5.3.2.1 Unimodal functions (Sphere, Elliptic and Schwefel 1.02)

We begin first with a discussion on the results obtained for the unimodal problem set. The search traces of the different algorithms on Sphere and Elliptic functions are depicted in Figures 5.6 and 5.7. While CGA faces slow convergence on high dimensional problems as expected, the remaining algorithms managed to locate the global optimum solution, thanks to inclusion of individual learning. The average number of evaluations reported in Table 5.2 and 5.3 highlights that the computational costs incurred by the CMA variants in converging to the global optimum accurately on Sphere and Elliptic functions are relatively close. Note that in all the CMA variants considered, the individual learning phase is conducted immediately after the first search generation. On the other hand, Sphere and Elliptic problems are unimodal and convex, thus DSCG is able to converge to the global optimum regardless of the starting point used. This also explains why the results obtained by a stochastic multi-start individual learning is competitive to those obtained by MA.

In contrast, since the Schwefel function poses a challenge to the DSCG local search, MS-DSCG which generates random starting points fails to search on the function well as shown in Figure 5.8. It is also observed that CMA does not fare as well as CGA, suggesting that performing local search on the entire population is inefficient. That also explains why CMAR5 and ACMA5 fare better than ACMA10.

Figure 5.6: Search traces (average of 50 runs) for minimizing 10D Sphere function. (Square box indicates that the Global Optimum is found)

Figure 5.7: Search traces (average of 50 runs) for minimizing 30D Elliptic function. (Square box indicates that the Global Optimum is found)

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Figure 5.8: Search traces (average of 50 runs) for minimizing 30D Schwefel function.

Table 5.2: Fitness error values and Success Rate (30 Dimensional Sphere)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.000013	0.000081	0.000037	0.000014	0%
MS-DSCG	0.0 (344)	0.0 (6070)	0.0 (1365)	0.0 (1111)	100%
CMA	0.0 (451)	0.0 (3755)	0.0 (1227)	0.0 (852)	100%
CMAR5	0.0 (443)	0.0 (2862)	0.0 (910)	0.0 (552)	100%
ACMA5	0.0 (435)	0.0 (2857)	0.0 (742)	0.0 (603)	100%
ACMA10	0.0 (429)	0.0 (4345)	0.0 (700)	0.0 (805)	100%

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Table 5.3: Fitness error values and success rate (30 Dimensional Elliptic)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.296993	11.802538	2.094388	2.013883	0%
MS-DSCG	0.0 (344)	0.0 (8191)	0.0 (1688)	0.0 (1686)	100%
CMA	0.0 (442)	0.0 (9753)	0.0 (1645)	0.0 (1494)	100%
CMAR5	0.0 (444)	0.0 (3135)	0.0 (1250)	0.0 (803)	100%
ACMA5	0.0 (448)	0.0 (4454)	0.0 (1675)	0.0 (1139)	100%
ACMA10	0.0 (442)	0.0 (4312)	0.0 (1124)	0.0 (964)	100%

Table 5.4: Fitness error values and success rate (30 Dimensional Schwefel 1.02)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	1.083250	6.511442	3.507798	1.379640	0%
MS-DSCG	25017.911043	46222.875190	36613.290116	4981.768773	0%
CMA	338.664096	1820.302667	1120.620191	332.701733	0%
CMAR5	0.000302	0.015256	0.004470	0.003089	0%
ACMA5	0.000694	0.084662	0.013748	0.016618	0%
ACMA10	0.017464	3.542390	0.509196	0.548611	0%

5.3.2.2 Multimodal functions

Figure 5.9: Search traces (average of 50 runs) for minimizing 30D Ackley function. (Square box indicates that the Global Optimum is found)

Here, we consider next the set of separable multimodal problems, i.e., functions imbue with low or no epistasis. Figures 5.9 - 5.11 present the search trends of the algorithms on the 30 dimensional Ackley, Rastrigin and Griewank problems which has increasing level of multimodality. The results indicate that the ACMA5 and CMAR5 had converged to the global

Table 5.5: Fitness error values and success rate (30 Dimensional Ackley)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.000819	0.002120	0.001418	0.000325	0%
MS-DSCG	0.064219	2.561123	1.098980	0.672054	0%
CMA	0.0 (124294)	0.0 (124477)	0.0 (124393)	0.0 (32)	100%
CMAR5	0.0 (15462)	0.0 (20294)	0.0 (17699)	0.0 (1243)	100%
ACMA5	0.0 (14775)	0.0 (24435)	0.0 (21027)	0.0 (1846)	100%
ACMA10	0.0 (26780)	0.0 (39853)	0.0 (33909)	0.0 (3074)	100%

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Figure 5.10: Search traces (average of 50 runs) for minimizing 30D Rastrigin function. (Square box indicates that the Global Optimum is found)

Figure 5.11: Search traces (average of 50 runs) for minimizing 30D Griewank function. (Square box indicates that the Global Optimum is found)

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Table 5.6: Fitness error values and success rate (30 Dimensional Rastrigin)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	9.949604	36.813473	19.560903	5.540093	0%
MS-DSCG	0.0 (239)	0.0 (34266)	0.0 (9645)	0.0 (8302)	100%
CMA	0.0 (343)	0.0 (31291)	0.0 (7973)	0.0 (7676)	100%
CMAR5	0.0 (350)	0.0 (5291)	0.0 (2713)	0.0 (1325)	100%
ACMA5	0.0 (336)	0.0 (5291)	0.0 (2511)	0.0 (1111)	100%
ACMA10	0.0 (332)	0.0 (4297)	0.0 (2695)	0.0 (1797)	100%

Table 5.7: Fitness error values and success rate (30 Dimensional Griewank)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.000020	0.019786	0.004263	0.005972	0%
MS-DSCG	5.792934	35.850984	12.249027	4.521426	0%
CMA	0.0 (62666)	0.0 (101777)	0.0 (84336)	0.0 (12715)	100%
CMAR5	0.0 (8213)	0.0 (35290)	0.0 (14494)	0.0 (5274)	100%
ACMA5	0.0 (7788)	0.0 (32194)	0.0 (15053)	0.0 (4962)	100%
ACMA10	0.0 (12057)	0.0 (34789)	0.0 (20027)	0.0 (6302)	100%

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

optimum while CGA fails to do so for the same termination condition, suggesting the DSCG local search synergizes well with the CGA to bring about search improvements. Since the DSCG local search operates on each dimension separately (see Figure B.6), it functions efficiently on the separable problems. This suggests why the MS-DSCG also managed to find the global optimum on Rastrigin problem on all the 50 runs.

Further, consistent with the results shown on the unimodal problems, CMAR5, ACMA5 and ACMA10 again attained superior search performances over the CMA. Clearly, this is attributed to the introduction of selective local search schemes into the CMA. To quantify the significance in improvements by the adaptive CMAs, the search statistics of the algorithms are also summarized in Tables 5.5-5.7. It is worth noting that besides a faster convergence speed, CMAR5, ACMA5 and ACMA10 also exhibit better search robustness as indicated by the lower standard deviation of the results obtained across the 50 independent runs.

Figure 5.12: Search traces (average of 50 runs) for minimizing 30D Rosenbrock function.

Rosenbrock is a multi-modal function plague with strong epistasis. Although its fitness landscape is not as rugged as the other three previously considered benchmark problems, the

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Table 5.8: Fitness error values and success rate (30 Dimensional Rosenbrock)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.100618	80.102006	34.995072	25.813214	0%
MS-DSCG	145.614786	2681.761437	681.762756	390.338469	0%
CMA	0.000089	15.750893	1.436206	2.725633	0%
CMAR5	0.007964	67.471566	7.762101	10.577379	0%
ACMA5	0.000072	16.184286	2.288657	3.677642	0%
ACMA10	0.000466	15.473135	1.230378	2.738209	0%

strong interactions between the genes is often deem to make it hard for many optimization approaches including GA or CGA. This suggests why the algorithms failed to locate the global optimum accurately, i.e., success hit rate of 0% on all the 50 independent runs (see Figure 5.12 and Table 5.8). Nevertheless, the results obtained highlighted that the inclusion of local search had contributed to improvements in the search performance of the CGA, i.e., CMA, CMAR5, ACMA5 and ACMA10 produces improved performance than the CGA. Further, it is observed that CMA and ACMA10 had performed better than CMAR5 and ACMA5, suggesting greater exploitation is beneficial on the Rosenbrock problem. On the other hand, ACMA5 is observed to be superior over the CMAR5 further suggests the latter facing difficulties with the strong epistasis in the problem. Hence a selection scheme with some uniformity such as the stratified scheme is deem to be more appropriate than a simple random selection.

The extremely rugged surface of Weierstrass functions is generally a greater challenge for CMAs than the CGA. The significantly large number of basins reduce severely the efficacy of CMAs as the local search is easily trapped in the local optima and most of the computational efforts spent are generally wasted. This suggests why the CMA which conducts a local refinement on each individual in the CGA population fares poorly when compared to CMAR5, ACMA5 and ACMA10, for the same computational budget, see Table 5.9. It is worth noting that even though CMAR5 is able to achieve excellent search performance, the simple random

Figure 5.13: Search traces (average of 50 runs) for minimizing 30D Weierstrass function.

Table 5.9: Fitness error values and success rate (30 Dimensional Weierstrass)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.045592	3.530791	0.449901	0.759902	0%
MS-DSCG	35.269832	41.501266	39.144839	1.360618	0%
CMA	0.011849	2.412587	0.333060	0.601024	0%
CMAR5	0.002224	1.973621	0.345337	0.540117	0%
ACMA5	0.002177	0.897336	0.035927	0.161454	0%
ACMA10	0.001824	0.986958	0.097758	0.287558	0%

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

selection scheme suffers from poor robustness as observed in the large standard deviations of the results on the Weierstrass problem (see Figure 5.13 and Table 5.9). In contrast, the ACMAs fare the best on the Weierstrass problem.

Figure 5.14: Search traces (average of 50 runs) for minimizing 30D Scaffer function.

On the Scaffer problem, it is observed in Figure 5.14 and Table 5.10 that the local search procedure, i.e., DSCG, does not bring about significant improvement to the search, since the

Table 5.10: Fitness error values and success rate (30 Dimensional Scaffer)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.253642	1.509344	0.553448	0.297457	0%
MS-DSCG	0.671650	1.840983	1.294955	0.274535	0%
CMA	0.208198	0.783990	0.395896	0.150297	0%
CMAR5	0.077727	1.318264	0.467296	0.263380	0%
ACMA5	0.038864	0.936715	0.430312	0.223992	0%
ACMA10	0.097159	1.276680	0.371344	0.213785	0%

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Table 5.11: Fitness error values and success rate (FMS)(note that the best performing algorithm is highlighted in bold typeface)

Algorithm	Fitness error values and success rate				
	Best	Worst	Mean	Std	Success
CGA	0.000063	25.470558	15.810940	7.312292	0%
MS-DSCG	0.049222	17.229564	6.981543	4.753076	0%
CMA	0.0 (71098)	14.778976	7.192541	5.606119	36%
CMAR5	0.0 (10670)	22.383644	13.097359	7.465197	20%
ACMA5	0.0 (4884)	21.483843	4.168509	6.189833	66%
ACMA10	0.0 (6621)	17.382780	3.679700	5.595474	68%

CGA displays a search trend that is close to those of the CMAs. This suggests why MS-DSCG performs badly on the Scaffer problem. The search performances of the CMA, CMAR5, ACMA5 and ACMA10, on the other hand, do not differ significantly. Nevertheless, CMA and ACMA10 appear to fare slightly better than CMAR5 and ACMA5 in terms of solution quality and robustness.

Figure 5.15: Search traces (average of 50 runs) for minimizing FMS function.

Lats but not least, we discuss the results of the search on the multimodal and highly epistatic FMS problem. Figure 5.15 and Table 5.11 indicate that ACMAs outperform the other counterparts significantly in terms of solution quality (best solution achievable), robustness and success rate. The consistently good performance of the ACMAs on this highly complex problem further emphasizes the importance of selection local search in CMAs.

In summary, it is now widely accepted that synergy between global and local search can bring about improved search performance. Nevertheless, our study has shown that a good balance through global exploration and local exploitation represents a crucial factor to achieving high and robust quality solutions efficiently. Exhaustive local search in CMA may lead to ineffective search due to the inefficient use of computational resources and premature fall in diversity during the search. On the other hand, random selection of individuals for local improvement may affect the robustness of the search. In such case, a diversity-based dynamic adaptive scheme such as the stratified scheme proposed would be more appropriate.

5.3.3 Cellular MAs with Comparison to Other Evolutionary and Memetic Approaches

In the following section, we present a comparison with other advanced evolutionary algorithms using the same set of benchmark problems studied in section 5.3.1. ACMA5 and ACMA10 is compared to the Orthogonal GA (OGA/Q) [84] and Hybrid Taguchi-Genetic Algorithm [143] on the 30D Sphere, Ackley, Rastrigin and Griewank. Figure 5.16 reports the number of function evaluation calls incurred by each algorithm in converging to a fitness error of 10^{-8} on the benchmark problems. The detailed statistical comparison between ACMA5 and the two algorithms are summarized in Tables 5.12 and 5.13. It is noted that ACMA5 and ACMA10 outperform the OGA/Q significantly at 99% confidence level on all the benchmark problems. At the same time, ACMA5 and ACMA10 also outperform HTGA on all the problems considered significantly, except the Ackley function.

CHAPTER 5. CELLULAR MEMETIC ALGORITHMS

Figure 5.16: Number of function evaluations used by different algorithms in solving the 30D benchmark functions.

Table 5.12: Statistic test on comparing ACMA5 vs. OGA/Q (Null hypothesis: ACMA5 and OGA/Q incur equal number of evaluations vs. alternative hypothesis: ACMA5 and OGA/Q incur different number of evaluations to reach to the global optimum)

Problems	ACMA5	OGA/Q	<i>t</i> -value	<i>p</i> -value	Conclusion
Sphere	742 ± 603	112559	−1311.2	$5.00e^{-113}$	ACMA5 is better
Ackley	21027 ± 1846	112421	−350.1	$6.25e^{-85}$	ACMA5 is better
Rastrigin	2511 ± 1111	224710	−1414.2	$1.235e^{-114}$	ACMA5 is better
Griewank	15053 ± 4962	134000	−169.1	$1.65e^{-69}$	ACMA5 is better

Table 5.13: Statistic test on comparing ACMA5 vs. HTGA (Null hypothesis: ACMA5 and HTGA incur equal number of evaluations vs. alternative hypothesis: ACMA5 and HTGA incur different number of evaluations to reach to the global optimum)

Problems	ACMA5	HTGA	<i>t</i> -value	<i>p</i> -value	Conclusion
Sphere	742 ± 603	20844	−235.7	$1.61e^{-76}$	ACMA5 is better
Ackley	21027 ± 1846	16632	16.8	$2.60e^{-22}$	HTGA is better
Rastrigin	2511 ± 1111	16267	−87.6	$1.67e^{-55}$	ACMA5 is better
Griewank	15053 ± 4962	20999	−8.5	$3.64e^{-11}$	ACMA5 is better

5.4 Conclusion

Previous studies have shown that the Cellular GA possesses better population diversity and exploration capacity than the standard GA. The impact of reducing the effect of premature convergence however raises the issue of slow convergence rate in CGAs. This has been identified as a core challenge for CGA especially when used in solving optimization problems that are computationally demanding. Since it is common knowledge that local search methods can locate local optima effectively and efficiently, it can potentially complement the CGA as a means of increasing its exploitation ability.

In this chapter, the performance of the cellular GA against the canonical cellular MA has been analyzed using a series of benchmark problems representing classes of unimodal, multimodal test functions having different level of epistasis. Empirical results on the test functions show that ACMA locates global optimum more often than the CGA; generating better success hit rates and giving the best optimum solution with lower computational effort, i.e., a lower minimum error threshold and a smaller number of function evaluations. However, due to the aggressive exploitative nature of the cellular MA, it is more susceptible to getting stuck in local optima than the cellular GA.

To achieve a desirable level of performance in a search algorithm, neither exploitation nor exploration should dominate. It is shown that adaptive cellular MAs (ACMAs) with selective individual learning mechanism can balance the degree of exploitation and exploration in the search under limited computational budget. Empirical study shows that the proposed ACMA significantly outperforms the Cellular GA, canonical Cellular MA and the Cellular MA with random selection on all the benchmark problems tested, in terms of solution quality, search efficiency and robustness.

Chapter 6

Diffusion Memetic Algorithms

In this chapter, a study of 2nd generation Memetic Algorithm based on the theory of “Universal Darwinism” is presented. While Genetic Algorithm (GA) tries to emulate biological evolution, MA is said to mimic cultural evolution. The theory of “Universal Darwinism” was coined by Richard Dawkins in 1989 [36] to provide a unifying framework governing the evolution of any complex systems. In particular, “Universal Darwinism” suggests that evolution is not exclusive to biological systems, i.e., it is not confined to the narrow context of the genes, but applicable to any complex systems that exhibit the principles of inheritance, variation and selection, fulfilling the traits of an evolving system. The new science of memetics represents the mind-universe analogue to genetics in culture evolution that stretches across the fields of biology, cognition and psychology, which has attracted significant attention in the last decades. The term “meme” was introduced and defined by Dawkins [36] as “the basic unit of cultural transmission, or imitation”, and in the English Oxford Dictionary as “an element of culture that may be considered to be passed on by non-genetic means”. To understand how a meme is transmitted and selected, one can refer to the corresponding models in gene evolution. A major difference that one tends to find is in the mechanisms of culture transmission, which is far more varied than gene transmission. Parent-child (vertical) transmission, (e.g., direct inheritance), is present in both. On the other hand, peer transmission (horizontal), (e.g., imitation, learning, infection), exhibits the characteristics of cultural transmission, but is practically absent in genetics. Hence,

CHAPTER 6. DIFFUSION MEMETIC ALGORITHMS

the variety of existing cultural transmission mechanisms bring about far more flexibilities than genetic evolution [28]. Recently, the concepts and theories in the study of human culture and memetics have been derived in the form of computational intelligence and adapted into operational algorithms for solving real-world problems in the fields of arts, digital media, business, finance, science and engineering, with great deal of successes [56, 117, 118, 135, 75, 90].

As discussed in Chapter 3, the 2nd generation Memetic Algorithm mimics the mechanisms of inheritance/memetic transmission and meme selection in their design. For the sake of clarity, the general details of a 2nd generation MA is emphasized in Algorithm 8. It is worth highlighting that the change from the 1st to 2nd generation MA lies in the additional meme selection process.

Algorithm 8 Memetic Algorithm (2nd generation)

```

1: Generate an initial population
2: Initialize the meme pool
3: while Stopping conditions are not satisfied do
4:   Evaluate all individuals in the population
5:   Evolve a new population using stochastic search operators
6:   Select the subset of individuals,  $\Omega_{il}$ , that should undergo the individual improvement
   procedure
7:   for each individual in  $\Omega_{il}$  do
8:     Select a meme from meme pool
9:     Perform individual learning using the selected meme
10:    Proceed with Lamarckian or Baldwinian learning
11:   end for
12: end while

```

In the context of optimization, parent-child (vertical) transmission takes place only when the offspring is generated as two or more individuals (referred to as parents) mates. It is also noted that memetic transmission from parents to children may take place through genetic or non-genetic means, i.e., memes may or may not be encoded as part of the genotype. On the other hand, meme can transfer from one individual to another (horizontal transmission) at anytime throughout the life cycle of an individual, suggesting that this form of transmission mechanism may pose a greater impact on the distribution of memes in the population.

The inheritance mechanisms in multi-meme [74] and co-evolution [132] MAs are purely genetic the transmission of genetic and cultural traits is achieved only through the underlying alleles. On the other hand, hyper-heuristic [67] and meta-Lamarckian [113] model the mechanisms of non-genetic inheritance since the rule of inheritance is based on fitness of the memes in generating individual learning improvement. Despite these works, it is worth noting that there exists a plethora of non-genetic transfers, which may include migration, diffusion, direct teaching, imitation, individual learning, and many others. For example, if a meme can be transmitted within a local vicinity, which is commonly referred as a *diffusion* process or on the other hand, an offspring might receive the meme directly from its parent as a result of *direct learning*.

To the best of our knowledge, till today relatively less effort has been spent on studying the non-genetic transmission of meme in the context of evolutionary and memetic computation. It would be interesting to note whether by communicating with a large number of peers (individuals within the same population), learning would be more effective and beneficial than just inheriting directly from parents. In the sections that follow, we will demonstrate how such a form of learning can be mimicked in the context of evolutionary and memetic optimization. Various factors that affect the learning and selection process of meme(s) will also be discussed.

6.1 Diffusion Memetic Algorithm

In nature, both genetic and memetic transmission take place among individuals which are geographically near to each other. Cellular Genetic Algorithm (CGA) [5] mimics that behavior by using a decentralized structure where each chromosome can only interact (mate) with other chromosomes within a particular neighborhood (see Figure 6.1). In a CGA, each individual has its respective pool of potential mates defined by neighboring individuals; while at the same time, each individual serves as mates in multiple pools. In this way, one-dimensional (1-D) or

CHAPTER 6. DIFFUSION MEMETIC ALGORITHMS

two-dimensional (2-D) structures with overlapped neighborhoods are then used to provide a smooth diffusion of good solutions across the grid.

Figure 6.1: Neighborhood structure in Cellular Genetic Algorithm

Based on the structure of CGA, we introduce a Diffusion-MA (DMA) for studying the non-genetic transfer of meme in the context of evolutionary optimization. A 2^{nd} generation memetic algorithm with meme diffusion is outlined in Algorithm 9. In DMA, the population is organized on a two-dimensional grid of $WIDTH \times HEIGHT$, each individual being located on a grid cell. The main difference between DMA and Cellular Memetic Algorithm is that individual may be associated or tagged with a meme that will be used to perform individual learning on it. In the first generation, the meme attached with each individual is randomly initialized. Note that individuals may also be initialized without any meme tagged to it.

Subsequently, the individuals undergo the evolutionary process. At each generation, individual in each cell of the grid mates with one of its neighbors to produce a new offspring. The mating neighbor is selected by means of natural selection, for example biased roulette wheel. The offspring then replaces the original parent. Subsequently, meme learning takes place to decide on the choice of meme that will operate on an offspring. The individual learning process is performed for every α generations, with each individual in the population refined by the meme associated to it. Here, α refers to the individual learning interval, which balances the degree of evolutionary and individual learning in the search.

CHAPTER 6. DIFFUSION MEMETIC ALGORITHMS

Algorithm 9 Diffusion Memetic Algorithm

```

1: procedure DIFFUSIONMA
2:   Initialize-Meme-Pool;
3:   pop = Create-Grid (WIDTH * HEIGHT)
4:   for x = 1 to WIDTH do
5:     for y = 1 to HEIGHT do
6:       initialize pop(x, y)
7:       pop(x, y).fitness = Evaluate(pop(x, y))
8:       pop(x, y).meme = Random-Meme;
9:     end for
10:  end for
11:  while termination condition is not satisfied do
12:    for x = 1 to WIDTH do
13:      for y = 1 to HEIGHT do
14:        /*Gene transmission*/
15:        parent1 = pop(x, y)
16:        parent2 = Select(Neighbors(x, y));
17:        child = Crossover(parent1, parent2)
18:        child = Mutate(child)
19:        child.fitness = Evaluate(child)
20:        /*Meme transmission*/
21:        child.meme = Meme-Selection()
22:        if (generation mod  $\alpha$  = 1) then
23:          /*Individual learning*/
24:          child = Individual-learning(child)
25:          if child.fitness > child.fitness then
26:            replace pop(x, y) with child
27:          else
28:            replace pop(x, y) with child
29:          end if
30:        else
31:          replace pop(x, y) with child
32:        end if
33:      end for
34:    end for
35:  end while
36: end procedure

```

CHAPTER 6. DIFFUSION MEMETIC ALGORITHMS

Algorithm 10 Meme Selection

```
1: function MEME-SELECTION(neighbor-list)
2:   for all meme  $m$  in meme-pool do
3:      $m.reward = 0$ 
4:      $m.count = 0$ 
5:   end for
6:   for all neighbor  $n$  in neighbor-list do
7:      $m = n.meme$ 
8:      $m.reward = m.reward + n.fitness$ 
9:      $m.count = m.count + 1$ 
10:  end for
11:  for all meme  $m$  in meme-pool do
12:     $m.reward = m.reward / m.count$ 
13:  end for
14:  return meme with highest reward
15: end function
```

It is noted that in general, meme can either be inherited from its parent (parent-child transmission) or learned from other individuals in the population (peer transmission). In DMA, the offspring learns the meme from its neighbors in the grid instead of inheriting directly from the parents. Since meme information can only be transmitted from one individual to its neighbors, the process of meme transmission is achieved via “diffusion”. Algorithm 10 illustrates how a meme associated with the offspring can be determined by learning from its neighbors based on some rewarding scheme. A possible instantiation of meme selection can be defined by the fitness of an individual’s neighbors. For example, the reward of a meme defined by the average fitness of neighbors that share the same meme is considered in Algorithm 10. Subsequently, memes having higher rewards are then equipped with greater chance of survival. It is also worth noting the reward of meme can also be defined as the average fitness of all the individuals in the current population that share the same meme, this approach, however, may suffer from several drawback addressed by Hansen and Ostermeier[52]. Another popular reward scheme is based on the performance of the memes [113], i.e., the total fitness improvement achieved when the meme is applied to perform individual learning during the previous stage of the search. Last but not least, complicated reward function that takes into consideration of various performance

Table 6.1: Benchmark functions used in the study

Function	Range	Characteristics	
		Epistasis	Multi-modality
$F_{Rastrigin}$	$[-5.12, 5.12]^{30}$	none	high
$F_{Griewank}$	$[-600, 600]^{30}$	weak	high

factors of the meme can also be employed [67].

6.2 Experimental Study

In this section, we present a numerical study to analyze non-genetic transmission of memes by diffusion in the context of 2nd generation memetic algorithm. Particularly studies will be made based on the Diffusion Memetic Algorithm. Two commonly used continuous parametric benchmark test problems already extensively discussed in the literature are considered (Appendices A.10 and A.9). The benchmark problems used represent classes of multimodal and epistatic / non-epistatic test functions. Table 6.1 tabulates the two test functions with their notable characteristics. Each run continues until the global optimum is found or a maximum of 300,000 function evaluations is reached. In addition, the algorithms terminate upon convergence when a fitness error of 10^{-8} with respect to the global optimum is reached.

In the present study, the Cellular GA parameters for all the algorithms are configured consistently with grid size of 10 x 10 real-coded solutions, Gaussian mutation and uniform crossover are used with probability settings of 0.03 and 0.9, respectively, while biased roulette wheel is used for selection. For individual learning procedures or memes, we consider the i) procedure of Davies, Swann, and Campey with Gram-Schmidt orthogonalization (DSCG) [131], ii) Davidon, Fletcher and Powell's Quasi-Newton Strategy (DFP) [34] and iii) the Simplex Method by Nelder and Mead (Simplex) [103] which are representatives of first and zeroth order exact individual learning methods commonly found in the literature. On all the CMA

CHAPTER 6. DIFFUSION MEMETIC ALGORITHMS

Figure 6.2: Memetic map across different search generations of DMA (DSCG + DFP) on 30D Griewank

and DMA variants considered, the individual learning interval, labeled as α in Algorithm 9, is set to 10 generations, i.e., individual learning phase is applied for every 10 generations. In each individual learning phase, a maximum computational budget of 100 functions evaluations is used. For each experiment, the average of fifty independent search runs are presented.

6.2.1 Meme Diffusion in MA, DMA

In this subsection, we first demonstrate the idea of meme diffusion in the context of memetic algorithm. In particular, we illustrate the effect of using a non-genetic transmission of memes by diffusion based on the DMA when searching on the 30-dimensional Griewank problem (Appendix A.9). In this example, we considered a pool of two memes based on the individual learning procedures DSCG and DFP. GA population is initialized with four individuals associated with a meme while all remaining individuals in the population do not associate with any meme. In particular, individuals at positions (3, 3) and (8, 3) of the grid are assigned with DSCG meme while those at positions (3, 8) and (8, 8) have the DFP meme (see Figure 6.2.a).

Figure 6.2 illustrates the diffusion process of the memes at different instances of the DMA search, i.e., at generations 1, 10, 14, 30 and 40. It is observed that as the evolution begins, memes that are deemed to generate better search improvements or solution qualities based on some reward metrics, are given higher chance of surviving, thus they spread or diffuse across its neighbors through the offspring, (see for example, Figure 6.2.b). As the search evolves further, most cells in the grid are shown to have been infected with a meme at generation 14 (Figure 6.2.c). In addition, it is noted that the DSCG meme are spreading faster than DFP since the number of individuals infected with the DSCG meme are much higher than those with DFP meme. This suggests the stronger individual learning or refining capabilities of DSCG over the DFP meme on the problem of interest. Subsequently when most the cells have been infected with a meme, individuals in a neighborhood may be associated with different memes and these memes will have to compete for survival to take greater ownership of the entire grid (see

Figure 6.2.d). Weak memes will eventually fade away and may cease to exist due to their poor performance. Finally, at the end of generation 40, nearly all the cells in the grid are shown to have been infected by the DSCG meme (Figure 6.2.e), demonstrating the greater efficacy of the DSCG over DFP.

6.2.2 First generation Cellular Memetic Algorithms

To begin we first present the search traces of the 1st generation Cellular Memetic Algorithms (CMAs) when used to search on the benchmark problems in Figures 6.3 and 6.4. Note that the different CMAs presented are formed by a synergy of the canonical CGA and a meme. From the results shown, it is clear that no single CMA always performed best on the two test functions considered. Even worse, the CMA is shown to perform poorer than the CGA on 30D Griewank. This is expected since it is generally acknowledged that culture evolution does not always bring about fitness maximization as one would wish to attain.

For example, Hughes [57] highlighted the negative effect of human cultural evolution in genetic evolution: *“on the average, the wealthiest Western European families in the 1600s had six children, reared four to adulthood, but only married off two per family (again on average). By circa 1700 Western Europe’s elites had begun to reduce the very high death rates from which their infants and children traditionally suffered; but as death rates for the young fell, so did birth rates. Demographic contraction [among the elites] continued throughout the 1800s despite improved survivorship”*.

6.2.3 Benefits of Meme Diffusion

Next, we study the impact of using meme diffusion on evolutionary search performance by comparing the Diffusion Memetic Algorithm with 1st generation Cellular Memetic Algorithms (CMAs), to determine whether the non-genetic transfer mechanism considered could translate to practical benefits in the context of optimization.

Figure 6.3: Search performance of CGA and CMAs on 30D Griewank

Figure 6.4: Search performance of CGA and CMAs on 30D Rastrigin

Figure 6.5: Search performance of CGA and DMAs on 30D Griewank

Figure 6.6: Search performance of CGA and DMAs on 30D Rastrigin

CHAPTER 6. DIFFUSION MEMETIC ALGORITHMS

Here, we consider several instantiations of meme pools. This involves experimental studies on DMA with meme pool consisting of different combinations of meme pairs, i.e, DSCG + DFP, DFP + Simplex or Simplex + DSCG, and the combination of all three memes, i.e, DSCG + Simplex + DFP to search on 30D Griewank and Rastrigin functions. The search traces plotted in Figures 6.5 and 6.6 indicate that both the DSCG and DFP memes spread faster than the Simplex meme, which explains why the search performances for DMAs with (DSCG + Simplex) or (DFP + Simplex) are similar to that of CMAs using DSCG or DFP, respectively, on both the benchmark problems. On the other hand, it is interesting to observe that DMA (DSCG + DFP) and DMA (DSCG + Simplex + DFP) fare better than both CMA(DSCG) and CMA(DFP) on the Griewank function. Note that in DMA, an individual may be infected with different memes throughout the entire search. This has the benefits of giving each individual the opportunity to be refined by different individual learning procedures, hence the possibility of better results.

Figure 6.7: Meme distribution during the search process of DMA (DSCG + Simplex + DFP) on 30D Griewank

Figure 6.8: Meme distribution during the search process of DMA (DSCG + Simplex + DFP) on 30D Rastrigin

More details on the diffusion process of DMA (DSCG + Simplex + DFP) on the benchmark functions are depicted in Figures 6.7 and 6.8. On Griewank function, DSCG meme outperforms the other two memes (refer to Figure 6.3), therefore the number of individuals infected with the DSCG meme at the final stage of the search is much higher (see Figure 6.7). In this case, most of the runs ended up with the DSCG meme spreading over the entire population. On the other hand, the DSCG and DFP memes fare equally well on the Rastrigin function (refer to Figure 6.4), thus both memes possess relatively equal number of infected individuals at the end of the search process. Overall, the Simplex meme is deemed to be incapable of bringing about benefits to the search on both the Griewank and Rastrigin functions.

6.2.4 The neighborhood size in cellular structure

One of the core parameters of both CGA or DMA is the neighborhood size used in the diffusion process. Here, the neighbors of an individual or a cell is defined by the threshold used,

Table 6.2: Search performance of DMA on 30D Griewank

	100,000 evaluations			300,000 evaluations		
	d=1	d=2	d=3	d=1	d=2	d=3
Best	0	0	0	0	0	0
Worst	1.3262	1.7741	2.2600	0	0	0
Mean	0.3777	0.5292	0.4332	0	0	0
Std.	0.4423	0.4518	0.5476	0	0	0
Success	12%	8%	10%	100%	100%	100%

in the form of (i.e. $|dx| + |dy|$). Figure 6.9 illustrates the neighbor set of one individual with the distance threshold set to 1 and 2. Following the convention of Cellular GA, the neighborhood structure is circularly wrapped, i.e., individuals in the last row of the grid are neighbors with distance 1 from the individuals in the first row of the grid. Such a rule help enforce all individuals in the population to have equal number of neighbors.

Figure 6.9: Neighborhood structure in meme selection algorithm (a) Distance = 1 (b) Distance = 1 and 2

Here, we study the effect of neighborhood sizes for distance thresholds $d = 1, 2$ and 3 on DMA (DSCG + DFP). Table 6.2 presents the statistical results of DMA (DSCG + DFP)

Figure 6.10: Meme distribution of DSCG during the search process of DMA (DSCG + DFP) on 30D Griewank

searching on the Griewank function, i.e., the best and worst, together with mean and standard deviation of the best fitness values found at 100,000 and 300,000 evaluations, across 50 independent runs, are reported. “Success” refers to the percentage of run instances where global optimum is successfully located.

Figure 6.10 depicts the distribution of individuals that is infected with DSCG meme along the DMA (DSCG + DFP) search for different neighborhood sizes. Since the neighbors of an individual increases with Manhattan distance threshold used, DMA of larger neighborhood sizes would generally diffuse the memes more rapidly. For example, an individual may infect up to a maximum of 24 neighbors when a distance threshold d of 3 is considered, while influences only 12 and 4 neighbors for distance threshold of 1 and 2, respectively.

From Table 6.2 it can be observed that the best results were obtained when the neighborhood size is smallest, i.e., the distant threshold $d = 1$. When the neighborhood size increases, the performances decreases. The results suggests that higher meme spreading speed

may produce a negative impact to the search performance. That can be explained by noting that neighbors which are too far away from the current individual actually share a little similarity in genetic structure with the current individual, therefore meme information learnt from those neighbors may not be suitable for refining the current individual, resulting in a degraded performance.

6.3 Conclusions

In nature, it is widely accepted that individuals are products of the interaction between genetic evolution and cultural evolution. In the context of optimization, the 2nd and 3rd generations of Memetic Algorithm mimic the process of cultural transmission and evolution with the purpose of increasing the chance of employing appropriate memes, while at the same time, yielding robust and improved search performance.

In this chapter we have studied an instance of non-genetic transfer of meme in the context of evolutionary optimization. A 2nd generation Memetic Algorithm with meme diffusion (DMA) is also proposed and investigated. Using the natural overlapping neighborhoods of cellular structure, memes are allowed to diffuse or spread across the population. Empirical study on DMA using two commonly used test functions shows that non-genetic transmission of memes via diffusion not only helps promote spread of memes appropriate for the problem of interest, but also facilitates collaboration between diverse memes, thus bringing about benefits in terms of improved search performance which would not be possibly achieved when only single meme are considered.

In the last two chapters, it has been shown that the use of adaptive approach to configure the algorithmic parameters brings about greater flexibility in the MAs as well as improves the overall search performance. Nevertheless, it is worth noting that most adaptive methods proposed are heuristic in nature and would require additional parameters to be handled (such as the number of group α in the proposed ACMA and DMA). Therefore, it would be useful if

CHAPTER 6. DIFFUSION MEMETIC ALGORITHMS

those adaptive methods can be theoretically justified so that the parameters can be mathematically formulated and defined. In the next two chapters, we will concentrate more on theoretical analysis of the memetic framework. In particular, Chapter 7 presents a probabilistic model of the memetic framework while Chapter 8 provides a complexity analysis of numerical methods used as individual learning procedure in Memetic Algorithms.

Chapter 7

A Probabilistic Memetic Framework

This chapter presents a formal Probabilistic Memetic Framework (PrMF) that governs at run-time whether evolution or individual learning should be favored, with the primary objective of accelerating MA search without negatively affecting the accuracy. Note that existing strategies for controlling the MA parameters are mostly designed based on heuristics that comes with little theoretical motivation and considers each design issue of MA independently. In contrast to earlier work, a theoretical bound for individual learning intensity or t_{il} within the PrMF is derived. Section 7.1 outlines the formulation of the probabilistic models used in deriving the theoretical bound. Subsequently, a novel Approximate Probabilistic Memetic Framework (APrMF) that adapts by governing the learning intensity of each individual according to a theoretical upper bound while the search progresses is proposed. Section 7.2 describes the APrMF in details. Section 7.3 presents the numerical study on search performance of APrMF while Section 7.4 compares the performance of APrMF with several recent state-of-the-art advance evolutionary, memetic and hybrid algorithms. Finally, Section 7.5 concludes with some recommendations for further research.

7.1 A Probabilistic Model for Memetic Algorithm

7.1.1 Memetic Algorithm

As previously emphasized, MA can be defined as a synergy of evolution and individual learning at the macro level. In the case of a minimization problem, the aim of the memetic algorithm is to locate the global *minimum* $\mathbf{x}^*$ such that $\mathbf{x}^* = \arg \min f(\mathbf{x})$, without violating the constraints imposed.

Without loss of generality, evolution usually involves some stochastic operators while individual learning is in the form of local heuristics [32] or conventional exact enumerative methods [131, 103, 125, 155]. Examples of the stochastic evolutionary search include Monte Carlo Algorithm, Simulated Annealing, Genetic Algorithm, Swarm Algorithm and others. In practice, the objective function of a problem does not display convexity, thus several local optima in the form of minima and/or maxima may exist. In this case, a local minimum is defined as a point for which there exists some $\delta > 0$ so that for all $\mathbf{x}$ such that $\|\mathbf{x} - \mathbf{x}_1\| \leq \delta$, the expression $f(\mathbf{x}_1) \leq f(\mathbf{x})$ holds. Examples of individual learning strategies include the hill climbing, Simplex method, Newton/Quasi-Newton method, interior point methods, conjugate gradient method, line search and other local heuristics.

In what follows, the theoretical formulations of probabilistic models for evolution and individual learning in MA is presented. In particular, MA is modeled as a process involving the decision of whether evolution or individual learning should be employed at a specific stage of the search, based on the probability that each process is able to locate the global optimum. To begin, basic definitions used in the formulations are outlined.

- *Definition 1:* The optimization problem is **considered solved** or **reached the global optimum** if at least one solution $\mathbf{x}^+$ satisfies the condition:

$$f(\mathbf{x}^+) \leq f(\mathbf{x}^*) + \varepsilon \quad (\text{Eq. 7.1})$$

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

where ε denotes a very small value.

- **Definition 2:** A **type I point** satisfies inequality (Eq. 7.1). As shown in Figure 7.1, a point is of type I if it lies in the segment AB of the graph.
- **Definition 3:** Given a search space S , a solution set $S' \subseteq S$, a neighborhood relation $N \subseteq S \times S$ and an evaluation function $f : S \rightarrow \mathbb{R}$. A **local minimum** is a candidate solution $\mathbf{x}^+ \in S'$ such that $\forall \mathbf{x} \in N(\mathbf{x}^+) : f(\mathbf{x}^+) \leq f(\mathbf{x})$. An **individual learning operator** is defined by $IL : (\mathbf{x} \in S') \rightarrow (\mathbf{x}^+ \in S')$ such that $\mathbf{x}^+$ is a local minimum. A basin of attraction w.r.t. local optimum $\mathbf{x}^+$ and individual learning operator IL , is the set of candidate solution $S_B \subseteq S'$, such that $\forall \mathbf{x} \in S_B : IL(\mathbf{x}) = \mathbf{x}^+$.
- **Definition 4:** A **type II point** w.r.t. an individual learning method IL is a point that lies in a basin of attraction w.r.t. any of type I points and the individual learning IL . Based on Figure 7.1, consider gradient based individual learning, all points lying in segment CD are type II points.
- **Definition 5:** p_1 or p_2 is the probability that an individual in the population of candidate solutions is a type I or type II point, respectively.

Figure 7.1: Illustrations of type I and type II points

Note that it follows from above definitions that all individual learning approaches, starting from a type II point, always converge to a type I point within tractable computational budget. In

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Figure 7.1 the basins are shown as valley, but without loss of generality, the basin of attraction can be in the form of any neighborhood/modality structure.

Here, *evolution* and *individual learning* in MA are modeled as independent processes and the probability that each process is able locating type I point(s) is analyzed. Consider current generation k , the probability of finding at least one type I as a result of individual learning can be derived as:

$$P_l = 1 - (1 - p_2^{(k)})^{f_{il} * n} \quad (\text{Eq. 7.2})$$

The total computational cost, C_{il} (in terms of function evaluation counts/calls), of individual learning in each search generation can be written as

$$C_{il} = t_{il} * f_{il} * n \quad (\text{Eq. 7.3})$$

where n represents the MA population size, t_{il} is the computational budget allocated to a single iteration of individual learning, and f_{il} is the frequency of individual learning that defines the portion of a population that undergoes individual learning.

For the same computational budget expended, the number of search generations Δ_g that one may replace individual learning with stochastic evolutionary search is given by:

$$\Delta_g = \frac{t_{il} * f_{il} * n}{t_{gs}} \quad (\text{Eq. 7.4})$$

where t_{gs} is the computational cost incurred by evolution in generating the subsequent populations. For the sake of consistency, t_{gs} is considered in terms of function evaluations incurred per generation. By assuming weak dependency among the value of $p_1^{(k)}$ across Δ_g generations, the probability, P_g , of having at least one type I point among the population is given by:

$$P_g = 1 - \prod_{i=1}^{\Delta_g} (1 - p_1^{(k+i)})^n \quad (\text{Eq. 7.5})$$

where $p_1^{(k+i)}$ is the probability that an individual in the population is a type I point at generation $k + i$.

7.1.2 A Theoretical Upper Bound for t_{il} in MA

Here, the design of an MA is generalized as a decision of evolution against individual learning at each point in time. Clearly, individual learning should be used if it has a higher probability of reaching a type I point over stochastic evolution under equal computational budget, i.e.,

$$\begin{aligned} P_l &\geq P_g \\ \Leftrightarrow 1 - (1 - p_2^{(k)})^{f_{il} \cdot n} &\geq 1 - \prod_{i=1}^{\Delta_g} (1 - p_1^{(k+i)})^n \\ \Leftrightarrow (1 - p_2^{(k)})^{f_{il} \cdot n} &\leq \prod_{i=1}^{\Delta_g} (1 - p_1^{(k+i)})^n \\ \Leftrightarrow (1 - p_2^{(k)})^{f_{il}} &\leq \prod_{i=1}^{\Delta_g} (1 - p_1^{(k+i)}) \end{aligned} \quad (\text{Eq. 7.6})$$

Every search algorithm, except for uniform random search, introduces some unique form of bias, suitable for some classes of problems but not for others. Hence it is fair to assume that the global search method chosen for solving the problem of interest is one that is deemed as capable of directing the search towards the promising region containing Type I points as the search evolves, i.e.,

$$\prod_{i=1}^{\Delta_g} (1 - p_1^{(k+i)}) \leq (1 - p_1^{(k)})^{\Delta_g} \quad (\text{Eq. 7.7})$$

From Eq. 7.6 and Eq. 7.7 we have:

$$(1 - p_2^{(k)})^{f_{il}} \leq (1 - p_1^{(k)})^{\Delta_g} \quad (\text{Eq. 7.8})$$

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Taking logarithms of both sides, we arrive at:

$$\begin{aligned}
 f_{il} * \ln(1 - p_2^{(k)}) &\leq \Delta_g \ln(1 - p_1^{(k)}) \\
 \Leftrightarrow f_{il} * \ln(1 - p_2^{(k)}) &\leq \frac{t_{il} * f_{il} * n}{t_{gs}} \ln(1 - p_1^{(k)}) \\
 \Leftrightarrow \ln(1 - p_2^{(k)}) &\leq \frac{t_{il} * n}{t_{gs}} \ln(1 - p_1^{(k)})
 \end{aligned} \tag{Eq. 7.9}$$

Since $\ln(1 - p_1^{(k)}) < 0$, Eq. 7.8 becomes:

$$\begin{aligned}
 \ln(1 - p_2^{(k)}) &\leq \frac{t_{il} * n}{t_{gs}} \ln(1 - p_1^{(k)}) \\
 \Leftrightarrow t_{il} &\leq \frac{t_{gs} \ln(1 - p_2^{(k)})}{n \ln(1 - p_1^{(k)})} \\
 \Leftrightarrow t_{il}^{upper} &= \frac{t_{gs} \ln(1 - p_2^{(k)})}{n \ln(1 - p_1^{(k)})}
 \end{aligned} \tag{Eq. 7.10}$$

Eq. 7.10 thus provides a theoretical upper bound on the computational cost allowable for performing individual learning in MA. This implies that one should not allocate a computational budget greater than $\frac{t_{gs} \ln(1 - p_2^{(k)})}{n \ln(1 - p_1^{(k)})}$ for conducting individual learning when designing MAs. Indirectly, it also implies that if a meme or learning procedure requires a computational budget of more than t_{il}^{upper} to reach a type I point, then it should not be used since it does not offer any competitive advantage over the process of evolution in the MA search. It is also worth noting that Eq. 7.9 is independent of f_{il} indicating that the frequency of learning in MA does not affect the theoretical bound for t_{il} .

7.2 A Theoretical Analysis of the Probabilistic Memetic Framework

In this section, a Probabilistic Memetic Framework (PrMF) is formulated as a basis for adaptation. This is achieved by governing the maximum intensity of the individual learning using the

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

theoretical bound derived in the previous section, while the search progresses. Based on the taxonomy in [115], PrMF represents an online adaptive approach. The pseudo code for PrMF is outlined in Algorithm 11. In the first step, the PrMF population is initialized either randomly or using design of experiments techniques such as Latin hypercube sampling [93]. The evaluated populations of chromosomes then undergo evolution based on the stochastic operators. For instance, in a genetic based MA, the stochastic operators include mutation, crossover and selection.

Subsequently, in contrast to the canonical MA, the theoretical upper bound for learning intensity, t_{il}^{upper} , is estimated for each individual/chromosome in the newly evolved population, based on $p_1^{(k)}$, $p_2^{(k)}$ and $\frac{t_{gs}}{n}$ in Eq. 7.10. Each chromosome then undergoes individual learning using the specified meme according to their respective maximum allowable intensity. In the next subsection, the efficacy of the proposed PrMF is analyzed using two synthetic benchmark functions with known fitness landscape.

Algorithm 11 PrMF

```

1: Initialize: Generate an initial population;
2: while Stopping conditions are not satisfied do
3:   Evaluate all individuals in the population.
4:   for each individual in new population do
5:     /* Individual learning with  $t_{il}$  defined by the estimated theoretical upper bound*/
6:     Estimate the theoretical individual learning intensity bound,  $t_{il}^{upper} \leq \frac{t_{gs}}{n} \frac{\ln(1-p_2^{(k)})}{\ln(1-p_1^{(k)})}$ 
7:     Perform individual learning using the specified meme for  $t_{il}^{upper}$  evaluations
8:     Proceed with Lamarckian or Baldwinian learning
9:   end for
10:  Generate a new population using stochastic search operators.
11: end while

```

7.2.1 Unimodal Sphere function

First, consider the Sphere function where the one-dimensional landscape depicted in Figure 7.2. For unimodal Sphere function, all decision vectors in the entire solution space including

Figure 7.2: Unimodal Sphere function

the global optimum lies in the single basin of attraction for a first or second-order derivative based individual learning strategy. Hence, the probabilities of hitting type I ($p_1^{(k)}$) or type II ($p_2^{(k)}$) points in the entire solution space can be easily estimated as $p_1^{(k)} < 1$ and $p_2^{(k)} = 1, \forall k$, respectively. Note that $p_1^{(k)}$ approaches zero with increasing dimensionality of the Sphere function. With $p_1 \rightarrow 0$ and $p_2^{(k)} = 1$ in Eq. 7.10, the theoretical upper bound of t_{il} approaches infinity. This implies that all individual learning involving first or second-order derivative based methods would search more efficiently than random sampling or stochastic search methods in reaching the global optimum. From Eq. 7.2, it can also be derived that the probability P_l of finding at least one type I point or the global optimum of the Sphere function in the case of a unimodal problem when using individual learning is 1 since $P_l = 1 - (1 - p_2^{(k)})^{(f_{il} * n)} = 1$, when $p_2^{(k)} = 1$. Statistically, it makes better sense to use a first or second-order derivative based method over stochastic evolutionary operators to search on the unimodal Sphere function.

7.2.2 Multi-Modal Step function

Next, consider the multi-modal Step function where the one-dimensional landscape is depicted in Figure 7.3. Consider again a first or second-order derivative based individual learning strategy as the individual learning procedure versus the stochastic evolutionary operators in the

Figure 7.3: Multi-Modal Step function

PrMF. From the landscape in Figure 7.3, it can be easily observed that the probabilities of hitting type I ($p_1^{(k)}$) or type II ($p_2^{(k)}$) points in the entire solution space are equal, i.e., $p_1^{(k)} = p_2^{(k)}, \forall k$. With $p_1^{(k)} = p_2^{(k)}$, the theoretical upper bound becomes $\frac{t_{gs}}{n}$ according to Eq. 7.10, i.e., $t_{il} \leq \frac{t_{gs}}{n} \frac{\ln(1-p_2^{(k)})}{\ln(1-p_1^{(k)})} = \frac{t_{gs}}{n}$. On the other hand, the condition $t_{gs} \leq n$ holds since the entire or only a portion of the population is generally replaced and evaluated in each search generation. With $t_{gs} < n$, t_{il}^{upper} becomes ≤ 1 which implies that no form of individual learning should be performed on the multi-modal Step function. Since any first or second-order derivative based method cannot generate any improvements starting from any of the chromosome in the population, any amount of computational budget allocated to individual learning is unlikely to contribute to the success of PrMF in search for the global optimum.

7.2.3 Approximate Probabilistic Memetic Framework

In the previous section, it is assumed that information on $p_1^{(k)}$ and $p_2^{(k)}$ for the problem in hand can be easily obtained or estimated. In reality, one usually has little or no prior knowledge of $p_1^{(k)}$ and $p_2^{(k)}$ in complex optimization problems. In this regards, a novel Approximate Probabilistic Memetic Framework (APrMF) that incorporates an Individual Learning Intensity Estimation Scheme for approximating $p_1^{(k)}, p_2^{(k)}$ and the theoretical upper bound of the problem

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

in hand is proposed. In doing so, APrMF adapts/governs the individual learning intensity of each individual using the approximated theoretical upper bound while the search progresses.

To begin, Algorithm 12 presents an outline of the APrMF. In the first step, the APrMF population is initialized either randomly or using design of experiments techniques such as Latin hypercube sampling. During this initial stage, the APrMF search operates similarly to the canonical MA with all chromosomes assigned equal computational budget for conducting individual learning. Nevertheless, in contrast to a canonical MA, the individual learning procedure or meme in APrMF is equipped with search tracking capability such that the search history and structure of each chromosome is archived in database Φ . The archived search history is then used for estimating the theoretical upper bound of future individual learning intensity.

After some pre-defined number of generations has elapsed, the adaptive mechanism for individual learning intensity, i.e., t_{il} , takes effect. For each chromosome/individual, denoted here as $\mathbf{x}(i)$, that undergoes individual learning, the upper learning intensity bound $t_{il}^{upper}(i)$ and expected learning intensity $t_{il}^{expected}(i)$ are estimated based on the *Individual Learning Intensity Estimation Scheme* outlined in Algorithm 13. Here, the *Individual Learning Intensity Estimation Scheme* is described as six main steps with the aid of the simple illustration in Figure 7.4:

- Step 1: Locate the k nearest neighbors of individual $\mathbf{x}(i)$ from database Φ based on some appropriate distance metric. For example, a simple Euclidean distance metric may be used in the case of a continuous parametric optimization problem, i.e., the two nearest neighboring points of individual $\mathbf{x}$ in Figure 7.4, are denoted by $\mathbf{x}_{neighbor1}$ and $\mathbf{x}_{neighbor2}$.
- Step 2: Identify the set Ω_{trace} of k learning search traces associated with the nearest k chromosomes. In Figure 7.4, the search traces for $\mathbf{x}_{neighbor1}$ and $\mathbf{x}_{neighbor2}$ are t_1 and t_2 , respectively.

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Algorithm 12 APrMF

```

1: Initialize: Generate an initial population;
2: for The first few generations do
3:   Evaluate all individuals in the population.
4:   for each individual  $\mathbf{x}(i)$  in new population do
5:     Perform individual learning using the specified meme with tracking capabilities for
       a maximum of  $t_{il}(i) = t_{il}^{initial}$  evaluations
6:     Proceed with Lamarckian or Baldwinian learning
7:   end for
8:   Generate a new population using stochastic search operators.
9: end for
10: while Stopping conditions are not satisfied do
11:   Evaluate all individuals in the population.
12:   for each individual  $\mathbf{x}(i)$  in new population do
13:     /* Individual learning with adaptive  $t_{il}$  set according to upper bound and the ex-
       pected value */
14:     Estimate the individual learning intensity bound,  $t_{il}^{upper}(i)$  and expected learning
       intensity  $t_{il}^{expected}(i)$  for individual  $\mathbf{x}(i)$  using the Individual Learning Intensity Estimation
       Scheme outlined in Algorithm 13.
15:     if ( $t_{il}^{expected}(i) < t_{il}^{upper}(i)$ ) then
16:       /* increase budget for individual learning */
17:        $t_{il}(i) = f(t_{il}^{upper}(i))$ 
18:       Perform individual learning using the specified meme for  $t_{il}(i)$  evaluations
19:       Proceed with Lamarckian or Baldwinian learning
20:     else
21:       /* Do not perform any individual learning */
22:     end if
23:   end for
24:   Generate a new population using stochastic search operators.
25: end while

```

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

- Step 3: Determine the fittest point, $\mathbf{x}_{best}$, in Ω_{trace} , i.e., $\mathbf{x}_{best} = \arg \min\{f(\mathbf{x}) : \mathbf{x} \in \Omega_{trace}\}$. $\mathbf{x}_{best}$ represents the best quality solution found so far within the local vicinity or neighborhood of $\mathbf{x}(i)$. $\mathbf{x}_{best}$ is denoted by point A in Figure 7.4.
- Step 4: Determine the range of type I point which is estimated based on the furthest ε -close point in Ω_{trace} to $\mathbf{x}_{best}$, denoted here as $\mathbf{x}_1$, i.e., $\mathbf{x}_1 = \arg \max\{\|\mathbf{x} - \mathbf{x}_{best}\| : f(\mathbf{x}) \leq f(\mathbf{x}_{best}) + \varepsilon\}$. In Figure 7.4, $\mathbf{x}_1$ is defined as point B.
- Step 5: The values of probabilities p_1 and p_2 for the neighborhood region of $\mathbf{x}(i)$ are approximated by:

$$p_2 = \frac{\|\mathbf{x} - \mathbf{x}_{best}\|^{ndim}}{volumeofsearchspace}$$

$$p_1 = \frac{\|\mathbf{x}_1 - \mathbf{x}_{best}\|^{ndim}}{volumeofsearchspace}$$

The upper learning intensity bound t_{il}^{upper} of the current individual is then estimated by:

$$\begin{aligned} t_{il}^{upper} &= \frac{t_{gs} \ln(1 - p_2)}{n \ln(1 - p_1)} \\ &= \frac{t_{gs} \ln(1 - \frac{\|\mathbf{x} - \mathbf{x}_{best}\|^{ndim}}{volumeofsearchspace})}{n \ln(1 - \frac{\|\mathbf{x}_1 - \mathbf{x}_{best}\|^{ndim}}{volumeofsearchspace})} \end{aligned}$$

Based on Taylor series expansion, $\ln(1 - \alpha^n) \approx -\alpha^n$ for small value of α , the above equation simplifies to:

$$t_{il}^{upper} = \frac{t_{gs}}{n} \frac{\|\mathbf{x} - \mathbf{x}_{best}\|^{ndim}}{\|\mathbf{x}_1 - \mathbf{x}_{best}\|^{ndim}} \quad (\text{Eq. 7.11})$$

For the illustration example in Figure 7.4

$$t_{il}^{upper} = \frac{CA}{BA} = 5 \text{ for the case I while}$$

$$t_{il}^{upper} = \frac{CA}{BA} = 3 \text{ for the case II.}$$

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

- Step 6: Estimate the expected learning intensity $t_{il}^{expected}$ as an average of the intensity for the k nearest neighbors. In Figure 7.4, the expected intensity for individual x is 3 since its neighbors, defined by $x_{neighbor1}$ and $x_{neighbor2}$ took 3 function evaluations on average in their individual learning processes.

Algorithm 13 Individual Learning Intensity Estimation Scheme

- 1: /* Estimate p_1 and p_2 */
- 2: **Identify** set Ω , the k nearest chromosomes of x in database
- 3: **Identify** set Ω_{trace} , the k learning search traces associated with the nearest k chromosomes.
- 4: **Find** x_{best} , the fittest individual in Ω_{trace} , i.e., $x_{best} = \arg \min \{f(x) : x \in \Omega_{trace}\}$.
- 5: **Find** x_1 , the furthest ε -close point to x_{best} , i.e., $x_1 = \arg \max \{\|x - x_{best}\| : f(x) \leq f(x_{best}) + \varepsilon\}$.
- 6: **Estimate** the upper bound for learning intensity t_{il}^{upper} using Eq. 7.11
- 7: **Estimate** the expected value for learning intensity, $t_{il}^{expected}$.

Figure 7.4: Illustration of algorithm (a) Narrow, deep basin (b) Wide, shallow basin

For individual $x(i)$, APrMF then proceeds with individual learning for a maximum computational budget defined by the estimated $t_{il}^{upper}(i)$ and $t_{il}^{expected}(i)$, if the current expected learning intensity of the current neighborhood does not exceed the estimated learning intensity

Table 7.1: Benchmark functions used in the study.

Function	Range	Characteristics	
		Epistasis	Multi-modality
F_{Sphere}	$[-100, 100]^{30}$	none	none
F_{Step}	$[-5.12, 5.12]^{30}$	none	weak
$F_{Rastrigin}$	$[-5.12, 5.12]^{30}$	none	high
$F_{Griewank}$	$[-600, 600]^{30}$	weak	high
F_{Ackley}	$[-32, 32]^{30}$	weak	moderate

upper bound, i.e., $t_{il}^{expected}(i) \leq t_{il}^{upper}(i)$. This is followed by a replacement of the genotype and fitness of $\mathbf{x}(i)$ in the population or only the latter action with the improved solution in the spirit of Lamarckian or Baldwinian learning, respectively. Otherwise, no form of individual learning would be made should $t_{il}^{expected}(i) > t_{il}^{upper}(i)$ happens. The stochastic GA operators are then used to create the next population. The entire process repeats until the specified stopping criteria are satisfied.

7.3 Empirical study

In this section, a numerical study to analyze the search behavior and performance of the Approximate Probabilistic Memetic Framework is presented. Table 7.1 tabulates these benchmark test functions with their notable characteristics.

To see how the proposed APrMF improves the efficiency of a search, diverse candidates of MA formed by the synergy of different stochastic and individual learning strategies on the representative benchmark problems are considered. In particular, the i) Canonical Genetic Algorithm (GA) [35], or ii) Differential Evolution (DE) [137] or iii) Evolutionary Strategy (ES) [18] are employed as candidate stochastic search strategies here. For individual learning or meme, the i) strategy of Davies, Swann, and Campey with Gram-Schmidt orthogonalization (DSCG) [131], ii) Davidon, Fletcher and Powell Strategy (DFP) [125] and iii) Simplex strategy

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

of Nelder and Mead [103] are considered. These are representative of second, first and zeroth order exact individual learning methods commonly found in the literature.

To gain a better understanding of APrMF, results are analyzed and compared with the canonical MA having different fixed individual learning intensity according to the following aspects:

- Search Quality and Efficiency - the capability of the strategy to provide high search quality and efficiency over different problem types.
- Computational Cost - the amount of extra CPU effort incurred over and above canonical memetic algorithm.
- Robustness - the capability of the framework/algorithm to generate performances that is reliable over different problems.
- Simplicity & ease of implementation - Simple framework/algorithm should require minimum effort to develop, as well as a minimum numbers of control parameters that need to be managed.

7.3.1 Search quality and efficiency

The average convergence trends obtained by APrMF based on GA and DFP individual learning procedure (GA-DFP) for solving the 30-dimension Sphere and Step problems are depicted in Figures 7.5 and 7.6, as a function of the total number of fitness evaluation calls. All results presented are averages over 25 independent runs. Each run continues until the global optimum was found or a maximum of 100,000 function evaluations was reached. In each run, the control parameters of the GA used in APrMF for solving the benchmark problems were set as follows: population size of 50, bit-flip mutation rate of 0.03, two-point crossover with a rate of 0.7 and 32-bit binary encoding. The initial individual learning intensity used in APrMF is initialized

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

to 100 evaluations. In Figures 7.5 and 7.6, canonical MA implies that throughout the entire search, a fixed individual learning intensity of 100 is used, i.e., no form of adaptation for t_{il} is made. From the results reported in the Figures, APrMF is shown to converge to the global optimum of both the Sphere and Step functions more efficiently than the canonical MA.

Besides the best fitness convergence trends obtained along the APrMF search, Figure 7.5(ii) also depicts the respective individual learning/global search ratio against fitness evaluation calls,. Note that the individual learning/global search ratio reflects the degree of local against global efforts expended in the search so far. It is worth noting that during the initial stage of the plots in Figures 7.5 and 7.6 on learning/global search ratio against function evaluation call, both the APrMF and canonical MA exhibit similar trend or traces since the former operates exactly like the canonical MA during this initial learning phase of two search generations before its adaptive strategy begins to bite. From Figure 7.5(ii), it is observed that in contrast to the canonical MA (with a fixed individual learning intensity of 100) which reaches a stable state of individual learning/global search ratio, the individual learning/global search ratio of the APrMF increases significantly as the search progresses on the Sphere function. This implies that the expected individual learning intensity of the entire population increases adaptively once the knowledge about individual learning actually leads to greater benefits than the stochastic evolutionary method is gained. In contrast, the plot in Figure 7.6(ii) illustrates a case where the individual learning/global search ratio of the population drops since the converse is true in the case of a Step function, i.e., the use of the stochastic GA operators leads to greater benefits than individual learning. In both Figures, APrMF is shown to converge to the global optimum more efficiently than canonical MAs. Note that these results obtained by APrMF based on GA and DFP is consistent with our theoretical analysis presented earlier. Hence, this shows that the proposed Individual Learning Intensity Estimation Scheme in APrMF is capable of generating good approximation of p_1 , p_2 and t_{il}^{upper} at search runtime.

Figure 7.5: Search trends of Canonical MA (with a fixed individual learning intensity, $t_{il} = 100$) and APrMF (GA-DFP) on the Unimodal Sphere function

Figure 7.6: Search trends of Canonical MA (with a fixed individual learning intensity, $t_{il} = 100$) and APrMF (GA-DFP) on the Step function

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Next, the search behavior of the APrMF for different evolutionary and individual learning procedures or memes is further analyzed on greater benchmark problems that include the 30-dimensional Ackley, Griewank and Rastrigin functions. In particular, the following configurations of APrMFs and canonical MAs are investigated:

- APrMF with the initial individual learning intensity configured to 100 (APrMF1)
- APrMF with the initial individual learning intensity configured to 200 (APrMF2)
- Canonical Memetic Algorithm with fixed individual learning intensity of 100 (MA1)
- Canonical Memetic Algorithm with fixed individual learning intensity of 200 (MA2)

All parameters of the GA are kept consistent as before. Table 7.2 summarizes the parameter settings of the other search procedures used in the present experimental study.

The averaged convergence trends of various MAs obtained for the 30-dimensional benchmark problems as a function of the total number of function evaluations are summarized in

Table 7.2: Parameter setting of APrMF.

General parameters	
Global search	GA, DE and ES
Local search	DSCG, DFP and Simplex
Stopping criteria	100,000 evaluations or convergence to global optimum
Population size	50
Genetic Algorithm parameters	
Encoding scheme	Real-encoded
Selection scheme	Roulette wheel
Crossover operator	Two point crossover, $p_c = 0.7$
Mutation operator	Gaussian mutation, $p_m = 0.03$
Differential Evolution parameters	
Crossover probability	$p_c = 0.9$
ES parameters	
Selection method	$\mu + \alpha$, $\mu = 50$, $\alpha = 50$
Mutation operator	Gaussian mutation
Local search parameters	
Initial individual learning intensity $t_{il}^{initial}$	100 or 200 evaluations

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Figures 7.8-7.10. For the sake of readability in our discussion, the numerical results obtained are grouped accordingly in Figures 7.8, 7.9 and 7.10, so as to highlight the different adaptive trends of the individual learning/global search ratio and individual learning intensity in the APrM framework.

Note that all the subplots in Figure 7.8 share similar upward trends in the individual learning/global search ratio as the search progresses for different level of synergy between evolution and individual learning procedures on the benchmark problems. With the APrM framework, knowledge on the greater benefits of using individual learning against stochastic evolutionary search for the problems in hand is gained. Hence over time, the expected individual learning intensity or individual learning/global search ratio increases adaptively as the search progresses.

On the other hand, Figure 7.9 displays upward trends in the individual learning/global search ratio on APrMF1 while experiencing a downward trend on APrMF2. Since APrMF1 and APrMF2 differ in the configurations of initial individual learning intensity at 100 and 200, respectively, such trends serve to indicate that the optimum configuration of individual learning intensity should lie somewhere between the region from 100 to 200. Note that in Figures 7.9(b) and 7.9(d), the individual learning/global search ratios of APrMF1 and APrMF2 converged to very similar values. This serves to demonstrate that the proposed APrM framework is capable of adapting the individual learning intensity to suit the search problem in hand, regardless of its initial settings as well as the evolutionary and individual learning procedures used.

Figure 7.10 illustrates the inability of the individual learning procedures in contributing to the optimization problems. For instance, the Simplex individual learning procedure is found to synergize poorly with ES or DE in contributing to the MA search on the Griewank function, see Figure 7.10(a) and (c). Similarly, ES also did not synergize well with Simplex to form an MA that is appropriate for solving the Ackley function, as shown in Figure 7.10(b). Consequently, the APrM frameworks, i.e., APrMF1 and APrMF2, adapt by reducing the expected individual learning intensity of the population as the search proceeds, when knowledge on the inability

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

of individual learning compared to the stochastic GA operators is learned. Thus a downward trend in the individual learning/global search ratio can be observed in Figures 7.10(a)-(c).

Figure 7.7: Legend for Figures 7.8-7.10

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

7.8.a: DE DSCG Rastrigin

7.8.b: DE Simplex Ackley

7.8.c: GA Simplex Rastrigin

7.8.d: GA DSCG Griewank

Figure 7.8: APrMF with Upward Trends in Individual learning/global search Ratio and t_{il}

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

7.9.a: DE DSCG Ackley

7.9.b: ES Simplex Rastrigin

7.9.c: GA DSCG Ackley

7.9.d: GA Simplex Griewank

Figure 7.9: APrMF Adapting the Individual learning/global search Ratio to its Optimal Configuration

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

7.10.a: DE Simplex Griewank

7.10.b: ES Simplex Ackley

7.10.c: ES Simplex Griewank

Figure 7.10: APrMF with Downward Trends in Individual learning/global search Ratio and t_{il}

7.3.2 Computational complexity

The time complexity of a canonical Memetic Algorithm can be easily derived as the order of $O([n_{gen} * t_{gs} * \tau_{eval}] + [n_{gen} * n * t_{il} * \tau_{eval}])$, where n_{gen} is the maximum number of generations evolved during the search process, τ_{eval} is the time taken for one objective function evaluation while t_{gs} , n and t_{il} have been defined in the previous sections. Hence, $n_{gen} * n$ represents the total number of chromosomes that undergo individual learning in the entire search.

APrMF enhanced the canonical MA by introducing the *Individual Learning Intensity Estimation Scheme* which is made up of two main parts:

- (i) Identify the k nearest chromosomes, for which the time complexity is of the order of $O((n_{gen} * n)^2 * \tau_{dis})$ where τ_{dis} is the time taken to perform the distance measure between any two chromosomes.
- (ii) Estimate the upper bound and expected learning intensity in constant time denoted by γ . The total time complexity incurred throughout the APrMF search is of the order of $O((n_{gen} * n) * \gamma)$

Hence, the total time complexity incurred by APrMF over the canonical MA is of the order of $O([(n_{gen} * n)^2 * \tau_{dis}] + [(n_{gen} * n) * \gamma])$. Note that since τ_{dis} and γ are relatively small (order of μsec) especially on complex optimization problems where function evaluations are computationally expensive [146], i.e., τ_{eval} is usually large (order of minutes or more), the extra cost incurred may be considered to be negligible.

7.3.3 Simplicity & Robustness

In contrast to the f_{il} , Ω_{il} and t_{il} control parameters in a Canonical MA, APrMF represents a much simpler framework since it has only a single $t_{il}^{initial}$ parameter. Furthermore, from the results reported in Figures 11-13, the APrMF has been demonstrated to adapt the individual

intensity t_{il} competently at runtime, regardless of the initial setting for $t_{il}^{initial}$, thus producing search performances that are superior to the canonical MA counterparts on all the benchmark problems considered. For instance, for the same initial configuration, APrMF is shown to increase the value of t_{il} to suit the Sphere landscape, while the reverse is observed on the Step function. In another example, both the APrMF1 and APrMF2 converged to robust search performances and global/individual learning ratio on the Griewank function, see Figure 7.9d. Hence, APrMF not only converges to robust solutions, but also offers ease of implementation as an added advantage.

7.4 APrMF with Comparison to Other Evolutionary and Memetic Approaches

In this section, a comprehensive empirical study of APrMF is provided and compared to several recent state-of-the-art evolutionary, memetic and/or hybrid evolutionary individual learning approaches using the test suite proposed in the Congress on Evolutionary Computation (CEC'05) [87]. Readers are referred to Appendix A for the details description of these functions.

The search performance of APrMF based on GA-DSCG for the 10D and 30D versions of the benchmark functions are summarized in Tables 5 and 6, respectively. In the tables, results are presented for 10^3 , 10^4 , 10^5 and $n * 10^4$ function evaluations (n is the dimensionality of the benchmark problems considered). At each of the evaluation points, the objective function errors of 25 independent runs are sorted and the best, the 7th, the median, the 19th and the worst results together with the mean and standard deviation are presented in the tables. The parametric configuration of the GA-DSCG used is also summarized in Table 7.3.

Table 7.3: Parameter Configuration of APrMF based on GA-DSCG.

	10D functions	30D functions
Stopping criteria	100,000 evaluations	300,000 evaluations
Global search	Genetic algorithm	
Local search	DSCG	
Population size	50	
Encoding scheme	Real-encoded	
Selection scheme	Roulette wheel	
Crossover operator	One point crossover $p_c = 0.7$	
Mutation operator	Gaussian mutation $p_m = 0.03$	
Initial individual learning intensity	100 evaluations	300 evaluations

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Table 7.4: Optimization result using AP_rMF (GA-DSCG) on 10D benchmark functions. FEC represents Function Evaluation Calls incurred to converge at the global optimum solution of each respective function.

FEC		F_{Sph}	$F_{Sch1.2}$	F_{Elp}	$F_{Sch1.2N}$	$F_{Sch2.6}$	F_{Ros}	F_{Ras}	F_{Gri-R}
1e3	min	0	1.91e+3	2.59e-028	9.17e+3	2.09e+3	8.45e+0	0	7.63e-1
	7th	0	2.83e+3	1.42e-027	1.28e+4	5.74e+3	4.50e+1	0	9.57e-1
	med.	0	2.83e+3	5.25e-027	1.43e+4	6.70e+3	5.86e+1	0	9.57e-1
	19th	0	2.83e+3	1.13e-026	17420	7.71e+3	1.49e+2	0	9.57e-1
	max	0	3.50e+3	3.64e-026	2.28e+4	9.28e+3	1.39e+3	0	1.34e+0
	mean	0	2.80e+3	8.13e-027	1.50e+4	6.41e+3	2.82e+2	0	9.84e-1
	std	0	3.07e+2	8.33e-027	3.31e+3	1.80e+3	4.58e+2	0	1.11e-1
1e4	min	-	1.33e-6	-	2.27e+3	9.23e+1	0	-	0
	7th	-	6.34e-5	-	4.63e+3	7.35e+2	0	-	1.82e-10
	med.	-	6.34e-5	-	6.18e+3	1.66e+3	2.02e-7	-	2.46e-2
	19th	-	9.54e-5	-	7.15e+3	2.14e+3	1.03e+0	-	4.67e-2
	max	-	6.45e-3	-	1.02e+4	3.89e+3	4.94e+0	-	1.13e-1
	mean	-	4.01e-4	-	5.92e+3	1.54e+3	8.16e-1	-	2.95e-2
	std	-	1.28e-3	-	2.02e+3	9.26e+2	1.46e+0	-	2.96e-2
1e5	min	-	-	-	1.34e+2	7.36e-4	-	-	0
	7th	-	-	-	8.70e+2	6.48e+1	-	-	0
	med.	-	-	-	1.22e+3	1.53e+2	-	-	0
	19th	-	-	-	1.81e+3	2.57e+2	-	-	9.86e-3
	max	-	-	-	4.71e+3	1.20e+3	-	-	3.69e-2
	mean	-	-	-	1.48e+3	2.23e+2	-	-	8.07e-3
	std	-	-	-	1.01e+3	2.58e+2	-	-	1.14e-2

FEC		F_{Ack-R}	F_{Ras-R}	F_{Wei-R}	$F_{Sch2.13}$	F_{GriRos}	F_{Scf}	F_{Hyb1}	F_{Hyb2}
1e3	min	2.02e+1	6.78e+1	5.50e+0	1.35e+3	2.83e+0	1.23e-1	2.26e+2	9.10e+2
	7th	2.04e+1	7.98e+1	8.12e+0	2.13e+3	4.56e+0	3.97e-1	2.98e+2	1.03e+3
	med.	2.05e+1	7.98e+1	9.02e+0	3167	5.15e+0	4.42e-1	3.20e+2	1.07e+3
	19th	2.06e+1	7.98e+1	9.58e+0	3.78e+3	5.15e+0	6.16e-1	3.73e+2	1.11e+3
	max	2.08e+1	9.66e+1	1.08e+1	3.91e+3	6.25e+0	1.08e+0	4.71e+2	1.16e+3
	mean	2.05e+1	7.96e+1	8.67e+0	2.83e+3	4.81e+0	4.89e-1	3.37e+2	1.06e+3
	std	1.43e-1	5.43e+0	1.44e+0	8.52e+2	7.15e-1	2.21e-1	6.81e+1	6.72e+1
1e4	min	2.01e+1	2.79e+1	4.79e+0	1.01e-1	6.23e-1	3.89e-2	1.33e+2	6.76e+2
	7th	2.02e+1	3.88e+1	5.68e+0	2.80e+1	8.83e-1	5.83e-2	1.68e+2	8.42e+2
	med.	2.03e+1	4.49e+1	6.90e+0	5.73e+1	1.09e+0	7.79e-2	1.96e+2	9.09e+2
	19th	2.04e+1	5.55e+1	7.27e+0	9.25e+1	1.51e+0	1.e-1	2.11e+2	9.59e+2
	max	2.05e+1	7.98e+1	8.39e+0	3.99e+2	3.39e+0	1.66e-1	2.73e+2	1.03e+3
	mean	2.03e+1	4.88e+1	6.63e+0	8.36e+1	1.27e+0	8.49e-2	1.95e+2	8.85e+2
	std	1.11e-1	1.34e+1	1.04e+0	8.82e+1	6.41e-1	3.05e-2	3.65e+1	9.50e+1
1e5	min	2.0e+1	9.95e+0	4.17e+0	0	1.92e-1	1.94e-2	1.16e+2	4.32e+2
	7th	2.02e+1	1.99e+1	5.17e+0	0	5.84e-1	3.89e-2	1.31e+2	6.64e+2
	med.	2.02e+1	2.29e+1	5.60e+0	0	6.91e-1	3.89e-2	1.37e+2	7.90e+2
	19th	2.03e+1	2.80e+1	6.32e+0	1.84e-4	8.27e-1	7.77e-2	1.44e+2	8.e+2
	max	2.04e+1	3.38e+1	7.92e+0	3.52e-1	1.16e+0	9.72e-2	1.59e+2	8.26e+2
	mean	2.02e+1	2.30e+1	5.70e+0	2.38e-2	7.07e-1	4.99e-2	1.37e+2	7.21e+2
	std	1.05e-1	6.35e+0	1.01e+0	7.84e-2	2.07e-1	2.66e-2	1.18e+1	1.20e+2

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Table 7.5: Optimization result using APrMF (GA-DSCG) on 30D benchmark functions. FEC represents Function Evaluation Calls incurred to converge at the global optimum solution of each respective function.

FEC		F_{Sph}	$F_{Sch1.2}$	F_{Elp}	$F_{Sch1.2N}$	$F_{Sch2.6}$	F_{Ros}	F_{Ras}	F_{Gri-R}
1e3	min	3.11e-27	3.85e+4	1.33e-24	7.67e+4	1.49e+4	2.60e+2	0	3.72e+0
	7th	1.31e-26	71670	1.30e-23	1569	1.94e+4	3.09e+3	4.01e+0	4.32e+0
	med.	3.26e-26	8.34e+4	2.71e-23	119853	2.17e+4	4.79e+3	5.31e+0	6.16e+0
	19th	3.26e-26	103222	8.15e-2	156792	2.45e+4	7.42e+3	2.89e+1	6.41e+0
	max	6.77e+2	107899	1.17e+1	173645	3.28e+4	1.15e+4	6.25e+1	6.41e+0
	mean	2.71e+1	8.30e+4	2.42e+0	1.27e+5	2.23e+4	5.29e+3	1.57e+1	5.51e+0
1e4	std	1.35e+2	2.15e+4	4.55e+0	3.05e+4	4.32e+3	2.96e+3	1.80e+1	1.09e+0
	min	-	7849	-	6.56e+4	1.06e+4	5.81e+1	0	4.22e-3
	7th	-	1.26e+4	-	8.52e+4	1.31e+4	2.25e+2	0	5.47e-1
	med.	-	1.41e+4	-	9.06e+4	1.43e+4	4.98e+2	0	6.21e-1
	19th	-	1.62e+4	-	107937	1.62e+4	1.14e+3	2.77e-2	7.23e-1
	max	-	2.70e+4	-	156792	2.41e+4	2.14e+3	5.98e+0	1.05e+0
1e5	mean	-	1.52e+4	-	1.01e+5	1.47e+4	6.93e+2	6.66e-1	6.06e-1
	std	-	5.17e+3	-	2.54e+4	2.94e+3	5.84e+2	1.60e+0	2.41e-1
	min	-	0	-	3.14e+4	5.93e+3	1.52e-7	-	0
	7th	-	0	-	41743	7.23e+3	8.82e+0	-	0
	med.	-	0	-	4.37e+4	8.37e+3	1.56e+1	-	0
	19th	-	0	-	4.90e+4	9.71e+3	2.86e+1	-	0
3e5	max	-	8.41e+3	-	6.43e+4	1.13e+4	1.73e+2	-	9.86e-3
	mean	-	3.36e+2	-	4.55e+4	8.48e+3	3.30e+1	-	1.04e-3
	std	-	1.68e+3	-	6.91e+3	1.57e+3	4.10e+1	-	2.45e-3
	min	-	-	-	2.57e+4	4.45e+3	0	-	0
	7th	-	-	-	3.15e+4	6.03e+3	1.34e-6	-	0
	med.	-	-	-	3.59e+4	6.39e+3	1.10e+0	-	0
1e3	19th	-	-	-	4.03e+4	7.21e+3	1.32e+1	-	0
	max	-	-	-	5.42e+4	9.14e+3	5.79e+1	-	9.86e-3
	mean	-	-	-	3.61e+4	6.66e+3	8.27e+0	-	8.76e-4
	std	-	-	-	6.63e+3	1.28e+3	1.29e+1	-	2.36e-3
FEC		F_{Ack-R}	F_{Ras-R}	F_{Wei-R}	$F_{Sch2.13}$	F_{GriRos}	F_{Scf}	F_{Hyb1}	F_{Hyb2}
1e3	min	2.07e+1	3.83e+2	3.09e+1	4.22e+4	2.87e+1	8.51e-1	1.33e+2	8.47e+2
	7th	2.08e+1	4.41e+2	3.38e+1	7.07e+4	5.88e+1	2.82e+0	1.48e+2	9.07e+2
	med.	2.09e+1	4.80e+2	3.58e+1	120430	8.39e+4	3.14e+0	1.53e+2	9.31e+2
	19th	2.11e+1	4.86e+2	3.79e+1	140507	160182	3.93e+0	2.47e+2	9.57e+2
	max	2.11e+1	5.16e+2	3.86e+1	182762	162441	4.84e+0	5.21e+2	9.68e+2
	mean	2.09e+1	4.66e+2	3.58e+1	1.11e+5	7.58e+4	3.31e+0	2.16e+2	9.28e+2
1e4	std	1.21e-1	3.71e+1	2.14e+0	4.e+4	7.14e+4	8.94e-1	1.19e+2	3.05e+1
	min	2.05e+1	2.49e+2	2.10e+1	1.60e+4	7.98e+0	8.51e-1	1.16e+2	5.26e+2
	7th	2.07e+1	3.39e+2	3.04e+1	3.42e+4	1.37e+1	1.27e+0	1.23e+2	7.70e+2
	med.	2.08e+1	3.39e+2	3.12e+1	4.63e+4	2.06e+1	1.54e+0	1.27e+2	8.34e+2
	19th	2.08e+1	3.92e+2	3.23e+1	51674	3.16e+1	1.85e+0	1.31e+2	8.94e+2
	max	2.09e+1	4.69e+2	3.58e+1	7.44e+4	2.13e+2	2.67e+0	1.39e+2	9.45e+2
1e5	mean	2.08e+1	3.64e+2	3.12e+1	4.44e+4	3.76e+1	1.59e+0	1.27e+2	8.17e+2
	std	9.90e-2	5.36e+1	2.79e+0	1.32e+4	5.30e+1	4.67e-1	6.21e+0	1.05e+2
	min	2.e+1	1.79e+2	2.10e+1	3.29e+0	4.71e+0	2.45e-1	1.03e+2	5.03e+2
	7th	2.06e+1	2.34e+2	2.93e+1	4.41e+2	6.84e+0	6.29e-1	1.12e+2	5.75e+2
	med.	2.07e+1	2.51e+2	3.02e+1	1.63e+3	8.68e+0	7.72e-1	1.17e+2	5.98e+2
	19th	2.08e+1	2.82e+2	3.12e+1	2.79e+3	1.25e+1	9.29e-1	1.19e+2	6.56e+2
3e5	max	2.09e+1	3.38e+2	3.32e+1	5.87e+3	2.22e+1	1.54e+0	1.21e+2	7.43e+2
	mean	2.07e+1	2.55e+2	2.98e+1	1.87e+3	9.89e+0	8.16e-1	1.15e+2	6.12e+2
	std	1.86e-1	4.29e+1	2.49e+0	1.70e+3	4.13e+0	3.26e-1	4.81e+0	5.87e+1
	min	2.e+1	1.71e+2	2.10e+1	2.42e-6	4.71e+0	2.34e-1	1.03e+2	4.78e+2
	7th	2.06e+1	1.92e+2	2.82e+1	8.22e+0	6.28e+0	4.31e-1	1.07e+2	4.98e+2
	med.	2.07e+1	2.14e+2	2.99e+1	1.40e+2	6.84e+0	5.21e-1	1.13e+2	5.26e+2
1e3	19th	2.08e+1	2.33e+2	3.04e+1	6.64e+2	8.16e+0	6.85e-1	1.16e+2	5.51e+2
	max	2.08e+1	2.91e+2	3.32e+1	2.83e+3	1.38e+1	8.76e-1	1.19e+2	6.45e+2
	mean	2.06e+1	2.17e+2	2.93e+1	4.42e+2	7.72e+0	5.41e-1	1.12e+2	5.34e+2
	std	1.91e-1	3.36e+1	2.54e+0	6.72e+2	2.26e+0	1.88e-1	5.12e+0	4.31e+1

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Using the results obtained in Tables 7.4 and 7.5, APrMF is pitted against other memetic and hybrid evolutionary individual learning approaches. The list of algorithms used here for comparisons were reported in the recent CEC'05 which significantly outperform many others in the literature. Table 7.6 provides a brief description of these algorithms, while Tables 7.7, 7.8, 7.9 and 7.10 summarize their search performance compared to the APrMF in solving the same set of benchmark problems. Entries in each table are the average number of function evaluations for all the successful runs (among 25 independent runs). For fair comparisons, the accuracy level of convergence, ϵ , is set to 10^{-6} for functions 1 to 5 and 10^{-2} for all others, similar to that used in the CEC 2005 benchmark studies. Overall, APrMF outperforms all the other methods on the unimodal, multimodal, discrete, continuous, epistatic, and non-epistatic 10 and 30 dimensional benchmark problems. On the hybridized problems, i.e., functions A.15 in Appendix A, competitive results have been obtained for APrMF compared to the other approaches considered.

Table 7.6: Memetic Algorithms or Hybrid EA-Local search used in Comparison.

Algorithm name	Description
BLX-GL50	Hybrid Real-Coded Genetic Algorithms [42]
BLX-MA	Real-coded memetic algorithm with adaptive local search probability and local search length [99]
DMS-L-PSO	Dynamic multi-swarm particle swarm optimizer with local search [86]
EDA	Estimation of Distribution Algorithm [154]
DEshcSPX	Differential evolution with crossover-based local search [110]

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Table 7.7: Success measure of the algorithms in solving the 10D benchmark functions. For instance, 0.6 (25) on F_{Sphere} implies that APrMF incurred an average of (0.6*1000) function evaluation calls on 25 successful independent runs. A '-' entry in the table implies that the respective algorithm fails to converge to the global optimum successfully. Bold italic also highlights the best search performance among the approaches considered (based on pair-wise t-test between each respective algorithm pairs).

	F_{Sph}	$F_{Sch1.2}$	F_{Elp}	F_{Ros}	F_{Gri-R}	F_{Ras}	$F_{Sch2.13}$
APrMF	0.6(25)	11.0(25)	0.6(25)	10.5(25)	17.6(20)	1.0(25)	31.9(16)
BLX-GL50	19.0(25)	41.04(25)	-	51.8(25)	20.8(9)	20.4(3)	51.59(13)
BLX-MA	12.0(25)	36.96(25)	-	-	-	69.8(18)	-
DMS-L-PSO	12.0(25)	12(25)	11.7(25)	54.7(25)	94.8(4)	35.7(25)	54.1(19)
EDA	10.0(25)	11.0(25)	16.3(23)	68.2(25)	75.9(1)	-	35.2(10)
DEshcSPX	22.9(25)	34.7(25)	89.2(20)	50.2(23)	97.3(21)	89.7(5)	-

Table 7.8: Result of t-test with 95% confidence level comparing statistical values for APrMF and those of the other algorithms in solving the 10D benchmark functions (s+, s-, and $\approx$ indicate that APrMF is significantly better, significantly worse, and indifferent, respectively).

	F_{Sph}	$F_{Sch1.2}$	F_{Elp}	F_{Ros}	F_{Gri-R}	F_{Ras}	$F_{Sch2.13}$
BLX-GL50	s+	s+	-	s+	s+	s+	s+
BLX-MA	s+	s+	-	-	-	s+	-
DMS-L-PSO	s+	s+	s+	s+	s+	s+	s+
EDA	s+	$\approx$	s+	s+	s+	-	s+
DEshcSPX	s+	s+	s+	s+	s+	s+	-

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Table 7.9: Success measure of the algorithms in solving the 30D benchmark functions. For instance, 87.0 (25) on $F_{Sch1.2}$ implies that APrMF incurred an average of (87*1000) function evaluation calls on 25 successful independent runs. A ‘-’ entry in the table implies that the respective algorithm fails to converge to the global optimum successfully. Bold italic also highlights the best search performance among the approaches considered.

	F_{Sph}	$F_{Sch1.2}$	F_{Elp}	F_{Gri-R}	F_{Ras}
APrMF	0.5(25)	87.0(25)	1.0(25)	25.5(25)	5.9(25)
BLX-GL50	58.05(25)	159.6(25)	-	66.3(25)	-
BLX-MA	32.13(25)	-	-	-	238.8(9)
DMS-L-PSO	5.13(25)	129.6(25)	285.3(21)	57.4(24)	-
EDA	150.1(25)	159.6(25)	219.3(25)	129.93(25)	-
DEshcSPX	89.4(25)	299.3(2)	-	148.1(21)	-

Table 7.10: Result of t-test with 95% confidence level comparing statistical values for APrMF and those of the other algorithms in solving the 30D benchmark functions (s+, s-, and $\approx$ indicate that APrMF is significantly better, significantly worse, and indifferent, respectively).

	F_{Sph}	$F_{Sch1.2}$	F_{Elp}	F_{Gri-R}	F_{Ras}
BLX-GL50	s+	s+	-	s+	-
BLX-MA	s+	s+	-	-	s+
DMS-L-PSO	s+	s+	s+	s+	-
EDA	s+	s+	s+	s+	-
DEshcSPX	s+	s+	-	s+	-

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

A further comparative study of APrMF (also based on GA-DSCG for consistency) to recent advance evolutionary algorithms is also provided here and the results are summarized in Figure 7.11. In particular, APrMF is compared against the Fast Evolutionary Strategy [151], Fast Evolutionary Programming [152] Orthogonal GA (OGA/Q) [84], and Hybrid Taguchi-Genetic Algorithm (HTGA) [143] on common benchmark functions used that includes the non-rotated Ackley, Griewank and Rastrigin functions. The results in Figure 7.11 indicate the superior performance of the APrMF in converging to the global optimum more efficiently than all the counterparts considered, three of which were published recently in the IEEE Transactions on Evolutionary Computation.

Figure 7.11: Number of function evaluations used by different algorithms in solving the 30D benchmark functions

Last but not least, Table 7.4 summarizes the performance of ACMA (presented in Chapter 5) and APrMF across the set of 30-dimension benchmark functions. Note that the purpose of DMA presented in Chapter 6 is for demonstrating the concept of meme transmission, therefore it is not necessary to be included in this comparison study. Pairwise t-test comparison is conducted with 95% confidence level and the winner at each function is highlighted in bold. It is

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

Problems	ACMA5	APMA
Sphere	(742 ± 603)	(587 ± 320)
Elliptic	(1675 ± 1139)	(1057 ± 412)
Schwefel 1.02	0.0137 ± 0.0166	0 ± 0(87032)
Rastrigin	(2511 ± 1111)	(5932 ± 2374)
Rosenbrock	2.289 ± 3.678	8.27 ± 12.9
Scaffer	0.430 ± 0.224	0.541 ± 0.188

Table 7.11: Statistic test on comparing ACMA5 vs. APrMF (Null hypothesis: ACMA5 and APrMF incur equal number of evaluations vs. alternative hypothesis: ACMA5 and APrMF incur different number of evaluations to reach to the global optimum)

observed that while APrMF outperformed ACMA significantly on unimodal functions, its does not fair as well as ACMA on multimodal function. This suggests the current approximation method in APrMF may not be sufficiently accurate for multimodal functions.

7.5 Conclusion

One major issue pertaining to adaptation is to determine during runtime whether evolution or individual learning should be directed, with the objective of accelerating MA search. In contrast to earlier works which rely on semi-adhoc or heuristics methods, the essential backbone of our framework is a probabilistic model derived for establishing the theoretical upper bound of individual learning intensity. In particular, MA is modeled as a process involving the decision of embracing the separate actions of evolution or individual learning and analyze the probability of each process in locating the global optimum. This leads to a novel Probabilistic Memetic Framework, a quantitative formalization for adaptation by governing the learning intensity of each individual according to the theoretical upper bound while the search progresses.

An important aspect of PrMF is the determination of certain probabilistic measures based on the neighborhood structure of a individual learning. It was showed in this chapter that in practice, such probabilistic measures can be estimated reliably. For this purpose, the formalization of an Approximate PrMF is outlined to demonstrate how the PrMF can be put into

CHAPTER 7. A PROBABILISTIC MEMETIC FRAMEWORK

practice. For validation, empirical and theoretical studies on representative benchmark problems commonly used in the literature are then presented to demonstrate the characteristics and efficacies of the probabilistic memetic framework. Subsequent comparisons to recent state-of-the-art evolutionary algorithms, memetic algorithms and hybrid evolutionary local demonstrate that the proposed framework converges to good solutions efficiently.

Chapter 8

Complexity Analysis of Individual Learning Methods in Memetic Algorithm

Several factors has been established to define the difficulty or hardness of an optimization problem, including the dimensionality of the search space and the presence of noise. The “curse of dimensionality” has been extensively studied as one of the main challenge to the search performance of EAs and MAs [37, 110] with recent special section in the 2008 IEEE World Congress on Computational Intelligence (CEC@WCCI - 2008) dedicated for large scale global optimization. Similarly, the presence of noise is found to cause stagnation in evolutionary algorithms as well as numerical search methods [8, 19]. It is therefore beneficial to isolate the influence by some of these and analyze each separately. In this chapter, a complexity analysis of some individual learning methods, suitable for handling nonlinear programming problems, is presented. In particular, the computation and space complexity of several numerical learning methods are analyzed theoretically and validated using the quadratic sphere model via empirical study. In addition, the effect of Gaussian noise on the scalability of the numerical methods is also investigated. For the sake of readability, the complexity study is organized with representatives of second, first, and zeroth order methods. It is noted that in this section, we assume the objective of the search is to minimize the fitness function.

8.1 2^{nd} order method: Newton's method in optimization

For a continuous function f which is twice-differentiable and with Hessian matrix positive definite, the search sequence $\{\mathbf{x}^{(n)}\}$ is defined by:

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - [f''(\mathbf{x}^{(k-1)})]^{-1} f'(\mathbf{x}^{(k-1)}) \quad (\text{Eq. 8.1})$$

Theorem 1: Since Newton's method (Appendix B.1) operates based on the Taylor expansion of $f(\mathbf{x})$ in the neighborhood of a local optimum $\mathbf{x}^*$, it is shown to converge quadratically [124], i.e.,

$$\lim_{k \rightarrow \infty} \frac{|\mathbf{x}^{(k)} - \mathbf{x}^*|}{|\mathbf{x}^{(k-1)} - \mathbf{x}^*|^2} \leq \beta \quad (\text{Eq. 8.2})$$

for some positive value β , where $|\mathbf{x}^{(k)} - \mathbf{x}^*|$ represents the distance from $\mathbf{x}^{(k)}$ to the optimum $\mathbf{x}^*$.

Eq. 8.2 implies that for sufficiently large value of k , the following holds:

$$|\mathbf{x}^{(k)} - \mathbf{x}^*| \leq \beta |\mathbf{x}^{(k-1)} - \mathbf{x}^*|^2 \quad (\text{Eq. 8.3})$$

Since scaling does not affect the search performance, one may assume that Eq. 8.3 holds for all $k \geq 1$ and $|\mathbf{x}^{(k)} - \mathbf{x}^*| < 1$, without loss of generality.

Theorem 2: From Eq. 8.1, the computational complexity of the Newton method, in terms of iterations, necessary to reduce the distance to the optimum by a factor of 2^b , is $O(\ln(b))$, for some positive constant b .

Proof: From Eq. 8.3,

CHAPTER 8. COMPLEXITY ANALYSIS OF INDIVIDUAL LEARNING METHODS IN MEMETIC ALGORITHM

$$\begin{aligned}
|\mathbf{x}^{(k)} - \mathbf{x}^*| &\leq \beta |\mathbf{x}^{(k-1)} - \mathbf{x}^*|^2 \\
&\leq \beta^2 |\mathbf{x}^{(k-2)} - \mathbf{x}^*|^{2^2} \\
&\dots \\
&\leq \beta^k |\mathbf{x}^{(0)} - \mathbf{x}^*|^{2^k}
\end{aligned} \tag{Eq. 8.4}$$

where $\mathbf{x}^{(0)}$ denotes the initial search point. To reduce the distance to the optimum by a factor of 2^b , the following condition must be satisfied:

$$|\mathbf{x}^{(k)} - \mathbf{x}^*| = \frac{|\mathbf{x}^{(0)} - \mathbf{x}^*|}{2^b} \tag{Eq. 8.5}$$

Substituting Eq. 8.5 into Eq. 8.4, we have:

$$\begin{aligned}
|\mathbf{x}^{(0)} - \mathbf{x}^*| * 2^{-b} &\leq \beta^k |\mathbf{x}^{(0)} - \mathbf{x}^*|^{2^k} \\
2^{-b} &\leq \beta^k |\mathbf{x}^{(0)} - \mathbf{x}^*|^{2^k - 1} \\
-b &\leq k * \log_2 \beta + (2^k - 1) * \log_2 |\mathbf{x}^{(0)} - \mathbf{x}^*|
\end{aligned} \tag{Eq. 8.6}$$

Let $c = -\log_2 |\mathbf{x}^{(0)} - \mathbf{x}^*| > 0$ since $|\mathbf{x}^{(0)} - \mathbf{x}^*| < 1$, Eq. 8.6 then becomes:

$$b \geq (2^k - 1) * c - k * \log_2 \beta \tag{Eq. 8.7}$$

Eq. 8.7 shows that $2^k = O(b)$ or $k = O(\ln(b))$.

In each iteration, the complexity involved in the calculation of the Hessian matrix inverse is $O(n^3)$ operations, hence the total computational complexity can be derived as $O(n^3 \ln(b))$.

In practice, the evaluations of $f'(\mathbf{x})$ and $f''(\mathbf{x})$ are often performed using finite difference due to the lack of mathematical formulations or analytical forms (i.e., $f(\mathbf{x})$ is a black box). Moreover, the computational cost of the objective function $f(\mathbf{x})$ is normally composed of significantly more complex arithmetic operations. Therefore, it is more favorable to define the computational complexity in terms of function evaluation calls necessary. For Newton's

method, the computational cost incurred in each iteration is $O(n^2)$ function evaluation calls, which is the effort necessary for calculating the Hessian matrix. Hence the computational complexity of Newton's method is $O(n^2 \ln(b))$, in terms of function evaluation calls.

8.2 1^{st} order methods - gradient descent

The gradient descent strategy denotes the representative of 1^{st} -order numerical methods that is considered in the present analysis study. In this method, the sequence of search point maybe defined by

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \gamma f'(\mathbf{x}^{(k-1)}) \quad (\text{Eq. 8.8})$$

Theorem 3: For a continuous differentiable function f , gradient descent method converges linearly [124], i.e.,

$$|\mathbf{x}^{(k)} - \mathbf{x}^*| \leq \beta |\mathbf{x}^{(k-1)} - \mathbf{x}^*| \quad (\text{Eq. 8.9})$$

for some positive value $\beta \in (0, 1)$.

Theorem 4: When using the gradient descent strategy to optimize a continuous differentiable function f , the number of iterations made before the distance to local optimum reduces by a factor of 2^b is $O(b)$, for some constant b .

Proof: The following linear convergence condition is satisfied:

$$\begin{aligned} |\mathbf{x}^{(k)} - \mathbf{x}^*| &\leq \beta |\mathbf{x}^{(k-1)} - \mathbf{x}^*| \\ &\leq \beta^2 |\mathbf{x}^{(k-2)} - \mathbf{x}^*| \\ &\dots \\ &\leq \beta^k |\mathbf{x}^{(0)} - \mathbf{x}^*| \end{aligned} \quad (\text{Eq. 8.10})$$

where $0 < \beta < 1$.

Substituting Eq. 8.5 into Eq. 8.10,

$$\begin{aligned}
\frac{|\mathbf{x}^{(0)} - \mathbf{x}^*|}{2^b} &\leq \beta^k |\mathbf{x}^{(0)} - \mathbf{x}^*| \\
\Leftrightarrow 2^{-b} &\leq \beta^k \\
\Leftrightarrow -b &\leq k \log_2(\beta)
\end{aligned} \tag{Eq. 8.11}$$

Since $\beta < 1$, $\log_2(\beta) < 0$, we have $k * (-\log_2(\beta)) < b$ or $k = O(b)$. On the other hand, for steepest descent and other 1st-order methods, the computational cost for each iteration is $O(n)$ operations or $O(n)$ function evaluation calls to obtain the gradient vector. Therefore, the computational complexity of 1st-order method can be defined as $O(nb)$.

8.3 Zeroth order methods - direct search

Direct search methods do not make use of any gradient or Hessian information. Instead, the move directions are inferred solely based on the objective values. In this section, we provide an analysis on both deterministic and stochastic direct search methods.

8.3.1 Deterministic direct search methods

Deterministic methods are normally classified into 3 core categories: pattern search, simplex search (appendix B.3) and strategies with adaptive set of directions. Both pattern search and simplex search requires an average $O(n)$ function evaluations to identify a downhill direction. Starting with some initial step length (i.e., the initial lattice resolution in pattern search and initial simplex edge size in simplex search), these methods increase the step length with some constant ratio after a successful move while reducing the step length if no successful move is obtained. Clearly, the number of iterations required to move from $\mathbf{x}_{\text{start}}$ to $\mathbf{x}_{\text{end}}$ is $O(\log(\mathbf{x}_{\text{start}} - \mathbf{x}_{\text{end}}))$. The number of iterations before the function value is reduced to 2^{-b} -fraction of the initial value is then $O(b)$. Since the number of evaluations in each iteration is $O(n)$, the computational complexity of pattern search and simplex strategy becomes $O(nb)$, in terms of

CHAPTER 8. COMPLEXITY ANALYSIS OF INDIVIDUAL LEARNING METHODS IN MEMETIC ALGORITHM

function evaluations. In terms of arithmetic operations, the handling of n directional vectors in both pattern and simplex search methods would require $O(n^2)$ computational operations, which translates to a total complexity of $O(n^2b)$.

On the other hand, strategies with adaptive set of directions, e.g., interval division, Fibonacci division methods or the more complex strategy of Davis, Swann and Campey (DSC) (appendix B.4), all work on dimension-by-dimension basis. Clearly, in order for the distance to the optimum to reduce by a factor of 2^b , at least one of the dimensions must fulfil this condition, which requires no less than $O(b)$ function evaluations. Hence, in general, strategies with adaptive set of directions require no less than $O(nb)$ function evaluations to reduce the objective values by a factor of 2^b . In terms of arithmetic operations, the orthogonalization procedure of Gram-Schmidt [48] incurs $O(n^3)$ operations while the improved orthogonalization procedure of Palmer [122] reduces the number of computational operations to $O(n^2)$, hence the complexity of this set of search strategies is $O(n^3b)$ or $O(n^2b)$ depending on the orthogonalization procedures employed.

8.3.2 Stochastic direct search methods

Thus far, there has been relatively little work or none that focuses on the use of stochastic search methods as individual learning, particularly in the context of non-linear programming. In [129], Salomon showed that the ES and gradient-based search methods, such that steepest decent or quasi-Newton, share great resemblance, in terms of search behavior and performance. Jansen and Wegener [61], on the other hand, studied simulated annealing and an instance of $(1 + 1)$ evolutionary algorithm using binary objective function and showed that both methods are robust and achieve polynomial convergence time on average. However, the study was limited to the specific class of binary function considered. On continuous functions, Beyer [17] and Jägersküpper [60] had conducted extensive studies on the performance of $(1, +\lambda)$ evolutionary strategy (ES) and both showed that when properly configured, ES can achieve

CHAPTER 8. COMPLEXITY ANALYSIS OF INDIVIDUAL LEARNING METHODS IN MEMETIC ALGORITHM

promising progress rate to local optima and exhibit good robustness in performance. In the present study, we consider the $(1 + \lambda)$ ES as a potential stochastic individual learning method for MA design. It is worth noting that ES operates the search based only on the fitness values of the objective function, hence it is categorized here under the class of direct search methods.

8.3.2.1 Evolutionary Strategy

Algorithm 14 Evolutionary Strategy Individual Learning (Starting point $\mathbf{x}$)

```

1: Initialize  $good = 0, bad = 0$ 
2: Initialize mutation strength  $\sigma = 1.0$ 
3: while (stopping conditions are not satisfied) do
4:   /*Check for stagnation*/
5:   if ( $\sigma \leq \epsilon$ ) then
6:     Reset  $\sigma = 1.0$ 
7:   end if
8:   Generate the mutation vector  $\mathbf{m} = \{m_i\}, m_i = N(0, \sigma)$ 
9:   Create a new search point using Gaussian mutation  $\mathbf{x}^i = \mathbf{x} + \mathbf{m}$ 
10:  if ( $f(\mathbf{x}^i) < f(\mathbf{x})$ ) then
11:     $good = good + 1$ 
12:  else
13:     $bad = bad + 1$ 
14:  end if
15:   $evaluationCount = evaluationCount + 1$ 
16:  if ( $good + bad = n$ ) then
17:    if ( $good \geq 1/5 \times n$ ) then
18:      Increase mutation strength  $\sigma = \sigma \times ratio$ 
19:    else
20:      Increase mutation strength  $\sigma = \sigma / ratio$ 
21:    end if
22:     $good = 0, bad = 0$ 
23:  end if
24:  if ( $evaluationCount \bmod lambda = 0$ ) then
25:    Replace  $\mathbf{x}$  with the best search point  $\mathbf{x} = \arg \min_{\mathbf{x}^i \in \{\mathbf{x}^1, \dots, \mathbf{x}^\lambda\}} f(\mathbf{x}^i)$ 
26:  end if
27: end while

```

ES differs from other evolutionary algorithms including genetic algorithms, differential evolution, particle swarm optimization, by not involving any crossover operation. In ES, only

CHAPTER 8. COMPLEXITY ANALYSIS OF INDIVIDUAL LEARNING METHODS IN MEMETIC ALGORITHM

the mutation operator, normally in the form of Gaussian or Cauchy mutation, is employed. This suggests that the evolutionary form of ES may be regarded as a form of individual learning.

In numerical optimization, one of the strict condition that must be hold is the property of convergence guarantee to local optima and if possible, achieving good progress rate. Hence, it maybe more favorable to consider the ES with explicit mutation strength adaptation, instead of the self-adaption scheme. Here, we consider the $(1 + \lambda)$ ES with Gaussian mutation adapted with the 1/5-rule adaptation mechanism since it bears some resemblances to direct search methods. On the other hand, note that the stochastic nature of ES enables it to escape from local optima, therefore we propose an additional restart mechanism to the ES-based individual learning. Algorithm 14 describes our proposed ES-based individual learning strategy, labeled here as ES.

8.3.2.2 Computational and space complexity

In this sub-section, the analysis on the rate of convergence of ES is presented. Here, the best solution in the population at generation k is denoted as $\mathbf{x}^{(k)}$ while the distance from $\mathbf{x}^{(k)}$ to the optimum is denoted as $r^{(k)}$:

$$r^{(k)} = |\mathbf{x}^{(k)} - \mathbf{x}^*|$$

The operation of ES guarantees that $r^{(k)}$ is monotonically decreasing. The progress rate $\phi(r^{(k)})$ at the k^{th} iteration is then defined as:

$$\phi(r^{(k)}) = r^{(k-1)} - r^{(k)} = |\mathbf{x}^{(k-1)} - \mathbf{x}^*| - |\mathbf{x}^{(k)} - \mathbf{x}^*| \quad (\text{Eq. 8.12})$$

Rechenberg [127] showed that the progress rate $\phi(r)$ follows the formula:

$$\phi(r^{(k)}) = c \frac{r^{(k)}}{n} \quad (\text{Eq. 8.13})$$

where c denotes a constant. Eq. 8.12 and Eq. 8.13 become:

CHAPTER 8. COMPLEXITY ANALYSIS OF INDIVIDUAL LEARNING METHODS IN MEMETIC ALGORITHM

Table 8.1: Computational and space complexity of numerical optimization methods

	Computation (arithmetic)	Computation (evaluations)	Storage
Newton method	$O(n^3 \ln(b))$	$O(n^2 \ln(b))$	$O(n^2)$
Gradient descent	$O(nb)$	$O(nb)$	$O(n)$
Quasi Newton	$O(n^2b)$	$O(nb)$	$O(n^2)$
Simplex	$O(n^2b)$	$O(nb)$	$O(n)$
Pattern search	$O(n^2b)$	$O(nb)$	$O(n)$
DSCG	$O(n^3b)$	$O(nb)$	$O(n^2)$
DSCP	$O(n^2b)$	$O(nb)$	$O(n^2)$
$(1 + \lambda)$ ES	$O(n\lambda b)$	$O(\lambda b)$	$O(n)$

$$\begin{aligned}
|\mathbf{x}^{(k-1)} - \mathbf{x}^*| - |\mathbf{x}^{(k)} - \mathbf{x}^*| &= c \frac{|\mathbf{x}^{(k-1)} - \mathbf{x}^*|}{n} \\
|\mathbf{x}^{(k)} - \mathbf{x}^*| &= |\mathbf{x}^{(k-1)} - \mathbf{x}^*| \left(1 - \frac{c}{n}\right) \\
|\mathbf{x}^{(k)} - \mathbf{x}^*| &= |\mathbf{x}^{(0)} - \mathbf{x}^*| \left(1 - \frac{c}{n}\right)^k
\end{aligned}$$

indicating that ES possesses a linear convergence rate. Based on the proof for first order deterministic methods presented in Section 8.2, ES thus requires $O(b)$ iterations to reach a reduction of distance to the local optimum at a factor of 2^b . Each iteration, $(1 + \lambda)$ ES makes λ function evaluation calls. Hence in total ES has a computational complexity of $O(\lambda b)$ in terms of function evaluation calls.

From the above analysis, it is clear that ES will fare equally well to any first and zeroth order numerical methods, in terms of computational complexity. It is also worth noting that ES incurs very small space complexity of $O(n)$, to store the trial solutions, compared to other deterministic methods which normally require a space complexity of $O(n^2)$, for accommodating the Hessian matrix, approximated Hessian matrix, or the set of search directions.

Table 8.1 summarizes the computational and space complexity of various numerical optimization methods discussed above. The results provide a theoretical means for selecting suitable individual learning method, in terms of factor involving problem dimensionality n , level of precision necessary as defined by b , and memory space requirement.

8.4 Empirical study

8.4.1 Numerical study on noiseless Sphere model

From the proofs given in Sections 8.2 and 8.3, the first and zeroth order methods, including ES share equivalent linear order of complexity, with respect to the dimensionality of search problem. This implies that from a starting point, the number of function evaluations required to reduce the distance by a factor of 2^b is proportional to the problem dimension. In this section, the theoretical convergence rates formulated is empirically verified using the quadratic sphere model (Appendix A.1):

$$f_{\text{Sphere}}(\mathbf{x}) = \mathbf{x}^T \mathbf{x} = \sum_{i=1}^n x_i^2$$

where n is the problem dimension.

To validate the theoretical results experimentally, the required number of iterations, line searches, objective function calls, and the computation time were measured with respect to the initial values suggested in [131]:

$$x_i^{(0)} = x_i^* + \frac{-1^i}{\sqrt{n}}, \quad \text{for } i = 1 \dots n$$

to the first trial solution that satisfies the condition:

$$|\mathbf{x}^{(k)} - \mathbf{x}^*| \leq \frac{1}{10} |\mathbf{x}^{(0)} - \mathbf{x}^*| \quad (\text{Eq. 8.14})$$

The above condition, i.e., Eq. 8.14 is checked at each iteration and the search is terminated upon satisfying the condition.

Here, three numerical methods, namely DSCG, DFP and ES, are considered in the numerical study for comparison. In addition, two instances of ES are also examined, one with the λ value configured as equal to problem dimension size (labeled as ES1) and the other with λ set to 50 (denoted as ES2). For both ES methods, the results from the averaged of 50 independent

Figure 8.1: Performance of numerical methods on Sphere function

runs are reported. Figure 8.1 depicts the average number of function evaluation calls made by each numerical method on problems of increasing dimensions, up to 10^4 .

It is observed from the Figure that all three methods exhibited a linear convergence rate on the numerical results, which agrees well with theoretical analysis in sections 8.2 and 8.3. In addition, DFP outperforms all the other methods considered, in terms of the actual number of function evaluation calls made. Between the two instances of ES considered, ES2 incurred lower computational budget on problem of higher dimensions as compared to the ES1, while both operate equivalently on dimensions lower than 80. However, in comparison to the deterministic search counterparts, ESs incurred significantly larger evaluation calls on average.

8.4.2 Empirical study on noisy Sphere model

External disturbances, normally in the form of Gaussian noise, is a factor that is of interest to the complex engineering design community. The study on presence of noise thus serves to examine the robustness of a numerical method. Therefore, a scalability study would not be complete without considering the influence of noise on search performance. In this subsection,

CHAPTER 8. COMPLEXITY ANALYSIS OF INDIVIDUAL LEARNING METHODS IN MEMETIC ALGORITHM

Figure 8.2: Performance of numerical methods on increasing problem dimension

experiment on the Sphere problem in the presence of noise is also performed and defined by:

$$f_{NoisySphere}(\mathbf{x}) = f_{Sphere}(\mathbf{x}) \times (1 + \alpha|N(0, 1)|)$$

where α defines the magnitude of the noise.

In this study, different noise level that varies from 0.1 to 0.9 is considered. For each noise level, the experiment is conducted for increasing dimensions of up to 10^4 . Due to the stochastic nature of the objective function, the average of 50 independent runs are also reported for DSCG and DFP strategies. The success rate, defined here as the frequency that a numerical method successfully reduces the initial objective value by 10 times (i.e., condition in Eq. 8.14), within a maximum computational budget of 10^6 function evaluations. Further, it is worth noting that for DFP, the gradient vector is calculated using finite differencing.

It is noticed that DFP failed to display any success on even the simple 2-dimensional Sphere with a noise level of 0.1, since the gradient vector determined based on finite differencing is heavily affected by the presence of noise, leading the DFP to proceed in the wrong direction.

CHAPTER 8. COMPLEXITY ANALYSIS OF INDIVIDUAL LEARNING METHODS IN MEMETIC ALGORITHM

As such, the DFP search stopped prematurely due to the insignificant progress made after several iterations.

Figure 8.2 depicts the largest dimensional size that a method managed to attain at least one success out of fifty runs for the different approaches considered. It is observed that most numerical methods face significant difficulties operating at high dimensions and the success rate is observed to decrease drastically with increasing noise strength. Nevertheless, ES1 and ES2 exhibited similar superior robustness as compared to DSCG across the increasing dimensions. In particular, at the higher noise level range of 0.5 - 1.0, DSCG failed to converge to a solution that satisfies Eq. 8.14 on landscapes with dimensionality larger than 10.

Next, we study the number of function evaluations incurred by each method. Figure 8.3 presents the performance of ES1, ES2, and DSCG on Sphere with increasing noise levels. It is interesting to observe that despite the presence of noise, the required function evaluation calls scale up linearly with increasing dimensionality on all the methods considered. It is also worth noting that when the objective function is plagued by noise, DSCG incurs significantly higher number of function evaluations as compared to its performance on noiseless sphere function. In contrast, the two variants of ES did not incur additional computational cost, which probably highlights the robustness of ES under noisy environment.

CHAPTER 8. COMPLEXITY ANALYSIS OF INDIVIDUAL LEARNING METHODS IN MEMETIC ALGORITHM

8.3.a: ES1 with noise level 0.1 - 0.5

8.3.b: ES1 with noise level 0.6 - 1.0

8.3.c: ES2 with noise level 0.1 - 0.5

8.3.d: ES2 with noise level 0.6 - 1.0

8.3.e: DSCG with noise level 0.1 - 0.5

8.3.f: DSCG with noise level 0.6 - 1.0

Figure 8.3: Performance of numerical methods on increasing noise levels

8.5 Conclusions

In this chapter, the time and space complexity of several numerical methods typically used as individual learning in memetic algorithm have been analyzed. Theoretical results revealed that while 2^{nd} -order methods possess high convergence rate, they incur a large computational complexity, in terms of both arithmetic operations ($O(n^3)$) and function evaluation calls ($O(n^2)$), due to the cost of handling the Hessian. On the other hand, the 1^{st} and zeroth-order methods exhibited linear complexity in terms of function evaluation calls, though they shared a similar complexity with 2^{nd} -order methods, in terms of arithmetic operations. This suggests that the 1^{st} and zeroth-order methods are more suitable for optimization problems that are plagued with computationally expensive fitness evaluations [116]. On the other hand, if the gradient vector and Hessian matrix can be cheaply obtained, e.g., there exist analytical forms or approximation models [119], 2^{nd} -order methods may be preferred over low order methods.

Last but not least, when the objective function is affected by noise (in the form of Gaussian), it is shown empirically that stochastic direct search methods are more robust than the deterministic counterparts. They do not only perform successfully at high dimensions, but the number of function evaluations requirements are also significantly lower than deterministic numerical methods.

Chapter 9

Conclusions & Future Work

9.1 Conclusions

This dissertation has presented the research work completed on “*Adaptive and Formal Memetic Frameworks*” research. In particular, the design issues and parameters of MA are automatically configured at each decision point based on the proposed heuristic adaptation mechanisms or via formal methods.

In Chapter 4, the impact of each individual design issue as well as its relative impact on memetic search performance have been studied on three commonly used benchmark functions. Empirical study has highlighted the need for adapting individual learning intensity (t_{il}) and frequency (f_{il}) while suggesting that stratified scheme for selecting chromosomes undergoing individual learning represents a more robust choice without sacrificing solution quality.

Taking the cue, Chapter 5 introduced the idea of adaptive stratified individual selection scheme where the population was divided into separate groups or clusters based on their fitness values, and subsequently single chromosome from each group would be randomly selected to undergo individual improvement. The fitness range of each group was also used to monitor the diversity of the current population and a random restart would be conducted when the diversity is deemed to be low.

Chapter 6 considered the selection of suitable local learning methods. While genetic transmission can only be achieved vertically, i.e., from generation to generation, it is shown that

CHAPTER 9. CONCLUSIONS & FUTURE WORK

memetic computation permits transmission both vertically and horizontally. Diffusion memetic algorithm, resembling the mechanism of virus spreading, demonstrated an instance of memetic algorithm where memetic materials were transferred horizontally, i.e., among members of the same population.

Subsequently, Chapter 7 presented a formal probabilistic model that considers the various design issues of MA simultaneously. Using the proposed formal model, a theoretical upper bound for individual learning intensity ($t_i l$) was derived for adapting the search while it progresses online.

Last but not least, Chapter 8 presented a formal study on the scalability of different numerical search methods on problems plagued with high dimensionality and noise. Computation and space complexity of each method with respect to the dimensionality of the optimization problem have been theoretically formulated and subsequently empirically validated.

9.2 Future Work

Despite the extensive work towards the development of adaptive and formal memetic frameworks for continuous optimization problems, many challenges remain in the field that deserve further research studies. In what follows, several potential future work are listed and briefly discussed.

9.2.1 Greater exploration on the probabilistic memetic framework

The probabilistic memetic framework (PMA) presented in Chapter 7 employs the k-nearest neighbor method in approximating the probabilities of success for evolutionary search and individual learning, represented by the values of p_1 and p_2 . It would be worthwhile to consider other learning techniques, including distribution estimation or regression methods, to achieve more accurate estimation and hence inspiring to attain better search performances. Another

option is to model the evolutionary learning in MA as a Markov chain, such that the expected value of p_1 and p_2 at any specific generation can be calculated in an exact manner.

Last but not least, the proposed PMA considered only deterministic individual learning methods. Therefore, it would certainly be interesting to extend the current framework to include stochastic learning methods based on the analytical study made in Chapter 8.

9.2.2 Probabilistic memetic framework for discrete optimization

Without loss of generality, the mathematical model and theoretically upper bound of individual learning intensity apply to both continuous and discrete optimizations. In Chapter 7, APrMF was demonstrated for solving non-linear programming problems, where $\mathbf{x} \in \mathbb{R}^n$. A short term future work would be to consider PMA for handling discrete optimization problems. The core challenge lies in the definition of suitable distance metric and deriving new bounds for individual learning methods in discrete optimization.

9.2.3 Applications to real-world problems

In this dissertation, the proposed algorithms and frameworks have been experimented on benchmark problems. With the diverse characteristics presented in the problem set, this has the advantage of demonstrating the robustness of each approach. However, these popular benchmark may never completely model every aspect of the complexities found in real world problem, for instance, deceptive problem landscape, undefined or ill-conditioned search region, and etc. By exposing the proposed algorithm to the world of practical engineering optimization tasks, it would better reveal the true strengths and weaknesses of any proposed algorithm.

9.2.4 Towards “memetic-driven” MA

In the second and third generation MA where the evolution of meme is considered, most existing work perform the promotion of memes based on their ability to refine the genetic material

CHAPTER 9. CONCLUSIONS & FUTURE WORK

to achieve better fitness value efficiently. Such form of MA can be termed as “genetic driven”, i.e., the only objective is to promote individuals with better genes, with respect to the fitness function. It may be worthwhile to consider other forms of “memetic driven” MA, where memetic and genetic materials evolve separately under different objectives. Such forms of MA may not only lead to good solution quality to the search problem of interest but also the high efficacy solvers.

9.2.5 Memetic algorithm frameworks for problems in uncertain environment

In practice, engineers and scientists often have to work with problems with time-variant search space, which are commonly found in financial/stock market or traffic flow control. Empirical study presented in Chapter 8 reveals the poor performance of deterministic individual learning methods under uncertain environments and with problems of high dimensionality. In contrast, using the evolutionary strategy as individual learning method led to more robust performance in the presence of noise and on high dimensional problems. This suggests that for large scale problems under uncertain environment, it maybe more favorable to employ stochastic strategies as the individual learning in MA.

References

- [1] H. A. Abbass. Speeding up backpropagation using multiobjective evolutionary algorithms. *Neural Computation*, 15(11):2705–2726, 2003.
- [2] E. Alba and B. Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 9(2):126–142, 2005.
- [3] E. Alba, B. Dorronsoro, and H. Alfonso. Cellular Memetic Algorithms. *Journal of Computer Science & Technology*, 2005.
- [4] E. Alba, M. Giacobini, M. Tomassini, and S. Romero. Comparing synchronous and asynchronous cellular genetic algorithms. *Parallel Problem Solving from Nature-PPSN VII*, 2439:601–610, 2002.
- [5] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 6(5):443–462, 2002.
- [6] E. Alba and J. M. Troya. Improving flexibility and efficiency by adding parallelism to genetic algorithms. *Statistics and Computing*, 12(2):91–114, 2002.
- [7] A. Arcuri and X. Yao. A memetic algorithm for test data generation of object-oriented software. *IEEE Congress on Evolutionary Computation (CEC)*, pages 2048–2055, 2007.
- [8] D. V. Arnold and H. G. Beyer. A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, 24(1):135–159, 2003.
- [9] D. V. Arnold and R. Salomon. Evolutionary Gradient Search Revisited. *IEEE Transactions on Evolutionary Computation*, 11(4):480, 2007.

REFERENCES

- [10] T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. Taylor & Francis, 1997.
- [11] T. Back, F. Hoffmeister, and H. Schwefel. Survey of Evolution Strategies. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, 1991.
- [12] J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proceeding of the First International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, USA, 1985.
- [13] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, New Jersey, USA, 1987. Lawrence Erlbaum Associates, Inc.
- [14] S. Baluja. Structure and performance of fine-grain parallelism in genetic search. *Proceeding of the Fifth International Conference on Genetic Algorithms*, pages 155–162, 1993.
- [15] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler. Systematic integration of parameterized local search into evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):137–155, 2004.
- [16] R. Baraglia, J. I. Hidalgo, and R. Perego. A hybrid heuristic for the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 5(6):613–622, 2001.
- [17] H. G. Beyer. Toward a theory of evolution strategies: Some asymptotical results from the $(1, + \lambda)$ -theory. *Evolutionary computation*, 1(2):165–188, 1993.
- [18] H. G. Beyer and H. P. Schwefel. Evolution strategies—A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.
- [19] H.G. Beyer. Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. *Computer methods in applied mechanics and engineering*, 186(2-4):239–267, 2000.

REFERENCES

- [20] P. P. Bonissone, R. Subbu, N. Eklund, T. R. Kiehl, G. E. G. Res, and N. Y. Niskayuna. Evolutionary algorithms+ domain knowledge= real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 10(3):256–280, 2006.
- [21] R. Bradwell, K. Brown, E. Cantu-Paz, and B. Punch. Parallel asynchronous memetic algorithms. *Evolutionary Computation*, pages 157–159, 1999.
- [22] C. G. Broyden. The Convergence of a Class of Double-Rank Minimization Algorithms, Parts 1 and 2. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90, 1970.
- [23] B. D. Bunday. *Basic Optimization Methods*. Edward Arnold (Publishers) Ltd, 1985.
- [24] L. Buriol, P. M. França, and P. Moscato. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, 10(5):483–506, 2004.
- [25] E. Burke, P. Cowling, P. De Causmaecker, and G. V. Berghe. A Memetic Approach to the Nurse Rostering Problem. *Applied Intelligence*, 15(3):199–214, 2001.
- [26] E. K. Burke and A. J. Smith. Hybrid evolutionary techniques for the maintenance scheduling problem. *IEEE transactions on power systems*, 15(1):122–128, 2000.
- [27] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner. A Fast Adaptive Memetic Algorithm for Online and Offline Control Design of PMSM Drives. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 37(1):28–41, 2007.
- [28] L. L. Cavalli-Sforza. Cultural Evolution. *American Zoologist*, 26(3):845–856, 1986.
- [29] C. Y. Cheong, K. C. Tan, and B. Veeravalli. A multi-objective evolutionary algorithm for examination timetabling. *Journal of Scheduling*, 12(2):121–146, 2009.
- [30] G. Ciuprina, D. Ioan, and I. Munteanu. Use of intelligent-particle swarm optimization in electromagnetics. *IEEE Transactions on Magnetics*, 38(2 Part 1):1037–1040, 2002.
- [31] C. A. C. Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.

REFERENCES

- [32] G. A. Croes. A Method for Solving Traveling-Salesman Problems. *Operations Research*, 6(6):791–812, 1958.
- [33] D. D. Daberkow and D. N. Marvis. New approaches to conceptual and preliminary aircraft design: A comparative assessment of a neural network formulation and a response surface methodology. In *Proceeding of the 1998 World Aviation Conference, AIAA*, Anaheim, CA, USA, September 1998, AIAA-98-5509.
- [34] W. C. Davidon. Variable Metric Method for Minimization. *SIAM Journal on Optimization*, 1:1, 1991.
- [35] L. Davis et al. *Handbook of genetic algorithms*. Van Nostrand Reinhold New York, 1991.
- [36] R. Dawkins. *The Selfish Gene*. Oxford University Press, USA, 1989.
- [37] D. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. In *Notes to accompany lecture at AMS Conference on Mathematical Challenges of the 21st Century*, pages 1–32, Los Angeles, USA, August 2000.
- [38] D. J. Earl and M. W. Deem. Evolvability is a selectable trait. *Proceedings of the National Academy of Sciences*, 101(32):11531–11536, 2004.
- [39] G. Folino, C. Pizzuti, and G. Spezzano. Combining cellular genetic algorithms and local search for solving satisfiability problems. *Tenth IEEE International Conference on Tools with Artificial Intelligence, 1998.*, pages 192–198, 1998.
- [40] P. M. Franca, A. Mendes, and P. Moscato. A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, 132(1):224–242, 2001.
- [41] P. J. Gage, I. M. Kroo, and I. P. Sobieski. Variable-complexity genetic algorithm for topological design. *AIAA Journal*, 33(11):2212–2217, 1995.
- [42] C. Garcia-Martinez and M. Lozano. Hybrid Real-Coded Genetic Algorithms with Female and Male Differentiation. *Proceeding of the IEEE Congress on Evolutionary Computation (CEC)*, 1, 2005.

REFERENCES

- [43] C. K. Goh. *Multi-objective Memetic Algorithms*. Springer Verlag, 2009.
- [44] C. K. Goh, H. Y. Quek, E. J. Teoh, and K. C. Tan. Evolution and incremental learning in the iterative prisoner's dilemma. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 3:2629–2636, 2005.
- [45] C. K. Goh, E. J. Teoh, and K. C. Tan. Hybrid multiobjective evolutionary design for artificial neural networks. *IEEE Transactions on Neural Networks*, 19(9):1531–1548, 2008.
- [46] D. E. Goldberg. *Genetic Algorithms in Search, Optimisation, and Machine Learning*. Addison-Wesley, 1989.
- [47] D. E. Goldberg and S. Voessner. Optimizing Global-Local Search Hybrids. *Urbana*, 51:61801, 1999.
- [48] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Univ Pr, 1996.
- [49] J. J. Grefenstette. Lamarckian learning in multi-agent environments. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.
- [50] P. Hajela and J. Lee. Genetic algorithms in multidisciplinary rotor blade design. In *Proc. the 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Material Conference*, pages 2187–2197, New Orleans, USA, April 1995.
- [51] S. D. Handoko, C. K. Kwoh, and Y. S. Ong. Feasibility structure modeling: An effective chaperon for constrained memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 2009.
- [52] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, 2001.
- [53] W. E. Hart. *Adaptive Global Optimization with Local Search*. PhD thesis, University of California, 1994.
- [54] W. E. Hart, N. Krasnogor, and J. E. Smith. *Recent advances in memetic algorithms*, volume 166 of *Studies in Fuzzyness and Soft Computing*. Springer Berlin Heidelberg New York, 2004. ISBN 3-540-22904-3.

REFERENCES

- [55] S. M. K. Hasan, R. Sarker, D. Essam, and D. Cornforth. Memetic algorithms for solving job-shop scheduling problems. *Memetic Computing*, 1(1):69–83, March 2009.
- [56] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.
- [57] A. L. Hughes. *Evolution and Human Kinship*. Oxford University Press US, 1988.
- [58] H. Ishibuchi and S. Kaige. Comparison of multiobjective memetic algorithms on 0/1 knapsack problems. *Proc. of 2003 Genetic and Evolutionary Computation Conference Workshop Program (WOMA IV)*, 12:222–227, 2003.
- [59] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, 2003.
- [60] J. Jägersküpper. Probabilistic runtime analysis of $(1+, \lambda)$, ES using isotropic mutations. *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 461–468, 2006.
- [61] T. Jansen and I. Wegener. On the local performance of simulated annealing and the $(1+1)$ evolutionary algorithm. *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, page 476, 2006.
- [62] A. Jaszkievicz. Do multiple-objective metaheuristics deliver on their promises? a computational experiment on the set-covering problem. *IEEE Transactions on Evolutionary Computation*, 7(2):133–143, 2003.
- [63] Y. Jin. *Knowledge Incorporation in Evolutionary Computation*. Springer, Berlin, 2005.
- [64] A. J. Keane. Experiences with optimizers in structural design. In *Proc. the First International Conference on Adaptive Computing in Engineering Design and Control*, pages 14–27, Plymouth, UK, September 1994.
- [65] A. J. Keane. The options design exploration system <http://www.soton.ac.uk/~ajk/options/welcome.html>, 1995.

REFERENCES

- [66] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method*. in 'Fundamentals of Algorithms' series, SIAM Press., 2003.
- [67] G. Kendall, E. Soubeiga, and P. Cowling. Choice function and random hyperheuristics. *4th Asia-Pacific Conference on Simulated Evolution And Learning, SEAL*, pages 667–671, 2002.
- [68] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671, 1983.
- [69] M. Kirley, D. Newth, and D.G. Green. A Tree Based Genetic Algorithm for Solving Open-Shop Scheduling Problems. *International Journal of Knowledge Based Intelligent Engineering Systems*, 4(3):143–149, 2000.
- [70] M. Kirley and R. Stewart. Multiobjective evolutionary algorithms on complex networks. *Lecture Notes in Computer Science*, 4403:81, 2007.
- [71] J. Knowles and D. Corne. Memetic algorithms for multiobjective optimization: Issues, methods and prospects. *Studies in Fuzziness and Soft Computing*, 166:313–354, 2004.
- [72] J. D. Knowles and D. W. Corne. A comparison of diverse approaches to memetic multiobjective combinatorial optimization. *Proceeding of the Genetic and Evolutionary Computation Conference Workshop Program*, pages 103–108, 2000.
- [73] U. Kohlmorgen, H. Schmeck, and K. Haase. Experiences with fine-grained parallel genetic algorithms. *Annals of Operations Research*, 90:203–219, 1999.
- [74] N. Krasnogor. Coevolution of genes and memes in memetic algorithms. *Graduate Student Workshop*, page 371, 13 1999.
- [75] N. Krasnogor, B. P. Blackburne, E. K. Burke, and J. D. Hirst. Multimeme algorithms for protein structure prediction. In *PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, pages 769–778, London, UK, 2002. Springer-Verlag.
- [76] N. Krasnogor and S. Gustafson. A study on the use of “self-generation” in memetic algorithms. *Natural Computing*, 3(1):53–76, 2004.

REFERENCES

- [77] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.
- [78] N. Krasnogor, J. Smith, et al. A memetic algorithm with self-adaptive local search: TSP as a case study. *Proceeding of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 987–994, 2000.
- [79] K.K. Kshetrapalapuram and M. Kirley. Mining classification rules using evolutionary multi-objective algorithms. *Lecture notes in computer science*, 3683:959, 2005.
- [80] K. W. C. Ku, M. W. Mak, and W. C. Siu. A study of the Lamarckian evolution of recurrent neural networks. *IEEE Transactions on Evolutionary Computation*, 4(1):31–42, 2000.
- [81] P. Lacomme, C. Prins, and W. Ramdane-Cherif. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(1):159–185, 2004.
- [82] E. Lamma, L. M. Pereira, and F. Riguzzi. Multi-agent logic aided lamarckian learning. *Technical Report DEIS-LIA-00-004, Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna (Italy), 2000. LIA Series*, 2000.
- [83] M. W. S. Land. *Evolutionary Algorithms with Local Search for Combinatorial Optimization*. PhD thesis, UNIVERSITY OF CALIFORNIA, 1998.
- [84] Y. W. Leung and Y. Wang. An orthogonal genetic algorithm with quantization for global-numerical optimization. *IEEE Transactions on Evolutionary Computation*, 5(1):41–53, 2001.
- [85] R. M. Lewis, V. Torczon, and M. W. Trosset. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, 124(1-2):191–207, 2000.
- [86] J. J. Liang and P. N. Suganthan. Dynamic multi-swarm particle swarm optimizer. *Proceeding of the Swarm Intelligence Symposium (SIS)*, pages 124–129, 2005.

REFERENCES

- [87] J. J. Liang, P. N. Suganthan, and K. Deb. Novel composition test functions for numerical global optimization. *Proceeding of the Swarm Intelligence Symposium (SIS)*, pages 68–75, 2005.
- [88] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, pages 498–516, 1973.
- [89] Y. C. Lin, K. S. Hwang, and F. S. Wang. Hybrid differential evolution with multiplier updating method for nonlinear constrained optimization problems. *Proceeding of the Congress on Evolutionary Computation (CEC)*, 2002.
- [90] B. Liu, L. Wang, and Y. H. Jin. An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 37(1):18–27, 2007.
- [91] W. N. Martin, J. Lienig, and J. P. Cohoon. Island (migration) models: evolutionary algorithms based on punctuated equilibria. *Handbook of Evolutionary Computation*, 6:3, 1997.
- [92] S. Z. Martínez and C. A. C. Coello. Hybridizing an evolutionary algorithm with mathematical programming techniques for multi-objective optimization. *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 769–770, 2008.
- [93] M. D. McKay, R. J. Beckman, and W. J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 42(1):55–61, 2000.
- [94] Y. Mei, K. Tang, and X. Yao. Improved Memetic Algorithm for Capacitated Arc Routing Problem. *IEEE Congress on Evolutionary Computation*, 2009.
- [95] P. Merz. Advanced Fitness Landscape Analysis and the Performance of Memetic Algorithms. *Evolutionary Computation*, 12(3):303–325, 2004.
- [96] P. Merz and K. Katayama. Memetic algorithms for the unconstrained binary quadratic programming problem. *Biosystems*, 78(1-3):99–118, 2004.

REFERENCES

- [97] R. Meuth, M. H. Lim, Y. S. Ong, and D. C. Wunsch. A proposition on memes and meta-memes in computing for higher-order learning. *Memetic Computing*, 1(2):85–100, 2009.
- [98] Z. Michalewicz. *Genetic algorithms+ data structures= evolution programs*. Springer-Verlag London, UK, 1996.
- [99] D. Molina, F. Herrera, and M. Lozano. Adaptive Local Search Parameters for Real-Coded Memetic Algorithms. *Proceeding of the IEEE Congress on Evolutionary Computation (CEC)*, 1, 2005.
- [100] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P Report*, 826, 1989.
- [101] H. Muhlenbein, M. Schomisch, and J. Born. The Parallel Genetic Algorithm as a Function Optimizer. *Proceeding of the Fourth International Conference on Genetic Algorithms*, pages 271–78, 1991.
- [102] P. B. Nair and A. J. Keane. Passive vibration suppression of flexible space structures via optimal geometric redesign. *American Institute of Aeronautics and Astronautics Journal*, 39(7):1338–1346, 2001.
- [103] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.
- [104] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga. Implementation of an effective hybrid GA for large-scale traveling salesman problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):92–99, 2007.
- [105] Q. C. Nguyen, Y. S. Ong, H. Soh, and J. L. Kuo. Multiscale Approach to Explore the Potential Energy Surface of Water Clusters $(H_2O)^n$ $n \leq 8$. *The Journal of Physical Chemistry A*, 112(28):6257–6261, 2008.
- [106] Q. H. Nguyen, Y. S. Ong, and N. Krasnogor. A study on the design issues of Memetic Algorithm. In *IEEE Congress on Evolutionary Computation, 2007. CEC 2007.*, pages 2390–2397, 2007.

REFERENCES

- [107] Q. H. Nguyen, Y. S. Ong, and M. H. Lim. Non-genetic transmission of memes by diffusion. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1017–1024. ACM New York, NY, USA, 2008.
- [108] Q. H. Nguyen, Y. S. Ong, and M. H. Lim. A probabilistic memetic framework. *Evolutionary Computation, IEEE Transactions on*, 13(3):604–623, 2009.
- [109] Q. H. Nguyen, Y. S. Ong, M. H. Lim, and N. Krasnogor. Adaptive cellular memetic algorithms. *Evolutionary Computation*, 17(2):231–256, 2009.
- [110] N. Noman and H. Iba. Enhancing differential evolution performance with local search for high dimensional function optimization. *Proceeding of the 2005 conference on Genetic and Evolutionary Computation*, pages 967–974, 2005.
- [111] N. Noman and H. Iba. Accelerating Differential Evolution Using an Adaptive Local Search. *Evolutionary Computation, IEEE Transactions on*, 12(1):107–125, 2008.
- [112] M. Olhofer, T. Arima, T. Sonoda, and B. Sendhoff. Optimisation of a stator blade used in a transonic compressor cascade with evolution strategies. *Adaptive Computing in Design and Manufacture*, pages 45–54, 2000.
- [113] Y. S. Ong and A. J. Keane. Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110, 2004.
- [114] Y. S. Ong, N. Krasnogor, and H. Ishibuchi. Special issue on memetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 2006.
- [115] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong. Classification of Adaptive Memetic Algorithms: A Comparative Study. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, 36(1):141, 2006.
- [116] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling. *AIAA Journal*, 41(4):687–696, 2003.

REFERENCES

- [117] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong. Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems. *Knowledge Incorporation in Evolutionary Computation, Studies in Fuzziness and Soft Computing Series*. Springer Verlag, 2004.
- [118] Y. S. Ong, P. B. Nair, and K. Y. Lum. Max-Min Surrogate-Assisted Evolutionary Algorithm for Robust Aerodynamic Design. *IEEE Transactions on Evolutionary Computation*, 10(4):392–404, 2006.
- [119] Y. S. Ong, P. B. Nair, and K. Y. Lum. Evolutionary algorithm with hermite radial basis function interpolants for computationally expensive adjoint solvers. *Computational Optimization and Applications*, in press, 2006.
- [120] E. Özcan, B. Bilgin, and E.E. Korkmaz. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23, 2008.
- [121] I. Paenke, Y. Jin, and J. Branke. Balancing Population-and Individual-Level Adaptation in Changing Environments. *Adaptive Behavior*, 17(2):153–174, 2009.
- [122] J. R. Palmer. An improved procedure for orthogonalising the search vectors in Rosenbrock’s and Swann’s direct search optimisation methods. *The Computer Journal*, 12(1):69, 1969.
- [123] C. Pettey. *Diffusion (cellular) models*. *Handbook of Evolutionary Computation*. IOP Publishing, 1997.
- [124] E. Polak. *Computational methods in optimization: a unified approach*. Academic Pr, 1971.
- [125] M. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155, 1964.
- [126] N. J. Radcliffe. The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 10(4):339–384, 1994.
- [127] I. Rechenberg. *Evolutionssstrategie*. Frommann-Holzboog, 1973.

REFERENCES

- [128] M. G. Resende. Parallel metaheuristics for combinatorial optimization. *International School on Advanced Algorithmic Techniques for Parallel Computation with Applications*, 1999.
- [129] R. Salomon. Evolutionary algorithms and gradient search: similarities and differences. *IEEE Transactions on Evolutionary Computation*, 2(2):45–55, 1998.
- [130] K. Schuchardt, B. Didier, and G. Black. Ecce - a problem solving environment's evolution toward grid service and a web architecture. *Concurrency Computation Practice and Experience*, 14(13-15):1221–1239, 2002.
- [131] H. P. Schwefel. *Evolution and optimum seeking*. Wiley New York, 1995.
- [132] J. E. Smith. Co-evolving Memetic Algorithms: Initial Investigations. *Parallel Problem Solving from Nature VII (PPSN)*, pages 537–546, 2002.
- [133] J. E. Smith. Co-evolving memetic algorithms: a learning approach to robust scalable optimisation. *Proceeding of the Congress on Evolutionary Computation (CEC)*, 1, 2003.
- [134] J. E. Smith. Coevolving Memetic Algorithms: A Review and Progress Report. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 37(1):6–17, 2007.
- [135] W. B. Song. *Shape Optimisation Of Turbine Blade Firtrees*. PhD thesis, Univ. of Southampton, 2002.
- [136] P. Spiessens and B. Manderick. A massively parallel genetic algorithm: Implementation and rst analysis. *Proceeding of the Fourth International Conference on Genetic Algorithms, San Diego, CA*, pages 279–287, 1991.
- [137] R. Storn and K. Price. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.
- [138] K. C. Tan, E. J. Teoh, Q. Yu, and K. C. Goh. A hybrid evolutionary algorithm for attribute selection in data mining. *Expert Systems With Applications*, 36(4):8616–8630, 2009.

REFERENCES

- [139] J. Tang, M. H. Lim, and Y. S. Ong. Parallel Memetic Algorithm with Selective Local Search for Large Scale Quadratic Assignment. *Intl. Journal of Innovative Computing, Information and Control*, 2(6):1399–1416, Dec 2006.
- [140] J. Tang, M. H. Lim, and Y. S. Ong. Diversity-Adaptive Parallel Memetic Algorithm for Solving Large Scale Combinatorial Optimization Problems. *Soft Computing Journal*, 11(9):873–888, July 2007.
- [141] S. Tong, D. Powell, and S. Goel. Integration of artificial intelligence and numerical optimization techniques for the design of complex aerospace systems. *AIAA, Aerospace Design Conference, Irvine, CA*, page 1992, 1992.
- [142] H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao. An evolutionary algorithm for large traveling salesman problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(4):1718–1729, 2004.
- [143] J. T. Tsai, T. K. Liu, and J. H. Chou. Hybrid Taguchi-genetic algorithm for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 8(4):365–377, 2004.
- [144] S. Tsutsui and Y. Fujimoto. Forking Genetic Algorithm with Blocking and Shrinking Modes (fGA). *Proceeding of the 5th International Conference on Genetic Algorithms table of contents*, pages 206–215, 1993.
- [145] A. S. S. M. B. Ullah, R. Sarker, D. Cornforth, and C. Lokan. An agent-based memetic algorithm (ama) for solving constrained optimization problems. *IEEE Congress on Evolutionary Computation, 2007. CEC 2007*, pages 999–1006, 2007.
- [146] A. Vicini and D. Quagliarella. Airfoil and wing design through hybrid optimization strategies. *AIAA Journal*, 37(5):634–641, 1999.
- [147] H. Wang, D. Wang, and S. Yang. A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 13(8):763–780, 2009.

REFERENCES

- [148] H. Wang, S. Yang, W. H. Ip, and D. Wang. Adaptive Primal–Dual Genetic Algorithms in Dynamic Environments. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society*, 2009.
- [149] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
- [150] X. Yao. Simulated annealing with extended neighborhood. *Int. Journal of Computer Mathematics*, 40:169–189, 1991.
- [151] X. Yao and Y. Liu. Fast evolution strategies. *Control and Cybernetics*, 26(3):467–496, 1997.
- [152] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, 1999.
- [153] X. Yu, K. Tang, T. Chen, and X. Yao. Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. *Memetic Computing*, 1(1):3–24, 2009.
- [154] B. Yuan and M. Gallagher. Experimental results for the special session on real-parameter optimization at CEC 2005: A simple, continuous EDA. *Proceeding of the IEEE Congress on Evolutionary Computation (CEC)*, 2005.
- [155] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.
- [156] P. Zou, Z. Zhou, G. Chen, X. Yao, and C. Hefei. A novel memetic algorithm with random multi-local-search: A case study of TSP. *Evolutionary Computation, 2004. CEC2004. Congress on*, 2, 2004.

Appendix A

Benchmark Test Problems

Some commonly used benchmark test problems already extensively discussed in the literature are used here in this research work. They represent classes of general constrained, unimodal and multimodal continuous parametric test problems. The unimodal benchmark test problems are bound-constrained Sphere, Step and Rosenbrock functions. All the benchmark test problems considered have diverse fitness landscapes for study. This is the set of benchmark functions recently proposed for evolutionary optimization [87].

In the equations, n is the dimensionality, $\mathbf{x}$ is the vector of design variables to be optimized. Note that most problems are symmetrical and their global optima locate at the origin. In order to avoid any biased of the search algorithms on exploiting this symmetric property of the benchmark functions, the design variables are shifted. $\mathbf{o}$ is the shifted global optimum and $\mathbf{z} = \mathbf{x} - \mathbf{o}$ is the vector of shifted variables. For rotated problems, $\mathbf{M}$ is the rotation matrix and $\mathbf{z} = \mathbf{M} \times (\mathbf{x} - \mathbf{o})$ is the vector of shifted rotated design variables.

A.1 Sphere function

The Sphere test problem is a smooth, symmetrical function and is used to provide a measure of the general efficiency of the proposed strategy. It has a single minimum located at $(0, \dots, 0)$. The function is defined as

$$F_{Sphere} = \sum_{i=1}^n (z_i^2) \quad (\text{A.1})$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} \in [-100, 100]^n.$$

Appendix A. Benchmark Test Problems

A.2 Step function

The Step function consist of flat plateaus with $slope = 0$ in an underlying continuous function. It is hard to find the global optimum because minor changes of the object variables do not affect the fitness. Therefore no conclusions about the search direction are possible. It is defined as

$$F_{step} = 6n + \sum_{i=1}^n \lfloor z_i \rfloor \quad (A.2)$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} \in [-5.12, 5.12]^n.$$

A.3 Schwefel's problem 1.2

$$F_{Schwefel1.2} = \sum_{i=1}^n (\sum_{j=1}^i z_j)^2 \quad (A.3)$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} \in [-100, 100]^n.$$

A.4 Elliptic function

$$F_{Elliptic} = \sum_{i=1}^n (10^6)^{\frac{i-1}{n-1}} \quad (A.4)$$

$$\mathbf{z} = \mathbf{M} \times (\mathbf{x} - \mathbf{o}), \quad \mathbf{x} \in [-100, 100]^n.$$

A.5 Schwefel's problem 1.2 with Noise in Fitness

$$F_{Schwefel1.2+Noise} = (\sum_{i=1}^n (\sum_{j=1}^i z_j)^2) \times (1 + 0.4|N(0, 1)|) \quad (A.5)$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} \in [-100, 100]^n.$$

Appendix A. Benchmark Test Problems

A.6 Schwefel's problem 2.6 with Global Optimum on Bounds

$$F_{Schwefel2.6} = \max |\mathbf{A}_i \times \mathbf{z}| \quad (\text{A.6})$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} \in [-100, 100]^n.$$

$$\mathbf{A}_i = \{a_{ij}\}, \quad i, j = 1, 2, \dots, n, \quad a_{ij} = \text{random}(-500, 500), \quad \det(\mathbf{A}) \neq 0.$$

A.7 Rosenbrock Test Function

Rosenbrock test function is nonseparable, with highly correlated decision variables. As an added difficulty the minimum is located at $(1, \dots, 1)$ in a long, flat-bottomed, curved, and narrow valley. It is defined as

$$F_{Rosenbrock} = \sum_{i=1}^{n-1} (100 \times (z_{i+1} - z_i^2)^2 + (1 - z_i)^2) \quad (\text{A.7})$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} \in [-100, 100]^n.$$

A.8 Ackley Test Function

Ackley test function is symmetric and very bumpy. Its number of local minima increases exponentially with the problem dimension. It has a global minimum located at $(1, \dots, 1)$ with very strong local features. It is defined as

$$F_{Ackley} = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n z_i^2}} - e^{\frac{1}{n}\sum_{i=1}^n \cos(2\pi z_i)} \quad (\text{A.8})$$

$$\mathbf{z} = \mathbf{M} \times (\mathbf{x} - \mathbf{o}), \quad \mathbf{x} \in [-32, 32]^n.$$

A.9 Griewank Test Function

The Griewank test problem is a high dimensional multimodal function with many local minima and a global minimum located at $(0, \dots, 0)$. It is defined as

$$F_{Griewank} = 1 + \sum_{i=1}^n z_i^2 / 4000 - \prod_{i=1}^n \cos(z_i / \sqrt{i}) \quad (\text{A.9})$$

$$\mathbf{z} = \mathbf{M} \times (\mathbf{x} - \mathbf{o}), \quad \mathbf{x} \in [-600, 600]^n.$$

Appendix A. Benchmark Test Problems

This function has inter-parameter linkage due to the presence of the product term. However, the effect decreases as the number of parameters increases. The Griewank function with 10 dimensions, $n = 10$, has more than 500 local minima in the hypercube $[-600, 600]^{10}$. It has a very rugged landscape and is difficult to optimize.

A.10 Rastrigin Test Function

The Rastrigin test problem is also a high dimensional multimodal function with many local minima and a global minimum located at $(0, \dots, 0)$. It also has a very rugged landscape and is difficult to optimize.

$$F_{Rastrigin} = 10n + \sum_{i=1}^n (z_i^2 - 10 \cos(2\pi z_i)) \quad (\text{A.10})$$

$$\mathbf{z} = \mathbf{M} \times (\mathbf{x} - \mathbf{o}), \quad \mathbf{x} \in [-600, 600]^n.$$

A.11 Weierstrass Test Function

$$F_{Weierstrass} = \sum_{i=1}^n (\sum_{k=0}^{k_{max}} (a^k \cos(2\pi b^k (z_i + 0.5)))) - n \sum_{k=0}^{k_{max}} (a^k \cos(\pi b^k)) \quad (\text{A.11})$$

$$\mathbf{z} = \mathbf{M} \times (\mathbf{x} - \mathbf{o}), \quad \mathbf{x} \in [-0.5, 0.5]^n.$$

A.12 Schwefel's problem 2.13

$$F_{Schwefel213} = \sum_{i=1}^n (A_i - B_i(\mathbf{x}))^2 \quad (\text{A.12})$$

$$A_i = \sum_{j=1}^n (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j), \quad B_i(\mathbf{x}) = \sum_{j=1}^n (a_{ij} \sin x_j + b_{ij} \cos x_j)$$

$$\mathbf{x} \in [-\pi, \pi]^n, \quad a_{ij}, b_{ij} = \text{random}(-100, 100), \quad \alpha_j = \text{random}(-\pi, \pi). \quad i, j = 1, 2, \dots, n.$$

Appendix A. Benchmark Test Problems

A.13 Expanded Griewank's plus Rosenbrock's function

$$F_{GrieRos} = \sum_{i=1}^n F_{Griewank}(F_{Rosenbrock}(z_i, z_{i+1})), \quad z_{n+1} = z_1 \quad (\text{A.13})$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \quad \mathbf{x} \in [-5, 5]^n$$

A.14 Expanded Scaffer's F6 function

$$F_{GrieRos} = \sum_{i=1}^n F(z_i, z_{i+1}), \quad z_{n+1} = z_1 \quad (\text{A.14})$$

$$F(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{(1 + 0.001(x^2 + y^2))^2}$$

$$\mathbf{z} = \mathbf{M} \times (\mathbf{x} - \mathbf{o}), \quad \mathbf{x} \in [-100, 100]^n$$

A.15 Composition functions

$$F_{Hybrid1} = f_{16} \text{ in [87].}$$

$$F_{Hybrid2} = f_{19} \text{ in [87].}$$

A.16 Frequency Modulation Sounds Parameter Identification Problem [144]

$$F_{FMS} = \sum_{t=0}^{100} (y(t) - y_0(t)) \quad (\text{A.15})$$

$$y(t) = a_1 \sin(w_1 t\theta + a_2 \sin(w_2 t\theta + a_3 \sin(w_3 t\theta)))$$

$$y_0(t) = 1.0 \sin(5.0 t\theta + 1.5 \sin(4.8 t\theta + 2.0 \sin(4.9 t\theta)))$$

$$a_i, w_i \in [-6.4, 6.35], \quad i = 1, 2, 3$$

Appendix B

Conventional numerical search methods: pseudo-code and flow charts

B.1 Newton method

Newton's method gives rise to a wide and important class of algorithms that require computation of the gradient vector

$$\Delta f(\mathbf{x}) = \begin{pmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \dots \\ \partial f / \partial x_n \end{pmatrix}$$

and the Hessian matrix

$$\Delta^2 f(\mathbf{x}) = (\partial^2 f / (\partial x_i \partial x_j))$$

Consider at iteration k :

$$f(\mathbf{x}^{(k)} + \mathbf{s}) = f(\mathbf{x}^{(k)} + \mathbf{s}) + \Delta f(\mathbf{x}^{(k)})^T \mathbf{s} + \frac{1}{2} \mathbf{s}^T \Delta^2 f(\mathbf{x}^{(k)}) \mathbf{s}$$

for some small $\mathbf{s}$. When the Hessian matrix is positive definite, the above equation has a unique minimizer, obtained by solving:

$$\Delta^2 f(\mathbf{x}^{(k)}) \mathbf{s} = -\Delta f(\mathbf{x}^{(k)})$$

And the next search point will be:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}$$

Appendix B. Conventional numerical search methods

The convergence rate is quadratic, that is:

$$|\mathbf{x}^{(k)} - \mathbf{x}^*| < \beta |\mathbf{x}^{(k+1)} - \mathbf{x}^*|^2$$

where $\mathbf{x}^*$ is the global optimum.

B.2 Quasi-Newton strategies: Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) methods

B.2.1 DFP method

Much greater interest has been shown for a group of second order gradient methods that attempt to approximate the Hessian matrix and its inverse during the iterations only from first order data. This now extensive class of quasi Newton strategies has grown out of the early work of Davidon and Fletcher and Powell. The Davidon-Fletcher-Powell or DFP method and some variants of it are also known as variable metric strategies. They are sometimes also regarded as conjugate gradient methods because in the quadratic case they generate conjugate directions. The basic recursion formula for the DFP method is very much similar to those 2nd-order methods:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + s^{(k)} \mathbf{v}^{(k)}$$

with the search direction $\mathbf{v}^{(k)}$ is obtained from:

$$\mathbf{v}^{(k)} = -H^{(k)T} f'(\mathbf{x}^{(k)})$$

where $H^{(k)}$ is the approximated inverse Hessian matrix and is updated iteratively using the rule:

$$H^{(k+1)} = H^{(k)} + A^{(k)}$$

With the correction matrix $A^{(k)}$ is derived from information collected during the last iteration.

Appendix B. Conventional numerical search methods

$$A^{(k)} = \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)T}}{\mathbf{y}^{(k)T}\mathbf{z}^{(k)}} - \frac{H^{(k)}\mathbf{z}^{(k)}\mathbf{z}^{(k)T}H^{(k)}}{\mathbf{z}^{(k)T}H^{(k)}\mathbf{z}^{(k)}}$$

where $\mathbf{y}^{(k)}$ is the change in the variable vector

$$\mathbf{y}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = s^{(k)}\mathbf{v}^{(k)}$$

and $\mathbf{z}^{(k)}$ is the change in the gradient vector

$$\mathbf{z}^{(k)} = f'(\mathbf{x}^{(k+1)}) - f'(\mathbf{x}^{(k)})$$

B.2.2 BFGS method

Broyden [22] proposed an alternative update formula:

$$A^{(k)} = \frac{\mathbf{z}^{(k)}\mathbf{z}^{(k)T}}{\mathbf{z}^{(k)T}\mathbf{y}^{(k)}} - \frac{H^{(k)}\mathbf{y}^{(k)}\mathbf{y}^{(k)T}H^{(k)}}{\mathbf{y}^{(k)T}H^{(k)}\mathbf{y}^{(k)}}$$

Quasi-Newton method using the above update rule is referred to as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) strategy.

B.3 Simplex method

B.3.1 The Basic Simplex Method

The optimization begins with the initial trials. The trial conditions are spread out efficiently in the form of a simplex. Note that the number of initial trials is equal to the number of control variables plus one. The shapes of the simplex in a one, a two and a three variable search space, are a line, a triangle or a tetrahedron as shown in Figure B.1.

The basic simplex algorithm consists of a few rules:

- The first rule is to reject the trial with the least favorable fitness value in the current simplex.

At each iteration, new search point is calculated, by reflection into the search space opposite the undesirable point. This new trial replaces the least favorable trial in the simplex and the process is repeated.

Appendix B. Conventional numerical search methods

Figure B.1: Geometric interpretation of lower dimensional simplices

- The second rule is never to return to the search points that have just been rejected.

If the generated search point is also the least favorable point, we will choose the second least favorable point in the simplex to replace. This rule is to avoid oscillating between two search points.

Figure B.2: An example of a typical optimization sequence with the basic simplex method. Change in the levels for two control variables with the response marked as contours.

Besides the two main rules, two more rules are also applied.

- Trials retained in the simplex for a specified number of steps are reevaluated. The reevaluation rule avoids the simplex to be stuck around a false favorable point.
- Trials generated outside the search range are not calculated. Instead a very unfavorable fitness value is assigned, forcing the simplex to move away from the boundary.

The calculations in the basic simplex algorithm are outlined in Figure B.3. For each simplex the following labels are used: **W** for the least favorable trial or the trial being rejected, **B** for the most favorable trial and **N_W** for the second least favorable trial (i.e. next-to-the worst).

Appendix B. Conventional numerical search methods

Figure B.3: Basic Simplex method flowchart.

B.3.2 The Modified Simplex Method [103]

The modified simplex method has much in common with the basic method, but can adjust its shape and size depending of the response in each step. This method is also called the variable-size simplex method. Several new rules are added to the basic simplex rules. These new rules make the simplex:

- Expand in a direction of more favorable conditions.
- Contract if a move was taken in a direction of less favorable conditions

The procedures for expansion and contraction enable the modified simplex both to accelerate along a successful track of improvement and to home in on the optimum conditions. Therefore the modified simplex will usually reach the optimum region quicker than with the basic method and pinpoint the optimum levels more closely. The degree of contraction depends on how unfavorable the new response is. Figure B.4 illustrates the different moves with the modified simplex method.

Figure B.5 outlines the modified simplex algorithm and some generally accepted modifications to the original version of the modified simplex method [103]. For each simplex the following labels are used: **W** for the least favorable trial or the trial being rejected, **B** for the most favorable trial and **N_W** for the second least favorable trial (i.e. next-to-the worst).

Appendix B. Conventional numerical search methods

Figure B.4: Different simplex moves from the rejected trial condition (W). R = reflection, E = expansion, C+ = positive contraction and C- = negative contraction.

Figure B.5: Modified Simplex method flowchart.

Appendix B. Conventional numerical search methods

B.4 Davies, Swann, and Campey method with Gram-Schmidt (DSCG) or Palmer (DSCP) orthogonalization

In the DSC procedure, the search begins with starting point $\mathbf{x}^{(k,i)} = \mathbf{x}^{(1,0)}$, where i and k are the direction and iteration counters, respectively. A line search is conducted on each independent n dimension along direction set $\{\mathbf{v}^{(k)}\}$. At the end of each iteration k , a new set of orthogonal search directions, $\mathbf{w}$, is generated. This process is repeated until convergence to a local optimum or the allowable computational budget has elapsed. For the details, the reader is referred to Figure B.6 which outlines the main steps of the DSCG/DSCP procedure.

B.4.1 Pseudo-code

The orthogonalization procedure of Gram-Schmidt is described as follows:

Let $\mathbf{a}_i = \sum_{j=1}^n d_j^{(k)} \mathbf{v}_j^{(k)}$, where n represents the number of dimensional variables while $d_j^{(k)}$ is the distance covered in direction $\mathbf{v}_j^{(k)}$ in the iteration k .

Next, the set of vectors $\{\mathbf{a}_i\}$ is orthogonalized using the equations:

$$\mathbf{w}_i = \begin{cases} \mathbf{a}_i & \text{if } i = 1 \\ \mathbf{a}_i - \sum_{j=1}^{i-1} (\mathbf{a}_i^T \mathbf{v}_j^{(k+1)}) \mathbf{v}_j^{(k+1)} & \text{otherwise} \end{cases}$$

Finally, the new set of search directions for the $(k+1)^{th}$ iteration is defined as:

$$\mathbf{v}_i^{(k+1)} = \frac{\mathbf{w}_i}{\|\mathbf{w}_i\|}$$

It can be easily verified that the set of vectors $\{\mathbf{w}_i\}$ are orthogonal to each other. Moreover, $\mathbf{w}_0$ denotes the direction from the starting point to final point of k^{th} iteration in the search. In this way, it is hope that the search directions in the subsequent iterations will align better with the search surface.

Appendix B. Conventional numerical search methods

Algorithm 15 DSCG

Initialization:

Starting point $\mathbf{x}^{(0,0)}$ Initial step length $s^{(0)}$ Accuracy ε First set of direction $\{\mathbf{v}_i^{(0)}\} = \{\mathbf{e}_i\}, \quad (i = 1, 2, \dots, n)$ Iteration counter $k = 0$ Direction counter $i = 1$ **loop**Run n line search along n direction vectors $\{\mathbf{v}_i^{(k)}\}, \quad (i = 1, 2, \dots, n)$:

$$F(\mathbf{x}^{(k,i)}) = \min_d F(\mathbf{x}^{(k,i-1)} + d\mathbf{v}_i^{(k)})$$

Calculate the distance vector from the first point and the last point:

$$\mathbf{z} = \mathbf{x}^{(k,n)} - \mathbf{x}^{(k,0)}$$

if ($\|\mathbf{z}\| > 0$) **then**Do one more line search along the direction of $\mathbf{z}$:

$$\mathbf{v}_{n+1}^{(k)} = \mathbf{z} / \|\mathbf{z}\|$$

$$F(\mathbf{x}^{(k,n+1)}) = \min_d F(\mathbf{x}^{(k,n)} + d\mathbf{v}_{n+1}^{(k)})$$

else

$$\mathbf{x}^{(k,n+1)} = \mathbf{x}^{(k,n)}$$

end if**if** ($\|\mathbf{x}^{(k,n+1)} - \mathbf{x}^{(k,0)}\| \geq s^{(k)}$) **then**Perform orthogonalization using Gram-Schmidt or Palmer process to get new set of direction vectors $\{\mathbf{v}_i^{(k+1)}\}$

$$s^{(k+1)} = s^{(k)}$$

$$\mathbf{x}^{(k+1,0)} = \mathbf{x}^{(k,n)}$$

$$\mathbf{x}^{(k+1,1)} = \mathbf{x}^{(k,n+1)}$$

$$k = k + 1, \quad i = 2$$

else

$$s^{(k+1)} = 0.1s^{(k)}$$

if ($s^{(k+1)} < \varepsilon$) **then**End the search, return the current best point $\mathbf{x}^{(k,n+1)}$ **else**

$$k = k + 1, \quad i = 1$$

end if**end if****end loop**

Appendix B. Conventional numerical search methods

B.4.2 Flow chart

Figure B.6: The strategy of Davies, Swann, and Campey with Gram-Schmidt orthogonalization (DSCG).

List of Related Publications

Journal

- H. Soh, Y. S. Ong, Q. C. Nguyen, **Q. H. Nguyen**, M. S. Habibullah, T. Hung, and J.-L. Kuo, Discovering Unique, Low-Energy Pure Water Isomers: Memetic Exploration, Optimization and Landscape Analysis, *IEEE Transactions on Evolutionary Computation*, 14(3):419-437, 2010.
- **Q. H. Nguyen**, Y. S. Ong, M. H. Lim and N. Krasnogor, Cellular Memetic Algorithms, *Evolutionary Computation Journal*, 17(2):231-256, 2009.
- **Q. H. Nguyen**, Y. S. Ong, and M. H. Lim, A Probabilistic Memetic Framework, *IEEE Transactions on Evolutionary Computation*, 13(3):604-623, 2009

Conference

- F. Liang, Y. S. Ong, **Q. H. Nguyen**, Probabilistic Memetic Algorithm for Capacitated Arc Routing Problem, *IEEE World Congress on Computational Intelligence*, Barcelona, Spain, 2010.
- **Q. H. Nguyen**, Y. S. Ong and M. H. Lim, Non-genetic transmission of memes by diffusion, *Genetic and Evolutionary Computation Conference*, Atlanta, Georgia, US, 12-16 July 2008, ACM Press.
- M. N. Le, Y. S. Ong and **Q. H. Nguyen**, Optinformatics for Schema Analysis of Binary Genetic Algorithms, *Genetic and Evolutionary Computation Conference*, Atlanta, Georgia, US, 12-16 July 2008, ACM Press.

Appendix B. Conventional numerical search methods

- **Q. H. Nguyen**, Y. S. Ong, and N. Krasnogor, A Study on the Design Issues of Memetic Algorithms. *IEEE Congress on Evolutionary Computation, CEC*, Singapore, September 2007.
- Y. S. Ong, **Q. H Nguyen**, M. H. Lim and T. Jing, A Development Platform for Memetic Algorithm Design, *Joint 3rd International Conference on Soft Computing and Intelligent Systems and 7th International Symposium on Advanced Intelligent Systems, SCIS&ISIS*, Japan, September 2006.