GLOBAL OPTIMIZATION
WITH HYBRID
EVOLUTIONARY COMPUTATION

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2014

By
Hassan A Bashir
School of Computer Science
Contents

Abstract 13
Declaration 14
Copyright 15
Acknowledgements 16
Nomenclature 17

1 Introduction 19
  1.1 Global Optimization – perspective .......................... 20
  1.2 Optimization Methods ........................................... 21
    1.2.1 Classical Optimization Methods ............................ 22
    1.2.2 Heuristic Optimization Methods ........................... 23
    1.2.3 Hybrid Optimization Methods ............................... 24
  1.3 Research Questions .............................................. 25
    1.3.1 Exploring new understandings for the stochastic and
deterministic optimization methods .................................. 26
    1.3.2 Investigating hybrid optimization frameworks and
their extensions ........................................................... 27
  1.4 Scope ............................................................... 27
  1.5 Contributions ...................................................... 28
  1.6 Thesis Structure .................................................. 29
  1.7 Publications and Project ......................................... 31
I  Stochastic and Deterministic Optimization Methods 33

2  Stochastic Optimization – An EC Perspective 34
   2.1 Overview ........................................... 34
   2.2 Biological plausibility of Simulated Evolution ............ 37
   2.3 Developmental Aspects and Nomenclature of EC ......... 38
   2.4 Simulated Evolution and Evolutionary Spaces .......... 42
   2.5 Parameterisation in Evolutionary Computation .......... 44
      2.5.1 A Peek into GA – Understanding How it Works .... 45
   2.6 The Building Blocks in EC ................................ 50
      2.6.1 Initialisation: Representation and Population Sizing ... 51
      2.6.2 The Selection Operator ................................. 56
      2.6.3 Selection Analytic Parameters ......................... 59
      2.6.4 Variation operators: Recombination and Mutation .... 60
      2.6.5 Replacement Strategies and Elitism .................. 63
      2.6.6 Adapting Elitism in EAs ............................... 65
   2.7 Summary ........................................... 68

3  Deterministic Optimization Method 69
   3.1 Local Optimization – Background ......................... 70
   3.2 Line Search based methods ............................... 72
   3.3 Search Directions and Step Sizes ......................... 73
      3.3.1 Evaluating Search Directions .......................... 73
      3.3.2 Evaluating Step Sizes .................................. 76
   3.4 Convergence Analysis in gradient-based methods ........ 78
   3.5 The Sequential Quadratic Programming Algorithm ........ 80
   3.6 Interior Point Method for QP Problems ................... 81
   3.7 Automatic Differentiation for exact Derivatives ........... 84
   3.8 Summary ........................................... 92

4  Convergence Analysis in EC 93
   4.1 From Diversity measure to Evolvability measure ........ 93
   4.2 Classical Convergence Assessment Methodologies ......... 94
      4.2.1 Rationale ........................................... 95
      4.2.2 Convergence Analysis – A New Perspective ........... 96
   4.3 Spatial Diversity Measure ................................. 97
4.3.1 Evaluating the Coefficient of Diversity .............................................. 97
4.3.2 Normalisation of the Coefficient of Diversity ...................................... 98
4.4 Measuring Evolvability using Price’s Theorem ........................................ 99
  4.4.1 Monitoring Evolvability using extended Price’s Equation .................... 102
  4.4.2 Analysis of Evolvability with extended Price’s Equation .................... 103
4.5 The Proposed Population Evolvability measure ....................................... 107
  4.5.1 Convergence threshold parameter .................................................. 109
4.6 Spatial Diversity vs. Evolvability measure .......................................... 110
  4.6.1 Experiment: Comparing Spatial Diversity and Evolvability Measure .......... 112
  4.6.2 Analysis and Interpretation of Results ............................................ 114
  4.6.3 Discussion .................................................................................. 117
  4.6.4 Combining Spatial Diversity and Population Evolvability Measures .......... 117
  4.6.5 Contribution and Predicted Impact ................................................. 118
4.7 Summary ....................................................................................... 119

5 Diversity Control in EC ........................................................................... 120
  5.1 Diversity in Evolutionary Computation ................................................. 121
  5.2 Diversity in the light of Exploration and Exploitation ............................ 121
  5.3 Taxonomy of Diversity Control Policies in EC ....................................... 123
    5.3.1 Heuristic Population Initialisation ............................................... 124
    5.3.2 Multipopulation Strategies ......................................................... 126
    5.3.3 Hybrids and Portfolios of Algorithms ........................................... 128
  5.4 A Dual-Pool EC Model for Diversity Control ........................................ 129
    5.4.1 Rationale ................................................................................. 129
    5.4.2 The Dual-Pool EC Architecture .................................................. 130
    5.4.3 The Evolution Pool – Creation and Working .................................... 131
    5.4.4 The Diversity Pool – Creation and Working .................................... 131
    5.4.5 Search Space Partitioning (SSP) Heuristic Initialisation .................... 132
    5.4.6 The Proposed Dual-Pool EC Model ............................................. 135
  5.5 Visualising Diversity in EC Models ....................................................... 137
    5.5.1 Diversity Visualisation with a Standard EC Model .......................... 137
    5.5.2 Diversity Visualisation with a Dual-Pool EC Model ......................... 138
  5.6 Evaluation of the Dual-Pool EC model .................................................. 140
    5.6.1 Benchmark Test Cases – Features and Significance ......................... 140
7.4.1 The Dynamic Rotation Peak (DRP) Benchmark 189
7.4.2 Evaluation of Dual-Pool HESA on the DRP Benchmark 190
7.5 Results – Analysis and Interpretation 191
  7.5.1 Pool size Sensitivity Analysis 192
  7.5.2 Relative Performances and Convergence Efficiency 194
  7.5.3 Absolute Error Analysis 197
7.6 Parameter Adaptation for the hybrid EA 198
  7.6.1 Adapting the Mutation operator 199
  7.6.2 Adapting the Recombination operator 203
7.7 Comparison of the Adaptive and Non-Adaptive HESA 208
  7.7.1 Experimental Goals 208
  7.7.2 Analysis and Interpretation of Adaptation effect 211
7.8 Theory Research Relevance Tree for HESA Model 214
7.9 Contribution and Remarks 214

8 Investigating Extended Hybrid EAs 216
  8.1 SCA Derivative-Free Algorithm 217
  8.2 Samples Generation for the SCA Algorithm 219
  8.3 The Proposed Extended HESA model 220
  8.4 Evaluating the Extended HESA Algorithm 221
    8.4.1 Comparison of the HESA and ext-HESA models 223
    8.4.2 Comparing the HESA models with Dynamic Optimization Competition Algorithms 225
  8.5 A Generalised Extended Hybrid Framework 230
  8.6 Contributions 234
  8.7 Conclusion 235

9 Conclusion 236
  9.1 Development of Ideas 236
  9.2 Summary 237
  9.3 What insights are learned from this thesis? 239
  9.4 Limitations of this study 243
  9.5 Future Research Directions 244

A Vectorised Automatic Differentiation 248
  A.1 Overloaded Operators & functions for AD Objects 248
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.2 Symbolic vs. vectorised Forward-mode AD</td>
<td>251</td>
</tr>
<tr>
<td>B Benchmark Test Case Studies</td>
<td>254</td>
</tr>
<tr>
<td>C Result Summary</td>
<td>255</td>
</tr>
<tr>
<td>D Evaluation Results for the Dual-Pool HESA</td>
<td>257</td>
</tr>
<tr>
<td>E Evaluation Results for the Adaptive HESA Model</td>
<td>259</td>
</tr>
</tbody>
</table>

Word Count: 60021
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Evolutionary Nomenclature</td>
<td>40</td>
</tr>
<tr>
<td>2.2</td>
<td>Standard Parameters of a Typical Genetic Algorithm</td>
<td>45</td>
</tr>
<tr>
<td>2.3</td>
<td>Initialisation and Evaluation of a hypothetical GA Sample Pool</td>
<td>47</td>
</tr>
<tr>
<td>2.4</td>
<td>Fitness proportionate selection: A demonstration of Roulette Wheel</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>fitness proportionate Selection for the Hypothetical GA</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>Tournament selection</td>
<td>49</td>
</tr>
<tr>
<td>2.6</td>
<td>Crossover and Mutation</td>
<td>50</td>
</tr>
<tr>
<td>2.7</td>
<td>A summary of commonly used representation methods in evolutionary</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>algorithms</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Comparison of the three major gradient based algorithms</td>
<td>79</td>
</tr>
<tr>
<td>3.2</td>
<td>A list of variables definitions for a 2-dimensional sample problem</td>
<td>85</td>
</tr>
<tr>
<td>3.3</td>
<td>A list of values and derivatives for the forward mode AD on a</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>sample problem</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Tabulation of the proposed vectorised forward mode AD on a sample</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>problem</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Parameter settings of the Binary Encoded EC model for convergence</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>analysis</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Parameter settings of the real-valued EC model for convergence</td>
<td>112</td>
</tr>
<tr>
<td></td>
<td>analysis</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Parameter Settings for the Dual-Pool EC Model</td>
<td>141</td>
</tr>
<tr>
<td>5.2</td>
<td>Statistical Tests: Parametric and nonparametric tests for the results on comparing the Dual-Pool and Standard EC models</td>
<td>145</td>
</tr>
<tr>
<td>5.3</td>
<td>Comparing the average optimization errors of the Dual-pool EC (DP-EC) algorithm and the TPC-GA algorithm on the CEC2013 Benchmarks</td>
<td>151</td>
</tr>
</tbody>
</table>
6.1 A survey of hybrid optimization algorithms in various applications domain ........................................ 161
6.2 Parameter settings of the proposed hybrid algorithm ........ 164
6.3 Optimization results for the hybrid EC/SQP Algorithm on the CEC2013 Unimodal and Multimodal Benchmarks .......... 173

7.1 Properties of the Dynamic Rotation Peaks Benchmark .... 190
7.2 Parameters of Dual-pool HESA model for Dynamic Experiments ... 191
7.3 Evaluation Parameters for the DRP Benchmark ........ 192
7.4 Absolute Error Analysis ........................................ 193
7.5 Relative Performance Analysis ........................................ 194

8.1 Comparison of the CEC2009 Dynamic Optimization Competition Algorithms with the HESA algorithms .......... 226

B.1 Global Benchmark (Basic) Functions ......................... 254

C.1 Computational cost required by the Dual-pool and Standard EC algorithms to converge to the solutions of six benchmark problems 256

D.1 Offline/online performances of the Dual-pool HESA model under varying pool sizes ........................................ 257
D.2 Absolute error of the Dual-pool HESA model under varying pool sizes ........................................ 258

E.1 Result summary: Evaluation of the Adaptive HESA model on DRP benchmarks ........................................ 259
# List of Figures

1.1 An illustration of local and global optimization landscapes 22
1.2 Retardation of fitness growth over successive generations in EC 25

2.1 Classification of optimization methodologies 36
2.2 Flowchart of a Typical Genetic Algorithm 39
2.3 Simulation of Genotype-Phenotype Mapping Functions 43
2.4 An illustration of sample of ice-cream cornets of various dimensions 47
2.5 Minimal Population sizes $N$ for 99.9% confidence interval of having all allele in the initial population of GA 56

3.1 Algorithmic evaluation graph for a 2-dimensional sample problem 87
3.2 An illustration of the proposed vectorised forward AD Design 89

4.1 Fitness comparison plots and the effect of genetic operators for optimizing a 2-dimensional Schwefel benchmark 105
4.2 Fitness comparison plots and the effect of genetic operators for optimizing a 2-dimensional Schwefel benchmark 109
4.3 Convergence detection by monitoring the $\pm \sigma$ interval of the contribution of crossover operator 111
4.4 Analysis of convergence measures on the smooth unimodal benchmark $f_0$ 115
4.5 Analysis of convergence measures on the Needle in Haystack (NiH) benchmark $f_3$ 116

5.1 A Research Relevance Tree for the Fundamental Diversity Control Approaches used in Evolutionary Computation 123
5.2 Theory Research Relevance Tree for Dual-Pool EC model 130
5.3 Comparing a typical Random initialisation with the SSP quasi-random heuristic 134
5.4 The Dual Pool EC model dynamically showing the merger of the evolution pool with the diversity pool.  
5.5 An illustration of the typical Spatial Diversity Dynamics for the standard EC model.  
5.6 An illustration of the typical Spatial Diversity Dynamics for the Dual-Pool EC model.  
5.7 Performance Comparison of the Dual-Pool EC with the Standard EC Algorithm on six global optimization benchmarks.  
5.8 Performance comparison summary for the standard EC and dual-pool EC models on the basis of functions evaluations.  

6.1 Crucial Considerations for Building Hybrid Models.  
6.2 Hybrid EC/SQP hybrid algorithm: Scalability test.  
6.3 Hybrid EC/SQP hybrid algorithm: Fitness comparison test.  
6.4 Comparison of the optimization errors of the EC/SQP algorithm with that of the CEC2013 Algorithms on Unimodal Benchmarks (F1 to F5).  
6.5 Comparison of the optimization errors of the EC/SQP algorithm with that of the CEC2013 Algorithms on Multimodal Benchmarks (F1 to F5).  

7.1 A framework for continuous optimization using the Hybrid Dual-pool EC/SQP Algorithm (HESA).  
7.2 A 3-D view of a 2-dimensional Dynamic rotation peaks (DRP) benchmark.  
7.3 Offline vs. Online Performances sensitivity analysis for HESA model.  
7.4 Offline vs. Online Performances of HESA model at Pool size of 50.  
7.5 Convergence characteristics of the HESA model for various dynamic changes.  
7.6 Absolute Error Analysis for the Dual-pool HESA model.  
7.7 Parameterisation Methodologies in Evolutionary Computation.  
7.8 A closed-loop adaptive mutation operator.  
7.9 Offspring Hypercube for Recombination operator.  
7.10 Offspring Hypercube for the proposed Recombination operator.  
7.12 Absolute Error Analysis: Adaptive vs. Non-adaptive HESA models
7.13 Convergence characteristics comparison: Adaptive vs. Non-adaptive HESA models
7.14 Theory Research Relevance Tree for the Dual-Pool HESA model
8.1 Search neighbourhood for the SCA algorithm
8.2 A framework for continuous optimization using the extended hybrid Algorithm
8.3 Relative performance comparison: ext-HESA vs. HESA Algorithms
8.4 Absolute Error Analysis: ext-HESA vs. HESA Algorithms
8.5 Convergence characteristics comparison: ext-HESA vs. HESA Algorithms
8.6 CEC2009 Dynamic Optimization Comparison Summary
8.7 Performance comparison: HESA vs. CEC:DSA Algorithm
8.8 Generalised Extended Hybrid Framework
8.9 Controller rules for the Generalised Extended Hybrid Framework
Abstract

GLOBAL OPTIMIZATION WITH HYBRID EVOLUTIONARY COMPUTATION

Hassan A Bashir

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy, 2014

An investigation has been made into hybrid systems which include stochastic and deterministic optimization. This thesis aims to provide new and relevant insights into the design of the nature-inspired hybrid optimization paradigms. It combines evolutionary and gradient-based methods. These hybrid evolutionary methods yield improved performance when applied to complex global optimization tasks and recent research has shown many of such hybridization policies.

The thesis has three broad contributions. Firstly, by examination of stochastic optimization, supported by case studies, we utilised the Price’s theorem to formulate a new population evolvability measure which assesses the dynamical characteristics of evolutionary operators. This leads to the development of a new convergence assessment method. A novel diversity control mechanism that uses heuristic initialisation and convergence detection mechanism is then proposed. Empirical support is provided to explicitly analyse the benefits of effective diversity control for continuous optimization.

Secondly, this study utilised research relevance trees to evolve hybrid systems which combine various evolutionary computation (EC) models with the sequential quadratic programming (SQP) algorithm in a collaborative manner. We reviewed the convergence characteristics of various numerical optimization methods, and the concept of automatic differentiation is applied to design a vectorised forward derivative accumulation technique; this enables provision of accurate derivatives to the SQP algorithm. The SQP serves as a local optimizer in the deterministic phase of the hybrid models. Through benchmarking on stationary and dynamic problems, results showed that the proposed models achieved sufficient diversity control, which suggests improved exploration-exploitation balance.

Thirdly, to mitigate the challenges of ‘inappropriate’ parameter settings, this thesis proposes closed-loop adaptive mechanisms which dynamically evolve effective step sizes for the evolutionary operators. It then examines the effect of incorporating a derivative-free algorithm which extends the hybrid model to a flexible and reusable algorithmic framework.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

First I would like to thank my supervisor Richard Neville, for his guidance and mentoring in many areas of research and teaching. I would not forget his helpful and generous advice which helped keep me focused while maintaining quality health and fitness during this programme; I will cherish this in my continuing academic career and beyond.

Thanks are due to everyone in the MLO research Lab. Special thanks to Darren, Richard M, and Joe for their friendship and the out-of-hours games, all of which are vital for maintaining an enjoyable and collaborative learning environment. Many thanks to Richard A. for giving me valuable feedback during early days of this research. Particular thanks to Geoffrey N. for taking the time to correspond with me during the final compilation of this thesis. I also wish to extend my gratitude to all the Staff in the corridor for their academic and administrative support.

My unreserved thanks go to my Parent, my entire brothers and sisters and everyone at home whom I dearly missed while embarking on my studies abroad, I would not have achieved any of this without your support and encouragement; warm thanks to my twin, Hussaini, for being always there in my life. I also would like to thank everyone who I have lived with here in Manchester, or met along the way, and all those who have assisted me, in anyway, during my studies over these years.

Finally, my sincere gratitude goes to the Petroleum Technology Development Fund (PTDF) of the Federal Republic of Nigeria for their financial support; their official backing has made my living away from home a memorable and enjoyable experience.
Nomenclature

- $t$: Iteration or generation counter
- $X, X^*$: An arbitrary sample solution, also called individual, such that $X \in \mathbb{R}^n$, and the optimal solution point
- $x_i, x^*_i, \bar{x}_i, \underline{x}_i$: A given variable for a sample solution $X$, a variable for the optimal solution point, lower and upper bounds for variable $x_i$
- $x_i'$: Chromosome representation of sample solution $x_i$, e.g. in binary bits
- $P, P_0, P_t$: A set of candidate solution, i.e., evolutionary pool, the initial population, and the population at generation $t$
- $N, n$: Population size, i.e., the cardinality of $P$, which is $N = |P|$, and the problem size or dimensionality
- $\mathcal{D}, \mathcal{D}$: Search domain, i.e., the feasible boundaries for the search, and a search neighbourhood within the search domain, such that $\mathcal{D} \subset \mathcal{D}$
- $f, f^*, \bar{f}$: Fitness, i.e., a quality of candidate solution that translates to chances of survival during selection, the Fitness of optimal solution $x^*$, and the average fitness
- $f()$: Objective function
- $g()$: Constraints (or bound) set
- $\mathcal{M}, P_M$: Mutation operation, and Probability (rate) of mutation
- $\mathcal{X}, P_X$: Crossover operation, and Probability (rate) of crossover
- $P_{Elite}, N_{Elite}$: Elite population and the size of the Elite pool
- $\alpha, \alpha_t$: Step length parameter, and its value at iteration $t$
- $d, d_t$: Search direction vector, and its value at iteration $t$
- $H$: Hessian matrix $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$
- $\varphi(), q(), \mathcal{L}()$: Merit function, Quadratic problem, and Lagrange function
- $C_{Div}, \tilde{C}_{Div}$: A vector of the coefficient of diversity, and its normalised value
\( \Delta Q, \Delta q_i \) \hspace{5mm} The change in average fitness \( Q_i \) across successive generations, and the change between parent’s average fitness \( q_i \) and that of its offspring \( q'_i \)

\( \sigma_{xover} \) \hspace{5mm} Crossover envelope, also called population or crossover evolvability which is twice the standard deviation interval \((2\sigma)\) of the crossover’s contribution to the fitness growth

\( \alpha^c, \mu^M \) \hspace{5mm} Recombination or crossover weighting parameter, and Mutation effective precision parameter; both these parameters useful for operator adaptation in EC model having real-valued data structure.

\( P_{Evo}, P_{Div} \) \hspace{5mm} Evolution pool and Diversity pool

\( \kappa, \rho(\kappa), \phi_{\kappa} \) \hspace{5mm} Number of search space partitions, Vector of partition sizes, and the subspaces (or partitions)

\( E^{last} \) \hspace{5mm} Absolute error in the final solution obtained just before a dynamic change in problem’s landscape.

\( x_{EC}, x_{SQP} \) \hspace{5mm} Optimal solution returned by the EC algorithm, and that returned by the SQP algorithm

\( s, s_f \) \hspace{5mm} Sampling period, and sampling frequency

\( r^i(t), r^s(t) \) \hspace{5mm} Relative performance at iteration \( t \), and its value at sampling period \( s \)

\( r^{offline}, r^{online} \) \hspace{5mm} Offline and online relative performance measures

\( T_i \) \hspace{5mm} Dynamic change types \( T_i = T_1, T_2, \ldots, T_6 \)

\( \delta^i_R \) \hspace{5mm} Radius of search neighbourhood along dimension \( i \)

\( X_D, f(X_D) \) \hspace{5mm} A set of sample solutions within a given neighbourhood \( D \), and a set of their respective fitness values

\( \delta_{cost} \) \hspace{5mm} Derivative cost ratio defined as the ratio of the cost of function evaluation to that of evaluating its derivatives, \( \delta_{cost} \in [0, 1] \)
Chapter 1

Introduction

The growing human aspirations for improved livelihood mean that whenever we are presented with diverse and numerous (sometimes uncountable) alternatives, we generally opt for the best possible one. Numerous situations arise naturally and require us to seek the best possible option that would maximise our gains within the confines of our reaches. For instance, the daily morning commute to workplace or school is a classical example of a task that requires seeking the best (safest, shortest, cheapest, most suitable, etc.) route. Likewise, a manufacturer seeking to boost production rate invariably opts for the most cost effective production strategy within the available production resources. In essence, our daily aspirations result in an increasing desire to maximise gain, profit, quality, etc. and/or minimise loss, cost, energy, time, or utility, etc.

Technically, the process of selecting the best out of the many possible decisions (options) is called Optimization; the best possible decision (or option) is called the optimal solution; and the optimal solution task is called an optimization problem.

The technologies available today and the recent growth in the development of new algorithms and modelling techniques mean that optimization can apply to various disciplines. In fact, the applications of optimization have gone beyond the confines of applied and computational mathematics to all areas of engineering, life sciences, finance and other sciences. Optimization, as a tool, received wide acceptance from both industry and academia. Perhaps, it is this multidisciplinary nature of the field, coupled with its wealth and diverse applicability, that made some early pioneers believe that “Optimization is a cornerstone for the development of modern civilisation” (Sun and Yuan, 2006).” Hence, there continue to be increasing need for thorough understanding of optimization problems and their
solution methodologies.

This thesis aims to investigate global optimization from the perspectives of hybrid evolutionary computation. This chapter presents a top-level overview of optimization (i.e., the problem domain) in Section 1.1. It then discusses the various solution methods in Section 1.2. Section 1.3 of this chapter presents the motivations, in form of research questions, for studying this domain. Section 1.4 outlines the scope of this study and Section 1.5 summarises the contributions of this research. Finally, Section 1.6 delineates the organisation of this thesis.

1.1 Global Optimization – perspective

Optimization methods search for optimal solutions to problems in continuous, discrete (or combinatorial), and/or multiobjective domains. This study focuses on nonlinear continuous optimization problems that are generally global in nature. This is because there are many application problems – from various domains – that are modelled as continuous optimization tasks, examples include:

- *Industrial control systems*: Controlling a chemical process or an (electro)mechanical device, such as a robot arm, to meet certain performance requirements, like robustness and/or efficiency;

- *Finance*: Optimizing the (often statistical) models that predict market fluctuations (sales forecast), or in designing an investment portfolio to maximise expected return within some acceptable levels of risk;

- *Seeking optimal path*: Finding an optimal trajectory for an aircraft or a robot arm;

- *Aerodynamic design*: Determining the optimal shape of an automobile component in a manufacturing system, this is especially critical in aircraft and sport utility equipment; and

- *Task scheduling*: Planning of operations in manufacturing plants to maximise production level within the limited available resources which often requires meeting quality standards while satisfying customer demands.

Notice that, common to all these optimization tasks are the following three important aspects:
1. **Objective**: This is a measure that assesses the extent to which the optimization goal is realised, and it is typically modelled as a mathematical (cost) function;

2. **Design variables**: These form the set of all possible choices that must be made to ensure successful realisation of the overall objective. These implicit choices are technically referred to as decision or design variables and are the parameters around which the optimization task can be formulated; and

3. **Constraints**: These are the requirements within which the optimization objective is limited. It can be a limitation due to resource, time or space and or acceptable error levels or tolerance. Typically, the domain for each design variable is bounded so as to explicitly define what constitute feasible solutions.

Generally, optimization problems involve multiple, often conflicting, design requirements. As a result, the search for an optimum solution is challenging. Figure 1.1 illustrates typical views of the solution spaces in local and global optimization problems. For the local optimization problems (Figure 1.1a), only one optimal solution exists, but such problems are mostly idealistic. The global problems (Figure 1.1b), on the other hand, involve a combination of many local and global optimal solutions. In fact, the majority of optimization tasks are of the global type. Regardless, optimization methods are expected to find the global optimum solution within minimum possible computational cost.

### 1.2 Optimization Methods

Several solution techniques exist for the different types of the aforementioned optimization problems. These techniques can be broadly classified into two main categories: classical (analytic) and heuristic methods. As will be described in the following sections, the classical methods are usually numerical and deterministic\(^1\) in nature, whereas the heuristic methods are generally population-based stochastic\(^2\) strategies.

---

\( ^1\)A search or optimization algorithm is said to be *deterministic* if it consistently follows the same execution path (and returns the same solution) when it is repeatedly run from the same starting solution.

\( ^2\)Stochastic methods are the opposite of deterministic methods and are therefore probabilistic (random); they may not follow the same execution path (or return the same solution) when
CHAPTER 1. INTRODUCTION

1.2.1 Classical Optimization Methods

Classical solution approaches have their origins in linear programming methods invented by Dantzig (1963) during the late 1940s. Subsequently, further numerical algorithms, including gradient-based methods, conjugate gradient methods, Newton and quasi-Newton methods, are applied to solve a variety of large scale numerical optimization problems. Classical methods are also classified as exact or complete methods.

Due to their deterministic nature, classical methods are most efficient in locating local optima; however, their designs generally lack heuristics to prioritise some solutions over others. Therefore, they are inclined to converge to the optimum within (or nearest to) the neighbourhood of their initialisation region. In Chapter 3, this thesis provides an in-depth analysis of the convergence dynamics of such gradient-based methods. Chapter 3 also presents a technique (developed during this research) that speed-up the process of evaluating derivatives for these methods.

repeatedly run from the same starting solution.
1.2.2 Heuristic Optimization Methods

Recognising the inherent limitations of classical methods, researchers proposed a variety of methods which aim to solve global optimization by using various heuristics\(^3\). These algorithms are also called approximate algorithms due to their stochastic (probabilistic) construction. These approximate methods are usually easier to implement than their exact counterparts, and although they are stochastic, they differ from pure random searches since their randomness is guided through intelligent and informative decisions.

Heuristic methods are mostly based on analogies with natural processes (Zhigljavsky and Žilinskas, 2008) and can be successfully applied to a wide range of optimization problems with little or no modifications in order to adapt to any specific problem. Heuristics methods based on the concept of natural selection (i.e. the principles of evolutionary genetics) have come to be identified as evolutionary computation (EC). Prominent examples are genetic algorithms (GA), genetic programming (GP), evolutionary programming (EP) and evolutionary strategy (ES) (Fogel, 1994). Other stochastic heuristics include simulated annealing, tabu search, ant colony optimization, artificial immune system, scatter search, estimation of distribution algorithms, multi-start and iterated local search algorithms. All these are typical examples of metaheuristics that fall under the category of approximate algorithms.

Among the many heuristic methods, GAs are probably the most widely used types of evolutionary algorithms (EAs) (Goldberg, 1989). GAs are originally inspired by Holland (1975, 1992) and they simulate the processes of natural evolution. The inspirational basis for such nature inspired methods stems from the fact that it is not unreasonable to assume that natural mechanisms will have done a pretty good job in optimizing the way living things interact and adapt to their environment. It is therefore reasonable to look into natural systems and see how they succeed in such long term adaptation through evolution.

Genetic algorithms may require a large amount of computation due to their population-based design, but they have desirable characteristics. For example, GAs can discard a local optimal solution in order to evolve a better sub-optimal

---

\(^3\)Meta(heuristics) is a term used to delineate a universal algorithmic framework designed to solve optimization problems based on probabilistic decisions made during the search process, see Glover (1986).
solution. Generally, GAs have higher potentials of converging to optimum solution(s) especially for complex global optimization tasks (Gen and Cheng, 1997). As is discussed in Chapter 2 genetic algorithms are non-problem dependent, non-derivative and quite flexible methods.

Despite the inherent flexibilities and robustness of EAs, their success has constantly been hindered by the eventual retardation in search progress towards the end of the evolutionary cycle. By means of the probabilistic evolutionary processes, EAs significantly improve the average fitness (i.e., quality) of the search pool during the early stages of the evolution, but this inevitably fades, sometimes drastically, as the average fitness of the search pool grows. The illustration in Figure 1.2 demonstrates the typical fitness characteristics of the standard genetic algorithm on solving a 2-dimensional global optimization problem.

Assuming a maximisation problem, Figure 1.2 shows that the gradient ($\Delta$) of the average fitness curve is quite steep (having $n$-fold fitness increase) at the initial stage, but this almost monotonically shrinks during the course of the evolution ($\Delta_1 > \Delta_2 > \cdots > \Delta_n$). This eventual collapse in speed as the search nears convergence is a major limitation to successful use of EAs in online and safety critical applications.

### 1.2.3 Hybrid Optimization Methods

It has been realised for some time that the effort to solve majority of large scale global optimization tasks has continued to face huge challenges, this is regardless of whether an exact or heuristic (approximate) method is used. As a result, there is a growing interest in the concepts that combine various algorithmic paradigms from different branches of the field. The act of combining various algorithms has now been popularised as “hybrid algorithms” or “hybrid metaheuristics”. A skilful hybridization of algorithms is believed to provide a more flexible and efficient solution method that is suitable for large scale real-world problems (Blum and Roli, 2008). Hence, as argued by Raidl (2006) and Blum et al. (2010) amongst others, the hybrid framework motivates the design of robust systems that simultaneously harness the benefits of the individual algorithms and discard their inherent weaknesses.

A central objective of this study is to propose an efficient and robust optimization method on the hybrid framework. Thus, although the chapters in Part
CHAPTER 1. INTRODUCTION

Figure 1.2: Retardation of fitness growth over successive generations in a typical evolutionary optimization model. The curve shows the characteristic growth in the average population’s fitness due to objective function $f(x)$ over successive generations.

I of this thesis mainly focus on understanding and analysing the individual algorithms, the investigations in Part II specifically address the hybridization aspect from the domains of stationary and non-stationary problems.

1.3 Research Questions

Global optimization methods have found applications in many domains (Werbos, 2011), and the literature contains a great deal of work on investigating the various optimization problems and their solution methods. However, in a wide scope, this study is motivated by the growing need to gain further insights into the underlying processes of the various optimization methods and to improve their applicability on various global optimization tasks. In particular, this thesis attempts to address the following research questions denoted by $Q_n$ which span the classical, heuristic and hybrid optimization domains.
1.3.1 Exploring new understandings for the stochastic and deterministic optimization methods

$Q_1$: From the classical (deterministic) optimization side, the commonly used gradient-based methods are by design tailored to solve local optimization tasks. Since this study aims to hybridize such algorithm with a stochastic method to facilitate global optimization, the first question here is:

“Could the convergence rate of the gradient-based algorithm be improved in the presence of accurate and inexpensive derivatives? What impact would that have on the global convergence of the overall hybrid system?”

$Q_2$: On the stochastic (evolutionary) optimization side, in the 1970s, Price (1972) has formulated an interesting theorem that allows investigating the individual characteristics of the evolutionary operators. But this has so far only been used to analyse the effect of the evolutionary operators on the fitness growth during the search. Thus, the question here is, by using Price’s theorem:

“Could the dynamical characteristics of evolutionary operators yield new insights into the convergence status of the evolutionary pool?”

Exploring this leads to the next question:

“Could such information be exploited to facilitate robust convergence analysis in EC?”

$Q_3$: In view of the crucial role (i.e., ensuring evolutionary progress) the presence of diversity plays in an evolutionary search, this study examines the standard EC framework to envisage how population diversity fares during evolution. The question that triggers this is:

“Could better understanding of convergence and diversity relationships (that is, evolvability) provide useful insights for improving the present convergence detection methods?”

Having achieved this, another question of interest is:

“Could a diversity control technique that utilises convergence detection facilitate continuous evolution in hybrid EAs?”
1.3.2 Investigating hybrid frameworks and their extensions

$Q_4$: There is a rising interest in the research for effective hybrid systems. This sparks some obvious questions:

“Upon what heuristics or assumptions are the current hybrid systems built and why?”

But, rather more targeted question is:

“Could real-time convergence information be used as underlying control mechanism to design effective collaborative hybrid systems?”

$Q_5$: A number of parameter control/adaptation mechanisms have been reported in the literature and are essentially meant to minimise the need for manual parameter tuning (Eiben et al., 1999); in addition, such adaptive methods also improve robustness but occasionally has a slight cost in efficiency. Thus, this thesis investigates:

“How would a closed-loop parameter adaptation strategy that monitors the instantaneous state of a search pool affect the performance of a hybrid EA? Besides minimising manual parameter tuning, what impact would this have on the robustness and efficiency of the hybrid EA?”

$Q_6$: Because of the robust nature of evolutionary optimization systems, EAs are becoming popular tools for optimization of dynamic problems (Li et al., 2008). Having analysed the performance of hybrid EAs on the stationary global optimization tasks, it is of interest to examine:

“How do the proposed hybrid algorithms (both the collaborative and its extended version) compare to the state-of-the-art EAs which are specifically built for dynamic problems?”

1.4 Scope

From the problem domain perspective, the scope of this study was to address global (linear/nonlinear) optimization tasks in the real-parameter (continuous)
domain. Thus, all the analyses herein involve such optimization problems modelled as continuous optimization problems. For the investigations however, the strategy adopted in this study was largely experimental and analytical. Therefore, in order to validate the hypotheses (research questions), it was necessary to design and evolve data structures and algorithms in this work; also, to validate the resultant hybrids, a collection of IEEE global optimization benchmarks that are mathematically formulated as continuous (differentiable with at least second derivatives available) functions is used.

1.5 Contributions

This study focuses on gaining new understandings for the stochastic and deterministic optimization methods so as to design effective hybrids utilising the two domains. The main contributions are in the development of new convergence and diversity assessment criteria (Part I), and then applying them to design collaborative hybrid systems (Part II) – though throughout the thesis many other contributions are presented. This section summarises the research contributions, but specific details are given at the end of every chapter, and a more detailed account is devoted to the Conclusion chapter (Chapter 9).

Following the research undertaken on hybrid evolutionary optimization during this study, in respect to the abovementioned research questions, this thesis makes the following contributions:

1) **Population Evolvability:** An identification of new role for the dynamical characteristics of evolutionary operators (Chapter 4); this leads to the proposal of a novel *population evolvability measure* (using Price’s theorem) that facilitates convergence detection.

2) **Diversity Control Mechanism:** An empirical analysis of the dynamics of population diversity (Chapter 5) during evolution, demonstrating how to sustain continuous optimization using a novel *diversity control mechanism*, which couples dual-populations and heuristic initialisation through the robust convergence measure proposed in Chapter 4.

3) **Collaborative Hybrid Framework:** A proposal for a novel collaborative hybrid framework (Chapter 6), which combines the global and local algorithms, this exploits the convergence information (Chapter 4) of the global algorithm to
dynamically opt for local refinement of the current best solutions. This model is then extended into a framework for continuous optimization – making it suitable for optimization in both stationary and non-stationary environments (Chapter 7).

4) **Closed-loop Parameter Adaptation**: This thesis proposes a *closed-loop parameter adaptation* mechanism for the hybrid EC models. We examined the impact of such adaptation method on evolutionary progress with regard to robustness and convergence efficiency (Section 7.6).

5) **Extended Hybrid System**: An extension of the proposed hybrid framework to include two direct methods (combining a derivative-based and derivative-free local search methods) and benchmarking its performance against several state-of-the-art EAs on IEEE dynamic optimization test-beds (Chapter 8).

But aside from that, on surveying the vast EC literature, the thesis evolves the state-of-the-art documentation by identifying an ongoing debate on the connection between simulated and biological evolution (Chapter 2). In particular, on whether to include more aspects of biological evolution (such as, *novelty search*) into the simulated evolution. While the benefit (if any) of prioritising novelty over fitness quality is still debatable in this field, we argue that more focus should be on improving optimization capabilities rather than trying to replicate the exact biological evolutionary phenomenon.

### 1.6 Thesis Structure

This thesis is partitioned into two parts. Part I (Chapters 2-5) overviews and investigates the two main optimization domains, namely stochastic and deterministic optimization. Then, Part II (Chapters 6-8) concentrates on hybrid modelling of these models for effective optimization of both stationary and non-stationary problems. Across the various chapters of this thesis an assortment of overviews – in form of *taxonomies* and what we called *theory research relevance trees* (TRRT) – is used to review the several aspects of evolutionary optimization, see the introduction to Chapter 2 and Sections 5.3, 6.2, and 7.3.
PART I:

Chapter 2 is of an introductory nature, it classifies the field of optimization from both the stochastic and deterministic perspectives. Chapter 2 lays an extensive background on stochastic optimization from the viewpoint of evolutionary computation; it highlights the impact of parameterisation in evolutionary optimization with examples from both theoretical and practical viewpoints. The chapter reports widely on the variety of elitist replacement strategies.

Chapter 3 introduces the deterministic optimization methodologies. It analyses the search directions, step sizes and convergence rates of the various gradient-based local search algorithms. Chapter 3 investigates the role of accurate derivatives to a gradient-based local search algorithm. It then presents an algorithmic approach to derivative evaluation with a view to provide a cost effective way to evaluating exact derivatives for continuous (smooth) problems.

Chapter 4 is experimentally based, it investigates the contributions of evolutionary operators in isolation; it then demonstrates how to utilise the effect of individual operators to analyse and assess convergence of evolutionary pool in real-time.

Chapter 5 analyses the various diversity measures used in evolutionary computation (EC). It then demonstrates how to achieve effective diversity control by coupling dual search pools through a robust convergence detection mechanism. The chapter introduces a framework suitable for continuous optimization.

PART II:

Chapter 6 overviews the current state of hybridization with EAs and provides a taxonomy of the trend in the hybrid optimization. To boosts overall search efficiency without compromising global convergence characteristics, Chapter 6 utilises the robust convergence detection mechanism (earlier proposed in Chapter 4) to propose a collaborative hybrid of global and local search methods.

Chapter 7 adopts the diversity control mechanisms proposed in Chapter 5 (i.e., dual-populations with heuristic initialisation) and extends the hybrid model into a framework for continuous optimization. Then, Chapter 7 evaluates the new hybrid frameworks on global optimization benchmarks in dynamic environments. Chapter 7 also developed a closed-loop adaptive parameterisation method for the mutation and recombination operators in EC. To analyse the effect of adaptation
on hybrid evolutionary models, Chapter 7 empirically compares the new adaptive hybrid model with its non-adaptive counterpart on the same set of dynamic optimization benchmarks.

Chapter 8 analyses the effect of incorporating a derivative-free local search method into the hybrid model. The chapter proposes an extended hybrid system built on a flexible algorithmic framework. This extended approach allows hybridizing the global (EC) with a diverse set of local refinement methods for robust and effective optimization across various domains.

Chapter 9 summaries this thesis. It provides extensive remarks on the contributions of this research vis-à-vis the aforementioned research questions. The chapter concludes the study by suggesting a number of future research avenues from theoretic, design, and implementation viewpoints.

1.7 Publications and Project

This thesis is partly from a collection of papers and project conducted during this research. This section details the publications, articles in preparation, and some previous publications related to this study.

Conference Proceedings


In preparation (Journal)


Selected previous publications

Some other previously published works (with other collaborators), which have contents related to this research.


The Evolution in Action (EiA) Project

To facilitate thorough understanding of EC processes for the experimentations in this study, the evolution in action (EiA) tool has been developed. The design of EiA combines real-time data (extracted during the modelling and simulation) into a real-time video of the evolutionary optimization process. The project was originally aimed at providing a visualisation that could enhance our understanding of the underlying principles governing the design of efficient and robust evolutionary optimization model. It is, however, now a useful tool that could facilitate teaching and research on optimization using evolutionary computation methodologies in general. In fact, EiA turns out to yield invaluable insights which stimulate many of the various proposals across the different parts of this thesis.
Part I

Stochastic and Deterministic Optimization Methods
Chapter 2

Stochastic Optimization – An Evolutionary Perspective

This chapter introduces stochastic optimization and presents a general classification for the field of optimization in Section 2.1. The chapter also reviews simulated evolution, its developmental aspects and the notion of evolutionary spaces in Sections 2.2 to 2.4. Section 2.5 overviews parameterisation in evolutionary computation (EC) and examines a practical example of an evolutionary process. The chapter further reviews the theoretical building blocks of evolutionary optimization in Section 2.6. Section 2.6.6 reports widely on the replacement strategy based on elitism principle; it presents an adaptive elitist replacement scheme which is underpinned in the parameterisation of the evolutionary models proposed in the later part of this thesis. Finally, the chapter concludes with a summary in Section 2.7.

2.1 Overview

In a typical real-world scenario, optimization tasks mostly involve large, rough and/or sparse search spaces. In most cases, it would be inconceivable to accommodate the computational requirements of the traditional operations research or mathematical programming approaches to exhaustively search for an optimal solution. Thus, researchers shift from applying deterministic techniques on “hard” problems to using approximate search methodologies that seek for a “good enough” solution in a reasonable computational time.

The classification tree in Figure 2.1 shows that several solution techniques
exist for the different types of optimization problems. Numerical algorithms are the traditional solution; typified by the popular linear programming simplex algorithm (Dantzig, 1963). Such algorithms are categorised as exact or complete methods and constitute the start-of-the-art approaches for solving various types of optimization problems in the diverse fields of mathematical programming (MP) and operations research (OR).

The development of new paradigms, based on various heuristics\textsuperscript{1}, follows the realisation that traditional methods generally lack global perspective in exploring the problem search space. In addition, the convergence guarantees of the traditional approaches are often limited to a locally optimal solution in the vicinity of their initialisation point. Thus, the emergence of new paradigms, popularly called approximate methods (Figure 2.1) opens up new possibilities for the development of successful approaches that can be applied to a wide range of optimization problems. The heuristics utilised by the approximate methods require little or no modifications when applied to various problem domains.

Approximate methods can be categorised into special- and general-purpose methods. On one hand, the design of special-purpose approximate methods mainly seeks to exploit some structural pattern in the problem under consideration. Consequently, such bespoke models tend to perform extraordinarily well on the particular set of problems for which they are optimized. However, the tailored design renders such methods inefficient even on similar problems. On the other hand, the general purpose approximate heuristics constitute the so-called stochastic optimization methods. Their distinctiveness centred on their ability to optimize in domains where:

- there is lack of a well-defined functional model relating the input variables to the output response of a given system;
- a model exists but its evaluation cost is prohibitively expensive; or
- the optimization surface (i.e., the fitness landscape) is complex, rugged (with plateaus), or has many locally optimal solutions (highly multimodal) with no exploitable structure.

\textsuperscript{1}The term (meta)heuristics is a generic term that was introduced to delineate a universal algorithmic framework designed to solve different optimization problems based on probabilistic decisions made during the search process.
Figure 2.1: Classification of optimization methodologies.

Algorithms that fall in the category of approximate methods come from a large number of heuristics established on different backgrounds and theoretical paradigms. The evolutionary computation (EC) methods, such as genetic algorithms (GA), genetic programming (GP) and evolutionary strategy (ES), are only a few examples of the naturally inspired stochastic heuristics. A comprehensive survey with historical details can be found in Fogel (1994). Since this thesis aims to investigate global optimization problems and propose solutions from the global perspective\(^2\), the review in this chapter is focused on the population based stochastic optimization frameworks under the paradigm of *evolutionary computation*.

While the stochastic optimization methods are usually easier to implement compared to their exact counterparts, on the downside, their lack of “sound” theoretical convergence guarantee often hinders their direct (online) use in mission-critical and safety-critical applications.

However, striving for more theoretically rigorous justifications, the EC community has, in the last couple of decades, seen several investigations that seek to formally describe the success of the evolutionary-based stochastic optimization methods. This probably began with the work of Vose (1995) on modelling genetic algorithms by Markov chain theory. Later, Rudolph (2000) analytically

\(^2\)In a global search space, stochastic local optimization approaches, such as stochastic hill-climber easily fall into the trap of local optima.
estimated the *takeover times* (see Section 2.6.3) for a variety of tournament selection methods as Markov chains and describes how takeover time is considered as an absorption time of a Markov chain having only one absorbing state. Greenhalgh and Marshall (2000), also modelled GA as Markov chain and determined upper as well as lower bounds for maximum iteration limit with a probability $\delta$ of having seen the optimum point. These approaches have already provided a remarkable insight into the convergence properties and dynamical behaviour of the nature-inspired optimization approaches. They have also led to the development of the so-called dynamical models of evolutionary algorithms by Rowe (2001). More recent work involve formalisation of the computational complexities of evolutionary systems and their convergence characteristics using a statistical means of estimating computational complexity called “drift analysis”. The investigations by He and Yao (2004) and Zhang et al. (2010) formulated the drift conditions based upon which an evolutionary algorithm can be said to possess a polynomial/exponential order.

### 2.2 Biological plausibility of Simulated Evolution – The ongoing debate

The pioneers of the various *simulated evolutionary*\(^3\) methodologies were inspired by the natural phenomenon of evolution. However, the extent to which these methodologies should conform to the biological notion of evolutionary genetics has remained a subject of contention in the EC domain. On one hand, a number of researches assert that successful EC designs should mimic and comply with the various phenomena of natural evolution. This includes the ability to sustain and, in some cases, prioritise population evolvability over fitness growth. Obviously the maintenance of evolvability is a long-term goal inherent in natural evolution. Therefore, aligned to this viewpoint is the notion of “novelty search” (Lehman and Stanley, 2011). Novelty search is an optimization method that favours intense exploration by prioritising diversification over fitness exploitation. Thus, supporting the insights gained from natural phenomenon of evolution, Bäck et al. (1997) posits:

---

\(^3\)Simulated evolution, as defined by Fogel et al. (1966), is “the process of duplicating certain aspects of the evolutionary system”.

“...we are far from using all potentially helpful features of evolution within evolutionary algorithms” (Bäck et al., 1997).

However, other investigations suggest utilising the very minimal concepts of natural evolution necessary to realise simulated evolutionary system suitable for optimization purposes, thus, Yao (2002) argue that:

“Although research in evolutionary computation could help us understand some natural phenomena better, its primary aim is not to build biologically plausible models” (Yao, 2002).

In this regard, the two viewpoints fundamentally seek to balance the optimization goals of fitness exploitation and search diversification via simulated evolution. Often called exploration and exploitation trade-off, this issue forms a key argument put forward in Chapter 5 of this thesis. In fact, preserving the ability to evolve new solutions while effectively harnessing the already found high quality ones is crucial to any successful global optimization approach.

### 2.3 Developmental Aspects of Evolutionary Computation

Evolution is a process that originated from the biologically inspired neo-Darwinian paradigm, i.e., the principle of survival of the fittest (Fogel, 1997). Evolutionary algorithms (EAs) are designed to mimic the intrinsic mechanisms of natural evolution and to progressively yield improved solutions to a wide range of optimization problems. The domain independent design of evolutionary algorithms facilitates their successful application on various problem domains without any incorporation of domain specific information.

As highlighted in the introductory chapter, the more popular evolutionary computation algorithms are the genetic algorithm (GA)\(^4\) (Holland, 1975), evolutionary strategies (ES) genetic programming (GP), and evolutionary programming (EP). These techniques largely share common basis and differ mainly in

---

\(^4\)Genetic algorithm was first introduced by Holland (1975) in the early 70s and it will be used in this study. Unless otherwise stated, any subsequent mentions of EC or EA in this thesis will be referring to the genetic algorithm.
some of their data structures, and in variation and reproduction operators. The terminologies used to describe the processes of simulated evolution, in most of parts of this thesis, are mainly borrowed from the fields of evolutionary genetics. Thus, a summary of their succinct meanings as used herein is given in Table 2.1.

In general, a genetic algorithm is an iterative procedure (Figure 2.2) that evolves a pool of candidate solutions across generations. If \( t \) is the generation counter, then GA starts with a fixed sized initial population \( P(t) : |P(t)| = N \).

A candidate solution point \( x_i \) is called an individual, and represents a single possible solution to the problem under consideration, i.e., in the phenotype space \( x_i \in \mathcal{P} \). A candidate solution, \( x'_i \in \mathcal{G} \), is a representation of an individual by a computational data structure called chromosome in the genotype space \( \mathcal{G} \).

Usually, a chromosome is encoded as a string of symbols of finite-length called genes. An encoded chromosome may be in the form of binary bit string, real-valued or some other specialised representations (Holland, 1975).

The sample solutions in the initial population are usually created randomly or via a simple heuristic construction, more detail on this is in Sections 5.3.1 and 5.4.5. At every generation, a stochastic selection process (Section 2.6.2) is applied on the initial population to choose better solutions following an evaluation that is based on some measures of fitness. The solutions that survive the selection process constitute a new set called parent and are qualified to take part in the remaining evolutionary processes.
### Table 2.1: Evolutionary Nomenclature

<table>
<thead>
<tr>
<th>Terminology</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual</td>
<td>$x$</td>
<td>An arbitrary sample solution, also called candidate solution</td>
</tr>
<tr>
<td>Gene</td>
<td>$-$</td>
<td>In biology, it is a hereditary unit of creation, in simulated evolution it is a representation unit that makes up an individual solution</td>
</tr>
<tr>
<td>Chromosome</td>
<td>$-$</td>
<td>A data structure (e.g. a string of binary) of fixed length of genes encoding a solution</td>
</tr>
<tr>
<td>Allele</td>
<td>$-$</td>
<td>A possible value a gene can take at any locus(^a) in a chromosome, e.g. a value of 0 or 1 for a binary bit</td>
</tr>
<tr>
<td>Population</td>
<td>$P$</td>
<td>A set of candidate solution setup to undergo an evolutionary cycle</td>
</tr>
<tr>
<td>Parent</td>
<td>$Pa, Pa_i$</td>
<td>A set or any sample solution involved in an evolutionary process expected to produce new solution(s)</td>
</tr>
<tr>
<td>Offspring</td>
<td>$O, O_i$</td>
<td>A new solution or set of solutions formed via evolving a single or group of parent solutions</td>
</tr>
<tr>
<td>Specie</td>
<td>$-$</td>
<td>A category/variety of solutions amenable to undergo certain evolutionary process</td>
</tr>
<tr>
<td>Niche</td>
<td>$-$</td>
<td>A batch of localised solutions sharing common features and/or occupying a specific region of an evolutionary search space</td>
</tr>
<tr>
<td>Phenotype</td>
<td>$p$</td>
<td>A solution in its original form that can be evaluated</td>
</tr>
<tr>
<td>Phenotype space</td>
<td>$\mathcal{P}$</td>
<td>The original problem space in which solutions get evaluated</td>
</tr>
<tr>
<td>Genotype</td>
<td>$g$</td>
<td>A representation of solution in a given encoding useful for crossover and mutation</td>
</tr>
<tr>
<td>Genotype space</td>
<td>$\mathcal{G}$</td>
<td>The solution encoding or representation space, e.g. a binary, gray, real-valued space, etc.</td>
</tr>
<tr>
<td>Generation</td>
<td>$G$</td>
<td>A complete cycle/iteration/epoch of an evolutionary model</td>
</tr>
<tr>
<td>Fitness</td>
<td>$f$</td>
<td>A quality of candidate solution that translates to chances of survival during selection</td>
</tr>
</tbody>
</table>

\(^a\)Locus: i.e. the specific site of a particular gene in a chromosome.

In order to explore other areas of the search space, the parent solutions (chromosomes) undergo recombination and/or mutation operations (Section 2.6.4) to generate a new set of chromosomes called offspring. The recombination entails exchange of characteristics by merging two parent chromosomes using a crossover operator, whereas mutation operation is a genetic alteration of a randomly chosen parent chromosome by a mutation operator.
A new generation of chromosomes is then formed via a pre-defined replacement procedure (Section 2.6.5) by selecting from either the combined pool of parents and offspring or the offspring pool based on a prescribed fitness measure. Fitter individuals have higher chances of been selected and the average fitness of the population is expected to grow with successive generations. The evolutionary cycle (Figure 2.2) continues until a termination criterion is met. Typical termination conditions enforce a user-defined limit on function evaluations, execution time, or when the search pool sufficiently converges to the optimum – or at least a suboptimal – solution. Note the classification of the key GA components on the basis of their operating evolution spaces in Figure 2.2. Analysis of the important distinction between the phenotype and genotype spaces follows in Section 2.4.

The complete dynamics of GA is as depicted in Algorithm 2.1. Notice from Algorithm 2.1 that, at any generation \( t \), the parameters: \( P(t) \), \( Q_s(t) \), \( Q_r(t) \) and \( Q_m(t) \) respectively represent the data structures for the population at the initial generation, at the end of selection, and after the recombination and mutation operations.

Because of their simple and stochastic nature, EAs require only the evaluation of the objective function but not its gradients. Such a derivative-free nature relieved EAs from the computational burden of evaluating derivatives. This is especially beneficial when dealing with complex objective functions where derivatives are difficult to compute. Further, the randomness in EAs improves their versatility in escaping the trap of suboptimal solution which is the major drawback in most local optimization techniques.
Goldberg (1989) summarised the key features of evolutionary algorithms that made them robust optimization methods. These are:

- EAs search with a population of sample solutions, not a single solution;
- The evolutionary variation operators (i.e. recombination and mutation) work on the encoded solution set, not directly on the solutions;
- The evolution operator (i.e. selection) works with a fitness measure rather than a derivative or other auxiliary knowledge; and
- The progress of the evolutionary process relies on probabilistic transition rules, not deterministic rules, that is, evolutionary search is stochastic and hence global.

The following section provides an exposition on simulated evolutionary processes for evolving solutions via mapping from one evolutionary space to another.

### 2.4 Simulated Evolution and Evolutionary Spaces

Atmar (1994) argued that despite the simplicity in the informational physics of the processes governing evolutionary system, delineating which processes of the evolution occur in what space is often misunderstood (cf. Figure 2.2). Formally, the evolutionary cycle runs in two distinct spaces: phenotypic and genotypic spaces (Lewontin, 1974) (Figure 2.3). The phenotype space $\mathcal{P}$ represents the behavioural or physical characteristics of an individual or chromosome, i.e. the solution in its original form; whereas the genotype space $\mathcal{G}$ is the encoding space and represents the exact genetic makeup of a chromosome, i.e. the solution in its encoded form. Figure 2.3 shows a simple simulation of evolution processes within and across generations depicting various genotype-phenotype mapping functions.

Simulated evolutionary systems often mimic natural evolution by creating their initial population in form of chromosomes encoded in a genotype space. Thus, as shown in Figure 2.3, the process usually begins in a genotype space with samples $g_i \in \mathcal{G}$. As enumerated in the following, the initial population evolves over generations and finally ends with a solution set $p_i \in \mathcal{P}$ in a phenotype space.
i) The first mapping function $f_1 : g \rightarrow p$ decodes from $G$ to $P$ space such that each genotype $g_i$ is translated into phenotype $p_i$ and then evaluated. Thus, the function shifts the evolution from the genotype to the phenotype space.

ii) The second mapping function $f_2 : p \rightarrow p$ describes the selection operation. It is the process of choosing individuals $p$ for reproduction and it occurs entirely in the phenotype space $P$.

iii) The third mapping function $f_3 : p \rightarrow g$ describes the genotypic representation. It is the process of encoding the genotype prior to reproduction. It shifts the evolution back to genotype space $G$.

iv) The fourth mapping function $f_4 : g \rightarrow g$ represents the evolutionary variation process. It is where variation operations such as recombination and mutation take place on the genotypic samples $g$. $f_4$ incorporates the rules of random and directed coding alteration during the reproduction process. This process entirely happens in the genotype space $G$.

Lewontin (1974) emphasised that although the distinction between evolutionary spaces $P$ and $G$ is sometimes illusory, it is imperative to clearly understand and delineate which part of the evolutionary process takes place at which state space.

Note that this study is limited to optimization of continuous linear/nonlinear problems (Section 1.4); adhering to real-valued encoding eliminates the need for
phenotype-genotype mapping. Thus, as a suitable choice in this regard, real-valued encoding will be used throughout this study. However, for the purpose of illustrations, the example demonstrating evolution, at a glance, in the following section (Section 2.5.1) utilises a binary encoded EC model. In addition, Chapter 4 of this thesis analyses the dynamics of EAs under both binary and real-valued encodings. In particular, Chapter 4 examines the convergence characteristics of the two EC models on various set of global optimization benchmarks.

2.5 Parameterisation in Evolutionary Computation

As will be demonstrated in the following example, EC methodologies involve a large number of parameters upon which their overall success relies. Because the performance of an EA heavily relies on the correct settings of its parameters, tuning the parameters to their optimal values is by itself a complex optimization problem. Thus, Grefenstette (1986) suggests using a meta-level GA to derive optimal parameters for a genetic algorithm on some specified problem category. Grefenstette’s proposal uses a user parameterised inner genetic algorithm to tune the parameters of a main genetic algorithm.

Early works in genetic algorithms by Holland (1975) and later by De Jong (1975) have led to the development of the most widely used standard parameter sets (Table 2.2). Although these settings are mainly obtained from empirical experiments, many other theoretical studies (Bäck, 1993; Boyabaltli and Sabuncuoglu, 2007; Goldberg and Deb, 1991) have reinforced the validity of such parameter settings. Thus, the space of a simple genetic algorithm comprises at least the six basic parameters (Table 2.2) which could be categorised into two main classes:

i. **Structural Parameters**: These include choice of data type (i.e. representation scheme) and the types of genetic operators used.

ii. **Numerical Parameters**: These include but not limited to the population size, probabilities of mutation and crossover, maximum number of generations, replacement rate, etc.
Table 2.2: Standard Parameters of a Typical Genetic Algorithm.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Typical values/Description/Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>$N$</td>
<td>$N \in [50, 200] : N \in \mathbb{N}$</td>
</tr>
<tr>
<td>Selection Scheme</td>
<td>–</td>
<td>Fitness proportionate, Ranking, Tournament, etc.</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>$P_C$</td>
<td>$[0.6, 1.0] : P_C \in \mathbb{R}$</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>$P_M$</td>
<td>$[0.01, 0.05]$ or $1/l : P_M \in \mathbb{R}, l = \text{string length}$</td>
</tr>
<tr>
<td>Replacement Scheme</td>
<td>–</td>
<td>Generational, Elitist, Overlap, etc.</td>
</tr>
<tr>
<td>Termination Criteria</td>
<td>–</td>
<td>Maximum Runtime, Generations, Function evaluations or any heuristic convergence measure.</td>
</tr>
</tbody>
</table>

2.5.1 A Peek into GA – Understanding How it Works

This section presents a case study that explores the dynamics of a hypothetical binary coded genetic algorithm. A complete evolutionary cycle consisting of initialisation and evaluation, reproduction and variation is illustrated. Specifically, the case study demonstrates how the use of proportionate/tournament selection methods and, the crossover and mutation operators stochastically evolve a randomly created solution pool over generations.

Case Study: Ice Cream Cone Production

Problem Statement:
An ice cream manufacturer wishes to mass produce ice cream waffle cones (cornets). Inside of each cornet is coated with a layer of oil, sugar and chocolate to minimise leaks and improve taste. Assume that a cornet is required to have a capacity of at least one scoop of ice cream, equivalent of 25 cubic units. Assume also that an acceptable height range for cornet lies within $h \in \mathbb{Z} | 4 \leq h \leq 12$ units and the base radius lies within $r \in \mathbb{Z} | 1 \leq r \leq 3$ units. Let the cost of producing a unit area of the cornet be £0.36. The manufacturer wishes to find an optimum dimension for the cornet that would minimise the cost of producing the waffle.

Problem modelling:
From the manufacturer’s perspective the task is essentially an optimization problem. The goal is to design a cone (of course an open-ended one) that satisfies the following requirements:
Let the cost of a unit area of the waffle be \( K = 0.36 \). Let the cone height and radius be \( h \) and \( r \) respectively, then

\[
V = \frac{1}{3} \pi r^2 h, \quad \text{and} \quad A = \pi rl = \pi r \sqrt{r^2 + h^2},
\]

where \( l = \sqrt{r^2 + h^2} \) is the lateral height of the cone.

Since both the volume \( V \) and surface area \( A \) of a cone are simply functions of \( r \) and \( h \), the task is defined in terms of the following optimization problem:

\[
\begin{align*}
\text{Minimise:} & \quad f(r, h) = \pi r \sqrt{r^2 + h^2}, \\
\text{Subject to:} & \quad g(r, h) = \frac{1}{3} \pi r^2 h \geq 25, \\
\text{Bounded within:} & \quad \{r \in \mathbb{Z} \mid 1 \leq r \leq 3\}, \\
& \quad \{h \in \mathbb{Z} \mid 4 \leq h \leq 12\}.
\end{align*}
\]

(2.2)

**Encoding:**

For the purpose of illustration, simple binary encoding will be used. From the given variable bounds for \( r \) and \( h \) (2.2), the radius and height of any sample solution can be represented with binary bits of length 3 and 4 respectively. And by concatenation, a candidate solution (chromosome) will have a string length of 7 bits.

Suppose that an ice-cream cornet \( A \) has radius \( r = 3 \) and height \( h = 9 \) (see Figure 2.4 for dimension specifications), then, its binary encoded chromosome is:

\[
\text{Chromosome } A : \quad (r, h) = (3, 9) \equiv \underbrace{101}_{3} \underbrace{1001}_{9}, \quad \text{or simply} \quad 1011001.
\]

Notice that based on the adopted encoding, the random evolutionary operators may yield solutions which violate the constraints and/or bounds of the design requirements. Let us adopt the simple technique of penalising any infeasible solution by adding to its fitness a value proportional to its constraint violation\(^5\).

Then, besides the representation (encoding) task, the complete evolutionary cycle involves the following sequence of operations.

\(^5\)Constraint handling is a huge area of research in its own right, and the motive in this example is not to establish the best constraint handling strategy rather to demonstrate how a simple GA works.
Table 2.3: Initialisation and Evaluation of a hypothetical GA Sample Pool

<table>
<thead>
<tr>
<th>Chromosome (Binary)</th>
<th>Decoded Values</th>
<th>Surface Area (A)</th>
<th>Volume (g)</th>
<th>Fitness (f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100101</td>
<td>{2, 5}</td>
<td>12</td>
<td>21</td>
<td>62 = 12 + 50</td>
</tr>
<tr>
<td>110101</td>
<td>{3, 5}</td>
<td>20</td>
<td>47</td>
<td>20</td>
</tr>
<tr>
<td>010100</td>
<td>{1, 4}</td>
<td>5</td>
<td>4</td>
<td>55 = 5 + 50</td>
</tr>
<tr>
<td>100111</td>
<td>{2, 7}</td>
<td>16</td>
<td>29</td>
<td>16</td>
</tr>
<tr>
<td>110100</td>
<td>{3, 4}</td>
<td>17</td>
<td>38</td>
<td>17</td>
</tr>
<tr>
<td>111000</td>
<td>{3, 8}</td>
<td>29</td>
<td>75</td>
<td>29</td>
</tr>
</tbody>
</table>

Average Fitness $\bar{f} = 33.17$

*Initialisation and Evaluation:*

The conventional method of creating the initial population (pool) of candidate solutions (chromosomes) is random. Thus, for the purpose of illustration, a randomly generated pool of only 6 samples is utilised in this case (Figure 2.4). Table 2.3 shows the encoded samples constituting the initial pool as strings of binary bits and their decoded decimal values. The corresponding values for the surface areas and volumes of the resulting cornets are also evaluated based on the definitions in equations (2.1). Finally, their respective fitness values are evaluated according to the optimization problem defined in (2.2). The fitness function is
equal to the surface area whenever the volume is at least 25 unit cube, else a constant penalty of $P = 50$ is added to whatever is the current value for the surface area.

*Reproduction:*  
Selection operation is carried out such that fitter samples (i.e. those having minimum fitness values in this case) are prioritised. This section illustrates two of the most commonly used selection approaches. First, the conventional fitness proportionate (Roulette wheel) selection method is examined, and later, the use of tournament selection method is demonstrated.

1. **Fitness Proportionate selection:** As the name implies, this selection process selects individuals based on the proportion of their fitness with respect to the entire pool, (Section 2.6.2). Since the optimization goal is to minimise the objective function the selection will favour those samples carrying the least proportion of the population’s average fitness. The result of this is presented in Table 2.4.

2. **Tournament Selection:** In contrast to Roulette wheel selection method, tournament selection (Section 2.6.2) requires specification of a tournament size. Here, a tournament size of two (binary tournament selection) is utilised. Thus, in two rounds of contest, two randomly chosen samples (without replacement) compete and the best fit sample is chosen (Table 2.5). Since the optimisation objective in this case study is a minimisation one, the sample with the least fitness value wins. In this way, every sample in the solution pool competes in exactly two contests.

Notice from the Roulette wheel and Tournament selection methods that the selection process works on the decoded values, i.e. the phenotype space. In addition, selection process inherently favours fitter solutions which results in improvement in the average fitness of the sample pool. Of course, having the two selection approaches returning identical results as witnessed above is only a coincidence. And as will be discussed further in Section 2.6.2, the Roulette wheel method has some known limitations which are essentially avoided in the tournament methods. Subsequent to the selection process, the chosen samples constitute the *parent pool* and undergo the following evolutionary variation processes.
CHAPTER 2. STOCHASTIC OPTIMIZATION – AN EC PERSPECTIVE

Table 2.4: Fitness proportionate selection: A demonstration of Roulette Wheel fitness proportionate Selection for the Hypothetical GA.

<table>
<thead>
<tr>
<th>Initial Population</th>
<th>Selected Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoded values {$r, h$}</td>
<td>Fitness</td>
</tr>
<tr>
<td>{2, 5}</td>
<td>62</td>
</tr>
<tr>
<td>{3, 5}</td>
<td>20</td>
</tr>
<tr>
<td>{1, 4}</td>
<td>55</td>
</tr>
<tr>
<td>{2, 7}</td>
<td>16</td>
</tr>
<tr>
<td>{3, 4}</td>
<td>17</td>
</tr>
<tr>
<td>{3, 8}</td>
<td>29</td>
</tr>
</tbody>
</table>

Average Fitness $\bar{f} = 33.17$

Average Fitness $\bar{f} = 19.17$

Table 2.5: Tournament selection

Comparing the samples’ Fitness ($f$)

<table>
<thead>
<tr>
<th>Tournaments</th>
<th>Winners</th>
</tr>
</thead>
<tbody>
<tr>
<td>{62 vs 17}</td>
<td>17</td>
</tr>
<tr>
<td>{20 vs 29}</td>
<td>20</td>
</tr>
<tr>
<td>{55 vs 16}</td>
<td>16</td>
</tr>
<tr>
<td>{16 vs 62}</td>
<td>16</td>
</tr>
<tr>
<td>{17 vs 20}</td>
<td>17</td>
</tr>
<tr>
<td>{29 vs 55}</td>
<td>29</td>
</tr>
</tbody>
</table>

Average Fitness $\bar{f} = 19.17$

Variation: Crossover and Mutation Operators

The working principles of the most commonly used evolutionary variation operators, i.e., crossover and mutation, are illustrated in this example. In general, variation operators work on the encoded values (binary in this case) of the sample solutions, i.e. in the genotype space. Assuming all selected samples (parents) undergo the crossover operation (i.e., $P_C = 1.0$) and the probability of mutation is $P_M = 0.01$. Then, the parent pool is shuffled to form a mating pool for the variation processes (Table 2.6).

First, the mating pool undergoes the crossover operation – with every two randomly chosen parent chromosomes exchanging their bits to the right of a randomly chosen crossover point. Then, mutation operator is applied on the resulting offspring at a rate equal to the probability of mutation $P_M$. It can be observed that the average fitness of the pool has improved from the initial pool having 33.17 (Table 2.3), to the new pool having 23.50 (Table 2.6) at the end of the first evolutionary cycle, i.e., a generation. Although this is not absolutely necessary, in
most cases the average fitness of the pool is expected to improve with subsequent generations. The fitness improvement with generations is not accidental. This is because the samples undergoing the evolutionary cycle have already passed a selection process and possess the right qualities to evolve better offspring. This is why GA does better than a mere random/brute force search\(^6\). Hence, as demonstrated in this case study, the continuing cycle of reproduction and variation processes eventually results in newer samples having improved fitness levels.

The following section presents detailed formalisations and descriptions for the several simulated evolutionary processes introduced thus far.

### 2.6 The Building Blocks in EC

To properly set up an EC model for solving global optimization problems, a number of questions arise. First, appropriate choice of suitable data structure, initialisation method and population sizing are of primary importance. This then lays a suitable ground for the choice and parameterisation of the evolutionary reproduction and variation processes. In this section, each of these aspects is examined in turn, and their respective roles in building successful EC models (as will be seen in later chapters) are highlighted.

\(^6\)In random search there is no information transfer from the previously seen solution to the next. However, in stochastic heuristics like EAs, knowledge gathered from previously seen solutions is *intelligently* used to guide the generation of new ones.
2.6.1 Initialisation: Representation and Population Sizing

A. Representation

A suitable representation scheme is crucial to the overall performance (efficiency and robustness) of an EA. Holland (1975) advocated using binary representation. The motivation was to ensure that the genetic variation processes operate in a domain (i.e. space) that is distinct from that of the original problem. This enhances the robustness of evolutionary algorithms by making them more problem-independent. Furthermore, binary representation eases the design and implementation of the major evolutionary operators. This section presents binary encoding and the mapping between genotype and phenotype spaces in a typical evolutionary computation algorithm.

Suppose an optimization problem is defined as:

\[
\begin{align*}
\text{maximise/minimise: } & \quad y = f(x), \\
\text{Subject to: } & \quad x \in X,
\end{align*}
\]

then the fitness function \( f : X \rightarrow Y \) assigns a cost value to each element in the decision variable \( X \) by mapping it to a corresponding element in the solution space \( Y \in \mathbb{R} \). For any variable \( x_i \in \mathbb{R} \mid a_i \leq x_i \leq b_i \), assuming the precision requirement is \( p = 10^{-2} \), then the length \( l_i \) of the binary bits required to map the real variable \( x_i \) into a corresponding binary variable \( x'_i \) is derived from:

\[
2^{l_i-1} \leq \frac{b_i - a_i}{p} \leq 2^l : l_i \in \mathbb{N}.
\]

Now, for any multidimensional function having \( \{x_i \in \mathbb{R} \mid i \in \mathbb{N}\} \) real variables. If each of these variables is mapped to its corresponding binary variable \( \{x'_i \in \mathbb{B} \mid i \in \mathbb{N}\} \), then for a population of size \( N \) an individual binary chromosome \( \tilde{x}_k \) is obtained by concatenating all the binary variables as follows:

\[
\tilde{x}_k = x'_1 | x'_2 | \cdots | x'_n : k = 1, 2, \ldots, N.
\]

Hence, the total length \( L \) of the resulting binary chromosome \( \tilde{x}_k \) is equal to the sum of the bit length \( l_i \) of all the \( n \) binary variables \( x'_i \) (2.6),

\[
L = \sum_{i=1}^{n} l_i.
\]
Note that the precision requirement for the decision variables may differ from one variable to another within a given problem. Thus, in general, for any decision variable defined as $x_i \in \mathbb{R} : a_i \leq x_i \leq b_i$, to map $x_i$ into a binary string of length $l_i$, the precision requirement is:

$$p_i = \frac{b_i - a_i}{2^{l_i} - 1}. \quad (2.7)$$

With the problem encoded into binary (i.e. genotype) space, decoding the chromosomes back to the phenotype space is a reverse process and it is necessary for evaluating their fitness before selection. This process entails the following two steps:

**First:** Decomposing the binary chromosome $\tilde{x}$ into its constituent binary variables $x'_i$. This requires splitting the $L$ bits of $\tilde{x}$ into chunks of $l_i$ bits corresponding to the $x'_i$ binary variables. Then, the corresponding real variables $x_i$ are derived via binary to decimal transformation of the $l_i$ bits of $x'_i$, such that:

$$x_i = \sum_{j=1}^{l_i} b_j 2^{l_i-j} : i = 1, 2, \ldots, n. \quad (2.8)$$

where $b_j$ are the binary bits of $x'_i$, $l_i$ is its length and $n$ is the total number of these variables.

**Second:** Mapping the obtained real variables $x_i$ to conform to their originally defined ranges $a_i \leq x_i \leq b_i$, such that:

$$x_i = a_i + \frac{b_i - a_i}{2^{l_i} - 1} x'_i : \forall i = 1, 2, \ldots, n. \quad (2.9)$$

Occasionally, situations arise where binary representation is not only promising but also the natural choice. The knapsack problem in operations research is a typical example. A 0-1 knapsack problem consists of a set of $n$ items to be packed into a knapsack of size $K$ units. If each item has a weight $w_i$ and is of size $k_i$ units, then the goal is to maximise the weight for a given subset $I$ of the items such that:

$$\max \sum_{i \in I} w_i : \sum_{i \in I} k_i \leq K. \quad (2.10)$$
Reeves and Rowe (2004) showed that the knapsack problem can be reformulated as an integer programming problem, and as a result, a sample solution can be represented as a binary string of length \( n \). Consequently, with no distinction between the genotype and phenotype spaces, mapping functions are simply eliminated.

Previously, the general view in the EC community regarding problem’s data structure and the choice of suitable EC algorithm was to match the problem under consideration to a suitable EC model. For example, the designs of evolutionary strategies are centred on real valued representation which is suitable for continuous problems. Similarly, the designs of genetic algorithms proposed by Goldberg (1989) focused on discrete optimization and use binary representation as a norm. However, many of the later researches (Blum and Roli, 2008; Chambers, 1995; Raidl, 2005) revealed that each of these algorithms is been successfully used with all kinds of representations for various optimization problems.

Table 2.7 compares and summarises the various representation techniques reviewed from various domains of the evolutionary computation. It highlights the cases when phenotype-genotype mapping functions is necessary and pinpoints a suitable category of the genetic operators to adopt.

Various types of representations for genetic algorithms are echoed in the literature for different problem domains. For some optimization problems, the binary representation is inadequate or even unsuitable for appropriate encoding of the problem under investigation. Greenhalgh and Marshall (2000) argue that although Goldberg’s (Goldberg, 1989) notion of implicit parallelism in genetic processing favours binary representation, practitioners report better performance with non-binary representations in many real-world applications (Davis, 1989). In fact, in some situations the use of problem dependent representations is necessary. Often there exist apparent representation schemes that can best suit the problem to be modelled. Optimization of permutation problems is a typical example where there is a natural choice for representation. In such a case, the representation can directly be defined over the range of all the possible permutations. Similarly, for optimization in continuous (real-valued) domain, which is the problem domain of interest in this thesis, use of floating-point representation is an appropriate choice.
Table 2.7: A summary of commonly used representation methods in EAs

<table>
<thead>
<tr>
<th>Problem type</th>
<th>Example problem</th>
<th>Representation type</th>
<th>Mapping function</th>
<th>Genetic operator type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
<td>Knapsack problem (Reeves and Rowe, 2004)</td>
<td>Direct binary</td>
<td>×</td>
<td>Generic</td>
</tr>
<tr>
<td>(binary)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network distribution (Mendoza et al., 2006; Sivanagaraju et al., 2006)</td>
<td>Direct binary</td>
<td>×</td>
<td>Generic</td>
</tr>
<tr>
<td></td>
<td>Feature selection (Oh et al., 2004)</td>
<td>Direct binary</td>
<td>×</td>
<td>Generic</td>
</tr>
<tr>
<td></td>
<td>Rotor stacking (McKee and Reed, 1987)</td>
<td>Direct binary, range of permutation</td>
<td>×</td>
<td>Specialised</td>
</tr>
<tr>
<td></td>
<td>Internet reconfiguration (Queiroz and Lyra, 2009)</td>
<td>q-ary encoding, q &gt; 2</td>
<td></td>
<td>Specialised</td>
</tr>
<tr>
<td>Permutation</td>
<td>Flowshop sequencing (Reeves, 1995)</td>
<td>Direct integer, range of permutation</td>
<td>×</td>
<td>Specialised</td>
</tr>
<tr>
<td></td>
<td>TSP problem (Radcliffe and Surry, 1994)</td>
<td>Allelic, using ordered pairs</td>
<td></td>
<td>Specialised</td>
</tr>
<tr>
<td></td>
<td>Process scheduling problem (Lin and Yang, 1999)</td>
<td>Direct integer</td>
<td></td>
<td>Specialised</td>
</tr>
<tr>
<td>Combinatorial</td>
<td>Parameterised scheduling strategy (Chambers, 1995)</td>
<td>Binary encoding</td>
<td></td>
<td>Generic</td>
</tr>
<tr>
<td>Continuous</td>
<td>Continues</td>
<td>Real encoding</td>
<td>✓</td>
<td>Generic</td>
</tr>
<tr>
<td>non-discrete</td>
<td>linear/nonlinear</td>
<td>Binary encoding</td>
<td>✓</td>
<td>Generic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gray encoding</td>
<td>✓</td>
<td>Generic</td>
</tr>
</tbody>
</table>

B. Population Sizing

Let the initial population $P_0$ of size $N$ represent a set of points in the search space of all possible populations, then every evolutionary generation is expected to shift the initial population $P_0$ to a different set of points $P_t$ in the search space. However, deriving optimal population size for EAs still remains an area of further research (Rowe, 2001). This is because, while larger population sizes facilitate wider exploration of problem search space, they impair the efficiency of
the search process due to increased evaluation of low quality samples. On the other hand, too small population size would not permit adequate exploration of all the promising areas of the search space and may risk premature convergence to a suboptimal solution. Arguably, larger population sizes allow EAs to easily discriminate between good and bad building blocks. Thus, Goldberg et al. (1989) argue that the parallel processing and recombination of good building blocks permit effective solutions of large and deceptive global optimization problems. Ultimately, in spite of the several theoretical viewpoints on optimum population sizing, the underlying trade-off between efficiency and effectiveness remains.

In a series of empirical investigations, De Jong (1975) showed that for a standard binary encoded genetic algorithm population sizes of 50 to 100 are sufficient for wide range of optimization problems. A theoretical study by Reeves (1993) revealed an interesting finding on what constitute a minimum population size for a binary encoded EA. Reeves (1993) showed that for every point in the search space to be reachable from the initial population by a recombina
tive genetic algorithm (i.e. a GA using only the crossover operator), there must be at least one instance of every allele at each locus in the whole population of strings. For binary representation, the probability that at least one allele is present at each locus is:

$$P = \left( 1 - \left( \frac{1}{2} \right)^{N-1} \right)^l \approx \exp \left( -\frac{l}{2^{N-1}} \right), \quad (2.11)$$

where $N$ is population size and $l$ is the chromosome or string length.

Thus, for 99.9% confidence level, the minimum population size $N$ is:

$$N \approx \left\lceil 1 + \frac{\log \left( -\frac{l}{\ln P} \right)}{\log 2} \right\rceil \approx \left\lceil 1 + \frac{\log(999.5 \times l)}{\log 2} \right\rceil. \quad (2.12)$$

Figure 2.5 shows the characteristic curves for the approximation of $N$ (2.12) for higher cardinality ($q$-ary) representations. Figure 2.5 reveals at least two important findings. First, the minimum population size required when a binary representation (i.e. when $q = 2$) is used, with up to a string length of 200 bits, is not more than 20 individuals. Second, the threshold for the minimum population

---

7The Building Block (BB) terminology was used by Holland (1975) to represent the short, low order (or low defining-length) schemata. Building Block Hypothesis (BBH) suggests that GA performs adaptation efficiently by combining and processing these short, low order schemata (BB) which have above average fitness (Goldberg, 1989).
size grows as the encoding cardinality increases. Note that the above expression (2.12) does not prescribe an optimum value for the population size. However, it suggests a threshold below which a population may not guarantee adequate exploration of the problem space by a binary-coded EA. Similarly, the values shown by the curves (Figure 2.5) are not the optimal values for the population size, neither are they sufficient for a realistic global search. However, such findings could justify the occasional convergence of EAs to global optimal solutions with “extraordinarily” small population sizes. Thus, for all the experimentations in thesis, wherein binary encoding is utilised (e.g. in Section 4.4.2), a pool size larger than the above minimum threshold is adopted to avoid under-sampling the search space.

2.6.2 The Selection Operator

Selection process is often seen as the driving force in evolutionary algorithms (Blum and Roli, 2008; Hancock, 1994). Selection methods in simulated evolution operate on the original solution (phenotype) space and, they imitate the process of natural selection by favouring solutions that encode successful structures. Successful solutions refer to those individuals that have higher fitness values relative to their counterparts. Although this notion (seeking for the best out of many) undoubtedly suites the key idea of optimization, it does seem to contradict some
viewpoint of natural evolution\(^8\), leaving evolutionary theorist to raise questions like:

“... does natural selection always favour those behavioural strategies that seek to minimise expected loss?” (Fogel, 2008).

In practice, there are varieties of selection methods in evolutionary computing. Choosing appropriate selection method can be difficult as it involves deciding on crucial parameters like selection pressure, selection intensity, growth rate, takeover time etc., which collectively dictate the mode and rate of convergence of the EC model. Thus, selection plays an important role in the parameterisation of EC and is critical to the overall success of an EA. The following sections examine the principles of the various selection approaches in EC.

A. Fitness Proportionate Selection Methods

Roulette wheel selection (RWS) is the simplest and most commonly used fitness proportionate (FPS) means of selection in evolutionary computations. RWS has a probability distribution such that the probability of choosing an individual is always directly proportional to its fitness. This probability intuitively corresponds to the area of a sector of a roulette wheel; the larger the sector, the higher the chances of selecting a given sample solution. Hence, the name of the roulette wheel selection method. For any population \( P \) consisting of a set of \( \{x_i \in X | X \in \mathbb{R}; |X| = N\} \) samples, let the fitness evaluation function be \( f : X \rightarrow Y \), then the total fitness of the population \( F \) is:

\[
F = \sum_{i=1}^{N} f(x_i),
\]  \hspace{1cm} (2.13)

the fitness proportion of a solution \( x_i \) corresponds to its probability \( p(x_i) \):

\[
p(x_i) = \frac{f(x_i)}{F} : i = 1, 2, \ldots, N;
\]  \hspace{1cm} (2.14)

its cumulative probability \( q(x_i) \) is:

\[
q(x_i) = \sum_{j=1}^{k} p(x_j) : k = 1, 2, \ldots, N; \quad q(x_k) \in [0, 1].
\]  \hspace{1cm} (2.15)

\(^8\)The role of selection is also part of the argument on the biological plausibility of simulated evolution (Section 2.2).
Selection of $N$ samples requires spinning the wheel $N$-times. This corresponds to $N$-times sampling (with replacement) from a pseudo-random sequence $rand \sim U[0,1]$. After every spin, sample $x_k$ is selected if the random number falls within the interval of its cumulative probability and that of its predecessor, i.e. if $q(x_{k-1}) \leq rand \leq q(x_k)$. As a result, samples with higher fitness values have larger cumulative probabilities and hence higher chances of been selected.

Although RWS scheme enjoys a great level of simplicity, it suffers from issues relating to sampling error and scaling. The lack of scaling often leads to a sudden collapse in selection pressure (Section 2.6.3) as population average fitness improves. The commonly used measures to tackle the lack of scaling in RWS include windowing and (linear/sigma) scaling (Chambers, 1995; Hancock, 1994). Advanced remedies include the use of rank-based selection method, proposed by Baker (1985). It entails ranking sample solutions in order of their fitness values.

B. Tournament Selection Methods

Tournament selection method is inspired by the natural mating contest in which a group of individuals compete for reproduction. It entails setting up a contest for $k < N$ individuals sampled independently without replacement from the main pool. The sample with higher fitness value wins the contest and the process is repeated until the required number of samples is selected. Selecting the contestants can be carried out with or without replacement. Further, the tournament size $k$ can be varied such that larger $k$ leads to increased selection pressure. Hence, this method is believed to yield improved control on the level of selection pressure and the approach is immune to scaling problems (Chambers, 1995; Reeves and Rowe, 2004). A tournament size of $k = 2$ yields the so-called binary tournament selection, which is probably the most commonly used implementation of tournament selection and is utilised in most of the EC models examined in this thesis.

Stochastic tournament selection is a variant of the strict tournament selection whereby the winner of any contest is only selected at certain probabilistic rate $p$. Thus, the rate of having the best sample being selected has now reduced to $kp$ instead of $k$ times. A value of $p = 0.5$ turns the process into a random selection. Note that even with $p > 0.5$, stochastic tournament implementations will remain susceptible to occasional sampling errors. Other selection schemes include truncation selection (Thierens and Goldberg, 1994), steady state GAs or Genitor (Goldberg and Deb, 1991; Hancock, 1994), and the ($\lambda, \mu$) and ($\lambda + \mu$)
methods originally inherited from evolutionary programming and evolutionary strategies (Goldberg and Deb, 1991).

2.6.3 Selection Analytic Parameters

As highlighted above, when investigating selection schemes it is imperative to understand the following crucial parameters (some of these parameters have their origin from field of quantitative genetics (Thierens, 1998)) that govern the choice of appropriate selection scheme.

i. Response to selection $R(t)$: This quantifies the difference in the average fitness of population between two successive generations.

$$R(t) = \bar{f}(t) - \bar{f}(t - 1), \quad (2.16)$$

where $\bar{f}(t)$ is the average fitness of the population at generation $t$.

ii. Selection Differential $S(t)$: This measures the difference between the average fitness of the parent set $\bar{f}_p(t)$ (i.e. individuals selected for reproduction) at generation $t$ and the average fitness of the entire population $\bar{f}(t)$ also at generation $t$.

$$S(t) = \bar{f}_p(t) - \bar{f}(t), \quad (2.17)$$

where $\bar{f}(t)$ and $\bar{f}_p(t)$ are the average fitness of the entire population and that of the parent population respectively.

iii. Selection Intensity $I(t)$: This is a dimensionless quantity derived by taking the ratio of the selection differential $S(t)$ with the population standard deviation also at generation $t$ such that:

$$I(t) = \frac{S(t)}{\sigma(t)} = \frac{f_p(t) - f(t - 1)}{\sigma(t)}, \quad (2.18)$$

where $\sigma(t)$ is the population’s standard deviation at generation $t$.

iv. Selection Pressure $P_s$: Selection pressure\(^\text{9}\) is another measure that estimates the expected number of offspring a best fit individual will have after selection.

\(^\text{9}\)Selection pressure is conventionally denoted by $P$, but since $P$ has already been used to denote population in this thesis, we assign a subscript to denote selection pressure as $P_s$. 
Although selection intensity $I(t)$ can be useful for such comparison purposes, it is application is limited to generational GAs where there is a clear set of parent population. Thus, $I(t)$ cannot be used to analyse elitist or steady state GAs. Selection pressure is highly related to convergence rate and can be defined in various ways (Hancock, 1994). A straightforward definition for $P_s$ is in terms of the ratio of probabilities:

$$P_s = \frac{P(\text{selecting best string})}{P(\text{average string})}.$$  

(2.19)

v. Takeover time: This is a measure that estimates how long it will take the best individual to take over the entire population. For any selection method applied on a population of size $N$ consisting of a single copy of the best individual, takeover time can be determined by evaluating the expected number of generations before getting a population vector that entirely consists of the copies of the best individual from the initial population. The measure furnishes some valuable insight into the complexity, growth rate and many other characteristics of a selection method.

Since tournament size is related to selection pressure (Reeves and Rowe, 2004), with tournament selection one could regulate the level of selection pressure, which influences the convergence of an evolutionary process to optimal solution. Thus, tournament selection is adopted for the EC models proposed in this thesis.

2.6.4 Variation operators: Recombination and Mutation

Following the reproduction (selection) process, the second phase of evolution involves the variation processes. The variation processes consist of two main operations, namely recombination (i.e. crossover) and mutation.

It is important to note that while the ultimate aim of the variation operation is the same whether in binary or real-valued encoded space, this section mainly introduces the concepts of crossover and mutation for binary encoded samples; specific details for the design of crossover and mutation operators in real-valued space will be presented later\(^{10}\) in Section 7.6.1 and 7.6.2.

Generally, variation operators are the main sources for introducing new individuals into a solution pool. Further, the operations in the variation phase are

---

\(^{10}\)Chapter 7 addresses adaptation of variation operators in real-valued space; thus, specific details on designing real-valued variation operators are provided there to avoid duplication.
often completely detached from the original problem formulation (phenotype). In other words, all the operations take place in the encoding (genotype) space. This feature essentially made the evolutionary algorithms problem-independent and robust in global optimization.

A. Recombination – The Crossover Operator

In its simplest form, the crossover operator generates two offspring by exchange of genetic materials between two parent samples subject to probabilistic decisions. As demonstrated earlier (Section 2.5.1), when individuals are represented as binary strings, parent samples (e.g., \( Pa_1 \) and \( Pa_2 \) in (2.20)) undergo binary crossover operation by exchanging the bits to the right of a randomly chosen locus called crossover point to yield two new offspring \( (O_1 \) and \( O_2) \).

\[
\begin{align*}
Pa_1 &= 101|0110, & O_1 &= 101|1101, \\
Pa_2 &= 010|1101, & O_2 &= 010|0110.
\end{align*}
\] (2.20)

This crossover scheme (2.20) is called single point (or \( 1X \)) crossover. It can easily be extended to a number of variants by creating a number of random crossover points after which the parent strings exchange their bits. Extended versions of single point crossover are called multi-point (\( m \)-point) crossover, with \( m > 1 \).

Crossover plays two key roles in evolution. First it provides a chance for further examination (exploitation) of the already explored hyperplane, like the offspring \( O_1 \) simply continues with the exploitation of the hyperplane \( 101\star\star\star\star11 \). Second, it allows exploration of new areas of the search space like the hyperplane \( \star1001\star \) in offspring \( O_2 \). Hence, every evaluation of a string of length \( l \) guides the search process by adding knowledge of \( 2^l \) hyperplanes. These two complementary roles of crossover are critical for a successful evolutionary search.

A popular alternative to \( m \)-point crossover is the so-called uniform crossover. Uniform crossover seeks to eliminate the bias\(^{12}\) in \( m \)-point crossover by making the operation completely random. The process requires representing the crossover operator as a binary mask obtained from a typical Bernoulli distribution.

Conversely to the crossover designs above, when real-valued encoding is utilised,

---

\(^{11}\)The \( \star \) fields stand for any possible allele value, e.g. 1 or 0.

\(^{12}\)The single point crossover operator is said to suffer positional bias because it tends to favour sub-strings of contiguous bits (Eshelman et al., 1989).
the most commonly used crossover operators are the discrete and intermediate recombination operators. While the former involves direct exchange of component between the parent samples, the latter involves linear combination (or averaging) of the parent solutions to generate new ones; further details on real-valued crossover follows in Section 8.3. Nevertheless, in line with the argument put forward by Lin and Yang (1999) – that conventional crossover operators do not perform well on complex optimization problems as they lack problem-specific knowledge in their encoding – many problem dependent crossover operators have been proposed (Jeurissen and van den Berg, 2008; Jih and Yung-Jen Hsu, 1999; Xiao and Tan, 2008) and successfully applied on various problem domains.

In general, crossover operator is applied on a population of individuals based on some probabilistic decisions. A survey of various theoretical analyses by Bäck et al. (1997) has shown that crossover probabilities of $P_C = [0.6, 1.0]$ are considered optimal for most global optimization problems. Although such recommendations have proven to be quite effective on wide range of global optimization problems, several recombination operators that automatically adapt their rates (Jiang et al., 2008a) have also been proposed, examples include a statistics-based adaptive non-uniform crossover (Herrera and Lozano, 2000) and punctuated crossover (Bäck et al., 1997).

B. Mutation Operator

Besides crossover, mutation is the second variation operator that also works in the genotype space and is capable of producing a new individual from a single parent string. Unlike the crossover operator, mutation can effectively guide the exploration of the search space when applied to a small population for large number of generations. In principle, it involves a probabilistic addition (at a mutation rate $P_M$) of random noise to sample solutions. Operationally, mutation involves simple bit flip operations for binary encoded strings or a random addition of Gaussian noise to a real-valued encoded sample. Detail treatment of real-valued mutation implementations follows in Section 8.2.

A number of empirical experimentations (Grefenstette, 1986) have shown that a mutation probability within the range of $P_M = [0.01, 0.05]$ is sufficient for wide range of global optimization problems. The so-called universal/standard setting (De Jong, 1975) which recommends a $P_M = 1/l$ has also received some wide acceptance (Bäck et al., 1997) in the genetic algorithms community. These
settings are not without some issues though. Investigations by Salomon (1996) show that small mutation rates are mostly suitable for problems of unimodal or pseudo-unimodal\textsuperscript{13} type, but multimodal problems require larger mutation rates. Also, Bäck (1993) criticised the universal rate as being too independent and unaware of the actual problem’s fitness landscape. Of course, the above concerns are direct calls for some form of adaptation, an issue this thesis examined at length in Chapter 7.

Summing up, the crossover and mutation operators remain crucial sources of variation in EC. If appropriately utilised, they can aid exploration and reachability of search spaces in both simple problems and those having rough and complicated landscapes, in so doing the variation operators improve the average fitness of a search pool over generations. As emphasised by Bassett et al. (2004), it is not the average individuals that drive the evolution forward but the exceptional individuals occasionally created by crossover or mutation.

\subsection*{2.6.5 Replacement Strategies and Elitism}

The field of evolutionary computation as stated in the introduction to this chapter has three major sub-fields. These include genetic algorithms, evolutionary strategies and evolutionary programming. The standard replacement strategy in genetic algorithms recommended by Holland (1975) was the generational replacement scheme. This scheme mimics the natural evolution in such a way that subsequent to a reproduction phase, the new samples (offspring population) completely replace the parent population. This strategy is called ($\lambda, \mu$) replacement scheme in the ES and EP communities; in addition to ($\lambda, \mu$), the ($\lambda + \mu$) strategy selects the best samples from a combine set of parent and offspring populations such that the original population size ($N$) is maintained. Nevertheless, there seems to be no demarcation in the way the replacement schemes are used across the different EC communities at the moment.

Although the generational replacement method is closer to the notion of natural evolution, it risks losing the best solution across generations due to occasional disruptions in the samples caused by crossover and mutation. This issue negates the actual optimization purpose where the ultimate goal is to seek and preserve

\textsuperscript{13}Pseudo-unimodal problems are multimodal problems having a global convex topology, example is the well known Griewank benchmark function (Whitley et al., 1996).
the best solution found so far\textsuperscript{14}. Realisation of the risk of premature lost of high quality solutions has led to the proposals of elitism (De Jong, 1975) and its variants (Greenhalgh and Marshall, 2000; Reeves and Rowe, 2004).

Elitism is a replacement strategy originally proposed by De Jong (1975) after the critical analysis of the behavioural trend in simulated evolution based on several empirical experiments. The concept aimed at preserving the candidate solution having the best fitness value \((f^*)\) as population evolve over successive generations. Suppose that a previously seen population is \(P(t - 1)\), then the sample(s) in the set

\[
\{x(t - 1) \in P(t - 1) \mid f(x(t - 1)) = f^*(t - 1)\}
\] (2.21)

are called the elites. The elites are directly copied to the current population \(P(t)\) unless the current population contains similar or fitter samples in its set, i.e.,

\[
\exists \{x(t) \in P(t) \mid f(x(t)) \geq f^*(t - 1)\}.
\] (2.22)

Other variants of the elitist strategy quickly surface. A generalised categorisation of replacement schemes by Bäck et al. (1997) is:

i. Generational: Also called non-elitist simple GA (sGA) and corresponds to the ES’s \((\lambda, \mu)\) strategy.

ii. Elitist: This follows the standard elitism and mildly corresponds to the ES’s \((\lambda + \mu)\) strategy.

iii. Overlapping: This generalises other variants of the elitist strategy and can simply be represented as \((\lambda, k_o, \mu)\) strategy where \(k_o\) is an ageing parameter \(1 \leq k_o \leq \infty\) signifying an individual’s life span, which is the maximum number of generation an individual (or sample solution) can survive.

The following section presents an approach that introduces adaptation into the concept of elitism. This proposal allows an EA to adapt the proportion of sample solutions to be used as elites during the evolution.

\textsuperscript{14}The challenges associated with the choice of replacement strategy are also in line with the ongoing debate on how biologically plausible should simulated evolution be (Section 2.2).
2.6.6 Adapting Elitism in EAs

As is reported in the following, the EC literature has seen several investigations which, empirically and theoretically, report on the effects of elitism on various types of EAs. Although many implementations of elitism still involve inheriting a single best solution from the previous generation as originally suggested by De Jong (1975), there have been several proposals which suggest utilising an elite pool of any size (Chambers, 1995; Reeves and Rowe, 2004). The proportion of the elite pool relative to the main population size is often called a generation gap in the ES terminology and steady-state EAs, or an elite fraction in GP.

As reported by Poli et al. (2008), the EC literature has witnessed several implementations of elitism which use a range of generation gaps (elite-fractions) from 0% up to 20% of the total population size. In an extensive study on the effect of elitism on GP, Poli et al. (2008) examine the effect of varying the elite-fractions (from 0 to 50% of total pool size) on the success rate and mean best fitness in the search pool. On a variety of test problem categories, the results in Poli et al. (2008) show that utilising some percentage of elite-fraction (such as 1 to 5%) generally improves performance in GP. They also concluded that, while elitism avoids the risk of premature lost of best solution due to stochastic sampling error (especially with small-sized pools), it also increases success rates when large population sizes are utilised.

On multi-objective EAs, Aguirre and Tanaka (2005) studied the effect of elitist selection by comparing the original NSGA-II ($\lambda + \mu$) against its ($\lambda, \mu$) version. They observe that lack of elitism in NSGA-II ($\lambda + \mu$) seems to severely affect its overall performance as compared to the elitist ($\lambda, \mu$) version. However, their results also indicate that excessive elitism can affect the efficacy of the search process. In particular, Aguirre and Tanaka (2005) noted that the presence of elitism tends to increase selection pressure making elitist algorithms more prone to the effects of genetic drift – a phenomenon that is known to affect negatively the performance of EAs.

Similarly, an empirical study by Laumanns et al. (2001) examines the impact of elitism relative to mutation rate. As a key finding, Laumanns et al. (2001) reported that best overall performance can be reached with strong elitism in combination with high mutation rates. They suggested that the high mutation rate is essential to overcome the risk of diversity collapse following the increase in selection pressure due to elitism. Another related study which examines the
effect of elitism on a GA can be found in Ishibuchi et al. (2008).

Importantly, there are already some proposals that suggest dynamically adjusting the size of the elite pool as the evolution progresses. Lee and Moon (2009) proposed a method that quantifies the fitness change in a non-stationary landscape and adapts the size of the elite population accordingly. Lee and Moon (2009) reported performance improvement, on the onemax problem, over the well-known random immigrants method. Similarly, Leung and Liang (2003) introduced a distance threshold parameter which is used to control the size of the elite by adjusting the population size according to the number of multiple optima in a problem. In comparison to other EAs, they found that the GA with adaptive elitism was effective in finding multiple optima in a multimodal landscape.

Based on the insights derived from the preceding investigations on the impacts of elitism on EAs it has become clear that, on one hand, retaining only a single elite individual (standard elitism) may limit the influence of elitism on the population’s fitness growth, i.e. a scalability problem. On the other hand increasing the size of the elite population could lead to rapid loss of diversity in the evolutionary pool. This is due to the domination of the new samples by the elites. Nevertheless, from the above reviews, performance improvement have been reported when the size of the elite samples is dynamically adjusted during the evolution – which is the root of the notion of adapting elitism in EAs.

As the benefits of adaptive elitism have been already demonstrated, this thesis utilises a variant of this strategy. In order to keep this chapter’s theme as introductory as possible, this section only introduces the adaptive elitist model as it underpins the replacement scheme used in all the EC models proposed later in this thesis. Unless otherwise stated, all future references to standard EC model in this thesis refer to the EA in Algorithm 2.2.

As is described in the following, the adaptive elitism is principally an overlapping replacement method that dynamically adjusts the size of its generation gap parameter ($\omega$) across generations. Assuming a maximisation problem, at every $t$th generation, the elite pool $P_{Elite}$ consists of a set of sample solutions having fitness values equal or approaching that of the best solution point $x^*(t)$, i.e.:

$$P_{Elite} = \{x(t) \in P(t) \mid f(x(t)) \rightarrow f(x^*(t))\}.$$  \hspace{1cm} (2.23)

During the initialisation stages of the evolution, see Algorithm 2.2 (line 3), the size of the elite pool is $|P_{Elite}| = N_{Elite}$ which is set to ($\omega\%$) of the total
Algorithm 2.2 Standard EA with Adaptive Elitism Replacement Strategy

1: \( t \leftarrow 0 \)
2: \( N \leftarrow |P(t)| \)
3: \( N_{\text{Elite}} \leftarrow \lceil \omega \% \text{ of } N \rceil \) // note that \( N_{\text{Elite}}, \omega \in \mathbb{Z}^+ \) and \( \omega \) is user defined
4: initialise \( P(t) \)
5: \( f_P(t) \leftarrow \) evaluate and rank \( P(t) \)
6: \( \bar{f}_P(t) \leftarrow \text{average of } f_P(t) \)
7: \( \text{Var}f_P(t) \leftarrow \text{variance of } f_P(t) \)
8: \( \text{while not termination do} \)
9: \( Q(t) \leftarrow \text{evolve } P(t) \)
10: \( f_Q(t) \leftarrow \text{evaluate and rank } Q(t) \)
11: \( \bar{f}_Q(t) \leftarrow \text{average of } f_Q(t) \)
12: \( \text{Var}f_Q(t) \leftarrow \text{variance of } f_Q(t) \)
13: \( \text{if } \bar{f}_Q(t) > \bar{f}_P(t) \text{ and } \text{Var}f_Q(t) > \text{Var}f_P(t) \text{ then} \)
14: \( \omega \leftarrow \frac{1}{2} \omega \) // shrink the generation gap
15: \( N_{\text{Elite}} \leftarrow \lceil \omega \% \text{ of } N \rceil \) // evaluate the new size of elite pool
16: \( \text{end if} \)
17: \( P_{\text{Elite}}(t) \leftarrow \text{Top } N_{\text{Elite}} \text{ of } P(t) \)
18: \( P(t+1) \leftarrow Q(t) \cup P_{\text{Elite}}(t) \)
19: \( t \leftarrow t + 1 \)
20: \( \text{end while} \)

population size \( N \); the number of elites \( (N_{\text{Elite}}) \) remains fixed if:

i. the new pool’s average fitness has not improved, i.e., \( \bar{f}(t) \leq \bar{f}(t-1) \), and;

ii. the fitness variance in the new pool has not fallen, i.e., \( \text{Var}f(t) \geq \text{Var}f(t-1) \).

However, \( N_{\text{Elite}} \) gets continuously halved at the end of every generation if the above two conditions are not satisfied, i.e., when there is a growth in the population’s average fitness and/or the fitness variance in the pool shrinks (lines 13-16).

After the initialisation stages (lines 1-4), the initial population is evaluated and ranked based on the fitness of the candidate solutions (line 5). Lines (6-7) compute and store the average fitness and fitness variance of the current population. Thereafter, the evolution process (line 9) which includes the selection, crossover and mutation continues iteratively until some termination condition is satisfied (lines 8-20). Notice that the elite population \( (P_{\text{Elite}}) \) constitutes the top best \( (N_{\text{Elite}}) \) individuals in the parent pool (line 17).

Note that the choice of generation gap of \( (\omega = 5\%) \) was empirical. Also, since a strict binary tournament selection is adopted, the successive halving of \( \omega \) (line 14) never stops the best individual from surviving across generations.
2.7 Summary

This chapter has overviewed the rationale behind the increasing acceptance of stochastic optimization approaches over their traditional (numerical) counterparts when solving global optimization problems. Key initialisation aspects such as data structure encoding, creation and sizing of the initial population are highlighted. Major parameterisation issues and evolutionary building blocks such as the choice of selection scheme, the variation operators: crossover and mutation and their probabilities, and replacement strategies are examined. Finally, the chapter reviewed the elitists replacement scheme and presented its variant which adapts the size of the elites based on the fitness variance in the search pool. Note that although the adaptive elitism presented herein utilises new heuristics to adjust the size of the elite samples, its overall significance as to the theme of this thesis is secondary.

The many insights derived from the investigations in this chapter are vital and pave the way for deeper investigations into the core aspects of evolutionary optimization (Chapter 4 and 5). Furthermore, the insights will eventually aid the design and development of effective hybrid EAs in Part II of this thesis. The next chapter focuses on the deterministic optimization methodologies with emphasis on gradient-based algorithms for local optimization.
Chapter 3

Deterministic Optimization Methods

The preceding chapter focused mainly on the background aspect of evolutionary computation methods. Such algorithms are *approximate* global search heuristics where success relies upon some stochastic heuristics. As a result they lack rigorous theoretical convergence guarantees. While the key objective of this study is to propose an effective approach for solving global optimization problems, global optimization in itself is studied across several different communities. Therefore, the hybrid model proposed in Part II of this thesis combines algorithms with features from evolutionary computation (EC) which are *stochastic* heuristic approaches, and mathematical programming (MP) techniques which are *deterministic* numerical methods. Whilst the EC and MP communities share the same objective, making the fastest possible progress towards the best possible solution, the MP approaches are often characterised with more theoretically rigorous properties. Notably, the field of numerical *global optimization* predates that of evolutionary computation, which essentially surfaces after around the 1950s to 1970s.

There are two main methods to numerical global optimization: (i) the discretisation approaches, and (ii) convex relaxation (also called convex underestimation) approaches. On one hand, for any given nonlinear problem (NLP) that is at least twice differentiable convex relaxation approaches relax the NLP by adding a suitable quadratic term. The idea is to enforce convexity over the entire search space of interest (Chachuat and Latifi, 2003). In principle, such approaches require that the Hessians for the combined quadratic terms and the original NLP are at least positive semi-definite. There are various methods for
ensuring convexity in such relaxed problems. One commonly used technique is the $\alpha$BB relaxation (Androulakis et al., 1995) in which the nonconvexity of the original NLP is overcome by appropriately setting the weighting parameters ($\alpha$) of the added quadratic terms. The discretisation methods, on the other hand, typically involve several concepts such as the branch-and-bound, interval, and cutting-plane methodologies.

Since the focus of the overviews in this chapter is on the deterministic numerical algorithms for local optimization, a deep overview on numerical global optimization methods is beyond the scope of this thesis. For extensive surveys of such methods the reader is referred to Floudas and Pardalos (2003); Floudas and Gounaris (2009); Papamichail and Adjiman (2002); Pintér (1995).

Firstly, this chapter overviews in a broad scope the deterministic numerical algorithms for local optimization (Section 3.1). It then introduces the line search based gradient methods in Section 3.2 and reviews their methods of evaluating search directions and step sizes, and convergence rates in Sections 3.3 and 3.4 respectively. Section 3.5 presents the Newton based sequential quadratic programming (SQP) algorithm which utilises interior point method (IPM) to solve its quadratic programming subproblems (Section 3.6). Finally, Section 3.7 presents an improved automatic differentiation technique that facilitates evaluation of accurate derivatives for the SQP algorithm.

3.1 Local Optimization – Background

Local optimization is concerned with searching for a solution that is optimal (either maximal or minimal) within a neighbouring set of solutions. A local search algorithm starts from an initial solution point and then iteratively moves to a neighbour solution. The neighbourhood relation is defined based on the search space; since every candidate solution has more than one neighbour, the choice of which one to move to is taken using only information about the solutions in the neighbourhood of the current one, hence the name local search or local optimization. The choice of the neighbour solution is essentially based on some maximisation or minimisation criterion. Thus, when no improving configurations are present in the neighbourhood, the search process is considered to be stuck at a locally optimal point called the local optimum solution.

There are at least two key categories of local optimization approaches, these
include: i) the classical mathematical programming (MP) based numerical methods; and ii) the many heuristic based deterministic approaches. Examples of the heuristic based deterministic methods include interval search, branch and bound, cutting plane methods etc. But as highlighted above, the focus in this thesis is on the gradient based mathematical programming methods which although have their own issues\footnote{Gradient based methods require direct/indirect evaluation of derivatives and, like with any local search methods, the final solution is strongly dependent on the chosen starting solution.} possess sound theoretical convergence characteristics.

Since this study is limited to \textit{continuous} optimization problems (see Section 1.4), the objective function to be optimized $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is defined in terms of a vector of the design variables $x$ of length $n$ that is in the set of real numbers $\mathbb{R}$. Suppose that a general expression for an $n$-dimensional continuous optimization problem is:

$$\text{Maximise: } f(x) : x \in \mathbb{R}^n; \ n \geq 1; \quad (3.1)$$

then, gradient based optimization algorithms sequentially generate, at every iteration $t$, a vector of new solution points $x_t$. Eventually, $x_t$ is expected to terminate at $x^\ast$ (i.e the optimal solution) when either no more progress can be made or when the optimal solution has been attained with sufficient accuracy. Beginning with an arbitrarily chosen initial solution point $x_0$, the iterative progression from any point $x_t$ to $x_{t+1}$ depends on: (i) the nature of the gradient of the objective function $f$ at the current solution point $x_t$; and (ii) some additional information about the previously visited solutions $x_{t-1}, x_{t-2}, \ldots, x_0$.

Typically, every new solution point $x_{t+1}$ is expected to yield a better function value than its predecessor $x_t$. In fact, a critical distinction between the two classes of local optimization methods is on the nature of their successive iterations. A class of algorithms that insists on improvement in the function value at every iteration enforces $f(x_t) > f(x_{t-1})$, and constitutes the so-called \textit{greedy algorithms} (Bhatti, 2000). The Newton and quasi-Newton algorithms are part of this class. The other class that do not insist on improving the value of the objective function at the end of \textit{every} iteration usually enforces $f(x_t) > f(x_{t-j})$, where $j > 1$ is the maximum acceptable iterations without improving the objective function value. This is the class of \textit{non-monotone} algorithms which are non-greedy in nature.

Another major categorisation of the gradient based algorithms is based on their approach for advancing from one iteration point $x_t$ to the next $x_{t+1}$. The
CHAPTER 3. DETERMINISTIC OPTIMIZATION METHOD

The general expression for deriving the next iteration point \( x_{t+1} \) is dependent on two key parameters; the evaluated search direction \( d_t \in \mathbb{R}^n \) and an estimated value for the step length parameter \( \alpha_t : \{ \alpha_t \in \mathbb{R} \mid 0 < \alpha_t \leq 1 \} \) along the obtained direction such that:

\[
x_{t+1} = x_t + \alpha_t d_t.
\]

At every iteration, line search based algorithms first evaluate a search direction \( d_t \). Then, by estimating a suitable value for the step length parameter \( \alpha_t \), they decide on how long to search along that direction, i.e. the step size. On the contrary, trust region based algorithms start by defining a region around the current solution point \( x_t \). Within the defined neighbourhood, trust region methods assume that a model derived from a second order Taylor approximation of the objective function (see equation (3.4)) provides a good approximation of the actual objective function. Based on the size of the defined trust region, these algorithms then choose the search direction and the step size simultaneously.

It is important to note that for the line search based methods, if at any iteration the evaluated step length parameter \( \alpha_t \) does not lead to improvement in the value of the objective function, the line search simply tries to re-evaluate a feasible one. However, in the case of the trust region methodology both the search direction and the step length must be discarded and the size of the trust region must be contracted and the procedure repeated. This among other reasons made the line search based optimization algorithms computationally cheaper than their trust region counterparts. Nevertheless, for starting points that are at significant distance from the local optimum the trust region methods can be more reliable (Nocedal and Wright, 2006). The sequential quadratic programming (SQP) local search algorithm, introduced in Section 3.5, is in the category of line search methods. Therefore, the next few sections focus on the line search based local optimization techniques.

3.2 Line Search based methods

As highlighted earlier, the iterative progression for all line search based methods relied on the computed search direction \( d_t \) and the evaluated step length \( \alpha_t \). However, line search methods principally differ in the mode of computing their
search directions. Suppose that $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a smooth continuously differentiable function. Let $d \in \mathbb{R}^n$ be a vector of search direction in the neighbourhood of iteration point $x$. Then, for $x^*$ to be a local optimum of the function $f$ the necessary and sufficient conditions for optimality requires: $\nabla f(x^*)$ to be zero, and $\nabla^2 f(x^*) \in \mathbb{R}^{n \times n}$ to be a symmetric positive definite matrix. The Taylor expansion of $f$ yields:

$$f(x + d) = f(x) + \nabla f(x)^T d + \frac{1}{2} d^T \nabla^2 f(x) d + \cdots. \quad (3.3)$$

Gradient based methods assume that the objective function $f$ is differentiable and that it is approximately quadratic (i.e. convex) in the vicinity of the stationary point $x$. Therefore, the second-order Taylor approximation of (3.3) is:

$$f(x + d) \approx f(x) + \nabla f(x)^T d. \quad (3.4)$$

Setting the gradient of the approximation in (3.4) to zero at stationary point, and solving for the search direction $d$ yields:

$$d = - \frac{\nabla f(x)}{\nabla^2 f(x)^T} = - [H(x_k)]^{-1} \nabla f(x), \quad (3.5)$$

where $H(x) = \nabla^2 f(x)$ is the Hessian of the function $f$ and $d$ is the search direction which is required to be a direction of descent/ascent.

### 3.3 Search Directions and Step Sizes in gradient-based algorithms

Any step of the gradient based algorithms involves evaluation of the search direction $d_t$ and the step length parameter $\alpha_t$ such that the next iteration is defined as in equation (3.2) above. The following examines the distinctions in evaluations of the search directions and step sizes for various types of gradient based algorithms.

#### 3.3.1 Evaluating Search Directions

To sustain progress during the optimization process, gradient based algorithms generally utilise a search direction $d_t$ which at any given iteration $t$ is required to
be a direction of descent/ascent. This allows the directional derivative,
\[ d_t^T \nabla f(x) < 0, \]  
(3.6)
to improve the value of the objective function \( f \) along the chosen direction. From (3.5), a general form for the search directions \( d_t \) is:
\[ d_t = -B_t^{-1} \nabla f(x), \]  
(3.7)
where \( B_t \in \mathbb{R}^{n \times n} \) is a symmetric non-singular matrix. The most commonly used types of search directions for local optimization algorithms are as follows:

i. **Steepest Descent/Ascent Direction**: The search direction defined as \( d_t = -\nabla f(x) \) is called the *steepest descent/ascent direction*. For all the possible directions via which the search could move from point \( x_t \) to \( x_{t+1} \), this is the direction along which \( f \) improves most rapidly. Therefore, *steepest ascent algorithms* are line search methods that move along the steepest direction at every iteration. Notice that in this case the Hessian matrix in equation (3.7) is set to an identity matrix (i.e., \( B_t = I_n \)). Thus, the primary advantage of these algorithms is that they require only the computation of the gradient of the objective function; hence, they have low computational cost per iteration.

ii. **Newton Direction**: Another important search direction is the *Newton direction*. Derived from the second-order Taylor expansion of \( f \) (see equation (3.4)), the value of \( B_t = \nabla^2 f(x) \) in equation (3.7) is the true Hessian of the objective function. The Newton direction is a reliable direction of ascent when the Hessian \( \nabla^2 f(x) \) is sufficiently smooth so that the quadratic approximation of the objective function in (3.4) is sufficiently accurate. Thus, *Newton methods* are the algorithms that use Newton directions at every iteration with the condition that the Hessian \( \nabla^2 f(x) \) is symmetric and positive definite\(^2\). These methods typically have fastest convergence rate compared to all other line search based local optimization techniques. Moreover, when the current solution point, \( x_t \), is within the neighbourhood of the optimum solution quite a few iterations are required to converge to the solution point.

\(^2\)A positive definite matrix must have a positive determinant, i.e., it is always non-singular. Thus, a real and symmetric matrix is positive definite if all its eigenvalues are positive. It is positive semidefinite, negative semidefinite or negative definite if all of its eigenvalues are non-negative, non-positive or negative respectively.
CHAPTER 3. DETERMINISTIC OPTIMIZATION METHOD

with high accuracy.

The main drawback of Newton methods is the cost of Hessian \( \nabla^2 f(x) \) evaluation; the true Hessian is a matrix of second derivatives and its evaluation can be quite cumbersome, error prone and expensive especially for higher dimensional problems. Furthermore, whenever \( B_t = \nabla^2 f(x) \) is not positive definite, the Newton direction may not be defined since the inverse of \( B_t \) may be ill-conditioned and consequently yields a search direction \( d_t \) that violates the descent/ascent property in (3.6).

iii. Quasi-Newton Direction: To circumvent some of the complexities associated with direct computation of Newton directions, line search approaches that use quasi-Newton directions are a commonplace. In such methods, instead of evaluating the exact value of \( B_t \) (3.7), it is only set to an approximation of the true Hessian \( \nabla^2 f(x_t) \) matrix. The initial approximation of the Hessian \( (B_t) \) is usually an identity matrix \( I \), which is then updated iteratively to take into account the additional information derived from subsequent iterations. The Hessian update methods mainly rely on the fact that delta changes in the gradient of \( f \) from one iteration point to another provide vital information about the nature of its second derivative along the search direction. The algorithms that rely on this principle of Hessian update are called quasi-Newton algorithms and can achieve high rate of convergence (superlinear) without the expensive explicit evaluation of the Hessian matrix.

However, the drawback of quasi-Newton methods is that their rate of convergence is slower than that of Newton methods as they require running through several iterations. Also, after certain number of steps, the Hessian approximation often yields an ill-conditioned matrix that may cause the entire search process to diverge. This is usually tackled by resetting the Hessian update \( (B_t) \) to its initial value \( (B_t = I) \), but doing this significantly retards optimization progress. For the majority of the popular Hessian update techniques such as BFGS, DFP and SR1 (Bhatti, 2000; Powell, 1986), the initial approximation of \( B_t \) is an identity matrix \( I \). Thus, the search direction is always downgraded to a steepest ascent one after every reset of the Hessian.

iv. Conjugate Gradient Direction: This category of search directions are typically more effective than steepest ascent directions and are fairly computationally equivalent (Sherali and Ulular, 1990). The methods that use this
search direction are called conjugate gradient methods. Note that while the conjugate gradient methods utilise steepest direction in their first iteration, they use conjugate directions instead of the local gradient for subsequent iterations. Also, unlike in the Newton and quasi-Newton directions, conjugate direction requires no computations or storage of large Hessian matrices. Nevertheless, these methods are only first order and therefore do not achieve fast convergence rate as Newton and quasi-Newton methods. As argued by Powell (1977, 1984), conjugate gradient methods may fail to converge on non-convex problems and therefore need restart.

### 3.3.2 Evaluating Step Sizes

The step sizes taken at every stage of an optimization process are estimated via a step length parameter $\alpha \in \mathbb{R} \mid (0 < \alpha \leq 1)$. $\alpha$ is a positive scalar that is expected to substantially improve the function value. It is normally the optimizer of the merit function\(^3\) defined as:

$$
\varphi(\alpha) = f(x_t + \alpha_t d_t) : \alpha > 0.
$$

(3.8)

Thus, the step size $\alpha$ is

$$
\alpha := \arg \max_{\alpha} f(x_t + \alpha_t d_t).
$$

(3.9)

Generally, the ideal value for the step length parameter is the global optimizer of the merit function (3.8) itself. Determination of this requires several evaluations of the objective function $f$ and its gradient $\nabla f$. Therefore, since the evaluation of a local optimizer to the merit function (3.8) is in itself expensive, a trade-off is necessary. Thus, at every iteration $t$, inexact line search algorithms are used to try out a sequence of candidate values for $\alpha_t$. Based on some pre-defined termination conditions, a suitable value for $\alpha_t$ is accepted. A simple condition that ensures $\alpha_t$ provides a meaningful improvement in $f$ entails:

$$
f(x_t + \alpha_t d_t) > f(x_t).
$$

(3.10)

---

\(^3\)Originally used in regression, a merit function is a function that measures the agreement between data and the fitting model for a particular choice of the parameters (Weisstein, 2011). Parameters are adjusted based on the value of the merit function until a smallest value is obtained, the resulting parameters are known as the best-fit parameters.
Note that the step length $\alpha$ needs not to lie near the optimum of the merit function $\varphi(\alpha)$ for it to effectively yield a sufficient increase in the objective function.

Several inexact line search algorithms for estimating a suitable value for the step length parameter exist. The methods are usually named after the termination condition used. Thus, they include the method based on Armijo’s rule, Goldstein condition and the popular Wolf conditions (Bhatti, 2000). All these algorithms typically possess two key stages. The first stage is a backtracking procedure that finds an interval containing the desirable step lengths, while the second is an interpolation phase where a good value for $\alpha$ is computed within the obtained interval.

A commonly used termination criterion utilises the two Wolf conditions. Based upon this, it has been proven (Al-baali, 1985) that to every smooth continuous function there exist a value for the step length parameter that satisfies the following two Wolf conditions.

**First:** The step length $\alpha$ must provide sufficient increase in $f$ such that:

$$f(x_t + \alpha_t d_t) \geq f(x_t) + c_1 \alpha_t \nabla f^T_t d_t,$$

where $c_1 \in [0, 1]$ is a positive constant which is small in practice, typically $c_1 = 10^{-4}$. Thus, the improvement in $f$ should be proportional to both the value of $\alpha_t$ and the directional derivative $\nabla f^T_t d_t$.

**Second:** This is called the curvature condition, it ensures that the derivative of the merit function $\varphi(\alpha_t)$ (3.8) is greater than a constant $c_2$ times the derivative of $\varphi(0)$, i.e.:

$$\varphi'(\alpha_t) \geq c_2 \varphi'(0),$$

such that the new directional derivative satisfies:

$$\nabla f(x_t + \alpha_t d_t)^T d_t \geq c_2 \nabla f^T_t d_t,$$

where $c_2 \in [c_1, 1]$, its typical value for Newton and quasi-Newton methods is 0.9 and 0.1 for nonlinear conjugate gradient methods. For more details on these and other techniques for evaluating the step length parameter, see Al-baali (1985); Bashir and Ximing (2011); Betts (2001); Fletcher (1987); Hertog (1994).
3.4 Convergence Analysis in gradient-based algorithms

Convergence assessment is critical to evaluating the performance of any optimization algorithm. This section elaborates the rate of convergence and local convergence of gradient based algorithms. It compares the gradient based algorithms on the basis of their convergence characteristics.

While a number of measures exist for evaluating the rate of convergence of gradient based algorithms, a commonly used one is the Q-convergence (Q stands for quotient) measure which is defined in terms of the quotient of successive errors. Suppose that a sequence of solution points \( \{x_t\} : x \in \mathbb{R} \) iteratively converges to an optimum \( x^* \), then, the rate of convergence of gradient based algorithms is classified as follows.

**Q-linear:** The convergence is Q-linear if there exist a constant \( r \in [0, 1] \) such that:

\[
\frac{||x_{t+1} - x^*||}{||x_t - x^*||} \leq r; \quad \forall \ t \text{ sufficiently large.} \tag{3.14}
\]

**Q-superlinear:** An algorithm is said to converge Q-superlinearly if as iterations \( t \) tend to infinity, the error between two successive iteration points decays.

\[
\lim_{t \to \infty} \frac{||x_{t+1} - x^*||}{||x_t - x^*||} \to 0. \tag{3.15}
\]

**Q-quadratic:** These are algorithms which for any scalar \( M \in \mathbb{R} : M > 0 \), their rate of convergence satisfies:

\[
\frac{||x_{t+1} - x^*||}{||x_t - x^*||^2} \leq M; \quad \forall \ t \text{ sufficiently large.} \tag{3.16}
\]

Notice that higher order rate of convergence are also possible and the trend is such that a Q-quadratic algorithm will always converge faster than a Q-superlinear or Q-linear algorithm. While all Newton algorithms converge Q-quadratically, quasi-Newton methods converge Q-superlinearly and at the other extreme, all steepest ascent algorithms have Q-linear rate of convergence.

Conversely, the local convergence of gradient based algorithms requires not only a suitable estimate of the step length parameter, but also a carefully chosen
Table 3.1: Comparison of the three major gradient based algorithms: The nature of their search directions, step length parameter and rate of convergence.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Search direction</th>
<th>Hessian requirement</th>
<th>Step length parameter</th>
<th>Degree of computation</th>
<th>Rate of convergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steepest Descent</td>
<td>Not needed: $B_k = I$</td>
<td>$0 &lt; \alpha &lt; 1$</td>
<td>1st Order</td>
<td>$Q$-linear</td>
<td></td>
</tr>
<tr>
<td>Conjugate Gradient</td>
<td>Not needed: $B_k = I$</td>
<td>$0 &lt; \alpha &lt; 1$</td>
<td>1st Order</td>
<td>$Q$-linear, generally faster</td>
<td></td>
</tr>
<tr>
<td>Quasi-Newton</td>
<td>$-B_k^{-1} \nabla f(x_k)$</td>
<td>Approximation: $B_k \approx \nabla^2 f(x_k)$</td>
<td>$0 &lt; \alpha &lt; 1$</td>
<td>2nd Order</td>
<td>$Q$-superlinear</td>
</tr>
<tr>
<td>Newton method</td>
<td>Exact Hessian: $B_k = \nabla^2 f(x_k)$</td>
<td>$0 &lt; \alpha \leq 1$</td>
<td>2nd Order</td>
<td>$Q$-quadratic</td>
<td></td>
</tr>
</tbody>
</table>

search direction. In order to have good convergence characteristics, the evaluated search direction must not be orthogonal to the gradient $\nabla f$, i.e. at least steepest ascent steps must be taken regularly.

One undesirable behaviour associated with the general gradient based methods is that while the steepest ascent algorithms have good local convergence, their rate of convergence is slowest (only $Q$-linear). Also, while the Newton algorithms converge most rapidly (i.e., $Q$-quadratically), their local convergence is not guaranteed when the initial search point is not in the vicinity of the optimum solution. This is because when the search point is far away from the optimum Newton methods tend to yield search directions that are nearly orthogonal to the gradient of the objective function $\nabla f$; thus, cannot lead to any improvement in $f$. Therefore, whenever the starting point is not guaranteed to be near the optimum, a trade-off is necessary to ensure local convergence and high rate of convergence of gradient based algorithms.

Table 3.1 summarises the key features for the above major categories of gradient based algorithms. It can be deduced, from the comparison in Table 3.1, that the Newton based method is suitable for the hybrid optimization model proposed in Part II of this thesis (Chapter 6). This is because the initial starting point fed to this local algorithm is almost always in the vicinity of (sub-)optimal solution point. This is true given that the starting point is obtained from a sufficiently converged global optimization (EC) algorithm. Consequently, local convergence
of the Newton based method is assured. The next section presents the Newton-based sequential quadratic programming local search method.

3.5 The Sequential Quadratic Programming Algorithm

The sequential quadratic programming (SQP) algorithm is a Newton based optimization method that can be implemented either in the line search or trust region framework. SQP is useful in deriving locally optimal solutions for constrained nonlinear problems (NLP). A feature that is common to all SQP formulations is that the algorithms are divided into an outer linearisation and an inner optimization loop. The linearisation loop approximates the nonlinear objective function \( f(x) \) in the original optimization problem (see equation (3.1)) with a quadratic model; and the nonlinear constraints (if any) with their approximate linear expressions at the current solution point \( x_t \). The result of this linearisation allows representation of the original nonlinear problem (3.1) as a sequence of quadratic programming (QP) subproblems as in equation (3.17). The QP subproblems are then optimized using any QP solver.

\[
\max_{x \in \mathbb{R}^n} f(x)^T x + \frac{1}{2} x^T H x : H \in \mathbb{R}^{n \times n}. \tag{3.17}
\]

The framework of SQP is based on Newton methods which solve nonlinear problems via a sequence of Newton steps, hence the name sequential quadratic programming. For extensive treatment on SQP algorithm see Fletcher (1987); Nocedal and Wright (2006).

The standard SQP design uses the active set strategy (Gill and Robinson, 2011) which is based on the null-space and range-space methods to solve its QP subproblems. In contrast, the method adopted here is based on the earlier work in Bashir and Ximing (2011); Ximing et al. (2008) where an interior point method (see Section 3.6) is deployed to optimize the QP subproblems.
3.6 Interior Point Method for solving Quadratic Problems

Interior point methods (IPMs) have their origin from linear programming (LP). From the late 80s IPMs become important tools in mathematical programming, operations research and in many other areas of science. The main idea behind IPM algorithms is to approach the optimal solution of an LP problem through the interior of the feasible region. This is the opposite of the strategy employed by the well known simplex algorithm (Dantzig, 1963, 1961) which moves along the boundary of the feasible region. A lot of research has been reported on IPM algorithm and its variants, many of which are surveyed and referenced by Mizuno et al. (1995) and Wright (1997). Interior point methods have been successfully applied on general convex problems, including quadratic programming problems (Hertog, 1994; Hoppe et al., 2002; Nesterov and Nemirovskii, 1994).

Unlike the standard SQP wherein derivatives approximations are utilised, the true performance benefits for the IPMs are best seen when accurate second derivatives are provided. To curb the cost of derivative evaluation (see Section 3.7 for Automatic Differentiation methods), most IPMs exploit solvers designed for modern computer architectures. They are shown to efficiently solve a sequence of systems with fixed structure like the QP problem (3.17) (Gill and Robinson, 2011, 2012).

In this work, an IPM algorithm is employed to optimize the QP subproblems obtained by linearisation of the original nonlinear problem by the main SQP algorithm. Thus, an IPM algorithm will form the inner (optimization) loop of the proposed SQP model.

Suppose that a general quadratic programming problem is defined as:

$$\max_x q(x) = c^T x + \frac{1}{2} x^T Q x; \quad x \in \mathbb{R}^n, \quad c \in \mathbb{R}^n, \quad Q \in \mathbb{R}^{n \times n},$$

(3.18)

then, the function $q(x)$ (3.18) is convex if at least the matrix of quadratic terms $Q$ is symmetric positive semidefinite.

If the function to be optimized is subject to some constraints $g$, say

$$g_i(x) \geq 0 : x \in \mathbb{R}^n, \quad i = 1, 2, \ldots, m, \quad \nabla g \neq 0,$$

(3.19)

then, IPMs use the method of Lagrange multipliers to redefine the optimization
problem into a composite function $\mathcal{L}$ called Lagrange function (named after its inventor Joseph Louis Lagrange (Arfken and Weber, 2005)). The Lagrange function combines the objective function (3.18) and all the constraints (3.19) such that:

$$
\mathcal{L}(x, \lambda) = c^T x + \frac{1}{2} x^T Q x + \sum_{i=1}^{m} \lambda^T g_i(x),
$$

(3.20)

where $\lambda \geq 0$ is a vector of Lagrange multipliers.

At every iteration of the SQP algorithm, the matrix of the quadratic terms, $Q$, is derived from the Hessian of the original nonlinear problem. $Q$ is then used to construct the QP subproblem (3.18) to be solved in the inner loop of the SQP algorithm. The maximum of (3.20) is then obtained by taking its gradient with respect to all its variables (i.e., $\nabla_{x,\lambda} \mathcal{L}(x, \lambda)$). This is done without explicitly inverting $g$, which is the reason why the method of Lagrange multipliers can be quite handy (Arfken and Weber, 2005). Hence, maximising the Lagrange function yields the maximum of the originally constrained NLP.

A point of particular interest here is that the design of the standard SQP algorithm mainly relies on Hessian approximations; a Hessian matrix is initially approximated by an identity matrix $I$ and then updated iteratively via the update formula proposed by Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Betts, 2001; Nocedal and Wright, 2006; Powell, 1986). Conversely, the proposal in this work is to develop a new automatic differentiation package, which utilises a vectorisation approach to algorithmically evaluate “exact” values for the gradients and Hessians of any differentiable function at any given solution point $x_t$. The cost of evaluating the derivatives is more or less equivalent to that of evaluating the function values. In this way the computational burden of evaluating derivatives via the conventional symbolic or finite difference methods is alleviated. The availability of accurate Hessians is beneficial to the IPM QP solver; it also benefits the proposed SQP algorithm (Algorithm 3.1) in the following ways:

i. The exact Hessians upgrade the convergence characteristics of the standard SQP algorithm from that of a quasi-Newton algorithm – that takes steepest ascent steps – to that of a Newton algorithm taking full\(^4\) Newton steps at

\(^4\)Full step will mean having the value of the step length parameter $\alpha = 1$. Quasi-Newton methods barely accept $\alpha = 1$ and even when they did, they still run at a superlinear rate (Powell, 1986).
Algorithm 3.1 The SQP local optimization algorithm

1: begin
2: \( t \leftarrow 0; \)
3: \( x_t \leftarrow x_0; \quad // \ x_0 \) is the starting point returned by the EC algorithm
4: \( d_t \leftarrow d_0; \quad // \ \text{Initial search direction} \ d_0 \) is a vector of all ones
5: while \( \nabla f(x_t) > \text{Tol}_1 \) and \( ||d_t|| > \text{Tol}_2 \) and \( t < \text{MaxIter} \) do
6:  linearise (3.1) into a QP subproblem (3.18)
7:  \( H(x_t) \leftarrow \nabla^2 f(x_t) \quad // \text{computed via Automatic Differentiation} (\S 3.7) \)
8:  evaluate \( d_t \) by minimising QP subproblem (3.18)
9:  if setting \( \alpha = 1 \) satisfies Wolf conditions (3.11) and (3.12) then
10:   set \( \alpha_t \leftarrow 1 \)
11: else
12:    compute \( \alpha_t \) from (3.9) that satisfies (3.11) and (3.12)
13: end if
14: \( x_{t+1} \leftarrow x_t + \alpha_t d_t; \)
15: \( t \leftarrow t + 1; \)
16: end while
17: end

every iteration. Hence, the convergence rate is enhanced from superlinear to quadratic.

ii. Also, the frequent reset of the search direction due to ill-conditioned approximate Hessians could be avoided or at least minimised\(^5\). This means that both local convergence and quadratic rate of convergence are assured. As a result, the proposed method would converge to the optimum in remarkably few steps.

The framework for the proposed SQP local optimization algorithm is depicted in Algorithm 3.1. Besides the initialisation of the SQP parameters (lines 2-4), the while loop (line 5) contains both the linearisation (line 6) and the optimization (line 8) steps and, exits only if one of the three stopping conditions is satisfied. In the termination conditions (line 5), the first two tolerance parameters, \( \text{Tol}_1 \) and \( \text{Tol}_2 \), check whether the delta changes in the gradient and directional derivative of the problem have sufficiently approached zero. Typical values are \( 10^{-6} \) to \( 10^{-3} \).

The third parameter (MaxIter) is the maximum iteration limit which is also user defined. Finally, notice that instead of using Hessian approximations (line 7), exact Hessians \( \nabla^2 f(x_t) \) derived from an automatic differentiation algorithm

\(^5\)Resetting the Hessian to an identity matrix \( I \) is common in BFGS and other update procedures. If exact Hessians are available and the starting point is in the neighbourhood of the optimum, then the problem of resetting search direction is eliminated.
(Section 3.7) are utilised. Having evaluated the search direction (line 8) and the step size (lines 9-13), a new solution point is derived from the current one (line 14). The process is repeated until one of the termination conditions (line 5) is reached.

The following section presents the detail principles of the automatic differentiation for exact Hessian evaluation.

### 3.7 Automatic Differentiation for exact Derivatives Evaluation

Also called algorithmic differentiation, for any differentiable function, automatic differentiation (AD) evaluates the derivatives via a sequence of basic elementary operations involving unary/binary operators and their operands. The need for AD arises because conventional methods like the method of divided differences are prone to round-off errors when the differencing interval is small, and suffer from truncation errors when the interval is large. On the other side, the computational cost of symbolic method is high especially when the function of interest is of higher dimensions.

Automatic differentiation works by repeated application of the chain rule (3.21). Suppose that \( f : \mathbb{R}^n \to \mathbb{R} \) is a composite function defined in terms of a vector \( h \in \mathbb{R}^m \) which is in turn a function of \( x \in \mathbb{R}^n \), according to chain rule the derivative of \( f \) with respect to \( x \) is:

\[
\nabla_x f(h(x)) = \sum_{i=1}^{m} \frac{\partial f}{\partial h_i} \nabla h_i(x).
\]  

(3.21)

Then, AD apply chain rule on a computational representation (tree-like graphs) of \( f \) to generate analytic values for the function and its derivatives.

Historically, the basic ideas of AD have been around for long time (Bücker et al., 2006; Rall, 1981, 1986; Werbos, 1989, 1994). However, it was the extensive work of Griewank and Walther (2008) that revived the interest in the use of algorithmic differentiation methods. Thereafter, a number of research work have been published\(^6\) on AD principles and its applications in mathematics and

\(^{\text{6}}\)More about AD can be found in the AD portal (Bücker and Schiller, 2000).
Table 3.2: A list of variables definitions for a 2-dimensional sample problem: 
\[ f(x_1, x_2) = (x_1 x_2 + \sin x_1 + 4) (3x_2^2 - 6). \]

<table>
<thead>
<tr>
<th>Variables</th>
<th>Vertices</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( v_{-4} )</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>( v_{-3} )</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>( v_{-2} )</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>( v_{-1} )</td>
<td>( x_1 )</td>
</tr>
<tr>
<td></td>
<td>( v_0 )</td>
<td>( x_2 )</td>
</tr>
<tr>
<td>Intermediate</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( v_1 )</td>
<td>( v_{-1} v_0 )</td>
</tr>
<tr>
<td></td>
<td>( v_2 )</td>
<td>( \sin v_{-1} )</td>
</tr>
<tr>
<td></td>
<td>( v_3 )</td>
<td>( v_1 + v_2 )</td>
</tr>
<tr>
<td></td>
<td>( v_4 )</td>
<td>( v_3 + v_{-4} )</td>
</tr>
<tr>
<td></td>
<td>( v_5 )</td>
<td>( v_0^2 )</td>
</tr>
<tr>
<td></td>
<td>( v_6 )</td>
<td>( v_5 v_{-3} )</td>
</tr>
<tr>
<td></td>
<td>( v_7 )</td>
<td>( v_6 - v_{-2} )</td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( v_8 )</td>
<td>( v_4 v_7 )</td>
</tr>
<tr>
<td></td>
<td>( v_8 )</td>
<td>( f )</td>
</tr>
</tbody>
</table>

machine learning by Christianson (1992a,b, 1999) and recently in Bartholomew-Biggs et al. (2000); Bischof et al. (2002); Forth (2006); Ghate and Giles (2007). The two basic modes via which AD yields the values and derivatives of any differentiable function are the forward and reverse modes. Details on these approaches can be found in Griewank and Walther (2008). However, as is described in the following sections, the approach adopted here is based on the vectorised forward AD developed by Bashir (2011) which utilises operator overloading to derive the function value, gradient and Hessians in a single forward sweep.

A. Graphical representation and Algorithmic evaluation of functions

Given a two dimensional differentiable function (3.22), evaluating its value at any given point \( x_t = [x_1, x_2] \) entails an orderly execution of the elementary operations that made up the function.

\[ f(x_1, x_2) = (x_1 x_2 + \sin x_1 + 4)(3x_2^2 - 6). \] (3.22)

The elementary operations may be assigned to some intermediate variables
that link the input variables to the output. Note that all the input, intermediate
and the output variables can be represented as vertices of a tree \( v_i \in \{v_a, v_b, \ldots \} \)
that represent the entire function. Table 3.2 shows the trace for the evaluation
of all the variables from the input to the output side for function (3.22).

Consider the tree-like graph (Figure 3.1) for function (3.22). First, the com-
puter interprets the function as a sequence of elementary operations on the work
variables\(^7\) mapped as vertices \( v_i \) such that \( i \leq 0 \) for the input variables and \( i > 0 \)
for the intermediate and output variables (Figure 3.1). The leaves of the graph
represent the input variables. The intermediate vertices stand for the interme-
diate variables and the top vertex is the output variable. According to graph
theory, the ordering require that for any vertices \( a \) and \( b \) linked with an arc from
\( b \) to \( a \), \( a \) is the parent vertex and \( b \) is the child. Therefore, the value of the child
vertex \( b \) must be evaluated before the parent vertex \( a \), i.e. the values of all the
children vertices must be obtained prior to evaluating that of their parent. Hence,
the overall function value can be obtained by evaluating the vertices in the graph
from leaves through the root in an orderly manner. Notice that many of the
elementary evaluations are executed in parallel, see the AD graph in Figure 3.1;
the hierarchy of the evaluations implies parallel execution, e.g., from \( \{v_2, v_1, v_5\} \),
to \( \{v_3, v_6\} \) and then \( \{v_4, v_7\} \), etc.

**B. Forward mode of AD**

The *forward mode* or *forward accumulation* evaluates derivatives by sweeping
across the functional graph in the same direction as that of evaluating the function
value. Beginning from the leaves (Figure 3.1), both the function value and the
directional derivative\(^8\) at every vertex are simultaneously evaluated and carried
forward to the root. Thus, every forward sweep yields the function value and its
derivative with respect to a chosen independent (i.e., input) variable.

---

\(^7\)For the example problem (3.22), the work variables are: \( v_i = \{x_1, x_2, 4, 3, 6\} \).

\(^8\)Directional derivative is the derivative with respect to any given variable augmented by the
AD tool.
Therefore, for any $n$-dimensional\footnote{The dimensionality is determined by the number of independent variables, e.g. $x_1, x_2, \ldots, x_n$.} function, gradient evaluation entails $n$ forward sweeps. For problem (3.22), the function has 2 independent variables and needs 2 forward sweeps. Notice that the forward mode is more suitable for functions having several dependent variables and few independent variables, but less suitable otherwise. In other words, forward accumulation is superior to reverse accumulation for functions of the form:

$$f : \mathbb{R} \rightarrow \mathbb{R}^m; \quad m \gg 1.$$  \hfill (3.23)

Table 3.3 outlines the expressions for the forward derivatives of all the vertices propagated from the input, intermediate to the output vertex. At the beginning of every forward sweep, the derivative of one of the input vertices is seeded to 1 and all other inputs to zero. Thus, to differentiate function (3.22) with respect to $x_1$ the vertices are set to $\dot{v}_{-1} = 1$ and $\dot{v}_0 = 0$ and vice versa. Thereafter, the derivatives of the rest of the intermediate variables are evaluated by applying the chain rule (3.21). Hence at the end of every sweep the \textit{exact} numerical values for the function and its gradient with respect to one of the variables are accumulated.

Notice from Table 3.3 that if the value of $\dot{v}_{-1}$ is seeded to 1 and $\dot{v}_0$ to 0, then,
Table 3.3: A list of values and derivatives for the forward mode AD on problem: 
\[ f(x_1, x_2) = (x_1x_2 + \sin x_1 + 4)(3x_2^2 - 6). \]

<table>
<thead>
<tr>
<th>Variables</th>
<th>Vertices</th>
<th>Values</th>
<th>Forward Derivatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>( v_{-4} )</td>
<td>4</td>
<td>( \dot{v}_{-4} = 0 )</td>
</tr>
<tr>
<td></td>
<td>( v_{-3} )</td>
<td>3</td>
<td>( \dot{v}_{-3} = 0 )</td>
</tr>
<tr>
<td></td>
<td>( v_{-2} )</td>
<td>6</td>
<td>( \dot{v}_{-2} = 0 )</td>
</tr>
<tr>
<td></td>
<td>( v_{-1} )</td>
<td>( x_1 )</td>
<td>( \dot{v}_{-1} = \dot{x}_1 )</td>
</tr>
<tr>
<td></td>
<td>( v_0 )</td>
<td>( x_2 )</td>
<td>( \dot{v}_0 = \dot{x}_2 )</td>
</tr>
<tr>
<td>Intermediate</td>
<td>( v_1 )</td>
<td>( v_{-1}v_0 )</td>
<td>( \dot{v}<em>1 = \dot{v}</em>{-1}v_0 + \dot{v}_{-1}v_0 )</td>
</tr>
<tr>
<td></td>
<td>( v_2 )</td>
<td>( \sin v_{-1} )</td>
<td>( \dot{v}<em>2 = \dot{v}</em>{-1}\cos v_{-1} )</td>
</tr>
<tr>
<td></td>
<td>( v_3 )</td>
<td>( v_1 + v_2 )</td>
<td>( \dot{v}_3 = \dot{v}_1 + \dot{v}_2 )</td>
</tr>
<tr>
<td></td>
<td>( v_4 )</td>
<td>( v_3 + v_{-4} )</td>
<td>( \dot{v}_4 = \dot{v}<em>3 + \dot{v}</em>{-4} )</td>
</tr>
<tr>
<td></td>
<td>( v_5 )</td>
<td>( v_0^2 )</td>
<td>( \dot{v}_5 = 2\dot{v}_0v_0 )</td>
</tr>
<tr>
<td></td>
<td>( v_6 )</td>
<td>( v_5v_{-3} )</td>
<td>( \dot{v}<em>6 = \dot{v}<em>5v</em>{-3} + v_5\dot{v}</em>{-3} )</td>
</tr>
<tr>
<td></td>
<td>( v_7 )</td>
<td>( v_6 - v_{-2} )</td>
<td>( \dot{v}_7 = \dot{v}<em>6 - \dot{v}</em>{-2} )</td>
</tr>
<tr>
<td>Output</td>
<td>( v_8 )</td>
<td>( v_4v_7 )</td>
<td>( \dot{v}_8 = \dot{v}_4v_7 + v_4\dot{v}_7 )</td>
</tr>
<tr>
<td></td>
<td>( v_8 )</td>
<td>( f )</td>
<td>( \dot{v}_8 = \dot{f} )</td>
</tr>
</tbody>
</table>

the value of \( \dot{v}_8 \) is the derivative of the function with respect to \( x_1 \). Conversely, if \( \dot{v}_0 \) is seeded to 1 and \( \dot{v}_{-1} \) to 0, then the resulting value of \( \dot{v}_8 \) is the derivative of the function with respect to \( x_2 \). Thus, \( n \)-forward sweeps are required to evaluate the complete gradient of an \( n \)-dimensional function.

C. Vectorised Forward mode of AD

The above principle explains the basic forward accumulation method of evaluating derivatives. A new approach that uses vectorisation is used in this work. As illustrated by the simple AD graph in Figure 3.2a, the proposed method also defines every vertex \( (v_i) \) of its AD graph as an AD object containing its own value, gradient and Hessian fields, such that
CHAPTER 3. DETERMINISTIC OPTIMIZATION METHOD

Figure 3.2: An illustration of the proposed vectorised forward AD principle for a part of the 2-dimensional function: $f(x_1, x_2) = x_1x_2 + \sin x_1 + 4$. (a): Depicts a symbolic representation of the vectorised forward AD graph. (b): Shows the implementation and initialisation principle of the vectorised forward AD graph. Note: every vertex contains a value, gradient, and Hessian; $\dot{x} = dx/dy$, $\ddot{x} = d^2x/dy^2$. 
Table 3.4: Tabulation of the proposed vectorised forward mode AD for a part of problem (3.22): $f(x_1, x_2) = (x_1x_2 + \sin x_1 + 4)$. Every input vertex $v_i$ has its value, gradient and Hessian initialised in a vector form, $v_i = (v_i, \dot{v}_i, \ddot{v}_i)$, such that sweeping from the input vertices to the output vertex yields the overall function value, gradient and Hessian.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Vertices</th>
<th>Vectorised Forward Automatic Differentiation Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Values</td>
<td>Gradient</td>
</tr>
<tr>
<td>$v_{-2}$</td>
<td>$x_1$</td>
<td>$[\dot{x}_1 \ 0] = [1 \ 0]$</td>
</tr>
<tr>
<td>Input</td>
<td>$v_{-1}$</td>
<td>$x_2$</td>
</tr>
<tr>
<td></td>
<td>$v_0$</td>
<td></td>
</tr>
<tr>
<td>$v_1 = v_2 \times v_1$</td>
<td>$x_1x_2$</td>
<td>$[\dot{x}_1 x_2 \ \dot{x}_2 x_1] = [x_2 \ x_1]$</td>
</tr>
<tr>
<td>Intermediate</td>
<td>$v_2 = \sin v_2$</td>
<td>$\sin x_1$</td>
</tr>
<tr>
<td>$v_3 = v_1 + v_2 + v_0$</td>
<td>$x_1x_2 + \sin x_1 + 4$</td>
<td>$[x_2 \ x_1] + [\cos x_1 \ 0] + [0 \ 0]$</td>
</tr>
<tr>
<td>Output</td>
<td>$f = v_3$</td>
<td>$x_1x_2 + \sin x_1 + 4$</td>
</tr>
</tbody>
</table>
\[ v_i = (v_i, \dot{v}_i, \ddot{v}_i) . \] (3.24)

In addition, as shown in Figure 3.2b and Table 3.4, the initial seeds for the gradients of the vertices of the independent variables are initialised as vectors rather than scalars, such that

\[ \dot{v}_{-2} = [1 \ 0] \quad \text{and} \quad \dot{v}_{-1} = [0 \ 1]. \]

Notice that the two vectors \( \dot{v}_{-2} \) and \( \dot{v}_{-1} \) now correspond to the rows of an identity matrix \( I_n \in \mathbb{R}^{n \times n} \). Similarly, the gradient of the constant term vertex \( (\dot{v}_0) \) is also initialised as a vector of zeros

\[ \dot{v}_0 = [0 \ 0]. \]

Then, the Hessians for all the vertices are initialised as \( n \)-dimensional arrays of zeros

\[ \ddot{v}_{-2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} , \quad \ddot{v}_{-1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} , \quad \text{and} \quad \ddot{v}_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \]

Therefore, in contrast to the conventional forward mode of AD which uses a scalar-based initialisation (Figure 3.1 and Table 3.3), the illustrations in Figure 3.2 and Table 3.4 show that for any \( n \)-dimensional function, the proposed vectorisation permits computing the value, gradients and complete Hessians of any multivariate function in a single forward sweep.

While ascertaining the true effectiveness of this new approach will entail a head-to-head comparison with non-vectorised forward AD designs on mathematical programming problems, such experimentation goes beyond the scope of this thesis, which is on global optimization using evolutionary computation. Nevertheless, at the end of this thesis, Appendix A presents some overloaded arithmetic operators and functions (Section A.1), and using a simple case study Appendix A also compares the process of evaluating derivatives using the vectorised forward AD and the symbolic AD method (Section A.2).
3.8 Summary

This chapter reviewed a spectrum of classical optimization methodologies which are mainly deterministic in nature. In particular, it analysed the theoretical convergence characteristics, the methods for evaluating search directions and step sizes of the steepest ascent, Newton, quasi-Newton and the conjugate gradient methods. This has led to the choice of the sequential quadratic programming (SQP) algorithm. The SQP is a second order algorithm that utilises automatic differentiation (AD) to derive its gradients and Hessians during evaluation of its search directions.

It is vital to note that the theme of this chapter is centred on overview of the classical optimization methods. Therefore, further experiments may be vital to assess the full impact of the vectorised AD approach used in the SQP algorithm. Note that in itself the SQP algorithm presented in this chapter, which is a well-known nonlinear optimization method, is not claimed as novel (see Section 3.5). Similarly, the automatic differentiation technique for evaluating derivatives has been widely known to efficiently yield more accurate derivatives than both the numeric and symbolic differentiation methods (Bischof et al., 2008; Christianson, 1999), see also the illustrative case study in Section A.2 of Appendix A. Nevertheless, this chapter improves the AD through vectorisation and then used it in the SQP algorithm. Note that the development of the vectorised AD approach stands as only incidental addition to this thesis. Further experimental investigations would have provided additional support; but experiments on such derivative evaluation techniques for classical optimization methods are outside the scope of this thesis, which is centred on global optimization using evolutionary methods.

Finally, the SQP presented here will serve as a local search method for the hybrid optimization system (Chapter 6) proposed later in Part II of this thesis. The next chapter analyses the individual effects of evolutionary variation operators to examine the general concept of convergence in the global EC algorithm.
Chapter 4

Convergence Analysis in EC

4.1 From Diversity measure to Evolvability measure

In the field of evolutionary computation (EC) diversity refers to the degree of heterogeneity or homogeneity between individuals in a given sample search pool; whereas convergence refers to the phenomenon that is used to assess the potential of a population-based search method to maintain progress by yielding new high quality solutions over successive generations.

It is a common practice in the EC community (Corriveau et al., 2012; Louis and Rawlins, 1993) to assess convergence by temporally measuring the instantaneous spatial diversity in the search pool. This is probably due to its simplicity, low cost of evaluation and its applicability across various models having different data structures or encoding methodologies. Spatial diversity measures the spread of the search pool over the entire search space and within the feasible boundaries of the search domain. Thus, by evaluating the similarity among sample solutions in a search pool, a good estimate of the spread of the sample points can be obtained. For example, in the binary, gray, or any other symbolic encoding, the Hamming distance measure sufficiently yields a good estimate of the pool’s diversity at any given generation. Similarly, in integer, or real-parameter encodings, the Euclidean distance measure (or variants such as Mahalanobis distance1) is often utilised.

1Mahalanobis distance is a descriptive statistic that provides a relative measure of a data point’s distance (residual) from a common point. It is a dimensionless measure introduced by Mahalanobis (1936).
The notion of evolvability in this study refers to the ability of the sample solutions in an evolutionary pool to generate new and higher quality solutions across generations. This view, being tailored to optimization, is slightly more restrictive than the conventional view of evolvability in natural evolution (Lehman and Stanley, 2011). In natural evolution, evolvability refers to the ability of the sample solutions to evolve new (probably more diverse) solutions irrespective of the state of their fitness attributes.

Recall that the application domain in this thesis is the continuous (or real-parameter) optimization problems – wherein real-valued encoding is most appropriate\(^2\). Therefore, the evolvability measure proposed in this chapter (Section 4.5) is first applied to analyse the convergence characteristics of both binary encoded and real-valued EC models. Conversely, on the subsequent analysis that compares the proposed evolvability measure with the spatial diversity measure (Section 4.6), only real-valued EC model is utilised. Therefore, Euclidean distance measure is applied to assess the spread (spatial diversity) in the solution points in this case.

### 4.2 Classical Convergence Assessment Methodologies

Besides stopping evolution based on some user prescribed limits on evaluations, generations or execution time, more sophisticated convergence measures that are mainly based on population diversity are used to automatically stop the evolution. This follows the intuition that diversity collapse directly/indirectly signifies convergence of an evolutionary search. In other words, evolution converges when the candidate solutions in the population become identical. In this respect, population diversity could relate to the genotype, phenotype or even the average fitness of the population directly.

Whilst the convergence characteristics of all evolutionary algorithms rely on several factors, the most important factors are the types of selection mechanism, the variation operators (e.g. crossover and mutation) and the population size.

\(^2\)With real-valued encoding, the EC models proposed in this study would require no genotype-phenotype mapping. This is because both the evolution and solution spaces use similar data types. By skipping mapping function the computational cost for every generation in the evolution is minimised.
The selection operation, being an exploitation process, favours the fitter individuals at the expense of the weaker ones. Therefore, selection is often seen as a deriving force to convergence in EC. Thierens and Goldberg (1994) used the normal fitness distribution method to model the convergence characteristics of various selection methods. They estimated the true convergence behaviours of various selection methods using ordinary differential equation models. They argue that fitness proportionate selection (FPS) methods have the slowest convergence characteristics and it further slow down as the search approaches optimal solution.

On the other hand, variation operators also influence population convergence. As such, it should be noted that besides the types of the variation operators, the frequency of their application also play a major role. A new diversity measure that estimates the average Hamming distance in the population was proposed by Louis and Rawlins (1993). The authors noted that while mutation and its probability of application can severely influence the convergence rate, vast majority of the traditional crossover operators such as $n$-point, uniform and punctuated crossover have little effect on EC convergence. They analytically proved that the new diversity measure (i.e., average Hamming distance) is not affected by crossover operators but mostly influenced by the selection mechanism. The measure was then used to predict the average Hamming distance at convergence and thus defined an upper bound limit on the evolution run time.

Hybridization approaches also play a vital role to improving convergence rate. As reported by Miura and Tanaka (2000), for most nonlinear optimization problems, the time required by a genetic algorithm (GA) to converge to an optimal solution can be reduced by incorporating some information about the gradients of the problem’s variables. Thus, they propose hybridizing GA with a steepest descent algorithm and argue that this can positively influence the overall convergence rate. Similar proposals were made in Isaacs et al. (2007); Oh et al. (2004).

### 4.2.1 Rationale

There are several motivations behind the objective of proposing new approaches to convergence detection in EC. This is because virtually all of the above approaches to convergence detection are not without their pitfalls. For instance, in a binary encoded EC model, the search space reduces to a Hamming landscape...
CHAPTER 4. CONVERGENCE ANALYSIS IN EC

(Reeves, 2005) of bit-flip neighbours as convergence sets in; thus, using Hamming distance allows effective analysis of the diversity profile of the search pool. However, such a similarity measure works in the genotype (encoding) space, and lacks any knowledge of the dynamics in the solution (i.e., phenotype) space. Thus, depending on the employed mapping technique, a set of closely similar (converged) solution points in the genotype space may significantly differ (appear diverged) when mapped to the phenotype space. In addition, concurrent application of such similarity measures to the encoding and solution spaces is expensive; in fact, it might even trigger more convergence false alarms.

Similarly, on real-valued EC models, using the Euclidean distance measure gives a good account of the spatial spread of the sample solution points in the search space. However, this measure tells nothing about the fitness distribution of the solution points in the fitness landscape. Thus, this chapter proposes a new approach to convergence analysis that would ensure effective convergence detection across various global optimization landscapes and on a variety of EC model frameworks.

4.2.2 Convergence Analysis – A New Perspective

The collapse of population diversity signals convergence of an evolutionary search pool. Hence, besides measures that monitor stagnation in fitness progress, traditional approaches mostly rely on diversity measures to assess convergence. This chapter aims to address the following question:

“What aspect(s) of evolution lead to diversity collapse in EC and how would that be monitored/exploited to achieve robust convergence detection?”

Generally, progress in evolutionary search is internally driven by the actions/interactions of the genetic operators (i.e., the selection and variation operators like crossover and mutation). Thus, this chapter seeks to address the above questions by analysing the dynamics of the drivers of evolution. Therefore, the following section introduces the traditional spatial diversity coefficient used for convergence measurement. The subsequent section investigates the effect of evolutionary operators using Price’s theorem. A new population evolvability measure for convergence detection is then proposed.

3Experimental investigations and analyses in this chapter cover both binary- and real-coded EC models.
4.3 Spatial Diversity Measure

This section examines the spatial diversity model traditionally used for convergence detection in EC. Recall that, depending on the employed data structure encoding, spatial diversity accounts for the spread (similarity/dissimilarity) among the sample solutions in an evolutionary pool. Thus, at every generation of an evolutionary search, a coefficient of diversity is evaluated using either the Hamming distance measure on binary encoded models or the Euclidean distance measure on real-valued encoding. Since this study utilises real-valued encoding (cf. Section 2.6.1), the following sections present the evaluation of the coefficient of diversity using a Euclidean distance measure.

4.3.1 Evaluating the Coefficient of Diversity

At every generation $t$, the instantaneous diversity among the sample solutions $\{x_i\}$, in any search pool $P(t)$ of size $|P(t)| = N$, is measured using a Euclidean distance measure. The diversity is then expressed in terms of a coefficient of diversity $C_{\text{Div}}$. However, the approach to determining a suitable reference sample point from which the distance of every sample solution is measured varies across this research domain. On one hand, **locus** of the current best sample solution has been used as the reference point (Herrera and Lozano, 1996). On the other hand, a **centroid** point (i.e., a hypothetical average sample point position), re-evaluated at every generation is often used (Eshelman and Schaffer, 1991; McGinley et al., 2011; Park and Ryu, 2010). The latter approach yields a rather more unbiased estimate of the true spread of the solution points and is therefore adopted in this study. Thus, in the following, $C_{\text{Div}}$ is derived by evaluating the distance of every sample solution from the evaluated centroid point $C$.

Suppose that a search pool $P(t)$ of size $N$ consists of a set of sample solutions $\{X_i \in P(t) \mid X_i = x_{i1}, x_{i2}, \ldots, x_{in}\}$, where $n$ is the dimensionality of the problem, then at any given dimension $j$ the vertex $c_j$ of a centroid sample solution is

$$c_j = \frac{1}{N} \sum_{i=1}^{N} x_{ij}. \quad (4.1)$$

Therefore, for an $n$-dimensional problem, the position of the centroid sample $C$
at any instance is
\[ C = \frac{1}{N} \sum_{i=1}^{N} [x_{i1}, x_{i2}, \ldots, x_{in}] = (c_1, c_2, \ldots, c_n), \]  
(4.2)

where the instance is aligned to the temporal evolution of the pool across generations \( t \), such that as time evolves, the position of the centroid is tracked through the search space.

Consequently, the Euclidean distance \( \gamma_j \) between all the sample points \( N \) and the centroid across any dimension \( j \) is
\[ \gamma_j = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_{ij} - c_j)^2}. \]  
(4.3)

Hence, the instantaneous spatial diversity across all \( n \) dimensions for any sample pool \( P(t) \) is expressed as a coefficient of diversity \( C_{Div} \), such that
\[ C_{Div} = \frac{1}{n} \sum_{j=1}^{n} \gamma_j. \]  
(4.4)

Therefore, the vector of the overall temporal spatial diversity for all generations \( t = 1, 2, \ldots, k \) is
\[ C_{Div} = (C_{Div}^1, C_{Div}^2, \ldots, C_{Div}^k). \]  
(4.5)

Note that the majority of the conventional and problem dependent diversity measures suffer from either sensitivity to distribution of outlier samples, changes in pool sizes or changes in problem dimension. However, experimental findings on comparing various spatial diversity measures by Corriveau et al. (2012) have justified the suitability of the above measure (cf. equation (4.4)).

### 4.3.2 Normalisation of the Coefficient of Diversity

Normalisation of a diversity measure is essential to effectively compare diversity across populations and/or generations. Specifically, normalisation filters out the effects of variations in pool sizes, or changes in problem types, from the true diversity dynamics of a given EC model. Various methods exist for achieving this sort of normalisation.

To normalise the above coefficient of diversity, \( C_{Div} \), a running normalisation
with the maximum coefficient of diversity $C_{Div}^{\text{max}}$ is utilised. This technique relies on the intuition that since the initial pool is generally created from a random uniform distribution, unless a more diverse pool is found over the course of evolution, the initial pool is often the most diverse. Thus, the coefficient of diversity in the first generation $C_{Div}^1$ comes from the most diverse population and is used as the initial normalisation factor $C_{Div}^{\text{max}}$. Subsequently, if at any generation $t$ a more diverse pool is found, then the newly found $C_{Div}^t \models C_{Div}^t = \max_i (C_{Div}^i)$ is used to update the normalised coefficient of diversity vector, denoted by $\tilde{C}_{Div}$, as in equation (4.6)

$$\tilde{C}_{Div} = \frac{C_{Div}}{C_{Div}^{\text{max}}} = \frac{1}{C_{Div}^{\text{max}}}(C_{Div}^1, C_{Div}^2, \ldots, C_{Div}^k),$$  \hspace{1cm} (4.6)

where $C_{Div}^{\text{max}} = \max_i (C_{Div}^i)$.

The above approach is referred to as normalisation with maximum diversity so far (NMDF) (Corriveau et al., 2012). The technique is immune to variations in problem dimensions or pool sizes. Other normalisation measures, such as the landscape diagonal (LD) and coefficient of variation (McGinley et al., 2011) are also used in the literature. The LD measure uses the maximum distance between extreme corners of the search space to assess diversity. It was shown that the majority of such measures are affected by changes in problem dimension or population sizing, and are therefore prone to scalability problems.

### 4.4 Measuring Evolvability using Price’s Theorem

This section analyses fitness dynamics in EC by investigating the effect of evolutionary operators.

In order to examine the independent effect of EC operators while in interaction, Price (1972) formulated a theorem that permits decomposition of the evolutionary process to separate the genetic effect (or contribution) of the selection operator from that of other variation operators (i.e. crossover and mutation). Although Price’s work was mainly in the field of evolutionary genetics, Price’s equation (4.7) provides greater insight into general selection processes. Price’s
theorem states that:

$$\Delta Q = \frac{\text{Cov}(z, q)}{\bar{z}} + \sum_{i=1}^{N} \frac{z_i \Delta q_i}{N \bar{z}},$$

(4.7)

where $\Delta Q = Q_2 - Q_1$ is the change in the measured characteristics $Q$ which is fitness in this case, $N$ is the number of individuals in the parent population (i.e., population size), $z_i$ is the number of offspring of parent $i$, and $\bar{z} = \sum_i z_i/N$ is the average number of the offspring produced. Also, $\Delta q_i = q'_i - q_i$ where $q_i$ is the fitness of parent $i$ and $q'_i$ is the average fitness of its offspring. And finally, $z$ and $q$ are vectors of $z_i$ and $q_i$ respectively.

The two terms in the Price’s theorem (4.7) represent the contribution of different operators as the mean of the characteristic being measured. Since this study deploys Price’s equation to analyse convergence in EC, the measured characteristic, $Q$, in this case will be fitness. The first term represents the contribution of selection operator while the second term gives aggregated contributions of the variation operators involved. Notice that the effect due to selection is modelled in terms of the covariance between the individuals $z$ and their fitness $q$. This conforms to the Fisher’s fundamental theory of natural selection that relates the change in the mean fitness in a population to the population’s fitness variance. It also agrees with the Frank’s argument (Frank, 1997), which sees the covariance between the phenotypic values of individuals and their fitness as the cause of differential productivity that leads to the change in phenotype.

The following lemmas elaborate on fitness decomposition in evolution using Price’s theorem.

**Lemma 1:**
Suppose that the fitness $q_i \in \mathbb{R}$ of each member of the parent population in (4.7) is represented by a vector $q$ such that:

$$q = [q_1, q_2, ..., q_N]^T.$$  \hspace{1cm} (4.8)

Suppose also that the number of offspring $z_i \in \mathbb{Z}$ produced by each one of the $N$ parent is represented by a vector $z$ such that:

$$z = [z_1, z_2, ..., z_N]^T.$$  \hspace{1cm} (4.9)
Then, the following expansions of the two terms in (4.7) show that it is only the first term that represents the contribution of the selection operator.

First term:

\[
\frac{\text{Cov}(z, q)}{\overline{z}} = \text{mean} \left[ \begin{array}{c}
(z_1 - \mu_{z_1})(q_1 - \mu_{q_1}) \\
(z_2 - \mu_{z_2})(q_2 - \mu_{q_2}) \\
\vdots \\
(z_N - \mu_{z_N})(q_N - \mu_{q_N})
\end{array} \right] / \overline{z},
\]  

(4.10)

where \(\mu_{z_i}\) is the mean of the number of offspring of parent \(i\) and \(\mu_{q_i}\) is the mean fitness of the parents.

Second term:

\[
\sum_{i=1}^{N} z_i \Delta q_i = z_1 (q'_1 - q_1) + z_2 (q'_2 - q_2) + \cdots + z_N (q'_N - q_N) 
\]

(4.11)

Recall that following any typical evolutionary selection process, the resulting offspring have the same fitness as their parents (i.e. selection process adds no new solutions to the population), therefore, from (4.11),

\[
\Delta q_i = q'_i - q_i = 0.
\]  

(4.12)

Hence, for the selection operator, the summation in the second term, (4.11), equals to zero. Consequently, the contribution of selection operator is only in the first term of (4.7).

Lemma 2:

To investigate the contribution of the variation operators, assuming any traditional \(m\)-point crossover operator is employed; crossing any two parents yields two offspring. Thus, if the size of the parent population is even and they all undergo the crossover operation (i.e., when crossover probability \(P_C = 1.0\)), then the number of offspring produced is always equal to the number of the parents. Therefore, \(\mu_{z_i} = z_i : \forall i\) in equation (4.10). Hence, the first term (4.10) of the Price’s equation is equal to zero. This is also true for any bit-flip mutation operation where a parent chromosome yields a single offspring after mutation. A noteworthy exception is when \(P_C < 1\) or the population size \(N\) is odd, that is, when not all chromosomes undergo the crossover operation. In such a case, from
(4.10), \( \exists i : \mu z_i \neq z_i \) and thus:

\[
\frac{\text{Cov}(z, q)}{\bar{z}} \neq 0.
\]

Hence, unlike the traditional notion (Bassett et al., 2005) where it is often assumed that the contribution of the variation operators in the first term of Price’s equation is always zero, this investigation reveals the contrary if taking into account the special cases mentioned above; when \( p_C < 1 \).

Now, expanding the second term as in (4.11) reveals that the summation is non-zero since the fitness \( q_i \) of any parent \( i \) is often different from that of its entire offspring\(^4\), i.e., for most \( i, q'_i \neq q_i \). Hence, the contribution of the variation operators to the fitness dynamics is mainly via the second term of Price’s equation, but they might also contribute via the first term as proven above.

\[\square\]

### 4.4.1 Monitoring Evolvability using extended Price’s Equation

Bassett et al. (2004) extend the second term (4.11) of Price’s equation to allow monitoring of the individual contributions of the variation operators. This helps to ascertain their individual contributions to the population’s fitness growth at various stages of the evolution. The extended Price’s equation is defined as:

\[
\Delta Q = \frac{\text{Cov}(z, q)}{\bar{z}} + \sum_{j=1}^{k} \sum_{i=1}^{N} \frac{z_i \Delta q_{ij}}{N \bar{z}}; k = 1, 2, ..., (4.13)
\]

where \( k \) is the number of genetic operators; \( q_{ij} \) is the average value of the fitness of all the offspring of parent \( i \) after the application of operator \( j \); \( \Delta q_{ij} = q'_{ij} - q_{i(j-1)} \) is the difference between the average fitness \( q \) of the offspring of parent \( i \) measured before and after the application of operator \( j \).

Since this study utilises only crossover and mutation as variation operators, the number of operators \( k \) is 2. Therefore, in contrast to the extended Price’s equation in (4.13) which may have several terms, the following extension (4.14) to the Price’s equation contains only three terms; a term for the selection, crossover

\[\footnote{\text{Although it is possible for any or all the offspring resulting from crossover to have equal fitness as their parent, this is rare and usually only occur when the population’s diversity collapses. Thus it signals convergence.}}\]
(X) and mutation (M) respectively. Each of these terms estimates the changes in the mean of the population’s fitness (Δq) due to one of the three genetic operators.

\[
\Delta Q = f(\text{selection}) + f(\text{crossover}) + f(\text{mutation})
\]

\[
= \frac{\text{Cov}(z, q)}{\bar{z}} + \frac{\sum_{i=1}^{N} z_i \Delta q_{iX}}{N \bar{z}} + \frac{\sum_{i=1}^{N} z_i \Delta q_{iM}}{N \bar{z}}.
\]

(4.14)

The next section examines evolutionary progress by visualising the individual effect of the evolutionary processes using the extended Price’s equation.

### 4.4.2 Analysis of Evolvability with extended Price’s Equation

In order to analyse the effect of the proposed extension (4.14) to the Price’s theorem, this section experiments with a binary-encoded EC model (cf. Algorithm 2.2) having mostly standard parameters (De Jong, 1975) (Table 4.1). Various global optimization test problems are used; however, the results reported here\(^5\) are those obtained on simulating the solution process of:

i) 2-dimensional Schwefel function, \(f_2\) (see Table B.1 in Appendix B).

ii) 100-dimensions of Rastrigin function, \(f_1\) (see Table B.1 in Appendix B).

The above test problems are both multimodal and highly dispersed (Lunacek and Whitley, 2006; Salomon, 1996) global optimization benchmarks. The objectives for this experiment are:

i. To observe and analyse the fitness growth of the solution pool as the evolution progresses; and

ii. To investigate changes in fitness attributed to the selection operator, and changes due to the crossover and the mutation operators respectively.

Eventually, by observing the way in which the selection and variation operators collectively move the evolutionary search forward, this investigation would provide new insights into fitness dynamics and convergence in EC. The experiment might also yield additional insight on the role of the individual operators

\(^5\)Similar results are obtained with several other global optimization benchmarks of various dimensions (cf. Bashir (2011)), thus, omitted.
Table 4.1: Parameter settings of the Binary Encoded EC model for convergence analysis.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Values/Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>$N$</td>
<td>50 to 100</td>
</tr>
<tr>
<td>Representation</td>
<td>$-$</td>
<td>Binary encoding</td>
</tr>
<tr>
<td>Selection Scheme</td>
<td>$-$</td>
<td>Binary tournament selection (BTS)</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>$P_C$</td>
<td>1.0</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>$P_M$</td>
<td>$1/L$ where $L=$string length</td>
</tr>
<tr>
<td>Replacement Scheme</td>
<td>$-$</td>
<td>The proposed Adaptive elitism (Section 2.6.6)</td>
</tr>
<tr>
<td>Adaptive overlap size</td>
<td>$\omega$</td>
<td>0.05 i.e., 5% of population size $N$</td>
</tr>
<tr>
<td>Initial Population</td>
<td>$-$</td>
<td>Uniform at random</td>
</tr>
<tr>
<td>Termination Criteria</td>
<td>Max-FEs</td>
<td>Maximum function evaluation ($100 \times N$) FEs</td>
</tr>
</tbody>
</table>

towards balancing the exploitation and exploration of problem search space. Table 4.1 shows the settings for the mathematical model parameters of the EA used in this experiment.

A. Simulation Results

Figure 4.1 shows the behaviour of the EC model and the dynamics of the evolutionary search pool during the optimization process of the Schwefel\textsuperscript{6} benchmark $f_2$. The fitness comparison plot (Figure 4.1a) compares the fitness of the best, worst and the average samples in the search pool. The plot for the extended Price’s equation (Figure 4.1b) on the other hand depicts the changes in the averages of the population’s fitness due to each of the three genetic operators according to the definitions in (4.14). In other words, the three curves (Figure 4.1b) show the average contributions of the selection, crossover and mutation operators to the fitness progress during the course of the evolution. All results are averages of 100 independent runs.

B. Discussion and Analysis of Results

Notice that this section has so far presented the results on the Schwefel benchmark, but the next set of experiments in Section 4.6 extends the investigations to a unimodal, a Needle-in-Haystack and the Rastrigin benchmarks. Whilst all

\textsuperscript{6}The Benchmark functions are described in Table B.1 in Appendix B.
Figure 4.1: Simulation results for optimization of the 2-dimensional Schwefel benchmark ($f_2$) showing fitness comparison plots and the effect of genetic operators. (a): Fitness comparison plot for the best, worst and average fitness achieved over a maximum generation limit of 100. (b): Compares the effect of genetic operators on fitness growth using extended Price’s theorem (4.14). Notice from (b) that the approximate ranges for the evolutionary operators are: Selection $\approx [+25, +350] : sign[+, +]$, Crossover $\approx -50, +50] : sign[+, +]$, and Mutation $\approx [-25, -200] : sign[+, +]$. See Table 4.1 for parameter settings of all plots.

these are chosen to serve as representatives of the commonly encountered global optimization landscapes, these investigations are in no way exhaustive of the various global optimization landscapes. Thus, it is important to note that whether or not these observations will generalise to all other problem types is subject to further investigations. Nonetheless, the results (Figure 4.1) obtained from these experiments reveal the following:

1) The fitness curves in Figure 4.1a show that both the average and the best fit individuals in the population rapidly grow with a steep gradient to their peak values within the very early generations of the evolution. It is, however, noticed that the fitness level of the worst fit samples remains considerably low (Figure 4.1a) throughout the evolution. This indicates that the search pool still maintains some amount of diversity in its fitness distribution\(^7\).

2) It is observed from Figure 4.1b that the curve for the selection operator is

\(^7\)Diversity in fitness distribution is often called *phenotypic diversity*, it does not account for spread of samples.
always above zero on the fitness axis, i.e. no sign changes for the selection operator curve \( (\text{sign: } [+,-]) \). It is therefore evident that selection works as a biased process. It primarily guides the search towards the promising regions of the search space. In so doing, selection ramps up the population’s average fitness as seen in Figure 4.1a. Of course, this behaviour conforms to the characteristics of typical tournament and other ranked-based selection methods (Rudolph, 2000), and thus, it is as expected.

3) Notice also from the same figure (Figure 4.1b) that, although the effect of crossover operator appears to concentrate at the early generations, a close examination reveals that the curve for the crossover operator swings above and below zero on the fitness axis, i.e. crossover curve changes \( \text{sign: } [-,-]. \) This substantiates the supposition that this operator has both exploration and exploitation effects. In other words, while it improves the population’s average fitness via exploitation, it also lowers it during exploration. Further, it is observed that the effect of crossover lessens over generations, and eventually, neither its exploitation nor its exploration effect seems to influence the fitness growth as convergence sets in.

4) Contrary to the previous operators, notice that the curve for the mutation operator (Figure 4.1b) always lie beneath zero on the fitness axis, i.e. mutation does not change \( \text{sign: } [-,-]. \) This means that the effect of this operator is more likely to favour exploring new areas of the search space; and as a result, it improves population’s diversity. But in so doing, mutation tends to drastically lower the fitness growth in the population. Moreover, as can be seen from the mutation curve (Figure 4.1b), the effect of this operator on fitness growth is unaffected by convergence – as it remains fairly constant over the entire run.

From the above results, a clear separation of roles among the genetic operators is deduced. Whilst the selection operator undertakes an exploitation role, mutation ensures effective exploration and the crossover operator serves as a regulator by performing both exploration and exploitation roles.

Consequent on the above observations, it is thought that among all the three genetic operators the characteristics of crossover operator have more influence on convergence dynamics of the search pool. This is because while the effects of both mutation and selection operators continue throughout the evolution, that of the crossover operator appears to be exclusively aligned to the convergence
dynamics and the fitness growth in the population. Moreover, as highlighted in item (2) above, the selection operator is a mere biased process that favours highly fit solutions. It adds no new solutions into the population and is therefore not suitable for convergence detection.

The following section investigates how the dynamics of the crossover operator can be used to detect convergence of an evolutionary search pool.

\section*{4.5 The Proposed Population Evolvability measure for Convergence detection}

Recall, from the proposed extended Price’s equation (4.7), that $Q$ is only the collective change in the average fitness due to the three genetic operators. It was argued that (Bassett et al., 2004) observing changes in averages alone does not convey sufficient information on the true effect of an operator. In fact, average effects may not guarantee accurate conclusions on the convergence of the evolution since the average individuals are not the major driving force for the evolution. This is evident from the plot of the crossover operator in Figure 4.1b, in which it is difficult to appreciate crossover’s effect on the fitness dynamics even during the early generations of the evolution.

Therefore, it is essential to focus on analysing the spread in fitness (fitness variance) in the population. This would permit exploring the true contributions of the best, average and worst individuals mainly produced by the evolutionary variation operators. In this regard, the spread is measured by evaluating the change in fitness variance in the population due to a given genetic operator.

From (4.14), suppose that the change in fitness due to a genetic operator $j$ (crossover in this case) is:

$$
\Delta Q_{ij} = \frac{\sum_{i=1}^{N} z_i \Delta q_{ij}}{N \bar{z}}.
$$

(4.15)

Then, since the change in fitness, $\Delta q_{ij} = q'_{ij} - q_{ij}$, is the random variable of interest, the expectation and variance of the crossover term can be obtained by respectively taking the first and second moment of (4.15) with respect to the $q_{ij}$.
values, such that:

\[ E[\Delta q_{ij}] = \sum_{i=1}^{N} \frac{z_{i} \Delta q_{ij}}{\bar{z}} = \sum_{i=1}^{N} \frac{\sum z_{i} \Delta q_{ij}}{\bar{z}}, \]  

(4.16)

where \( \sum z_{i} \) is the sum over all the offspring of parent \( i \). Typically, there are two offspring for each set of parent when \( P_{C} = 1.0 \). All other parameters are as previously defined in Section 4.4. The variance, \( Var[\Delta q_{ij}] \) is:

\[ Var[\Delta q_{ij}] = E[\Delta q_{ij}^2] - [E(\Delta q_{ij})]^2, \]  

(4.17)

and the standard deviation is:

\[ \sigma[\Delta q_{ij}] = \sqrt{Var[\Delta q_{ij}].} \]  

(4.18)

Now, a shaded area for the one standard deviation interval (\( \pm \sigma \)) is overlaid on the previous plot of the \( \Delta Q \) average changes in fitness due to the crossover operator (Figure 4.1b) – yielding the crossover envelope shown in Figure 4.2a. Notice that Figure 4.2a gives a better impression of the true effect of the crossover operator on the fitness progress across generations. A similar technique is then applied to investigate the effect of the mutation operator (Figure 4.2b).

Figures 4.2(a and b) show the \( \pm \sigma \) (standard deviation) envelopes for both the crossover and mutation operators. The shaded area above the curve for the change in the average fitness due to crossover in Figure 4.2a reveal that crossover operator does indeed contribute to the fitness growth, but at the same time, the shaded area underneath it (which lie under zero on the fitness axis) indicates how much the operator contributed to producing low fit individuals. It is noticed that the crossover envelope shrinks towards zero around the point labelled A. This happens as the curve for the change in the average fitness due to crossover settles around zero on the fitness axis. This means that beyond point A, crossover operator barely contribute to the fitness progress. Hence, this provides an early signal into the cause of diversity collapse in an evolutionary pool\(^8\); and as hypothesised earlier, this would yield a sufficiently robust convergence detection mechanism for EC models.

\(^{8}\)As a caveat to this, when Roulette wheel selection (RWS) mechanism is utilised, it was found that (Bashir, 2011) monitoring the effect of the crossover operator may not suffice as a convergence measure.
Figure 4.2: Visualising one standard deviations (±σ) intervals for the effect of crossover and mutation operators on optimization of the 2-dimensional Schwefel benchmark $f_2$. (a): Shows the average and the ±σ (shaded areas) for the crossover operator. (b): Shows the average and the ±σ (shaded areas) for the mutation operator. Details of the parameter settings in Table 4.1.

On the other hand, an observation of Figure 4.2b reveals that the curve for the change in the **average** fitness due to mutation operator lie and remain beneath zero on the fitness axis throughout the evolution. Also, larger portion of the shaded area for the ±σ mutation envelope is beneath zero. The ±σ mutation envelope (Figure 4.2b) remains virtually uniform (without shrinking) throughout the evolution. This means that the effect of mutation on the fitness of the population continues for the entire evolution run. Consequently, mutation dynamics cannot be deployed to conduct any meaningful convergence detection.

### 4.5.1 Convergence threshold parameter

Having used one standard deviation (σ) interval to analyse the effect of genetic operators, the width of the ±σ envelope for the effect of an operator $j$ (crossover in this case) on fitness growth at every $t$th generation lies within the interval:

$$[\Delta Q_{jt} - \sigma_{jt}, \Delta Q_{jt} + \sigma_{jt}], \quad (4.19)$$
where $\Delta Q_{jt}$ is the change in the average fitness of the population at iteration $t$ due to operator $j$ and $\sigma_{jt}$ is the corresponding standard deviation. Let the width for the crossover operator be represented by $\sigma_{Xover}$, then it can be determined as follows:

$$
\sigma_{Xover} = (\Delta Q_{jt} + \sigma_{jt}) - (\Delta Q_{jt} - \sigma_{jt}) = 2\sigma_{jt}.
$$

(4.20)

Figure 4.3 shows the simulation results for the optimization of a 100-dimensional Rastrigin function ($f_1$) using the parameter settings shown in Table 4.1. It is observed that the dynamical characteristics of this result closely resemble that of the previously seen Schwefel function ($f_2$), see Figure 4.1 and Figure 4.2. Notice that the point labelled (A) on Figure 4.3a directly corresponds to the point labelled (A) on Figure 4.3c when read from the x-axis (i.e., the generations axis). It is also noticed that the more the width of the crossover envelope ($\sigma_{Xover}$) shrinks to zero (Figure 4.3c), the flatter is the gradient of the best and average fitness curves in Figure 4.3a. This means that there is a strong correlation between the crossover envelope and the gradient of the best and average fitness curves. Hence, the width of the one standard deviation interval, $\sigma_{Xover}$ (4.20), indirectly represents the available diversity in the population. Also, the generation at which $\sigma_{Xover}$ tends to zero signifies the end of evolvability – indicating an imminent convergence of the search pool.

Therefore, for the proposed convergence measure, a threshold value is prescribed for the parameter $\sigma_{Xover}$ such that whenever its value falls below this threshold, the search process can automatically terminate. Note that this threshold parameter is user defined and its appropriate value is determined empirically. Sensitivity analyses (Bashir, 2011) suggest that a value of $\sigma_{Xover} < 0.01$ is suitable for the standard EC parameter set. That is, when using crossover and mutation probabilities of $P_C = 1.0$ and $P_M = 1/L$ (for binary encoded EC), or $P_C = 1.0$ and $P_M = 0.01$ (for real-valued EC model) respectively.

### 4.6 Spatial Diversity measure vs. Evolvability measure

On assessing convergence status via population evolvability, i.e., the potential of a search pool to evolve better solutions over generations, the proposed evolvability
Figure 4.3: Convergence detection by monitoring the $\pm \sigma$ (standard deviation) interval of the contribution of crossover operator on a 100-dimensions Rastrigin benchmark $f_1$. (a): Shows a fitness comparison plot for the best, worst and average fitness. (b): Compares the effect of evolutionary operators using extended Price’s equation (4.14). Plots (c and d) show the average and the $\pm \sigma$ (shaded areas) for the crossover and mutation operators respectively. The label A on plot (c) marks the generation at which the evolution converges and it corresponds to label A on plot (a) where fitness progress stalls. Table 4.1 has detail parameter settings; results are averaged over 100 runs; plots are zoomed in for clarity.

The measure ($\sigma_{\text{Xover}}$) is hypothesised to play an important role as the previously introduced traditional coefficient of diversity (Section 4.3). Therefore, the following sections investigate the roles of the two parameters ($\sigma_{\text{Xover}}$ and $C_{\text{Div}}$) on analysing the convergence profile of an EC model on a different set of test problems.
Table 4.2: Parameter settings of the real-valued EC for convergence analysis.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Values/Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>$N$</td>
<td>50 to 100</td>
</tr>
<tr>
<td>Representation</td>
<td>−</td>
<td>Real-valued</td>
</tr>
<tr>
<td>Selection Scheme</td>
<td>−</td>
<td>Binary tournament selection (BTS)</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>$P_C$</td>
<td>1.0</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>$P_M$</td>
<td>0.01</td>
</tr>
<tr>
<td>Recombination Parameter</td>
<td>$\alpha^C$</td>
<td>Weighting parameter $\alpha^C = U(0, 1)$</td>
</tr>
<tr>
<td>Mutation Parameter</td>
<td>$\mu^M$</td>
<td>Step size parameter $\mu^M = U(0, 1)$</td>
</tr>
<tr>
<td>Replacement Scheme</td>
<td>−</td>
<td>Adaptive elitism (Section 2.6.6)</td>
</tr>
<tr>
<td>Adaptive overlap size</td>
<td>$\omega$</td>
<td>0.05 i.e., 5% of population size $N$</td>
</tr>
<tr>
<td>Initial Population</td>
<td>−</td>
<td>Uniform at random</td>
</tr>
<tr>
<td>Termination Criteria</td>
<td>Max-FEs</td>
<td>Maximum function evaluation $(100 \times N)$</td>
</tr>
</tbody>
</table>

4.6.1 Experiment: Comparing Spatial Diversity and Evolvability Measure

The analysis herein is tailored towards investigating the dynamical characteristics of both the newly proposed crossover evolvability measure $\sigma_{Xover}$ and the commonly used coefficient of spatial diversity $C_{Div}$ across various fitness landscapes. Unlike in the preceding experiment (Section 4.4.2), the EC model (cf. Algorithm 2.2) used here features real-valued encoding. Thus, this model utilises the parameter settings shown in Table 4.2. Furthermore, a set of global optimization benchmarks is used as case studies. This experiment aims to empirically compare and analyse the individual roles of these complementary convergence measures.

A. Experimental Goals

The objectives for this experiment are:

i. to investigate the effectiveness of the newly proposed crossover evolvability measure $\sigma_{Xover}$ in detecting convergence in real-valued EC model;

---

9Since the application domain in this study is on continuous optimization problems, real-valued encoding is an appropriate choice as it eliminates the need for mapping functions between genotype and phenotype spaces. Henceforth, all the EC models that will be proposed in this thesis will use real-valued encoding.
ii. to compare, using various test problem landscapes, the roles of $C_{Div}$ and $\sigma_{Xover}$ in convergence detection; and

iii. to investigate when and why concurrent use of these convergence measures ($C_{Div}$ and $\sigma_{Xover}$) is essential for effective convergence detection in EC.

During the simulation of the optimization process in this experiment, the following parameters are recorded at every generation $t$:

- the normalised spatial diversity coefficient $C_{Div}$;
- the normalised crossover evolvability measure $\sigma_{Xover}$;
- the best fitness value at every generation $f(x_{best})$; and
- the average fitness of the samples in the pool at every generation $\bar{f}(x)$.

Using the following two distinct case studies, this experiment investigates the relationships between the above two convergence measures (i.e., $C_{Div}$ and $\sigma_{Xover}$) and the two important evolutionary phenomena: (i) fitness growth over generations, and (ii) fitness stagnation due to convergence.

Case I: Optimization of a smooth unimodal test problem
This involves simulating the solution process of a simple nonlinear problem having a single peak and a valley. The goal is to search for the maximum value of the function $f_0$. Detail formulation of $f_0$ is given in Table B.1 in Appendix B.

Case II: Optimization of a Plateau-based (Needle in Haystack) test problem
This examines the optimization process of the 2-dimensional Easom’s global optimization benchmark ($f_3$). The landscape of $f_3$ is characterised with an entirely flat search space having a single needle-like peak as its only optimum. Similarly, the objective is to optimize for the global optimum solution which is at $x^* = 1$. See Table B.1 in Appendix B for details on $f_3$.

B. Experimental Results
The simulation results for the optimization process of the test problem benchmarks $f_0$ and $f_3$ are as shown in Figures 4.4 and 4.5 respectively. The figures
reveal that both the crossover evolvability measure, $\sigma_{Xover}$, and the spatial diversity coefficient, $C_{Div}$, have significant effect on the convergence characteristics and fitness dynamics during the evolutionary search. In the following, $f_0$ and $f_3$ benchmarks are analysed in turn.

### 4.6.2 Analysis and Interpretation of Results

Firstly (on the $f_0$ benchmark), by mapping the fitness progress curves in Figure 4.4b against that of the spatial diversity $C_{Div}$ (Figure 4.4a), it is observed that the instantaneous spatial diversity in the search pool has an inverse relation to the fitness growth. That is, as the average fitness in the search pool grows over generations (Figure 4.4b), the spatial diversity shrinks and approaches zero. This leads to diversity collapse at later stages of the evolutionary search (Figure 4.4a).

On the other hand (still on $f_0$), the shaded area (Figure 4.4a) depicts the dynamics of the population evolvability measure, $\sigma_{Xover}$. It reveals the instances at which the search pool has higher tendencies to evolve better solutions and when evolvability begins to stall. It can be noticed that, at the early generations, the high levels of $\sigma_{Xover}$ correspond to a sharp rise in the gradient of the fitness progress – indicating higher evolvability. This trend continues until the pool loses its ability to evolve higher quality solutions, then $\sigma_{Xover}$ gradually shrinks and collapses towards zeros when the population evolvability ceases.

Notice from the above that for the $f_0$ benchmark both $C_{Div}$ and $\sigma_{Xover}$ have agreed in the way they assess population’s fitness growth and convergence profile of the search pool. In addition, the characteristics of the two convergence measures ($C_{Div}$ and $\sigma_{Xover}$) have also agreed on the previously introduced $f_1$ and $f_2$ benchmarks (results omitted for brevity). These results (Figure 4.4) reveal an interesting relation between the newly proposed evolvability measure $\sigma_{Xover}$ and the population’s fitness. But the fact that the dynamical behaviour of the newly proposed evolvability measure ($\sigma_{Xover}$) is consistent on both binary and real-valued encoded EC models (from across the experiments in Sections 4.4 to 4.6) means that the proposed measure is representation independent. Consequently, it can work on different implementations of EC models – promoting its suitability to wider problem domains.

Secondly, on the simulation results (Figure 4.5) for the Needle in Haystack benchmark ($f_3$), there is a subtle difference in the characteristics of the two measures ($\sigma_{Xover}$ and $C_{Div}$) and their relations to fitness progress and convergence.
Figure 4.4: Analysis of convergence measures (crossover evolvability measure $\sigma_{Xover}$ and spatial diversity coefficient $C_{Div}$) on the smooth unimodal benchmark $f_0$. (a): Compares the characteristics of $\sigma_{Xover}$ and $C_{Div}$ over generations. (b): Compares their relation to fitness progress and fitness stagnation. All results are averages of 100 runs.

It is observed from Figure 4.5a that the evolvability measure is low both at the early generations (when the spatial diversity is at its highest) and at the later stages of the evolution (when the spatial diversity approaches zero). However, similar to the previous results (Figure 4.4a), the dynamical characteristics of the traditional spatial diversity measure ($C_{Div}$) has remained unchanged even on this kind of test problem, i.e., $f_3$. This is because $C_{Div}$ only accounts for the instantaneous spread of the samples in the search pool but not their ability to evolve
or their tendency to converge.

On the other hand, the characteristics of $\sigma_{\text{Xover}}$ (Figure 4.5a) is found to accurately describe the fitness dynamics (Figure 4.5b) and hence convergence of the evolutionary search pool on this kind of problem, $f_3$. To understand this, suppose that the total evolution period (Figure 4.5b) is divided into early, middle and last generations. Then, it is observed, from the early generations, that the combination of a plateau-based fitness landscape in $f_3$ and the high dispersion in the initial solution pools significantly flattens the gradient in the fitness progress
CHAPTER 4. CONVERGENCE ANALYSIS IN EC

curves. This effect is clearly indicated by the lower levels of evolvability in Figure 4.5a during the early generations. But as the search pool gathers sufficient information on the problem landscape, a steep gradient in the fitness curve is observed at the middle generations (Figure 4.5b). This effect is also reflected by the substantial amount of $\sigma_{Xover}$ recorded during the middle stages of the evolution (Figure 4.5a). Finally, as convergence sets in, the gradual decline in fitness progress during the last generations is also correctly anticipated by the gradual fall in $\sigma_{Xover}$.

4.6.3 Discussion

The above observations and analyses mean that for some categories of global optimization problems (the like of $f_0$, $f_1$, $f_2$), both $\sigma_{Xover}$ and $C_{Div}$ perform similar roles; but on other varieties of problems (the like of $f_3$), the proposed evolvability measure ($\sigma_{Xover}$) may yield a better account of the overall evolutionary progress and hence its convergence characteristics.

Further, as observed from the simulation results (Figure 4.4a and Figure 4.5a), the population evolvability $\sigma_{Xover}$ often remains high long after the spatial diversity $C_{Div}$ has fallen to zero. This means that a convergence detection mechanism that solely relies on spatial diversity may prematurely halt the evolution at the time a significant fitness progress is well possible. Similarly, lower levels of evolvability measure are sometimes seen at the early stages of the evolution (Figure 4.5a). Thus, if $\sigma_{Xover}$ is used as the only convergence measure, the search process may also terminate prematurely.

Therefore, it is essential to remark that combining $\sigma_{Xover}$ and $C_{Div}$ is essential to harness synergy between the two measures. The following section demonstrates the combination procedure for these two measures.

4.6.4 Combining the Spatial Diversity and Population Evolvability Measures

Thus far, the above experiments have revealed the potential strengths and weaknesses of each of the convergence detection methods, viz. spatial diversity ($C_{Div}$) and population evolvability ($\sigma_{Xover}$) measures. The model in (4.21) demonstrates

10Due to the plateau-based nature of the NiH fitness landscape, early fitness progress is barely possible. This made evolvability $\sigma_{Xover}$ remains low during early generations.
how these two measures can be combined to harness synergy between the two. The set-up requires both measures to be satisfied for an evolutionary pool to be deemed as converged. Therefore, a logical “AND” is utilised to ensure that the user defined thresholds \( C_{\text{Div}}^{\text{min}} \) and \( \sigma_{\text{Xover}}^{\text{min}} \) for the spatial diversity and evolvability measures have simultaneously been reached prior to declaring the pool as converged.

Equation (4.21) describes the convergence model:

\[
\left( \tilde{C}_{\text{Div}}(t) < C_{\text{Div}}^{\text{min}} \right) \text{ AND } \left( \sigma_{\text{Xover}}(t) < \sigma_{\text{Xover}}^{\text{min}} \right), \tag{4.21}
\]

where \( t \) is the current epoch in the evolutionary search. The setting for \( C_{\text{Div}}^{\text{min}} \) and \( \sigma_{\text{Xover}}^{\text{min}} \) is empirical, a value of \( 10^{-3} \) for both works well on the benchmark problems used in this thesis.

In the next chapter, the EC model proposed in Section 5.4.6 utilises the above convergence model. To illustrate how the above convergence model (4.21) would facilitate instantaneous assessment of the convergence status of an evolutionary search pool, Section 5.5 investigates how an EC model that utilises this convergence model, for diversity control, compares to a standard EA.

### 4.6.5 Contribution and Predicted Impact

Convergence detection and assessment is crucial to any successful design of evolutionary optimization model. The traditional approaches mainly monitor stalled generations and use a spatial diversity measure to assess the similarity (or dissimilarity) among samples in a search pool. The investigations in this chapter have revealed that while such measures are sufficient on some category of problems, they may be inadequate on other varieties of global optimization problems. This is especially the case when dealing with global optimization problems having sparsely scattered peaks/valleys in a wide plateau landscape\(^{11}\). In this chapter a new convergence measure based on extended Price’s theorem is proposed. The measure assesses population evolvability via monitoring the effect of crossover operator on the fitness dynamics of an evolutionary search pool.

Furthermore, sequel to the above experimental findings, a combined measure is proposed. This combines the traditional spatial diversity measure and the

\(^{11}\)The NiH problem \((f_3)\) is a typical example having sharp peak(s) scattered in a wide landscape; and it seems that many real-world problems possess such features (Das and Suganthan, 2010; Horner et al., 1993).
newly proposed population evolvability measure to facilitate robust convergence
detection. As is presented in the following chapters (see Sections 5.4.6 and 6.3),
the new convergence measure plays the following roles in the development of the
dual-pool and hybrid EC models.

(i) serving as convergence detection mechanism, the combined system is used
to build the adaptive switching mechanism that connects the global search
algorithm with a local search method for local refinement in the hybrid
model proposed in Section 6.3; and

(ii) it also controls the operations of the diversity control mechanisms used in
the dual-pool EC model proposed in the next chapter (Chapter 5).

4.7 Summary

This chapter has introduced the general notion and significance of convergence
analysis in evolutionary framework. First, the commonly used spatial diversity
measure has been examined in detail. Then, a new convergence measure that
uses extended Price’s theorem to assess population evolvability is proposed. In
a series of experiments on various global optimization benchmarks, the analyses
of the individual characteristics of these convergence measures yield remarkable
insights into their individual strengths and weaknesses. The insights motivated
the proposal of a robust convergence assessment methodology which combines the
two measures to ensure effective convergence detection. The following chapter
builds upon these insights and extends the notion of diversity management and
control in evolutionary computation.
Chapter 5

Diversity Control in EC

Diversity control is vital for effective global optimization using evolutionary computation (EC) techniques. This chapter aims to present a new architecture that ensures effective diversity maintenance throughout the optimization cycle. In general, this chapter classifies key diversity control policies in EC via a research relevance tree; it then addresses evolutionary diversity from the perspective of exploration and exploitation balance. But in particular, the chapter proposes new approaches to diversity restoration, and diversity measurement. The investigations in this chapter seek to verify the following hypothesis:

#H: A diversity control method that uses convergence detection mechanism enhances the exploration-exploitation balance and improve maintenance of useful diversity even under limited population sizes.

As a caveat to this hypothesis, diversity control is achieved via the use of restarted subpopulations with a separate pool for evolution and a pool of restarted samples for diversity. To achieve this objective, this chapter proposes a dual population-based diversity control technique. The chapter briefly introduces the general concept of diversity in evolutionary perspective in Section 5.1, it then aligns diversity to exploration and exploitation in Section 5.2. As an overview, the chapter classifies the commonly used successful approaches for improving diversity management across the various EC paradigms in Section 5.3. Section 5.4 presents the design and developmental aspects of this new method. On the diversity dynamics, Section 5.5 provides new insights into the differences between the standard EC and the proposed model. Then, Sections 5.6 and 5.7 present, respectively, experimental evaluations as well as performance comparisons on the CEC2013 benchmarks. The chapter concludes with a remark in Section 5.8.
5.1 Diversity in Evolutionary Computation

The notion of diversity in evolutionary computation is synonymous with that in other population-based search techniques. Diversity may be defined as the degree of entropy among all the sample solution points in a given pool. Population diversity reflects the extent to which the solution pool is heterogeneous or homogeneous. When diversity is assessed based on the spread (i.e., the genotypic distance) of the sample points within the feasible search space, it is referred to as spatial/genotypic diversity \citep{Corriveau2012}. Otherwise, when measured in the phenotype space, diversity reflects the fitness distribution in the solution space and is called phenotypic diversity. When diversity is mentioned in the EC literature it often refers to spatial diversity. Diversity is a natural source of power that crucially contributes to the inherent adaptive capabilities of evolutionary algorithm (EAs) \citep{Cobb1993}; diversity contributes to the wide applicability of EAs to a wide range of global optimization problems.

5.2 Diversity in the light of Exploration and Exploitation

Exploration and exploitation describe the underlying phenomenon that governs the success of the nature-inspired evolutionary computation methodologies. As in many other population-based (meta)heuristics \citep{Blum2008, Reeves1993}, the ability of EAs to solve stochastic optimization tasks is directly aligned to the efficacy of their exploration and exploitation strategies. Because the terms exploration and exploitation are frequently used in the EC literature, they are mostly regarded as mere norms, this is reflected in the sporadic appearance of their explicit definitions.

From the diverse terminologies and linguistics often used in EC, this study views exploration from the perspective of diversification or spread of the search over the entire search space, whereas exploitation is viewed as intensification or steering of the search to a given neighbourhood. This view is consistent with the definitions of exploration and exploitation reported in \cite{Blum2008, Jiang2008b, Thierens1998} and in \cite{Molina2010}.

It is noteworthy that although exploration and exploitation play complementary roles in achieving a robust global search mechanism, establishing a suitable
balance between the two can be a difficult and challenging task in the design of EAs. This might be attributed to the facts that:

(i) simultaneous optimization of both exploration and exploitation needs a careful treatment, and requires a trade-off, since the two goals are sometimes-conflicting; and

(ii) EAs are mainly applied to large scale, complex problems which are usually characterised with poorly understood objective landscapes.

Nevertheless, Corriveau et al. (2012) argue that viewing exploration and exploitation as opposing forces is rather naive because this may only be true in some special cases (such as when optimizing a unimodal landscape). Thus, it would be fair to view them as orthogonal forces, making it possible to improve both simultaneously.

Clear manifestations of qualitative features that seek to strike a balance between exploration and exploitation are inherent in the core processes of the majority of evolutionary paradigms. For instance, variation operators, such as crossover and especially mutation, are believed to enhance exploration by ensuring good coverage of the entire search space. The reproduction operator (i.e., selection) mainly favours exploitation of the promising region(s) of the explored search space (Greenhalgh and Marshall, 2000; Potter et al., 2003). And, as demonstrated by Bashir (2011); Bashir and Neville (2012a) and Li et al. (2010), the crossover operator has the innate tendency to simultaneously enhance both exploration and exploitation. However, it seems that such nature inspired population-based metaheuristics are mainly good for exploration of the search space and identification (but not exploitation) of the areas with high quality solutions (Blum et al., 2011). This stimulates the need for new methods that can reinvigorate the balance between exploration and exploitation. The aim has been to seek for optimum exploitation while maintaining useful level of diversity throughout the search process.

The focus here is to investigate some of the most effective related strategies utilised for diversity control in EC theory. Thus, Section 5.3 presents a hierarchical tree-like diagram, termed research relevance trees (RRT) to provide a broader

---

1Unimodal: A sequence is said to be unimodal if it is a finite sequence that first increases/decreases and then decreases/increases (Weisstein, 2013). A unimodal distribution refers to a distribution having a single local maximum.

2Useful Diversity refers to the population diversity that in some way helps produce good solutions (Lozano et al., 2008).
Figure 5.1: A Research Relevance Tree for the Fundamental Diversity Control Approaches used in Evolutionary Computation.

classification of the diversity control policies in evolutionary computation. Then, each of the relevant domains in the RRT is examined in a greater detail.

5.3 Taxonomy of Diversity Control Policies

This section investigates some of the well-known, commonly used diversity control strategies in evolutionary computation. The analysis herein is underpinned by a taxonomy of the key diversity control policies in EC via the research relevance tree in Figure 5.1. This theoretic research relevance tree paradigm permits separation of the fundamental research domain into a number of possible approaches. It aids in the design and development phases and translates into mathematical model parameters and data structures for the dual-pool EC model (see Section 5.4.6).

The possible research approaches in Figure 5.1 could align to the following frequently used methods:

(i) heuristic population (re-)initialisation strategies,

(ii) multipopulations models, and

(iii) hybrids and portfolios of algorithms

which are reviewed in due course.

Note, however, that the categorisation in Figure 5.1 is by no means exhaustive of the multitude of approaches that could be used in diversity control. In fact, further details on other approaches that maintain population diversity by
dynamically controlling evolutionary parameters, such as mutation and crossover rates can be found in Eiben et al. (1999); McGinley et al. (2011). The next section examines the various quasi-random initialisation approaches, which complement diversity control by employing various heuristics to generate an initial search pool.

5.3.1 Heuristic Population Initialisation

As in many other population-based heuristics, the initial pool in evolutionary algorithms is usually generated in a random manner, by means of a uniform distribution (De Jong, 1975). For any given \( n \)-dimensional search domain \( D \subseteq \mathbb{R}^n \), the sample solution points \( x \), are randomly created within the feasible boundaries such that the initial pool is

\[
P = \{ x \in \mathbb{R}^n \mid x_i \leq x_i \leq \bar{x}_i \}; \quad i = 1, 2, \ldots, n.
\] (5.1)

where \( \bar{x}_i \) and \( x_i \) are the lower and upper bounds of the \( i \)th dimension.

This random sampling (5.1) is believed to satisfactorily yield a problem independent means of kick-starting any population-based stochastic search process. However, from the last decade, a number of work (Maaranen et al., 2004; Rahnamayan et al., 2007; Tometzki and Engell, 2011) have suggested that using quasi-random heuristics for population initialisation can have a profound impact not only on the search efficiency, but also on the overall quality of the resulting final solution. This intuition comes from the fact that even with no \textit{a priori} information on the nature of the final solution, heuristic initialisation can ease the generation of more diverse and probably fitter samples.

In an attempt to examine the benefit of a more uniformly distributed sample over a mere randomly generated one, Maaranen et al. (2004) use quasi-random sequences to generate initial pool. Although the good distribution property of their quasi-random sequence seems to degrade with increase in dimension, a problem widely known as the curse of dimensionality (Bellman, 2003; Morokoff and Caflisch, 1994). But the authors found that the pools generated using quasi-random sequences which try to imitate points with a perfect uniform distribution, tend to cover the entire feasible search space more optimally. Similarly, Rahnamayan et al. (2007) propose a novel approach that uses opposition-based learning\(^3\)

\(^3\)Opposition-based learning works based on the theory of opposite numbers: For any real
CHAPTER 5. DIVERSITY CONTROL IN EC

... to generate the initial pool for a genetic algorithm. Rahnamayan et al. (2007) reported acceleration in the algorithm’s overall convergence speed. Recently, de Melo and Delbem (2012) propose a smart sampling technique for creation of the initial pool using a classifier that maps and characterises the search space.

Elsewhere, Morrison’s investigations (Morrison, 2003) lead to the conclusions that heuristic initialisations neither yield significant improvement in the quality of the final solution, nor do they reduce the required number of function evaluations. But compared to random initialisation, they can minimise stochastic effects in the end result of the evolution by reducing the variance in the solution quality across independent runs.

In general, pool re-initialisation techniques, also called restarted or multi-start procedures, work by repeated application of an optimization algorithm on a restarted search pool. Hickernell and Yuan (1997) proposed a multi-start algorithm that performs global search by using a restarted quasi-random sample pool of size $N$. In what they called concentration, the authors applied an inexpensive local search to optimize each of the quasi-random sample points. After every $p$ iterations of the local search, $q$ best samples are kept while the remaining $N - q$ are replaced with newly created quasi-random ones and the procedure is repeated. Notice that their restarted local search is only run for a fixed predefined number of iterations ($p$). Nevertheless, their experimental results on several nonlinear benchmarks have shown that the multi-start quasi-random samples achieved better coverage of the feasible search space which improved the performance of the resulting algorithm on wider problem types.

Recently, Martí et al. (2012) extensively surveyed such multi-start approaches. Importantly, the authors went further to suggest categorisation of these methods into memory-based and memoryless procedures.

Further details on heuristic initialisation can be found in Tometzki and Engell (2011) where three different initialisation approaches are used as a pre-processing phase. Their results (Tometzki and Engell, 2011) suggest that heuristic initialisations can potentially improve convergence speed and solution quality. The next section reviews a spectrum of multipopulation strategies for diversity control in various EC models.

---

\[ \text{number } x \in [a, b], \text{ the opposite of } x, (O_x) \text{ is defined as: } O_x = a + b - x. \] Rahnamayan et al. (2007) theoretically prove that to every point $P$ in $D$, there is a unique opposite point $O_P$. 

5.3.2 Multipopulation Strategies

A commonly used diversity control strategy is the multipopulation approach. This has its inspiration from the biological notion of niching and speciation\(^4\) wherein diversity is enforced by promoting species formation within a population. In this regard, Chen et al. (2011) classify all concepts utilising multipopulations, islands, or ensembles models as forms of cooperative hybridization, which potentially enhance the preservation of useful diversity in the search pool.

Some key variations amongst the various multipopulation strategies are in the processes of creating subpools and in the adopted migration policies. In the majority of these strategies, subpopulations are run concurrently and evolve by optimizing a common objective. Thus, they can be classified as “synchronous”, see Figure 5.1. In island models (Alba and Tomassini, 2002) for example, the search begins with two or more subpopulations which exchange information via periodic migrations. But the number and size of the subpopulations are mainly predetermined by the user and are then kept unchanged throughout the evolution (Alba and Tomassini, 2002; Branke, 1998).

Niching and crowding algorithms are multi-populations strategies that generally seek to minimise premature convergence and in particular facilitate convergence to multiple, highly fit solutions across the search space. De Jong and Spears (1992) developed a ‘crowding-factor’ which defines the size of the niche and suggests that a newly generated sample should replace samples from within its own niche rather than any random sample in the population. Goldberg and Richardson (1987) proposed a variant of niching called sharing; they defined a sharing function and used it to modify the fitness of the sample solutions across the search pool. The idea is that the fitness values of samples existing in a cluster are adjusted (devalued) relative to that of fairly isolated samples. Many other variations of niching can be found in Horn and Goldberg (1998).

In deterministic crowding (Mahfoud, 1995), an offspring sample contests in a tournament with its nearest parent sample and the highly fit sample is selected to the next generation. In probabilistic crowding however, the best sample wins the contest at a certain probabilistic rate defined by a survival likelihood function. Ballester and Carter (2003) compared deterministic and probabilistic

\(^4\)Niching, in simulated evolution, promotes formation of groups of individuals in a population, whereas speciation restricts reproduction to within a group of related individuals that share a common niche (Sareni and Krähenbühl, 1998).
niching methods on a set of five multi-modal test problems. From a series of experiments on 2-dimensional multi-modal test problems, the results in Ballester and Carter (2003) led to the conclusions that GAs using crowding replacement with random selection seem more robust and have good chance of converging to alternative optima for the class of evaluated test functions. By contrast, the GA that combines random replacement with tournament selection performs poorer.

Further empirical studies which compare various schemes – including probabilistic and deterministic crowding, niching and speciation, sharing and restricted tournament selection – on real-parameter optimization can be found in Ballester and Carter (2004); Singh and Deb (2006). For extensive analytic study on various niching and crowding techniques on discrete multi-modal problems, see Mengshoel and Goldberg (2008). Other synchronous multipopulation methods dynamically create subpopulations and adjust their numbers and sizes as the search progresses (Branke et al., 2000; Tenne and Armfield, 2005). Such methods aim to ensure maintenance of diversity by avoiding premature population convergence.

The other category that is not as universally exploited is the “asynchronous” operation of subpopulations (Figure 5.1). In this case, the creation and evolution objectives of the subpools differ. For instance, in a recent work, Park and Ryu (2010) propose a dual-population GA (DPGA) designed to enhance diversity control. The model consists of a main and a reservoir population. The main pool has the fitness of its samples evaluated based on the problem’s original objective function whereas, the reservoir pool is evaluated with an objective that exclusively optimizes diversity. DPGA was reported (Park and Ryu, 2010) to perform best on highly multimodal functions having densely populated peaks, but not as good on sparse landscapes.

The micro-GA algorithm proposed by Krishnakumar (1990) is shown to possess remarkable performance characteristics on both stationary and non-stationary problems. Although the micro-GA is not specifically titled a multi-pool approach, its design also falls in to the class of the asynchronous multi-pool methods. The approach runs a standard GA with a small sized pool (only 5 samples) until convergence. Then a new pool of 4 samples which are randomly created replaces all the samples in the previous pool except the best sample, i.e. the elite. Thus, similar to the dual-pool EC algorithm proposed later in Section 5.4, the micro-GA utilises standard parameter settings for its variation operators and uses the tournament selection method. However, it does not utilise any heuristic initialisation
to generate the new sample pool. This means that it is not improbable for the newly generated random samples to still be in the neighbourhood of the previously converged pool. In addition, micro-GA mainly relies on a distance measure (i.e. similarity in the genotype or phenotype samples) to assess the convergence of its search pool, whereas the dual-pool EC model (Section 5.4.6) will utilise both the spatial diversity and population evolvability measures.

Other related multipopulation-based EAs include the so-called multinational evolutionary algorithm (Ursem, 1999) and forking GA (Tsutsui et al., 1997). The following section reviews some recent hybrid methodologies deployed to enhance diversity control.

5.3.3 Hybrids and Portfolios of Algorithms

There is a general consensus that the population-based methods are robust in attaining global optimality via wide exploration (Blum et al., 2011; Joines and Kay, 2003; Michalewicz, 1994). Yet, their lack of intense exploitation capabilities limits their effectiveness in dealing with complex global optimization tasks. To strengthen the balance in the exploration and exploitation for optimal diversity management, various approaches which combine a number of algorithmic models in form of hybrids or memetic algorithms⁵ are developed. Such approaches usually combine EAs with various local improvement procedures made from local search algorithms.

Joines and Kay (2003) examine the behaviour of various instantiations of hybrid algorithms. They found that irrespective of the employed model for evolutionary learning (Baldwinian, Lamarckian or their mix)⁶; hybrid frameworks seem to achieve good exploration-exploitation balance as compared to their non-hybrid counterparts. Theoretical analyses on hybrid frameworks which seek to balance exploration-exploitation trade-off can be found in Lin and Chen (2011).

It is noteworthy that while hybrid methods consistently enhance the exploitation ability of EAs, their design requires careful treatment. This is because excessive application of an intense local search algorithm risks over-exploitation – disrupting the crucial exploration-exploitation balance.

⁵Algorithmic portfolios also called memetic algorithms (MA) (Moscato, 1999).
⁶In Baldwinian evolution the local search only alters the individual’s fitness (phenotype); whereas Lamarckian evolution changes both the phenotype and the genetic structure of an individual.
From the above reviewed methodologies (Sections 5.3.1 to 5.3.3), it is noticed that the challenges in designing an effective diversity control policy require a multifaceted approach. Thus, in the following (Section 5.4) a dual-pool model designed to use a dedicated pool for evolution and a restarted sample pool for diversity is proposed. The proposed model is then evaluated on various global optimization case studies in Section 5.6.

5.4 A Dual-Pool EC Model for effective Diversity Control

5.4.1 Rationale

Multipopulation EC models (Section 5.3.2) have continued to receive significant attention following the many successful investigations (Chen et al., 2011) into such systems. However, these models have always had to cope with the challenges surrounding the additional parameterisation involved. In general, designing multipopulation models requires a prior decision on the creation and management strategies of the multiple pools. Thus, one needs to decide, beforehand, on the criteria upon which the subpopulations evolve within themselves and communicate with one another, i.e. the migration policies among subpools. In addition, the minimum and maximum pool sizes, the initial number of pools, thresholds for the minimum and maximum number of subpools (when dynamic pool creation is utilised), etc., must be decided.

Crucially, the challenges involved in such traditional multipool approaches often result in huge overhead due to proliferation of secondary parameters. In fact, the parameter tuning task often becomes more problematic – and occasionally intractable – since the optimum settings for such additional parameters are problem dependent.

With the above considerations in mind, this section proposes a dual-pool EC model that enjoys the benefits of the multipool framework combined with a heuristic initialisation. Specifically, as is visualised in the theoretic research relevance tree in Figure 5.2, the proposed approach integrates a quasi-random heuristic initialisation, called search space partitioning (SSP), into a dual population architecture to facilitate temporal diversity control. The dual-pool model is made up of an evolution pool (i.e., the main pool), and a diversity pool. The evolution
pool primarily undergoes the evolutionary optimization process, whereas the diversity pool constitutes the newly restarted samples which are created on demand to reinstate diversity into the evolution pool. Preliminary to the development of the proposed EC model, the characteristic data sets and mathematical model parameters for the solution approaches (SSP initialisation and the dual-pool) are specified in Figure 5.2. Details of each of these follow in turn.

5.4.2 The Dual-Pool EC Architecture

In contrast to the conventional multipool architectures which are mainly synchronous in nature, the proposal herein is aimed at designing a framework (cf.
Figure 5.2) that combines an *asynchronous* dual-pool model with a heuristic population initialisation. This enables robust diversity control by minimising recurring communication overhead among subpools.

### 5.4.3 The Evolution Pool – Creation and Working

Playing the role of the main pool, the evolution pool $P_{Evo}$ serves as the initialisation point for the global search. Similar to any conventional random initialisation, $P_{Evo}$ is created using a pseudorandom number generator based on a uniform distribution. The size of this is equal to the actual population size ($N$) for the overall search. In order to ensure the feasibility of the initial samples, $P_{Evo}$ is created (uniform at random) within the feasible boundaries of the search space, such that:

$$P_{Evo} \leftarrow x \in [\underline{x}, \bar{x}] : x \in \mathbb{R}^n,$$

where $n$ is the problem dimension, $x$ is the vector of design variables, $\underline{x}$ and $\bar{x}$ are vectors of lower and upper bounds respectively.

### 5.4.4 The Diversity Pool – Creation and Working

The evolution pool and the restarted samples in the diversity pool form a multipopulations strategy that runs in an *asynchronous* mode (Figure 5.2). Thus, the restarted samples forming the diversity pool $P_{Div}$, are only occasionally used to restore useful diversity into the evolution pool. Consequently, $P_{Div}$ is created only after and whenever a sufficient convergence of the evolution pool is detected. For the details on how convergence is assessed (Bashir, 2012)\(^7\), see Section 4.3 for a spatial diversity measure ($\tilde{C}_{Div}$), and Section 4.5 for the evolvability measure ($\sigma_{Xover}$) originally proposed in Bashir and Neville (2012a).

Note that the occasional creation and injection of the newly restarted samples from the diversity pool in this proposal clearly contrasts with the majority of traditional multipool approaches. In such approaches, the dual (Park and Ryu, 2010) or multiple (Branke et al., 2000) pools are run concurrently throughout the evolution\(^8\). Thus they suffer a significant hike in computational cost due

---

\(^7\)A robust convergence detection mechanism that combines a spatial diversity measure with a population evolvability measure proposed in Section 4.5 is adopted in this model.

\(^8\)For any additional subpool of size $N_i$ in the traditional multipool methods, evaluating the samples in the subpool(s) at every generation $t$, leads to an additional $N_i \times t$ evaluations for every evolutionary cycle.
to continuous evaluation of the extra samples from several constituent pools. The following section presents a mathematical model that describes the creation process of the diversity pool, \( P_{Div} \).

### 5.4.5 Search Space Partitioning (SSP) Heuristic

**Initialisation**

In order to improve diversity by enforcing uniformity in the coverage of the entire feasible search space, a strategy that generates the diversity pool \( P_{Div} \) using a quasi-random heuristic called search space partitioning (SSP) is proposed. In principle, SSP partitions the search space into uniformly sized hypercubes and repeatedly samples one random sample from each hypercube until the required pool size \( (N) \) is reached.

Given any \( n \)-dimensional search space \( D \in \mathbb{R}^n \) (Algorithm 5.1), let each of its dimensions be segmented into \( \kappa \) equal partitions (Algorithm 5.1, line 6). Suppose that \( \rho(\kappa) = \{m_1, m_2, \ldots, m_n\} \) is the set of the resulting partition sizes for each of the \( j = 1, \ldots, n \) dimensions (line 7). Then, along each dimension \( j \), the partition sizes, \( m_j \), are assumed to be uniform. Therefore, SSP segments the original search space \( D \) into \( \phi_\kappa = \kappa^n \) equal-sized subspaces (hyper-cubes) (line 8). For each subspace \( \phi_\kappa \), let \( x_i = [\bar{x}_i, \underline{x}_i] \in \mathbb{R}^n \) be a uniformly distributed random sample generated within the boundaries of \( m_i \). Then, SSP uses a simple uniformly distributed pseudorandom generator to generate equal number of samples across the entirety of the partitioned search space \( \phi_\kappa \) (lines 9-10).

The following example compares the effect of SSP on a highly partitioned search space against an un-partitioned space.

**A. A Concrete Case**

Given a 2-dimensional search space, such that \( D \in \mathbb{R}^2 \). Let the required population size be \( N = 100 \) samples\(^9\), then SSP heuristic generates diverse sample solutions by partitioning the search space within the following two limiting cases:

**Case I:** If SSP is set to use a single partition, i.e., \( \kappa = 1 \) such that the partition size for each of the \( n \)-dimensions is \( \rho(1) = \{m_1, m_2, \ldots, m_n\} \), then SSP

---

\(^9\)SSP partitions each of the \( n \)-dimensions of the search space \( D \in \mathbb{R}^n \) into \( \sqrt[\kappa]{N} \) segments. e.g., for a 2-D space, if \( N = 100 \), then each of the 2-dimensions is partitioned into \( \sqrt{100} = 10 \) segments.
Algorithm 5.1 Search Space Partitioning Quasi-random Heuristic.

1: Define and set search space \( (D) \) parameters
2: dimensions \( n \);
3: total population size \( N \);
4: bounds: \( D \in \mathbb{R}^n = \{ x_j \in \mathbb{R} \mid x_j \leq x \leq \bar{x}_j; j = 1, 2, \ldots, n \} \);

5: Define and set partition parameters
6: set the number of partitions to \( \kappa \);
7: evaluate partition sizes \( m(\kappa) : \{ m_j = \frac{x_j - \bar{x}_j}{\kappa} \mid j = 1, 2, \ldots, n \} \);
8: segment \( D \) into equal sized subspaces: \( \phi_\kappa = \{ \kappa^n = \prod_{j=1}^{n} m_j \} \);
9: generate a random (uniform) sample \( \{ x_i = U(\phi_\kappa) \} \) from each \( \phi_\kappa \);
10: repeat (9) until all \( N \) samples are generated.

Note that segmenting every dimension of the original search space \( D \) into \( \kappa \) partitions yields \( \phi_\kappa = \kappa^n \) subspaces.

yields \( \phi_\kappa = \kappa^n = 1^n \) subspace. In this case, SSP generates all the \( N = 100 \) samples using a uniform distribution over the entire feasible search space (Algorithm 5.1, line 4), i.e., within the subspace bounds \( \phi_\kappa = \prod_{j=1}^{n} m_j \) which is equal to the size of the original search space \( D \).

Case II: If on the other hand the SSP is set to use multiple partitions with say, \( \kappa = 10 \), then SSP segments the search space \( D \) into 10 partitions, and throws into each of the \( \phi_k = \kappa^n = 10^2 \) subspaces a single sample point \( x_i = [x_i, \bar{x}_i] \). The position of a sample in each subspace is also determined by a uniform random distribution as in the previous case.

Notice from the above cases that whilst case I falls back to a conventional pseudorandom initialisation where only a single partition is utilised, case II randomly allocates every sample into a separate partition across the entire search space. Thus, in the proposed SSP heuristic, the required minimum population size \( N \) relates to the number of partitions \( \kappa \) according to the following model:

\[
N = \kappa^n, \tag{5.3}
\]

where \( n \) is the dimensionality of the search space \( D \).

It is deduced from (5.3) that the higher the number of partitions \( \kappa \), the larger the required pool size \( N \) to achieve maximum spread for a given dimension \( n \). This is because the two have an exponential relation with respect to the dimensionality.
Figure 5.3: Comparing a typical Random initialisation with the SSP quasi-random heuristic on a 2-dimensional search space. (a): Random Initialisation. (b): SSP heuristic Initialisation.

$n$. Hence, the SSP quasi-random heuristic is obviously not immune to scalability problems in high dimensional problems, a phenomenon popularly known as *curse of dimensionality*. A simple workaround is to regulate the required minimum $N$ by reducing the number of partitions as dimensionality grows.

Figures 5.3(a and b) compare the distribution of the samples generated using a typical random initialisation with that of the SSP based heuristic initialisation in a 2-dimensional search space.

Figure 5.3a depicts a typical random initialisation where the sample points are generated independently from a uniform distribution. Thus, inevitably, some portions of the search space are often over-sampled, leaving other portions under-sampled. This therefore results in formation of clusters, scattered across the search space. This effect is visible in Figure 5.3a, such as in partition \( \{ 2 < x < 3 \text{ and } 0 < y < 1 \} \) which has no sample points. There are approximately 20 partitions in Figure 5.3a which do not have any sample point. This leaves several other partitions with a cluster of more than one sample points. Notice that this corresponds to the SSP defined in Case I above where $\rho(\kappa) = \rho(1)$, which is equal to the original search space $\mathcal{D}$.

In contrast, Figure 5.3b shows when the SSP heuristic segments the search space into $\kappa = 10$ partitions (i.e. $\rho(\kappa) = \rho(10)$), SSP distributes its samples in a rather more uniformly distributed manner, avoiding unnecessary cluster formations. Note that this is achieved at barely any additional computational cost.
The formation of clusters in initial search pools may have undesirable consequences. In essence, any population-based algorithm like EA could leverage the availability of clusters in its initial pool if and only if they are formed based on some prior knowledge on the nature of the problem landscape. Although having such prior information could benefit the efficiency of these stochastic methodologies, it is often not available. Therefore, blind formation of clusters with no prior information often impedes the overall search efficiency by misleading the evolutionary variation processes during the crucial initialisation stages. Hence, SSP avoids unwanted clustering by enforcing some uniformity in the initial distribution of the sample pool. This allows the EA to freely kick start its search without the need for any prior knowledge about the problem landscape.

The preceding overview on heuristic initialisations (Section 5.3.1) revealed that such approaches are commonplace in the EC literature. However, this proposal for randomly sampling within a uniformly partitioned search space is a novel approach that could minimise the cost of quasi-random heuristic initialisations.

### 5.4.6 The Proposed Dual-Pool EC Model

Having designed the mathematical models for the evolution pool and the SSP-based restarted samples in the diversity pool, this section presents the proposed model for the dual-pool EC algorithm as outlined in Algorithm 5.2. This model closely resembles the EA (Algorithm 2.2) previously examined in Chapter 2. A key distinguishing feature is in the initialisation stage (line 2) where a separate evolution pool \( P_{\text{Evo}} \) is utilised. This is then later combined (line 11) with the restarted samples in \( P_{\text{Div}} \) (created using the SSP heuristic initialisation, see line 8) whenever convergence is detected (line 5). The evolutionary cycle ends (line 3) when a termination condition – such as accuracy threshold or maximum evaluation limit – is reached. Consequently, this model’s search process proceeds such that, whenever \( P_{\text{Evo}} \) converges, new samples from the diversity pool, \( P_{\text{Div}} \), are used to restore sufficient diversity into the search process.

To facilitate further understanding of the underlying processes behind the proposed dual-pool EC model, the flow diagram (Figure 5.4) demonstrates the dynamic merging process of the separate pools during the course of evolution. Figure 5.4 reveals that most of the evolutionary cycles are run solely with the samples from the evolution pool \( P_{\text{Evo}} \), the newly restarted samples in the diversity pool \( P_{\text{Div}} \) are only introduced when sufficient convergence is detected. Thus,
Algorithm 5.2 The Dual-Pool EC Algorithm

1: initialisation
   \( N \leftarrow \text{Pool size}; n \leftarrow \text{problem dimension}; \)
   \( t \leftarrow 0; \)
2: initialise the evolution pool
   \( P_{Evo}(t) \leftarrow \{ X_i \} : X \in [x_j, \pi_j], i = 1, ..., N, j = 1, ..., n; \)
3: while not termination do
4: run EC model and estimate convergence at every iteration
   \( P_{Evo}(t), \tilde{C}_{Div}(t), \sigma_{Xover}(t) \leftarrow \text{invoke EC}(P_{Evo}(t)); \)
5: check for convergence of \( P_{Evo}(t) \)
   \( \text{if} \left( \tilde{C}_{Div}(t) < C_{\text{min}}^{\text{Div}} \right) \text{ and } \left( \sigma_{Xover}(t) < \sigma_{\text{min}}^{\text{Xover}} \right) \)
6: \( P_{Evo}^{R}(t) \leftarrow \text{rank } f(P_{Evo}(t)); \)
7: get the top \( k\% \) best solutions (elite) in \( P_{Evo} \)
   \( P_{Evo}^{R}(t) \leftarrow k\% \left( P_{Evo}^{R}(t) \right); \)
8: (re-)initialise the diversity pool \( P_{Div} \) using SSP heuristics (Algorithm 5.1)
   \( P_{Div} \leftarrow \{ X_i \} : X \in [x_j, \pi_j], i = 1, ..., N, j = 1, ..., n; \)
9: evaluate and rank \( P_{Div} \) by distance from the elite
   \( P_{Div}^{R} \leftarrow \text{rank } \| X_{Div} - X_{Evo}^{E} \| : i = 1, ..., N; \)
10: get the farthest samples in \( P_{Div}^{R} \)
    \( P_{Div} \leftarrow (1 - k) \left( P_{Div}^{R} \right); \)
11: merge evolution and diversity pools to form new \( P_{Evo} \)
    \( P_{Evo}(t) \leftarrow \{ P_{Evo}^{R}(t) \cup P_{Div} \}; \)
12: end if
13: \( P_{Evo}(t + 1) \leftarrow P_{Evo}(t) \)
14: \( t \leftarrow t + 1; \)
15: end while

Figure 5.4: The Dual Pool EC model dynamically showing the merger of the distinct evolution pool \( (P_{Evo}) \) with the SSP created diversity pool \( (P_{Div}) \) over generations \( t \). The periodic merging process is adaptively controlled via a robust convergence detection strategy.

the model allows continuous optimization via a temporal exploration-exploitation cycle. This enhanced diversity control mechanism helps avoid premature convergence. Ultimately, the optimization system runs continuously until a user specified limit on function evaluations is reached.
Recall that the commonly used approaches to multi-population models (see Section 5.3.2) require concurrent and continuous runs of the multiple pools. However, the proposal herein suggests a new framework which enables injecting fresh samples from the diversity pool into the evolution pool. The interaction between these pools is controlled through a robust convergence measure. This is a vital feature that avoids the unnecessary communications overhead suffered by the traditional multipool and multi-start approaches.

5.5 Visualising Diversity in EC Models

Prior to the experimental evaluations, this section examines with the aid of a visualisation how spatial diversity fares under both the standard EC model and the newly proposed dual-pool EC architecture. From the perspective of evolutionary optimization, the availability of diverse samples at any instance in a search pool serves as a driving force for continuous evolution. Diversity allows the evolutionary operators to generate newer and possibly higher quality solutions. However, since EAs naturally tend to eventually converge to a region having high quality solutions in the search space, they inevitably lose the crucial diversity in their sample pool. Of course, it is noteworthy that the convergence rate depends partly on the settings for the mutation and crossover probabilities, and to a large extent on the selection pressure\(^\text{10}\) employed.

5.5.1 Diversity Visualisation with a Standard EC Model

An illustration of typical temporal dynamics of the spatial diversity ($\hat{C}_{\text{Div}}$) in an evolutionary pool of a standard EC algorithm (Algorithm 2.2) is as shown in Figure 5.5c. The result comes from an EA model, applied on the Schwefel benchmark (Section 5.6.1), having a randomly initialised real-valued sample pool of size $N = 100$. The model uses BGA\(^\text{11}\) mutation and intermediate crossover operators applied at the rates of $P_M = 0.01$ and $P_C = 1.0$ respectively. A strict binary tournament selection without replacement is utilised.

\(^{10}\)Selection pressure describes the convergence rate and it is often defined as the ratio of the probability of selecting the currently best sample solution to that of an average sample solution. See Section 2.6.3 for further details.

\(^{11}\)The adopted mutation strategy is based on the Breeder GA (BGA) mutation algorithm (Mühlenbein and Schlierkamp-Voosen, 1993). It is an advanced version of Gaussian mutation.
It is observed that the initially diverse samples in Figure 5.5a gradually converge towards a limited area of the search space over generations (Figure 5.5b). To some extent, the spatial diversity, $\tilde{C}_{Div}$, falls sublinearly with increasing function evaluations (Figure 5.5c). Although this phenomenon could have been avoided by increasing the probability of mutation, applying high rates of mutation slow down the evolutionary progress and could turn the search into a random one.

On the other hand, an EA with a converged pool (such as the one in Figure 5.5b) has lower chances of yielding any significantly different and higher quality solutions. This is because the converged pool handicapped the effect of the evolutionary variation operators. Consequently, it is a difficult and uncertain trade-off to set up a good balance in exploration and exploitation.

### 5.5.2 Diversity Visualisation with a Dual-Pool EC Model

In comparison to the diversity dynamics of the standard EC (Figure 5.5), Figure 5.6 depicts the dynamics for the proposed dual-pool EC model (Algorithm 5.2) on the same benchmark problem. The parameterisations of the dual-pool EC are as specified in Table 5.1. Notice that a smaller pool size, $N = 50$, is employed. In particular, Figure 5.6d depicts the dynamics of $\tilde{C}_{Div}$, while Figure 5.6(a-c) show the temporal interplay of the evolution and diversity pools in this model.

It was found that similar to the standard EC model, the dual-pool EC enjoys an exploratory initialisation with the samples in its evolution pool $P_{Evo}$ scouting the entire feasible search space (Figure 5.6a). Then, the evolution pool gradually converges to a high quality region (see the cluster in Figure 5.6b) to exploit the already learned global information of the search. The diversity dynamics in Figure 5.6d reveals that unlike with the standard EC model, the rate of convergence in this model relates more linearly with the number of function evaluations. Also, the degree to which the samples converge is considerably higher (see Figure 5.6b and the value of $\tilde{C}_{Div}$ at the point labelled B in Figure 5.6d). As compared to a rather weak exploitation previously seen in the standard EC model (Figure 5.5c), Figure 5.6d indicates the ability of the dual-pool EC model to allow deep exploitation of the promising areas of the search space. Whilst the two algorithms share the same underlying parameterisation, the deep exploitation witnessed here could be a result of using relatively smaller sized pools (see Table 5.1). This was possible since the new framework is able to maintain sufficient diversity even with small sample sizes.
Figure 5.5: An illustration of a typical spatial diversity ($\tilde{C}_{Div}$) dynamics for the standard EC model. (a): Depicts an instantaneous 2-D view of the distribution of initial sample pool scattered all over the search space. (b): Shows the distribution of the sample pool after several function evaluations at later stage of the evolution with the samples virtually converged (c): Shows the dynamics of $\tilde{C}_{Div}$ with the regions of high level of diversity (label A) and low level of diversity (label B) marking the exploration and exploitation stages, respectively. The vertical axis in plot (c) is normalised.

Figure 5.6: An illustration of the typical spatial diversity ($\tilde{C}_{Div}$) dynamics for the Dual-Pool EC model. (a): Depicts an instantaneous of the distribution of initial sample pool over the search space. (b): Shows the distribution of the sample pool during the exploitation stage with the samples virtually converged. (c): Shows that merger with the Diversity Pool restores better coverage of the search space. (d): Shows the dynamics $\tilde{C}_{Div}$ with the labels (A), (B) and (C) marking a high $\tilde{C}_{Div}$ for exploration by the initial evolution pool, the lowest $\tilde{C}_{Div}$ during exploitation, and a restored high level of $\tilde{C}_{Div}$ after merger with the diversity pool, respectively.
After $P_{Evo}$ merges with the new samples form $P_{Div}$ (Figure 5.6c), the newly introduced diverse samples restore a full-scale spatial diversity into the previously converged evolution pool. Note that while the new samples in $P_{Div}$ draw the evolutionary search towards exploring other unexplored regions of the search space, the previously learned information is carried forward in the elite samples $P_{Evo}^E$ inherited from the previous evolution pool (Algorithm 5.2, line 7). Hence, this sequence of exploration-exploitation phases encourages continuous global searching – by preserving diversity – even when a small sized pool is utilised.

5.6 Evaluation of the Dual-Pool EC on Multimodal Benchmarks

This section analyses and evaluates the performance of the proposed dual-pool EC model on a set of multimodal global optimization benchmarks. The aim is to analyse the effect of effective diversity control on optimization of highly multimodal problems under limited population size and computational budget. Detail parameterisations for the dual-pool EC model are presented in Table 5.1. Besides the specifications for the standard evolutionary parameters, Table 5.1 specifies the types of the evolutionary operators, their rates and step sizes. It also specifies the creation mode for the dual populations.

5.6.1 Benchmark Test Cases – Features and Significance

The proposed dual-pool strategy is benchmarked on a set of global optimization test problems. The experiments empirically compare the performance of the dual-pool EC model with that of a standard EC model. The comparison is on the basis of the required function evaluations to attain a close approximation (within $10^{-3}$ accuracy level) of the true optimal solution.

The test problems considered are categorised into two major classes. The first class is a set of three traditional global optimization benchmarks consisting of: (i) Rastrigin ($f_1$); (ii) Schwefel ($f_2$); and (iii) Easom ($f_3$), test problems (see Table B.1 in Appendix B for their detailed formulations).

- The Rastrigin and Schwefel functions have many local optimum solutions surrounding the global optimum, and hence they are highly multimodal.
### Table 5.1: Parameter Settings for the Dual-Pool EC Model

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Description/Values/Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>$N$</td>
<td>20 to 50</td>
</tr>
<tr>
<td>Initial Population</td>
<td>–</td>
<td>SSP Heuristic initialisation</td>
</tr>
<tr>
<td>Encoding</td>
<td>–</td>
<td>Real-valued</td>
</tr>
<tr>
<td>Selection Scheme</td>
<td>–</td>
<td>Binary Tournament</td>
</tr>
<tr>
<td>Evolution Pool size</td>
<td>$P_{Evo}$</td>
<td>$N$, i.e. the main population size</td>
</tr>
<tr>
<td>Diversity Pool size</td>
<td>$P_{Div}$</td>
<td>$(1 - k) \times N$, i.e., $k%$ smaller than $N$</td>
</tr>
<tr>
<td>Evolution Pool Elite</td>
<td>$P_{Evo}^E$</td>
<td>Only $k = 5%$ of the evolution pool are merged with $P_{Div}$ after $P_{Evo}$ converges</td>
</tr>
<tr>
<td>Operator type$^a$</td>
<td>$\mathcal{C}$</td>
<td>Crossover: Intermediate recombination</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>$P_C$</td>
<td>1.0</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>$P_M$</td>
<td>0.01</td>
</tr>
<tr>
<td>Recombination parameter</td>
<td>$\alpha$</td>
<td>Weighting parameter $\alpha = U(0, 1)$</td>
</tr>
<tr>
<td>Mutation Parameter</td>
<td>$\mu$</td>
<td>Step size parameter $\mu = U(0, 1)$</td>
</tr>
<tr>
<td>Replacement Scheme</td>
<td>–</td>
<td>Generational–Elitist</td>
</tr>
<tr>
<td>Termination Criteria</td>
<td>$E_{abs}$</td>
<td>Absolute Error $E_{abs} &lt; 10^{-3}$, or</td>
</tr>
<tr>
<td></td>
<td>Max-FE</td>
<td>Max. Function Evaluations $(10,000)$</td>
</tr>
</tbody>
</table>

$^a$The Intermediate recombination operator is an extended line recombination often used with real-valued representation. The BGA mutation is a variant of Gaussian mutation.

Notice, however, that Rastrigin function is symmetric and has a global convex topology$^{12}$, whereas the Schwefel function does not.

- The Easom function is characterised with a sharp peak situated in a wide plateau landscape. This function is quite challenging to deterministic or gradient based models because it yields no promising direction of descent/ascent. It is also popularly known as the Needle-in-Haystack (NiH) benchmark.

The second class also constitutes three test problems, namely: (i) Rastrigin2; (ii) Sphere2; and (iii) Hybrid, benchmarks. These are essentially modified versions of the traditional benchmarks. The key motivation for the development of the modified benchmarks was to deal with the “flawed” properties associated with

$^{12}$Multimodal functions having a convex global orientation are said to have global convex topology. Such functions although multimodal, appear to be GA-easy due to the unique nature of their landscapes. They are also classed as low dispersion problems, see Lunacek and Whitley (2006).
the traditional benchmarks; such properties include separability, global convexity, symmetry, i.e. having global optimum situated at the centre of search domain and so on. As demonstrated by Liang et al. (2005), such flaws can be mitigated by simple modifications involving shifting, rotation, stretching/compression of the conventional benchmarks. In addition, more test problems, built from compositions of several instances of a given benchmark (called a basis function) or a mix of basis functions (also called hybrid composite functions) were also suggested.

The modifications on the traditional test problems yield improved benchmarks which are believed to have most of the attributes of the real-world problems. The modified benchmarks were recently adopted in the various IEEE CEC competitions on real-parameter and dynamic optimization problems. Details on their development can be found in Li et al. (2008); Liang et al. (2005); Salomon (1996). In particular, the Rastrigin2 benchmark used in these experiments is a shifted and rotated version of the traditional Rastrigin function. The Sphere2 benchmark is a composition of 10 Sphere basis functions. The Hybrid benchmark is a composite of various basis functions. It consists of two basis functions from each of the Sphere, Ackley, Griewank, Rastrigin and Weierstrass benchmarks. See Table B.1 in Appendix B for detailed expressions of each of these basis functions.

5.6.2 Results: Analysis and Interpretation

The proposed dual-pool EC model is compared with a standard EC algorithm on a set of global optimization benchmarks (detailed in Section 5.6.1). The results of the evaluations of the sensitivity of these algorithms across six different pool sizes (20 to 1000) are as presented in Figure 5.7; the reported results (detailed in Table C.1, Appendix C) are averaged outcomes of 100 independent runs for statistical significance. For all the test problem types, the bar plots in Figure 5.7 show the average number of function evaluations required to reach the true optimal solutions within an absolute error of $E_{abs} = 10^{-3}$.

The horizontal dashed-lines at the top of the plots in Figure 5.7 mark the limit of $10^5$ function evaluations. This limit defines the maximum computational budget available for the algorithms to converge to within $10^{-3}$ accuracy level of the optimum solution. Consequently, an algorithm is considered to have converged to the true optimal solution of a given problem if and only if its bar graph has not hit the mark for the maximum function evaluation limit of $10^5$. 

Figure 5.7: Performance Comparison of the Dual-Pool EC with the Standard EC Algorithm on six global optimization benchmarks across various pool sizes. The vertical axes show the computational cost (function evaluations) in log scale. At the top of bar pairs, + symbol indicates a statistically significant difference; − symbol indicates an insignificant difference. A nonparametric Wilcoxon Rank sum test is used. The error bar on the bar graphs shows the standard error in the mean function evaluations. The horizontal dashed lines mark the maximum evaluation limit. Results are averages of 100 runs.
Notice that the two algorithms are assessed on both robustness and efficiency; robustness is judged based on how often an algorithm converges to the true optimal solution within the budgeted evaluations; efficiency is rated based on the number of function evaluations needed to converge to the optimal solution. Thus, the efficiency is indicated by the height of the bar graphs (the lower the better). Notably, while both the algorithms have found the true solution on many of the test problems considered, see Figure 5.7(a to d) for example, the standard EC algorithm has failed to converge on several test cases on the Sphere2 and Hybrid benchmarks (Figure 5.7(e and f)).

Furthermore, the error bars on the bar graphs (Figure 5.7) represent the standard errors in the mean number of function evaluations. At the top of the bar pairs in Figure 5.7, the pairs having statistically significant difference and those that have statistically insignificant difference are labelled + and −, respectively. The following section describes the two significance tests used.

5.6.3 Statistical Significance Testing and Analysis

To assess the statistical significance of the obtained results, both the parametric (paired t-test) and the nonparametric (Wilcoxon Rank sum\textsuperscript{13}) tests (Hollander and Wolfe, 1999), were utilised. The two algorithms are statistically compared at $\alpha = 0.01$ significance level for the various benchmarks.

A key distinguishing feature between the two statistical tests is in the assumptions made about the nature of the distribution of the populations and the measured parameter. For the parametric t-test, the null-hypothesis $H_0$ assumes that the results for the two algorithms are random samples from independent normal distributions with equal means and, the alternative hypothesis $H_1$ assumes that the means are not equal. But for the parametric test, the null-hypothesis $H_0$ assumes the random samples are from independent identically distributed continuous distributions with equal medians, and the alternative hypothesis $H_1$ assumes the medians are not equal. Parametric tests can be used when normality conditions are satisfied. However, the significance of nonparametric tests in analysing experimental results obtained through multiple independent runs of EAs for multiple problem analysis was discussed at length by García et al. (2009). Importantly, nonparametric tests require no normality restrictions.

\textsuperscript{13}Wilcoxon rank sum test is a pair-wise non-parametric statistical test designed to detect significant differences between two algorithms.
Table 5.2: Statistical Tests: Parametric (t-test)/nonparametric (wilcoxon) test for the results in Table C.1 comparing the Dual-Pool and Standard EC models.

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Statistical Tests on Benchmark Problems: t-test/Wilcoxon</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rastrigin</td>
</tr>
<tr>
<td>20</td>
<td>−/+</td>
</tr>
<tr>
<td>50</td>
<td>−/+</td>
</tr>
<tr>
<td>100</td>
<td>+/+</td>
</tr>
<tr>
<td>200</td>
<td>+/+</td>
</tr>
<tr>
<td>500</td>
<td>+/+</td>
</tr>
<tr>
<td>1000</td>
<td>+/+</td>
</tr>
</tbody>
</table>

Note: The + symbol indicates a statistically significant difference; − symbol indicates an insignificant difference between the performances of the two algorithms. The highlighted results show that the parametric and nonparametric tests have only disagreed on 8 out of 36 test cases. Thus, while the nonparametric test is generally appropriate across the varying pool sizes, the parametric test may not be suitable on small to medium pool sizes.

In this work, comparison of the two statistical tests (detailed in Table 5.2) reveals two important findings.

1) For the experiments conducted with populations of 20 and 50 samples, the t-test and Wilcoxon rank sum test differ on 8 out of the 12 test cases. This interesting finding reveals that when a small pool size (≤ 50) is used, there is more than 66% chance of having the t-test fail. This could be due to the fact that parametric tests, such as t-test, require normality assumption which does not always hold for small sample sizes.

2) However, for the rest of the experiments where search pools of more than 50 samples are utilised, the t-test and Wilcoxon rank sum test have demonstrated remarkable agreement (Table 5.2). This implies that for larger pool sizes (> 50) the distribution of the sample pool seems to satisfy the normality assumption.

The above findings signify that whenever evaluation of EAs involve using small to medium pool sizes, a nonparametric test must be used to ensure robust significance analysis. Else, for large pool sizes (> 50), a parametric test suffices. Therefore, since our experiments span small to large pool sizes, the statistical significance results reported in Figure 5.7 are only those based on the nonparametric Wilcoxon test.
5.6.4 Analysis of Simulation Results

From the above description of the benchmark test cases in Section 5.6.1, the simulation results in Figure 5.7 are discussed in two main sections. Firstly, this section analyses the results from the first category of the test problems – the traditional benchmarks (Rastrigin, Schwefel, and Easom) – shown in Figure 5.7(a to c). It is observed from Figure 5.7(a to c) that both the dual-pool and standard EC algorithms have reached the required accuracy within the available \(10^5\) function evaluations.

Besides the general plots for the evaluation results in Figure 5.7, a set of performance summary plots on these evaluation results is shown in Figure 5.8. Figure 5.8a summarises the computational cost of each algorithm across all the test problems. Figure 5.8b summarises the cost incurred by each algorithm when run with a pool of 20 to 1000 samples. The significance of Figure 5.8b is to provide additional insight into the overall sensitivities of the individual algorithms to varying pool sizes.

The summary plot in Figure 5.8a shows that for both models, the computational cost on the three traditional benchmarks is approximately around the first \(10^4\) function evaluations; whereas on the modified benchmarks both models needed approximately \(10^5\) function evaluations. This finding reveals that both models were able to discriminate the different complexities of the two categories of benchmarks.

Notice, however, that the proposed dual-pool EC algorithm needed more evaluations on some of the test cases on the traditional benchmarks (see, for example, Figure 5.7(a and c) and the label (A) on Figure 5.8a for the Easom benchmark). This is not unexpected because on low complexity problems such as the traditional benchmarks, the dual-population framework used in the proposed method may not always translate to efficiency improvements. In fact, the central design goal is to enhance robustness on wide range of global optimization problems. Thus, while the proposed method is expected to ensure convergence to optimal solution on low to high complexity problems, some efficiency lag is likely on less complex problems. Nevertheless, when summarised over all the pool sizes, the computational cost summary plot (Figure 5.8b) revealed that the dual-pool model seems to always converge to the optimum solution with fewer function evaluations. This generally shows improved efficiency over the standard EC model.

On the other hand, the simulation results (Figure 5.7) and the summary plot
Figure 5.8: Performance comparison summary for the standard EC and dual-pool EC models on the basis of functions evaluations, the lower the better. (a): Shows the computational cost (in log scale) of the two algorithms on the individual benchmark test problems averaged over all population sizes. (b): Pool size sensitivity analysis - showing the collective cost (for all test problems) accrued by the standard EC and Dual-Pool EC models across varying pool sizes.

(Figure 5.8b) show that the proposed algorithm performs best with pool sizes of 50 to 100. As expected, this shows the ability of the proposed framework to find its way to the required optimal solution using only small to medium sized pools.

Another noteworthy observation is that the variability in the performance characteristics of the two algorithms, across all the benchmark problems, shrinks as the pool size increases from 20 to 1000. This is indicated by the diminishing size of the standard error bars in Figure 5.7. Whilst this fall in variability is a vital outcome, the larger pool sizes have also led to an increase in the number of function evaluations required to converge to the specified accuracy. This effect is observed from Figure 5.8b, which shows that the two algorithms converge with fewer function evaluations when a pool size of 50 to 100 is utilised.

Overall, for the traditional benchmarks (Figure 5.7(a to c)), the convergence summary graph (see the Rastrigin to Easom bar plots in Figure 5.8a) reveals that while both algorithms have found the required optimal solution, they also compete head-to-head on the basis of convergence efficiency.

Secondly, the plots in Figure 5.7(d to f) compare the two algorithms on the modified benchmarks, namely Rastrigin2, Sphere2 and Hybrid. Although these modified benchmarks require a considerably higher amount of function evaluations, it was found that (Figure 5.7(d to f) and Figure 5.8a) the dual-pool model
has always converged to the optimum solution faster and mostly with a statistically significant difference as compared to the standard EC model. In fact, out of the 100 repeated runs, the standard EC model has consistently missed the optimal solution when run with pool sizes of 50 and 1000 on the Sphere2 benchmark (Figure 5.7e), and pool sizes of 50, 100, 500 and 1000 on the Hybrid benchmark (Figure 5.7f). While this is noticeable from the bar graphs of the standard EC algorithm reaching the function evaluations limit on both Figure 5.7(e and f), it is best seen from the complete results in Table C.1 of Appendix C.

Further, while the bar plots in Figure 5.8a reveal that the performance of both the two algorithms is clearly affected by the increased complexity of the test problems, i.e., from the simplest of the traditional benchmarks (Rastrigin) to the most difficult Hybrid composition benchmark; Figure 5.8b shows that the dual-pool EC significantly minimises the overall computational cost when run with small to medium pool sizes of 50 to 100. Importantly, even on the larger pool sizes (200 to 1000), the proposed model still maintains lower function evaluations as compared to the standard EC model. This finding is rather surprising given that the proposed dual-pool EC model is designed to enhance performance on small to medium pool sizes; it was not expected to compete as efficiently when a large pool size is utilised.

5.6.5 Discussion

The earlier review on diversity control policies (Section 5.3) revealed that use of multipopulation-based evolutionary algorithms is not entirely novel. However, the proposed criterion upon which a pool of newly restarted samples dynamically interacts with a dedicated evolution pool suggests a new framework. The approach allows harnessing the benefits of an asynchronous multipool architecture (Figure 5.2) while avoiding its intractable inter-population communications difficulties. Conventional approaches in the literature mostly involve concurrent and continuous runs of the multiple pools. Such approaches often result in a severe increase in function evaluations, which increased their overall computational cost.

Equally, from the preceding overview on various proposals on heuristic initialisations (Section 5.3.1), it might seem that such techniques may not have any impact on diversity control. However, from a number of investigations (Morrisson, 2003; Tometzki and Engell, 2011) conclusions have been drawn that these strategies can significantly improve the statistical significance of the final results.
by minimising stochastic effects (i.e., variability) in the overall behaviour of the resulting EC model. In fact, the observed improvements in diversity control exhibited by the dual-pool EC model (Figure 5.6d) could partly be credited to the proposed SSP heuristic initialisation (Section 5.4.5). The SSP seeks to minimise stochastic variability in the final solution by encouraging optimum uniformity in the distribution of the randomly created sample solution points (cf. Figure 5.3).

The earlier section on diversity control (Section 5.3) has made it clear that diversity management and control in EC is a challenging task. It seems that adherence to any solo strategy risks destabilising the fragile exploration-exploitation balance. Looking back at the challenges of each of the previously reviewed strategies in Section 5.3, it is thought that a multifaceted approach to designing a diversity control policy is vital for developing a successful EC model.

In particular, the above findings may support the following two remarks. Firstly, simulation results (Figure 5.8) have shown that the dual-pool EC algorithm converges to the optimal solutions on both categories of benchmarks with only small to medium pool sizes. This justifies its ability to maintain and restore useful diversity into its search pool, which suggests that it is robust. This robust performance also validates the efficacy of its diversity dynamics previously observed in Figure 5.6d. Secondly, since working with small sized populations often translates to reduced computational cost, the ability of the proposed model to sustain evolutionary search with small to medium sized pools signals its potentials in improving convergence efficiency.

5.7 Comparison with the TPC-GA algorithm from CEC2013 Competition

This section evaluates the performance of the proposed dual-pool EC model on the benchmarks used for the CEC2013 competition on real-parameter optimization. Several global optimization algorithms have participated in this competition (Liang et al., 2013; Loshchilov et al., 2013), but for the comparison here, the following EA which features diversity controlled mechanisms is of interest:

- TPC-GA algorithm (Elsayed et al., 2013) – a diversity controlled EA featuring a three-parent crossover, a diversity operator with archiving and multi-population diversity control techniques.
5.7.1 CEC2013 Benchmarks and Evaluation Criteria

There are a total of 28 (F1 to F28) benchmarks for the CEC2013 competition on real-parameter optimization (Liang et al., 2013). The first five have local orientation, the next fifteen are multimodal functions while the remaining eight are composition functions. The specifications for the benchmark functions (Liang et al., 2013) also provided their exact optimal solutions. Therefore, the evaluation criteria require optimizing all the benchmarks (F1 to F28) for \( n \in \{10, 30, 50\} \) dimensions in 51 repeated runs. Then for each run, 11 optimization error values

\[
\text{error} = (f_i(x) - f_i^*(x))
\]  

are recorded after \((0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0) \times \text{MaxFES}\) function evaluations, where \(f_{\text{error}}\) is the error between the current best solution \((f_i(x))\) and the true optimal solution \((f_i^*(x))\), \(\text{MaxFES} = 10,000 \times n\) is the maximum limit on function evaluations and \(n\) is the problem’s dimension. For each problem a summary statistics for the mean values of \(f_{\text{error}}\) are reported over 51 runs. Further details on the problem specifications can be found in Liang et al. (2013).

5.7.2 Results and Discussion

The detail design and parameter settings for the TPC-GA algorithm including the evaluation results are provided in Elsayed et al. (2013); Loshchilov et al. (2013). For the dual-pool EC algorithm, the parameter settings are as previously presented in Table 5.1, except that a pool size of 100 is used for the 10 dimensions and 200 for the 30 and 50 dimensions of these benchmarks.

To compare the dual-pool EC (DP-EC) and the TPC-GA algorithms, Table 5.3 presents the resulting average optimization errors \((f_{\text{errors}})\) achieved by each of these algorithms on the F1 to F28 benchmarks – over the optimization period of \(\text{MaxFES} = 10000 \times n\) function evaluations. Note that all the benchmarks in this CEC2013 competition are formulated as minimisation problems, thus, the optimization goal here is to minimise \(f_{\text{error}}\). Best results are highlighted in bold except where there is no significant difference or there is tie. For all the 28 test problems, the last row in Table 5.3 summarises the overall score and the percentage score of each of the two algorithms for the 10, 30 and 50 dimensions.

Firstly, for the unimodal benchmarks (F1 to F5), it is observed from the highlighted (bold face) values in Table 5.3 that the TPC-GA algorithm exhibits its
Table 5.3: Comparing the average optimization errors of the Dual-pool EC (DP-EC) algorithm and the TPC-GA algorithm on various categories of the CEC2013 Benchmarks for Real-parameter Optimization. Optimization period is $10000 \times n$ evaluations. Results are averages of 51 repeated runs. Best results are highlighted in bold except for ties.

<table>
<thead>
<tr>
<th>Problem type</th>
<th>Func.</th>
<th>Average Optimization Errors ($f_{errors}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dimensions $n = 10$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DP-EC</td>
</tr>
<tr>
<td>Unimodal</td>
<td>F1</td>
<td>0.00e+0</td>
</tr>
<tr>
<td></td>
<td>F2</td>
<td>6.35e-5</td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>1.92e-2</td>
</tr>
<tr>
<td></td>
<td>F4</td>
<td>1.91e-5</td>
</tr>
<tr>
<td></td>
<td>F5</td>
<td>1.49e-7</td>
</tr>
<tr>
<td>Multimodal</td>
<td>F6</td>
<td>1.42e+0</td>
</tr>
<tr>
<td></td>
<td>F7</td>
<td>3.03e-1</td>
</tr>
<tr>
<td></td>
<td>F8</td>
<td>2.03e+1</td>
</tr>
<tr>
<td></td>
<td>F9</td>
<td>1.26e+0</td>
</tr>
<tr>
<td></td>
<td>F10</td>
<td>9.76e-3</td>
</tr>
<tr>
<td></td>
<td>F11</td>
<td>2.22e-1</td>
</tr>
<tr>
<td></td>
<td>F12</td>
<td>5.34e-1</td>
</tr>
<tr>
<td></td>
<td>F13</td>
<td>3.62e+0</td>
</tr>
<tr>
<td></td>
<td>F14</td>
<td>1.45e+2</td>
</tr>
<tr>
<td></td>
<td>F15</td>
<td>3.00e+2</td>
</tr>
<tr>
<td></td>
<td>F16</td>
<td>8.75e+0</td>
</tr>
<tr>
<td></td>
<td>F17</td>
<td>1.03e+1</td>
</tr>
<tr>
<td></td>
<td>F18</td>
<td>1.04e+1</td>
</tr>
<tr>
<td></td>
<td>F19</td>
<td>1.01e+2</td>
</tr>
<tr>
<td></td>
<td>F20</td>
<td>1.03e+2</td>
</tr>
<tr>
<td>Composition</td>
<td>F21</td>
<td>5.00e+2</td>
</tr>
<tr>
<td></td>
<td>F22</td>
<td>2.46e+2</td>
</tr>
<tr>
<td></td>
<td>F23</td>
<td>4.48e+2</td>
</tr>
<tr>
<td></td>
<td>F24</td>
<td>1.88e+2</td>
</tr>
<tr>
<td></td>
<td>F25</td>
<td>1.92e+2</td>
</tr>
<tr>
<td></td>
<td>F26</td>
<td>1.39e+2</td>
</tr>
<tr>
<td></td>
<td>F27</td>
<td>3.29e+2</td>
</tr>
<tr>
<td></td>
<td>F28</td>
<td>2.68e+2</td>
</tr>
</tbody>
</table>

Total score (%Score): 14(50.0%) 12(42.9%) 19(67.9%) 8(28.6%) 19(67.9%) 8(28.6%)
best optimization performance on the lower dimensions (10 dimensions) of these benchmarks. This suggests that TPC-GA successfully exploits the global convex structure in these unimodal problems. Notice however that this performance is barely carried onto the higher (30 and 50) dimensions of these unimodal benchmarks. In fact, for the F2, F3 and F4 benchmarks, the TPC-GA’s performance deteriorates with increasing dimensionality. Relatively, the proposed algorithm (DP-EC) have lower performance on the 10 dimensions of these unimodal benchmarks. However, the DP-EC appears to maintain its convergence characteristics even when the problem sizes (for the unimodal benchmarks) increase to 30 and 50 dimensions.

Secondly, on the multimodal benchmarks (F6 to F20), relative to the proposed DP-EC algorithm, the TPC-GA algorithm performed consistently better on the F6, F16 and F19 over the varying problem dimensions. However, with the exception of F8 where the performance differences are insignificant, the DP-EC algorithm have the overall lead on these multimodal benchmarks. Crucially, the success of DP-EC largely comes from its ability to maintain good performance on larger problem sizes (30 and 50 dimensions).

Finally, as observed from the composition benchmarks (F21 to F28), there is significant performance breakdown for both algorithms. This reflects the high level of difficulty posed by these category of benchmarks. It is noticed that the performances of the TPC-GA and DP-EC algorithms seem uniform over the varying problem dimension in this category. This suggests that none of them seem to effectively exploit the lower dimensional versions of these composition benchmarks. Nevertheless, the proposed DP-EC yields better results, overall.

The summary of the above results (see the last row of Table 5.3) suggests that the TPC-GA algorithm enjoys good exploitation capabilities which seems to have led to its higher success rate on lower dimensional unimodal problems (F1 to F5). This is reflected by its percentage score of 42.9% on the 10 dimensional benchmarks in general. On the other hand, whilst the dual-pool EC algorithm (DP-EC) is not as exploitative, its diversity maintenance scheme seems more effective as exhibited by its higher success rate (67.9% score) on large sized problems.
CHAPTER 5. DIVERSITY CONTROL IN EC

5.8 Contribution and Remarks

This chapter has introduced an approach that aims to control diversity in evolutionary computation (EC) algorithms. The proposed model attempts to ameliorate the challenges associated with balancing exploration and exploitation trade-off by jointly using a restarted dual-pool strategy with a novel heuristic initialisation policy. The insights obtained from the investigations in this chapter have paved the way for the development of the proposed dual-pool EC architecture.

The search space partitioning (SSP) paradigm presented in this chapter was a direct consequence of the need to minimise sporadic sampling of search spaces by random initialisation. The proposed SSP was designed to address such problems which lead to over- or under-sampling of potentially crucial portions of the search space when random initialisation is utilised. The SSP was then used to restart the samples in the diversity pool which together with the evolution pool form a dual approach to diversity restoration. The newly proposed model utilises an adaptive control mechanism to profile the instantaneous diversity in a search pool. The mechanism relies on diversity measurement to trigger re-diversification whenever sufficient convergence is detected. The diversity measurement couples spatial diversity measure with population evolvability measure.

The effectiveness of the proposed model in mitigating premature loss of population diversity has been empirically examined under various categories of global optimization benchmarks (Sections 5.6 and 5.7). Experimental investigations have revealed that with enhanced diversity control, complex and rugged global optimization landscapes could be explored with limited population sizes (50 to 200 samples). This feature, which corroborates the key objective in this chapter, is vital for minimising the cost of solving computationally expensive problems.

In the second part of this thesis (Part II), the evolutionary algorithms investigated thus far\(^\text{14}\) will be put into a hybridization framework. This entails combining an evolutionary algorithm with a local search method. The intuition is that applying hybridization could improve the exploitation capabilities and the overall convergence efficiency of the EC model proposed in this chapter.

\(^{14}\text{The algorithms include the standard EC (Algorithm 2.2) and the newly proposed dual-pool EC model (Algorithm 5.2).}\)
Part II

Hybridizing Evolutionary Computation
Chapter 6

Hybridizing Evolutionary Computation

This chapter proposes a new hybrid algorithm for solving small to large-scale continuous global optimization problems. It comprises of an evolutionary computation (EC) algorithm, previously introduced in Chapter 2, and a sequential quadratic programming (SQP) algorithm (Chapter 3). The algorithms are combined in a collaborative manner with a validation procedure. The proposed hybrid design aim is to ensure that the two algorithms complement each other by effectively exploring and exploiting the problem search space.

Therefore, the goal in this chapter is to verify the following hypotheses:

**#H1:** A hybrid of global and local search methodologies whose switching is controlled through a robust convergence detection mechanism should speed-up convergence to the optimal solution.

**#H2:** The global and local algorithms can serve as a means to validate each other’s result.

This chapter presents the rationale for the development of effective optimization methods using a hybrid approach, it also highlights the current trends in hybridizing evolutionary algorithms. The chapter introduces a taxonomy of EC-based hybrid frameworks which are recently applied to a variety of global optimization problems (Section 6.2). Then, the chapter proposes a task switching method which combines the EC algorithm with a SQP algorithm in Section 6.3. This chapter concludes with two sets of experiments. The first (Section 6.4) analyses the performance characteristics of the proposed hybrid algorithm against
other standard evolutionary algorithms (EAs). The second (Section 6.5) evaluates the overall performance of the proposed method by comparing it with other state-of-the-art hybrid EAs.

6.1 Rationale

In the last few decades, it has become well understood that population based search methods like the evolutionary algorithms (EAs) (Goldberg, 1989; Krasnogor and Smith, 2005) are effective in exploring search spaces even when faced with problems having high-dimensionality, non-convexity, and multimodality. This is because as early as during their initialisation stage, these population-based heuristics set out to capture a global perspective of the search space. Eventually over the course of the evolution, they try to focus on the most promising regions of the search space. However, EAs are generally not efficient in converging to the best solutions in these high quality regions (Blum et al., 2011; Raidl, 2003). Conversely, local search methods (such as gradient-based algorithms) are generally more effective in exploiting specific regions of the search space, i.e., they can easily converge to better solutions in the vicinity of any given initial solution.

Furthermore, it has become evident that many real-world large scale optimization problems elude acceptable solutions via simple exact methods or even the heuristic approaches when applied independently (Pelikan, 2010). In recent years, there is growing interest in concepts that combine various algorithmic ideas from different branches of artificial intelligence and operations research. The combination of such algorithms is what is referred to as *hybrid algorithms*, *hybrid metaheuristics* (Blum and Roli, 2008), or *memetic algorithms* (MA) (Krasnogor, 2002; Radcliffe and Surry, 1994) in another linguistic. A primary motivation for hybridizing algorithms was to come up with robust systems that combine the benefits of the individual algorithms while discarding their inherent weaknesses.

Despite their popularity, the vast literature on hybridization frameworks requires more detailed categorisation regarding: the nature of the problem domain, the constituent algorithms, the coupling schema and the intended area of application. Therefore, prior to the design and development of any hybrid system, it is imperative to address the following concerns in order to ascertain whether a hybrid system is needed, and if so, which kind of hybridization approach is suitable for the problem under consideration; the issues are:
A clear understanding of the type of problem at hand and based on the optimization goal, deciding to use only a heuristic, an exact or a hybrid of such algorithms. Typically, for simple convex problems of lower dimensions, local algorithms like the gradient based methods suffice. Also, when solution quality and computational time are not critical, then stochastic heuristics like the evolutionary algorithms are usually sufficient for many of the low to medium sized non-convex problems. Thus, in most cases, only when very good solutions (unobtainable by an exact or heuristic method in a feasible time frame) are needed, use of hybrid algorithms is advised.

Determining what algorithms to combine and which type of combination of these algorithms might work well for the class of problem at hand and why.

Ascertaining what role enhancing the capabilities of the individual algorithms can play to the success of the hybrid system.

Determining how to effectively fine tune the hybridized system to optimality.

Unfortunately, not all of the above questions have direct or simple answers. In fact, a fundamental goal in the previous chapters (Part I of this thesis) has been to select and optimize the constituent algorithms for the hybrid system. Therefore, the objective in this chapter is to investigate current hybrid methodologies and propose a new approach to combining an evolutionary computation (EC) algorithm with the sequential quadratic programming (SQP) local search algorithm in a collaborative framework. The overall aim is to yield an efficient and robust algorithm suitable for medium to large scale global optimization problems in continuous domain.

6.2 Background

It is probably not possible to exhaustively enumerate the various types of hybrid algorithms in the EC literature (Blum and Roli, 2008) as the notion of hybridization in itself lacks a precise definition. It seems that there is no specific framework that clearly defines what should constitute the so-called hybrid algorithms. Although this may sound like a drawback, it is actually thought of as the reason behind the breakthroughs achieved with such systems thus far. As argued by
Blum et al. (2011), lack of precise boundary in the area of hybrid algorithms is what made the research field very rich and versatile.

Generally, designing successful hybrid frameworks requires making judicious decisions on a huge number of important considerations. The classification in Figure 6.1 aims to provide concise yet comprehensive insights into the many crucial aspects in designing hybrid evolutionary computation systems. Out of the eight major aspects in this classification (Figure 6.1), the first four could be considered as the primary considerations for building hybrid systems. They involve selecting the type of algorithms that compose the hybrid system, the switching/control method, the execution mode and the extent or degree to which these algorithms are coupled.

The remaining four aspects (Figure 6.1) which involve inheritance mode, use of domain knowledge and parameter control (adaptive and non-adaptive) may be considered secondary. However, these features could play vital role on how an evolutionary-based hybrid system fares on rugged and complex global optimization landscapes. The inheritance mode dictates how the learned information acquired through a local refinement algorithm get inculcated into the EA. Thus, for the Baldwinian inheritance model, the locally refined solution only alters the individual’s fitness (phenotype). However, the Lamarckian evolution changes both the phenotype and the genetic structure of an individual. Similarly, the incorporation of domain specific knowledge is often useful when a problem domain is well-defined. This may involve, for example, using problem dependent representation which may consequently require application of specialise evolutionary operators for reproduction. Last but not the least is parameter control; this involves issuing fixed specifications for the (hyper-)parameters, or using some form of adaptation (dynamic refinement) on some (or all) of the crucial parameters. Adaptation may lie within the evolutionary model itself (such as adapting operator step sizes and/or rates), and/or on the overall hybridization framework (such as dynamic adjustments to the rate and intensity of applying the local refinement).

While the above taxonomy may not provide an exhaustive list of all potentially crucial design considerations on building hybrids, it generally covered all major aspects of the evolutionary hybridization paradigm. Further related considerations can be found in a notable work by Raidl (2006). In this perspective, it is thought that a three-point view of classifying hybrid approaches entails the
CHAPTER 6. HYBRIDIZING EC FOR GLOBAL OPTIMIZATION

Figure 6.1: Crucial Considerations for Building Hybrid Models

following:

i. Hybrid metaheuristics: Though this could be seen as a superset of the forthcoming hybridization classes, it particularly refers to the combination of two or more heuristic algorithms. Thus, it involves combining various nature inspired algorithms like EAs and/or non-nature inspired algorithms like tabu
search, iterated local search etc. The motivation is to maximise the coverage of the search space for optimal exploration of the promising regions. Under this class, a common way of hybridizing EAs is through:

(a) the use of problem specific representation (Queiroz and Lyra, 2009);
(b) the use of additional genetic operators (Lin and Yang, 1999); and
(c) incorporation of domain specific knowledge or features of classical algorithms (Oh et al., 2004).

ii. Memetic Algorithms: May be seen as a subset of the above class, but they uniquely consist of combination of heuristic algorithms like EAs with local search methods like gradient based algorithms (Chen et al., 2011; Lin and Chen, 2011; Radcliffe and Surry, 1994). The main focus of this class is to facilitate rapid convergence to the optimum solution (i.e., exploitation of the promising regions) as soon as they are explored.

iii. Algorithms portfolio: This is based on the intuition that executing many short runs of one or more algorithms over a prescribed solution period can provide improvements in terms of overall performance. Gomes and Selman (1997) formulated the computational cost of a portfolio as a random variable having a probability distribution and evaluated its mean and standard deviation. They argue that a skilful schedule of multiple copies of a single algorithm can outperform a hybrid built from different algorithms. Recently, Streeter and Smith (2009) argued that a portfolio essentially comprises both scheduling and machine learning aspects and divided the scheduling aspect into either a restart or task-switching schedules. Other important advances in this direction include the proposal of a *multimethod search* algorithm (popularly known as AMALGAM) by Vrugt and Robinson (2007), a recent application of this method to noisy environment can be found in Bosman et al. (2013).

It is noteworthy that each of the above classes has its merits and demerits. The first class favours exploration of problem space at the expense of exploiting high quality regions. Although the second class tried to alleviate this problem, a naive design of a memetic algorithm that is overly focused on exploitation may risk premature convergence to sub-optimal solutions. Finally, the additional parameter in the third class (i.e., the scheduling aspect) is largely handled in a trial and error procedure.
Table 6.1: A chronological survey of some recent hybrid algorithms (across various applications domain) aligned to the taxonomy depicted in Figure 6.1.

<table>
<thead>
<tr>
<th>No.</th>
<th>Reference</th>
<th>Global algorithm</th>
<th>Local algorithm</th>
<th>Switching category</th>
<th>Key design goal</th>
<th>Validation/Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Radcliffe and Surry (1994)</td>
<td>GA</td>
<td>Hill climber</td>
<td>Integrative</td>
<td>Efficiency</td>
<td>TSP problems</td>
</tr>
<tr>
<td>2</td>
<td>Honggang and Jian-chao (1997)</td>
<td>GA</td>
<td>Feasible path method</td>
<td>Collaborative</td>
<td>Accuracy</td>
<td>Numerical problems</td>
</tr>
<tr>
<td>3</td>
<td>Tao (2004)</td>
<td>GA</td>
<td>Back propagation</td>
<td>Integrative</td>
<td>Efficiency</td>
<td>Fuzzy control</td>
</tr>
<tr>
<td>4</td>
<td>Oh et al. GA (2004)</td>
<td>GA</td>
<td>Search algorithm for feature selection</td>
<td>Integrative</td>
<td>Efficiency</td>
<td>Feature selection</td>
</tr>
<tr>
<td>5</td>
<td>Tenne and GA Armfield (2005)</td>
<td>GA</td>
<td>Derivative-free optimizer</td>
<td>Integrative</td>
<td>Accuracy</td>
<td>Numerical Problems</td>
</tr>
<tr>
<td>6</td>
<td>Isaacs et al. GA (2007)</td>
<td>GA</td>
<td>Simplex algorithm</td>
<td>Collaborative</td>
<td>Accuracy</td>
<td>Numerical Problems</td>
</tr>
<tr>
<td>7</td>
<td>Qing et al. GA (2008)</td>
<td>GA</td>
<td>Tabu search</td>
<td>Collaborative</td>
<td>Accuracy</td>
<td>Fuzzy scheduling problem</td>
</tr>
<tr>
<td>8</td>
<td>Kaur and GA Murugappan (2008)</td>
<td>GA</td>
<td>Nearest neighbour search</td>
<td>Collaborative</td>
<td>Accuracy/Efficiency</td>
<td>TSP problems</td>
</tr>
<tr>
<td>9</td>
<td>Hernandez-Diaz et al. NSGA II (2008)</td>
<td>GA</td>
<td>Steepest descent</td>
<td>Collaborative</td>
<td>Accuracy/Efficiency</td>
<td>Multiobjective optimization</td>
</tr>
<tr>
<td>10</td>
<td>Queiroz and Lyra GA (2009)</td>
<td>GA</td>
<td>Branch-exchange procedure</td>
<td>Integrative</td>
<td>Accuracy</td>
<td>Power distribution network</td>
</tr>
<tr>
<td>11</td>
<td>Guo et al. GA (2010)</td>
<td>GA</td>
<td>Interval search</td>
<td>Collaborative</td>
<td>Accuracy/Efficiency</td>
<td>Interval optimization</td>
</tr>
<tr>
<td>12</td>
<td>Pelikan (2010)</td>
<td>GA</td>
<td>Bayesian optimization hill climber algorithm</td>
<td>Integrative</td>
<td>Accuracy</td>
<td>NK landscape problems</td>
</tr>
</tbody>
</table>

The survey in Table 6.1 summarises some of the recent hybrid evolutionary-based optimization algorithms from various perspectives. From the primary considerations in building hybrids delineated by the taxonomy in Figure 6.1, Table
CHAPTER 6. HYBRIDIZING EC FOR GLOBAL OPTIMIZATION

6.1 covers aspects ranging from the nature/type and class of the combined algorithms, the adopted control/switching mode, the validation method employed and the intended application area. Note that the summary given here is only an attempt to highlight the large number of hybrid algorithms, reported over the last decade, in the field of evolutionary-based global optimization. Thus, Table 6.1 attempts to capture varieties of local search algorithms combined with EAs to enhance either the efficiency or accuracy of an optimization method for various application areas or problem domain.

6.3 The Proposed Task-Switching Hybrid EC Model

In order to investigate the efficacy of combining a global explorer (see the EC model in Algorithm 2.2) with a local exploiter (see the SQP local search algorithm in Algorithm 3.1) for global optimization, this section introduces a hybrid model that combines its constituent algorithms in a collaborative manner. Whilst an improved EC model such as the newly introduced dual pool model (see chapter 5) could be employed for this purpose, the rationale behind using a canonical EC model (Algorithm 2.2) in this chapter is to examine the impact of the proposed hybrid framework in isolation. Thus, the analysis herein aims to rule out the effect of any specialised features in the EC model.

Notice from the proposed hybrid framework shown in Algorithm 6.1 that the combined algorithms run in a sequential task-switching mode. The two algorithms retain their individual functionalities such that they complement each other by operating independently on the problem via data/information exchange.

Initially (cf. Algorithm 6.1), as a population based method, the EC algorithm is invoked with a randomly created initial population $P_0(t)$ of size $N$ from which the best solution ($x_{EC}$) found by the EC algorithm is derived (lines 3-4). Thus, this puts the EC algorithm in a position to provide the driving force for wide exploration of the search space. When the high quality regions are found and the EC algorithm has sufficiently converged\(^1\) to these areas, the search switches to the SQP algorithm which is fed with $x_{EC}$ as its initial solution (line 5). This switching to the local algorithm triggers exploitation of the highest quality region

\(^1\)The population evolvability measure proposed in Chapter 4 is utilised for the convergence detection.
Algorithm 6.1 The proposed hybrid EC/SQP algorithm

1: begin
2: initialise $t \leftarrow 0$;
3: initialise search pool $P_0(t)$: size = $N$;
4: while not converged // see convergence measure in Section 4.6.4
5:   Do Global Search: $x_{EC} \leftarrow$ invoke $EC(P_0(t))$; // (§2.6.6, Algorithm 2.2)
6: end while
7: Do Local Search: $x_{SQP} \leftarrow$ invoke $SQP(x_{EC})$; // (§3.5, Algorithm 3.1)
8: re-initialise search pool $P_1(t)$: size = $N - 2$;
9: validate solution: $x^* \leftarrow$ invoke $EC(P_1(t) \cup x_{SQP} \cup \bar{x}_{SQP})$;\footnote{Note: $\bar{x}_{SQP}$ is a mutated copy of the obtained global solution $x_{EC}$.}
10: end

previously explored by the EC algorithm.

In addition, since there is often a slight tendency for the final solution returned by the local search algorithm to be a sub-optimal one\footnote{This could happen when the EA has not properly explored the global optimum region and the local algorithm is therefore not supplied with a good starting solution.}, a validation routine is utilised to kick-start an additional round of a global search by the EC algorithm (lines 6-7). While in the validation routine, the initial population is also randomly created and is then seeded with a copy of the solution ($x_{SQP}$) returned by the local algorithm and its mutated version $\bar{x}_{SQP}$. Thus, if the required population size is $N$, then, $N - 2$ individuals are created randomly for the validation (line 6). Eventually, the search stops and returns $x^*$ as the optimal solution at the end of the validation process (lines 7). Notice that the validation routine involves only a re-run of the global EC algorithm, i.e., the local optimization – via the SQP algorithm – is not repeated.

6.4 Experiments with Standard EAs on Global Benchmarks

The proposed hybrid algorithm is evaluated by applying it on a set of numerical optimization benchmarks. See Table 6.2 for the parameter settings in these experiments. Of course, prior to evaluating any global optimization algorithm, it is important to seek a suite of test problems that would ensure effective evaluation. Thus, a suitable global optimization test suite should satisfy some crucial qualities which include the following properties:
Table 6.2: Parameter settings of the proposed hybrid algorithm

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Parameters</th>
<th>Symbols</th>
<th>Values/settings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Representation</td>
<td>–</td>
<td>Real-valued encoding</td>
</tr>
<tr>
<td>Global (EC)</td>
<td>Population size</td>
<td>$N$</td>
<td>100</td>
</tr>
<tr>
<td>(Algorithm 2.2)</td>
<td>Selection method</td>
<td>–</td>
<td>Binary tournament selection</td>
</tr>
<tr>
<td></td>
<td>Crossover: type, rate</td>
<td>$X, P_C$</td>
<td>Intermediate crossover, $P_C = 1.0$</td>
</tr>
<tr>
<td></td>
<td>Mutation: type, rate</td>
<td>$M, P_M$</td>
<td>BGA mutation, $P_M = 0.01$</td>
</tr>
<tr>
<td></td>
<td>Replacement scheme</td>
<td>–</td>
<td>Adaptive elitist strategy (§2.6.6)</td>
</tr>
<tr>
<td></td>
<td>Generation gap</td>
<td>$\omega$</td>
<td>$\omega = 5%$ of the population size $N$</td>
</tr>
<tr>
<td></td>
<td>Max. generations</td>
<td>MaxGen</td>
<td>when: counter $t \geq$ MaxGen</td>
</tr>
<tr>
<td></td>
<td>Stalled generations</td>
<td>–</td>
<td>when: convergence occurs (§4.5)</td>
</tr>
<tr>
<td></td>
<td>Search direction type</td>
<td>$d$</td>
<td>Newton directions (§3.3.1)</td>
</tr>
<tr>
<td></td>
<td>Search direction evaluation</td>
<td>–</td>
<td>Exact Hessians using Automatic differentiation $H(f(x)_k) = \nabla^2 f(x)$ (§3.7)</td>
</tr>
<tr>
<td>Local (SQP)</td>
<td>Step size</td>
<td>$\alpha$</td>
<td>$0 &lt; \alpha \leq 1$ using Wolf conditions (see §3.3.2)</td>
</tr>
<tr>
<td>(Algorithm 3.1)</td>
<td>Stopping criteria</td>
<td>–</td>
<td>when: gradient vanishes: $</td>
</tr>
</tbody>
</table>

Notation: $f =$ fitness, $t =$ generation or iteration counter.

- Nonlinearity;
- Scalability;
- Non-separability;
- Multimodality/non-convexity;
- Asymmetry; and
- Dispersion.

While the meanings of most of the above features can be directly derived from their names, with regards to the dispersion metric, a function having no convex-like global topology is considered highly dispersed. As demonstrated by Lunacek and Whitley (2006), high dispersion functions pose more difficulty to global optimization algorithms than ordinary multimodal functions having convex-like
global structure. Detailed review and analysis of the above features for global test problems can be found in Whitley et al. (1996).

To evaluate the performance of the proposed hybrid algorithm (cf. Algorithm 6.1), series of tests have been carried out on the Rastrigin ($f_1$), Schwefel ($f_2$) and Ackley ($f_7$) benchmark functions\(^3\) for global optimization. Although these test functions are all multimodal and have the majority of the qualities listed above, if their complexity is sorted based on levels of dispersion, the Ackley function is the least and the Schwefel function is the most difficult.

Two different sets of experiments were carried out. The first experiment investigates how the proposed hybrid algorithm behaves under increasing problem size. But the second aims to evaluate the performance of the hybrid algorithm (EC/SQP) by comparing it with a standard evolutionary computation (EC) algorithm and the well known covariance matrix adaptation algorithm (CMA-ES) on the three benchmark problems presented above. For the EC algorithm, elitism (Radcliffe and Surry, 1994) is utilised with other parameters as depicted in Table 6.2. Similarly, for the CMA-ES algorithm, the default parameters are used (see Hansen and Kern (2004) for details). For all experiments, a maximum of 30,000 function evaluations is allowed. All tests are run 100 times and averaged results are reported for statistical significance.

### 6.4.1 Experiment 1: Scalability Test

With the aim of evaluating the robustness of the proposed EC/SQP algorithm under increasing problem size, these experiments seek to optimize the global optimization benchmark problems described above with their problem sizes set to 2, 10 and 100 dimensions. The results of this experiment as shown in Figure 6.2 compare the fitness characteristics of the proposed hybrid model on (a) Ackley, (b) Rastrigin and (c) Schwefel functions respectively.

### 6.4.2 Experiment 2: Performance Comparison Test

This is a fitness comparison test where the proposed EC/SQP algorithm is compared to a standard EC algorithm and the CMA-ES algorithm, see Figure 6.3. Also, the goal is to optimize the above three benchmark test functions with each of the three algorithms allowed to run for 30,000 function evaluations.

\(^3\)Detail expressions for the Benchmark functions are provided in Table B.1 in Appendix B.
The results for experiments 1 and 2 are respectively shown in Figures 6.2 and 6.3 and the following sections analyse the outcomes of these investigations.

6.4.3 Analysis of Results: Experiment 1

A careful examination of Figure 6.2 reveals that for virtually all the varying sizes of the three test problems, up to 90% of the maximum attainable fitness level (i.e., the optimal solution) was reached within the first 5,000 functions evaluations. This is indicative of the speed at which the EC/SQP algorithm approaches the optimal solution – in other words, its convergence efficiency. Notice, however, that with the exception of the Ackley function in Figure 6.2(a), an increase in problem size increases the required function evaluations to reach the optimum solution. This shows that to some extent, larger problem sizes might inhibit the performance of the proposed algorithm. But this is consistent with the fact that an increase in problem size widens the search space (curse of dimensionality) making it harder to explore and narrow down to the region of the optimum solution. Yet, the observed performance decrease is rather minor considering the 10 times increase in the problem size.

A possible explanation of the discrepancy shown in Figure 6.2(a) by the Ackley function ($f_7$) might be related to the fact that it is a pseudo-convex function with several local optima induced via its cosine component (see Table B.1 in Appendix B). These local optimum valleys become shallow and smooth as its dimensionality increases. Thus higher dimensions of Ackley functions are smoother and seem easier to optimize. A similar behaviour exhibited by Griewangk benchmark function was reported by Whitley et al. (1996).

Overall, these test results have shown that the proposed hybrid EC/SQP algorithm appears to maintain good and fairly stable convergence efficiency across small to large scale problems. Hence, this outcome supports the suppositions made while introducing this chapter, that hybrid methods are robust to varying problem sizes, categories and/or levels of complexities.

6.4.4 Analysis of Results: Experiment 2

The results of this experiment are shown in Figure 6.3. For the three test problems under consideration, a critical analysis of the performance of each of the three algorithms requires cautious treatment. This is necessary because although all the
Figure 6.2: Hybrid EC/SQP hybrid algorithm evaluated under increasing problem sizes (2, 10, 100) of: (a) Ackley, (b) Rastrigin and (c) Schwefel benchmarks. The error bars are the standard errors of the mean; all results are averages of 100 independent runs. The plots (best viewed in colour) show that the performance of the EC/SQP algorithm is fairly stable and immune to varying problem sizes.
three test problems are nonlinear, multidimensional, and multimodal in nature; they are quite different in complexity when viewed on the basis of their dispersion. Thus, a possible categorisation of the plots in Figure 6.3 is to place the Rastrigin function (middle row) as a moderately dispersed problem, while the Ackley (top row) and Schwefel (bottom row) test functions are respectively the least and most dispersed problems. Consequently, they stand at the opposite extremes of complexity.

Beginning with the Rastrigin function (Figure 6.3(c and d)), the observed remarkable performance of all the three algorithms on this test problem may be attributed to its moderate level of dispersion. This could be why it seems to be an easy problem for the three different global optimization approaches. Nonetheless, it is noticed that (see the switch point indicating the instance at which the hybrid algorithm switches from global EC to the local SQP algorithm) the proposed EC/SQP algorithm appears to perform best followed by the CMA-ES and then the standard EC algorithm.

With regard to the Ackley function (Figure 6.3(a and b)), it might be noticed that while the CMA-ES and the hybrid EC/SQP algorithms clearly outperform the standard EC algorithm, the excellent efficiency exhibited by the CMA-ES algorithm (Figure 6.3(a)) seem to deteriorate following the 10 times increase in the problem size (Figure 6.3(b)). This, however, is not the case with the EC/SQP algorithm. Hence, it could be hypothesised that the hybrid nature of the EC/SQP algorithm might be the reason behind its immunity to the growth in the problem size.

Note that for all the test cases considered in this experiment (Figure 6.3), the standard EC algorithm always lags behind the other two algorithms except on the Schwefel function (Figure 6.3(e and f)). Surprisingly, this is where the CMA-ES algorithm exhibited its poor performance. This rather difficult to interpret result might have several possible explanations. It could be due to the fact that Schwefel function has the highest level of dispersion among all the three test cases. Therefore, it is a flat multimodal function that lacks any unimodal global topology. It remains unclear though, why high dispersion problems like Schwefel function seem to be easy for the standard EC algorithm. But a possible reason for the sudden decline in the performance of the CMA-ES algorithm may be due to its excessive evaluation of infeasible solutions during the early stages of the search process. It is therefore thought that CMA-ES algorithm relies on
Figure 6.3: Comparing fitness plots for Hybrid EC/SQP, standard EC and CMA algorithms. Test Problems (row-wise): Top: Ackley, Middle: Rastrigin, Bottom: Schwefel. Problem sizes (column-wise): Left: 10-Dimensions, Right: 100-Dimensions. The error bars are the standard errors of the mean; all results are averages of 100 runs. These plots show that for virtually all the test problems, the proposed EC/SQP algorithm is always as good if not better than any of the other two algorithms. Experiments are run for 30,000 function evaluations.
exploiting global convexity to be successful. Hence, its remarkable performance on the low dispersion Ackley function (Figure 6.3(a and b)) deteriorates when faced with high dispersion problems like Schwefel.

These results corroborate the findings of previous studies in this field. An investigation on the CMA-ES algorithm by Lunacek and Whitley (2006) revealed that the adaptive step-size heuristic, called *cumulation*, does not function as intended when the best regions of the search space are too spread out (such as in high dispersion problems like the Schwefel function). Nonetheless, it was reported elsewhere (Hansen and Kern, 2004), that although CMA-ES algorithm may need more than $10^5$ function evaluations for such high dispersion problems (which is way outside the budgeted limit of $30,000$ evaluations in these experiments), but it certainly will at some point converge to the true optimal solution if sufficient evaluations are granted.

Noteworthy, from virtually all the plots in Figure 6.3, the moment at which the EC/SQP algorithm switches from the global algorithm (EC) to the local algorithm (SQP) might easily be noticed; this dynamic switch-over is controlled by detecting the instantaneous convergence status of the search pool of the EC algorithm. Thus, exact switching times would vary across problem types and/or across different dimensions of the same problem. For the various benchmarks categories in Figure 6.3, the switch-over points provide evidence for the appreciable contribution of the local algorithm towards the overall speedup in the convergence efficiency of the hybrid EC/SQP algorithm.

In overall, for the three different test problems, the proposed EC/SQP algorithm always performs comparably or better than the standard EC and CMA-ES algorithms. Although some of the differences between the tested approaches (see Figure 6.3) may appear marginal, they are significant to within 99% confidence limit for both the 10 and 100 dimensions of Ackley (Figure 6.3(a and b)) and Schwefel (Figure 6.3(e and f)) test functions. Nevertheless, the test (Wilcoxon ranksum nonparametric test) showed that the samples from the 100 independent runs on the Rastrigin function (Figure 6.3(c and d)) are not sufficient to reject the null hypothesis. Thus, the performance difference on the Rastrigin function is not statistically significant.

Finally, these findings support the hypotheses put forward while introducing this chapter (Section 6); the findings provide additional evidence on the fact that
a skilful design of a hybrid system that combines the strengths of population-based and local search methods can boost the convergence efficiency of a global optimization system. In fact, the results presented herein demonstrate how the proposed approach can effectively complement the traditional techniques for solving global optimization problems of varying complexities.

6.5 Experiments with Hybrid EAs on CEC2013 Benchmarks

This section evaluates the performance of the proposed hybrid EC/SQP algorithm by comparing it with two hybrid algorithms from the CEC2013 competition on real-parameter optimization. Several global optimization algorithms have participated in this competition (Liang et al., 2013; Loshchilov et al., 2013), but the comparison here is focused on those algorithms with hybrid configurations. Thus, the performance of the EC/SQP hybrid algorithm is compared against that of:

i. iCMAES-ILS algorithm (Tianjun and Stützle, 2013) – a hybrid algorithm that couples the IPOP-CMAES algorithm with a new iterated local search (ILS), and

ii. DRMA-LSCh-CMA algorithm (Lacroix et al., 2013) – a dynamically updated region-based memetic algorithm with local search chaining and CMA-ES.

Importantly, although the choice of the above two algorithms, from the 21 reported participants in the CEC2013 competition, is based on the reason that they are both hybrid approaches, these algorithms are also among the top three highest ranked algorithms in the CEC2013 competition (Loshchilov et al., 2013).

6.5.1 CEC2013 Benchmarks and Evaluation Criteria

As earlier reported in Section 5.7.1, there are a total of 28 benchmarks for the CEC2013 competition on real-parameter optimization (Liang et al., 2013). Although the competition is on real-parameter optimization, the problem specifications in Liang et al. (2013) highlight that some of these benchmarks feature discontinuities which means they are not smooth (differentiable) over the entire feasible search space. As a result, since the gradient-based SQP algorithm in the proposed hybrid model requires smooth functions, only the first twelve (F1
CHAPTER 6. HYBRIDIZING EC FOR GLOBAL OPTIMIZATION

172
to F12) test problems are examined in this investigation. In other words, the benchmarks considered here comprised of F1 to F5 (having local orientation), and F6 to F12 (having multimodal landscape).

The evaluation criteria in this competition (Liang et al., 2013) require optimizing all the benchmarks (F1 to F12) for $n \in \{10, 30, 50\}$ dimensions in 51 repeated runs. Then for each run, 11 optimization error values

$$f_{error} = (f_i(x) - f_i^*(x))$$

are recorded after $(0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0) \times \text{MaxFES}$ function evaluations, where $f_{error}$ is the error between the current best solution ($f_i(x)$) and the true optimal solution ($f_i^*(x)$), MaxFES $= 10,000 \times n$ is the maximum limit on function evaluations and $n$ is the problem’s dimension. Then for each problem, a summary statistics consisting of the best, worst, mean, median and standard deviation values of $f_{error}$ are reported for the 51 runs. Finally, it is important to note that for this CEC2013 competition all benchmarks are formulated as minimisation problems. Therefore, the evaluation results presented in this section are of minimisation type.

6.5.2 Results and Discussion

The detail design and parameter settings for the iCMAES-ILS algorithm can be found in Tianjun and Stützle (2013), those for the DRMA-LSCh-CMA algorithm are in Lacroix et al. (2013). The evaluation results for each of these algorithms are obtained from their respective papers and are also reported in Loshchilov et al. (2013). Finally, the parameter settings for the proposed hybrid EC/SQP algorithm are as previously presented in Table 6.2. Note however that for the 30 and 50 dimensional problems, the population size used by the hybrid EC/SQP algorithm has been increased from 100 to 500 samples.

Table 6.3 presents a statistical summary of the optimization error ($f_{error}$) for the hybrid EC/SQP algorithm at the end of the evaluation period (MaxFES), which as mentioned earlier is $10000 \times n$, where $n$ is the problem dimension. Table 6.3 shows the results for the 10, 30 and 50 dimensions of the F1 to F12 benchmarks over 51 independent repeated runs.
Table 6.3: Optimization results for the hybrid EC/SQP Algorithm on the CEC2013 Unimodal and Multimodal Benchmarks at $10000 \times n$ maximum function evaluations. All results are averages of 51 repeated runs.

<table>
<thead>
<tr>
<th>Problem size Func.</th>
<th>Best</th>
<th>Worst</th>
<th>Median</th>
<th>Mean</th>
<th>Std.</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>0.0000e+0</td>
<td>1.4395e-7</td>
<td>0.0000e+0</td>
<td>2.8225e-9</td>
<td>2.0157e-8</td>
</tr>
<tr>
<td>F3</td>
<td>4.6434e-5</td>
<td>1.2701e-1</td>
<td>7.9846e-3</td>
<td>1.9120e-2</td>
<td>2.6999e-2</td>
</tr>
<tr>
<td>F4</td>
<td>4.5048e-6</td>
<td>5.8180e-5</td>
<td>1.5866e-5</td>
<td>1.9103e-5</td>
<td>1.1282e-5</td>
</tr>
<tr>
<td>F5</td>
<td>0.0000e+0</td>
<td>2.2405e-6</td>
<td>1.0780e-8</td>
<td>1.4894e-7</td>
<td>4.1788e-7</td>
</tr>
<tr>
<td>F6</td>
<td>1.2391e-5</td>
<td>9.8148e+0</td>
<td>5.0724e-2</td>
<td>1.4218e+0</td>
<td>3.3861e+0</td>
</tr>
<tr>
<td>Dimensions F7</td>
<td>7.8167e-2</td>
<td>9.6432e-1</td>
<td>2.5153e-1</td>
<td>3.0260e-1</td>
<td>1.9602e-1</td>
</tr>
<tr>
<td>F8</td>
<td>2.0112e+1</td>
<td>2.0371e+1</td>
<td>2.0283e+1</td>
<td>2.0271e+1</td>
<td>6.1175e-2</td>
</tr>
<tr>
<td>F9</td>
<td>5.7980e-2</td>
<td>3.7454e+0</td>
<td>1.0542e+0</td>
<td>1.2566e+0</td>
<td>8.3061e-1</td>
</tr>
<tr>
<td>F10</td>
<td>4.1817e-2</td>
<td>3.2970e-1</td>
<td>1.6230e-1</td>
<td>1.4640e-1</td>
<td>7.2910e-2</td>
</tr>
<tr>
<td>F11</td>
<td>0.0000e+0</td>
<td>6.7650e+0</td>
<td>2.9849e+0</td>
<td>3.3344e+0</td>
<td>1.5687e+0</td>
</tr>
<tr>
<td>F12</td>
<td>9.9496e-1</td>
<td>1.6914e+1</td>
<td>7.9597e+0</td>
<td>8.0136e+0</td>
<td>3.5412e+0</td>
</tr>
<tr>
<td>F1</td>
<td>0.0000e+0</td>
<td>9.1976e-5</td>
<td>8.6142e-7</td>
<td>7.8337e-6</td>
<td>1.6391e-5</td>
</tr>
<tr>
<td>F2</td>
<td>8.0435e-4</td>
<td>1.1049e-2</td>
<td>2.5310e-3</td>
<td>3.1852e-3</td>
<td>2.1794e-3</td>
</tr>
<tr>
<td>F3</td>
<td>1.4022e-2</td>
<td>1.8427e+0</td>
<td>1.0844e+0</td>
<td>2.9239e+0</td>
<td>4.2194e-1</td>
</tr>
<tr>
<td>F4</td>
<td>4.1418e-6</td>
<td>6.8820e-5</td>
<td>2.0067e-5</td>
<td>2.4071e-5</td>
<td>1.4857e-5</td>
</tr>
<tr>
<td>F5</td>
<td>4.9736e-8</td>
<td>3.4464e-4</td>
<td>3.9258e-6</td>
<td>3.7338e-5</td>
<td>7.7237e-5</td>
</tr>
<tr>
<td>F6</td>
<td>6.9838e-2</td>
<td>7.5985e+0</td>
<td>1.4556e+1</td>
<td>2.9513e+1</td>
<td>2.5570e+1</td>
</tr>
<tr>
<td>Dimensions F7</td>
<td>3.8864e+0</td>
<td>1.0074e+2</td>
<td>1.7884e+1</td>
<td>2.3820e+1</td>
<td>1.7852e+1</td>
</tr>
<tr>
<td>F8</td>
<td>2.0779e+1</td>
<td>2.0984e+1</td>
<td>2.0916e+1</td>
<td>2.0906e+1</td>
<td>4.5752e-2</td>
</tr>
<tr>
<td>F9</td>
<td>8.9192e-2</td>
<td>2.3747e+1</td>
<td>1.7727e+1</td>
<td>1.7803e+1</td>
<td>2.9258e+0</td>
</tr>
<tr>
<td>F10</td>
<td>1.5534e-1</td>
<td>9.3358e-1</td>
<td>3.8196e-1</td>
<td>4.1589e-1</td>
<td>1.9262e-1</td>
</tr>
<tr>
<td>F11</td>
<td>1.0945e-1</td>
<td>4.9565e+1</td>
<td>2.3879e+1</td>
<td>2.4978e+1</td>
<td>8.0383e+0</td>
</tr>
<tr>
<td>F12</td>
<td>5.9698e+0</td>
<td>1.6715e+2</td>
<td>6.6662e+1</td>
<td>7.2236e+1</td>
<td>4.2764e+1</td>
</tr>
<tr>
<td>F1</td>
<td>8.2309e-8</td>
<td>1.2086e-3</td>
<td>2.5963e-5</td>
<td>8.5225e-5</td>
<td>1.8714e-4</td>
</tr>
<tr>
<td>F2</td>
<td>7.5256e-4</td>
<td>3.6622e-3</td>
<td>1.6055e-3</td>
<td>1.6308e-3</td>
<td>5.8123e-4</td>
</tr>
<tr>
<td>F3</td>
<td>7.1014e-2</td>
<td>1.8405e+0</td>
<td>5.3814e-1</td>
<td>6.1811e-1</td>
<td>3.9879e-1</td>
</tr>
<tr>
<td>F4</td>
<td>3.3906e-5</td>
<td>1.3605e-4</td>
<td>7.0071e-5</td>
<td>6.9617e-5</td>
<td>2.3169e-5</td>
</tr>
<tr>
<td>F5</td>
<td>7.1704e-7</td>
<td>4.2792e-4</td>
<td>6.9361e-5</td>
<td>1.2422e-4</td>
<td>1.2622e-4</td>
</tr>
<tr>
<td>F6</td>
<td>1.6568e+1</td>
<td>8.8500e+1</td>
<td>4.3447e+1</td>
<td>5.2211e+1</td>
<td>2.3227e+1</td>
</tr>
<tr>
<td>Dimensions F7</td>
<td>1.1631e+1</td>
<td>6.6814e+1</td>
<td>3.8425e+1</td>
<td>3.8524e+1</td>
<td>1.2176e+1</td>
</tr>
<tr>
<td>F8</td>
<td>2.0984e+1</td>
<td>2.1163e+1</td>
<td>2.1108e+1</td>
<td>2.1103e+1</td>
<td>3.7513e-2</td>
</tr>
</tbody>
</table>

Continued on next page
Similar optimization results for the iCMAES-ILS algorithm are in Tianjun and Stützle (2013), and those for the DRMA-LSCh-CMA algorithm can be found in Lacroix et al. (2013). For comparison purposes, the evaluation results for the three hybrid algorithms are categorised into two:

- unimodal benchmarks (F1 to F5), and
- multimodal benchmarks (F6 to F12).

These are then graphically presented in Figures 6.4 and 6.5 respectively for the 10, 30 and 50 dimensions of each of these benchmarks. Notice that to facilitate visualisation and comparison all plots in these figures are semilog plots with the vertical axes (optimization error ($f_{\text{error}}$)) in log scale. Again, the optimization goal here is to minimise $f_{\text{error}}$. Also, as these results are averages of 51 repeated runs, the Kruskal-Wallis nonparametric test (Hollander and Wolfe, 1999) is applied in a multiple comparison procedure to verify statistical significance. It is found that at $\alpha = 0.01$ the differences between the performance characteristics of the three compared algorithms in Figures 6.4 and 6.5 are statistically significant.

Firstly, it is observed from the unimodal benchmarks (F1 to F5) in Figure 6.4 that the DRMA-LSCh-CMA and iCMAES-ILS algorithms generally outperform the EC/SQP algorithm. Both of these algorithms tend to make fast progress to the optimum solution during early stages of the evolution (see especially the 10 dimensions of F1 to F5 on the left column). However, the standard EC algorithm used in the EC/SQP hybrid model seems to make steady but mild progress until convergence. Nevertheless, significant progress is always observed as soon as the hybrid switching mechanism kicks off (switching from the EC to the SQP algorithm). While for the 10 dimensional problems (left column) the switching effect was only able to put the EC/SQP algorithm ahead on the F1 benchmark, it played more effective role on the 30 and 50 dimensions of the F3, F4 and F5...
Figure 6.4: Comparison of the optimization errors of the EC/SQP algorithm with that of two contestants of the CEC2013 competition (iCMAES-ILS and DRMA-LSCh-CMA algorithms) on Unimodal Benchmarks (F1 to F5). Results are averages of 51 runs; all plots are semilog with the vertical axes in log-scale.

benchmarks. The slow initial progress by the EC/SQP algorithm might have resulted from the large pools used on the 30 and 50 dimensional problems. But this was essential to minimise the risk of converging to sub-optimal point.
Figure 6.5: Comparison of the optimization errors of the EC/SQP algorithm with that of the iCMAES-ILS and DRMA-LSCh-CMA algorithms on the CEC2013 Multimodal Benchmarks (F6 to F12). Results are averages of 51 repeated runs.

Importantly, the above performance characteristics exhibited by the EC/SQP reveal that the switching from the population-based EC algorithm to the point-based SQP algorithm could crucially influence its overall optimization progress.
Overall, the two participants of the CEC2013 competition (DRMA-LSCh-CMA and the iCMAES-ILS) took the lead in this test, see especially their performances on the 30 and 50 dimensions of the F1 and F2 benchmarks.

Secondly, the evaluation results on the multimodal benchmarks (F6 to F12) in Figure 6.5 reveal some significant retardation in the convergence characteristics of all the three compared algorithms (see the flatter curves in Figure 6.5 as compared to those in Figure 6.4). As reported in the benchmark specifications for this competition (Liang et al., 2013), the difficulty levels of these benchmark problems cannot be overemphasised; besides increasing levels of multimodality, these benchmarks are shifted, rotated and asymmetric with vast number of local optimum solutions.

The plots in Figure 6.5 show that the EC/SQP algorithm closely trailed the other two algorithms in a number of cases (see the 10 dimensions of F9, F11 and F12; as well as 50 dimensions of F6 and F7 for example). However, it is only on four cases (10 dimensions of F6, F8; as well as 30 and 50 dimensions of F8) that the proposed EC/SQP clearly takes the lead in this test. In fact, as observed from Figure 6.5, the iCMAES-ILS algorithm performs best overall. This, however, does not come as a surprise since both the iCMAES-ILS and DRMA-LSCh-CMA algorithms are among the top three highest ranked contestants in the CEC2013 competition (Loshchilov et al., 2013). Importantly, both the iCMAES-ILS and DRMA-LSCh-CMA algorithms hybridize the CMA-ES algorithm, which is known to be quite effective (invariant against translation, rotation and scaling (Hansen, 2006)) on complex rotated landscapes; this could be the reason why their performances would be difficult to compete with on these benchmarks.

Ultimately, although the proposed EC/SQP algorithm does compete with these current (known) best performing hybrid algorithms, it unfortunately did not perform better on either of the two benchmark categories. Nevertheless, as each of these algorithms has a hybrid makeup, these findings could further justify the increasing interest in hybrid approaches in the EC domain; such algorithms tend to be more robust across different problem types and sizes.

6.6 Contribution

The attempt made in this chapter to realise the proposed hybrid optimization algorithm was possible following an intuitive amalgamation of the global (EC) and
the gradient based (SQP) algorithms, which are previously presented in Part I. The design goal was focused such that, tuning the combined parameters of the two algorithms while in collaboration enhanced the requisite balance in exploration and exploitation necessary for robust optimization. This would not have been possible without:

- the collaborative task-switching hybrid framework which dynamically shifts control from the EC to the SQP algorithm based on the robust convergence detection method, and
- the newly introduced validation routine (Section 6.3) which further substantiates the quality of the solution returned by the local algorithm.

### 6.7 Summary

This chapter has presented a broad review of various techniques of hybridizing evolutionary algorithms. In particular, following a taxonomy of the crucial considerations in building hybrid frameworks, a multidisciplinary survey of their applications in the recent years was presented. A novel hybridization approach that combines an evolutionary algorithm with a local search method in a collaborative manner was proposed. In a series of empirical experiments, the performance of the proposed hybrid algorithm was compared to that of the standard evolutionary algorithm and another with origin in evolutionary strategies. Additional experiments which compare the proposed EC/SQP algorithm with the two known best-performing hybrid EAs (on the CEC2013 benchmarks) are also presented. The results showed that the proposed hybrid EC/SQP algorithm achieved good optimization performance on these benchmarks – even though the other two hybrid EAs took the overall lead in these contests.

Importantly, the experimental results presented in this chapter have further identified the potentials of evolutionary-based hybrid frameworks in the field of stochastic global optimization. In other words, hybrid EAs could improve convergence efficiency without compromising robustness across various categories of global optimization landscapes. In the next chapter, the proposed hybrid framework will be enhanced to facilitate continuous optimization – making it possible to investigate its applicability on non-stationary optimization problems.
Chapter 7

Enabling Continuous Optimization for Global and Dynamic Environments

Having analysed the potentials of the task-switching hybrid framework proposed in the previous chapter (Sections 6.4 - 6.5), this chapter has three main objectives. First, this chapter enhances the initial hybrid framework (Algorithm 6.1) by replacing the canonical EC model (Algorithm 2.2) with the newly proposed dual-pool EC algorithm (Algorithm 5.2). The goal is to form an improved hybrid system which combines the dual-pool EC model with the improved Sequential Quadratic Programming (SQP) local search algorithm (Algorithm 3.1). Secondly, this chapter applies the newly proposed hybrid framework to global optimization problems in non-stationary environments – a set of dynamic optimization benchmarks is utilised. Finally, Sections 7.6 and 7.7 present and evaluate the impact of closed-loop parameter adaptation methods on the hybrid model.

7.1 The Dual Pool Hybrid EC/SQP Algorithm

A new hybrid framework built around the dual-pool EC model and the SQP algorithm (hybrid EC/SQP Algorithm (HESA)) is delineated in Algorithm 7.1. The new framework extends the previous hybrid model (Algorithm 6.1) by enabling continuous temporal collaboration between the global and local algorithms. The closed-loop structure of the flow diagram in Figure 7.1 illustrates how the new system is suitable for continuous optimization.
Algorithm 7.1 The Dual-Pool Hybrid Evolutionary Computation Algorithm – HESA

1: \( t \leftarrow 0; \)
2: \( N \leftarrow \text{Pool size (user defined)}; \)
3: \( P_{\text{Evo}}(t) \leftarrow \{X_i \mid X_{i,j} \in [x_j, \overline{x}_j], i = 1, \ldots, N, j = 1, \ldots, n \} \)  
   \( \text{// initialise Evolution Pool } P_{\text{Evo}} \)  
4: \( \text{while not termination do} \)
5: \( P_{\text{Evo}}(t), \sigma_{\text{Xover}}(t), \tilde{C}_{\text{Div}}(t) \leftarrow \text{invoke } \text{EC}(P_{\text{Evo}}(t)); \)
6: \( \text{if } \left( \tilde{C}_{\text{Div}}(t) < C_{\text{min}}^{\text{Div}} \right) \text{ and } (\sigma_{\text{Xover}}(t) < \sigma_{\text{Xover}}^{\text{min}}) \)  
5: \( P_{\text{Evo}}(t) \leftarrow \text{rank } f(P_{\text{Evo}}(t)); \)
7: \( P_{\text{Evo}}(t) \leftarrow k\% \left( P_{\text{Evo}}(t) \right); \)
8: \( x_{\text{SQP}} \leftarrow \text{invoke } \text{SQP}(X_{\text{Evo}}); \) where \( X_{\text{Evo}} \in P_{\text{Evo}} \)
9: \( X_{\text{Evo}} \leftarrow x_{\text{SQP}}; \)
10: \( P_{\text{Div}} \leftarrow \{X_i \mid X \in [\underline{x}_j, \overline{x}_j], i = 1, \ldots, N, j = 1, \ldots, n; \)  
11: \( P_{\text{Div}} \leftarrow \text{rank } \|X_{\text{Div}} - X_{\text{Evo}}^i\| : i = 1, \ldots, N; \)
12: \( P_{\text{Div}} \leftarrow (100 - k)\% \left( P_{\text{Div}} \right); \)
13: \( P_{\text{Evo}}(t) \leftarrow \{P_{\text{Evo}}(t) \cup P_{\text{Div}}\}; \)
14: \( \text{end if} \)
15: \( P_{\text{Evo}}(t + 1) \leftarrow P_{\text{Evo}}(t) \)
16: \( t \leftarrow t + 1; \)
17: \( P_{\text{Evo}}(t) \leftarrow \text{rank } \left| X_{\text{Div}}^i - X_{\text{Evo}}^i \right| : i = 1, \ldots, N; \)
18: \( \text{end while} \)  
19: \( \text{// initialise Diversity Pool } P_{\text{Div}} \)
20: \( \text{// keep evolving } P_{\text{Evo}} \text{ until maximum iteration limit is reached } *// \)
21: \( \text{// run EC model and estimate convergence at every iteration, see Algorithm 2.2 } *// \)
22: \( \text{// check for convergence of } P_{\text{Evo}}(t), \text{ see §4.5 } *// \)
23: \( \text{// evaluate and rank } P_{\text{Evo}}(t) \text{ by fitness } *// \)
24: \( \text{// get the top k% as elites } *// \)
25: \( \text{// run SQP algorithm with the best solution } \)
26: \( X_{\text{Evo}} \in P_{\text{Evo}} \text{ to get the local optimum } x_{\text{SQP}}, \)  
27: \( \text{Algorithm 3.1} *// \)
28: \( \text{// initialise Diversity Pool } P_{\text{Div}} *// \)
29: \( \text{// evaluate and rank } P_{\text{Div}} \text{ by distance from elite } *// \)
30: \( \text{// get the farthest samples in } P_{\text{Div}} *// \)
31: \( \text{// merge } P_{\text{Evo}} \text{ and } P_{\text{Div}} \text{ to form the new evolution pool } P_{\text{Evo}} *// \)
Figure 7.1: A framework for continuous optimization using the Hybrid Dual-pool EC/SQP Algorithm (HESA). While this framework stimulates continuous optimization cycle, the stopping criterion requires a user specified limit on function evaluations or iterations.

Importantly, the dual-pool EC model (Algorithm 5.2) remains generally preserved, but is augmented by the important addition of local refinement by the SQP algorithm (see Algorithm 7.1, line 9). Local refinement is dynamically applied to the best solution point returned by the dual-pool EC model whenever
convergence is detected (line 6). The criterion upon which the switching between the two – different\(^1\) but complementary – algorithmic paradigms is built is based on the robust convergence detection method earlier proposed in Section 4.5. The local optimization stage aims to improve the exploitation capabilities of the dual-pool EC model. Thus, the proposed HESA model ensures that the hybridized system benefits from synergy.

The flow diagram in Figure 7.1 visually outlines the dynamics of the proposed dual-pool HESA model. Figure 7.1 reveals that the search always begins with the global algorithm (dual-pool EC model); the convergence validation is aligned to two minimum thresholds: (i) the coefficient of spatial diversity ($\tilde{C}_{Div}$), and (ii) the population evolvability measure ($\sigma_{Xover}$), see Algorithm 7.1 (line 6).

The local algorithm (SQP) then processes a single solution point returned by the global algorithm. The single point was derived from the search pool of the global algorithm of size $N$, and it is the optimal solution ($x^*$) returned by the global algorithm.

The optimization process cycles through global and local searches until the available function evaluations are exhausted or the process is terminated by the user; this continuous cycle (Figure 7.1) enables continuous optimization in the context of global and dynamic environments.

To investigate the new hybrid framework, the rest of this chapter focuses on evaluation of the performance of the proposed HESA model on dynamic optimization problems. Section 7.2 introduces the dynamic optimization problems and highlights their significance in global optimization framework. Section 7.3 reviews four main approaches upon which the state-of-the-art models for dynamic optimization are designed. Then, Section 7.4 presents the dynamic rotation peaks (DRP) benchmark, which is then used – as a case study – to evaluate the performance of the proposed HESA model in Section 7.5. Finally, Section 7.9 summarises the contributions and concludes the chapter with remarks for further investigations. The following section introduces the aspects of optimization in non-stationary environments.

---

\(^1\)The hybridized algorithms differ in two important perspectives: the EC is global (population-based) and stochastic, whereas the SQP is local (individual-based) and deterministic.
7.2 The Rationale for Dynamic Optimization

Beyond the traditional goal of seeking to converge to a given global optimum solution in the stationary optimization setup, dynamic optimization requires continuous tracking of the optimal solution. This is necessary because in such environment, the locus and/or fitness of the global optimal point at a given time step \( t \) may slightly or significantly vary from that in the previous time step \((t - n)\); in other words, the geographic position of the solution points in the state space changes over time \((t)\). Therefore, in such environment, an optimization problem could be regarded as having an additional design parameter, i.e., the instantaneous time \( t \), such that the objective function is formulated as:

\[
\text{maximise: } y = f(x, t), \tag{7.1}
\]

where \( x \in \mathbb{R}^n \) is an \( n \)-dimensional vector of the problem’s design variables.

In practice, many real-world applications tend to exhibit some dynamic behaviour (subject to large amount of uncertainties); it is common for optimization model parameters to vary continuously during the optimization process. Thus, the demand for dynamic optimization algorithms in the field of sciences, engineering, financial management and prediction, mathematical modelling, industrial design and applications etc. has steadily increased in the recent years.

Typically, many control problems are subject to inherent dynamic behaviours. This could be as a result of system degeneration, gradual or abrupt changes in user specified control variables\(^2\). Such dynamic phenomena often lead to critical changes in the original model. Dynamism is also encountered in industrial productions and constructions where, changing customer demand, varying environmental conditions or a slight degradation in raw materials’ quality can significantly change the optimization goal. Similar cases occur in scheduling problems where the optimization goal changes following an unexpected machine breakdown or due to spontaneous addition/finishing of jobs.

Early researches in EC were focused on utilising these algorithms to solve complex (including, medium to large scale) multimodal but stationary global

\(^2\)The state of control systems may, in addition, experience dynamic changes due to temperature or pressure changes overtime, or as a result of changes in physical parameters like component wear.
optimization problems. It was after the realisation of their adaptive capabilities (De Jong, 1975), that EC algorithms have become widely accepted (Beyer and Sendhoff, 2007; Branke, 2001; de França and Von Zuben, 2009) as reliable candidates for searching and tracking of global optima in dynamically changing environments. Evolutionary algorithms (EAs), such as genetic algorithms (GAs), inherently have a high likelihood of withstanding changes in the objective function so long as their search pool remains diverse (i.e., fairly evenly distributed over the search space). Thus, with a diverse search pool, EAs would naturally adapt and refocus the search towards the neighbourhood currently favoured by the objective function.

However, as highlighted in the diversity dynamics of the standard EC model (see Section 5.5.1), EAs inevitably lose the diversity in their search pool over generations. This diversity collapse forces the search pool to converge to a currently explored promising region thereby paralysing the adaptive capabilities of these algorithms. Hence, the standard EAs generally fail to explore new areas of the search space when an eventual change in the problem’s optimal solution occurs. Therefore, following the brief overview in Section 7.3, Section 7.4 investigates the performance characteristics of the proposed HESA model (Algorithm 7.1) on dynamically changing environment. The HESA model’s collaborative framework will – at every time-step of the dynamically changing search space – control and maintain sufficient diversity in the search pool. The system uses a robust convergence detection mechanism to adaptively switch between its constituent algorithms.

### 7.3 Overview of key Dynamic Optimization Approaches

Although a number of early works on adaptive and dynamic optimization approaches date back to 1960s (Fogel et al., 1966; Goldberg and Smith, 1987), it is only in the last couple of decades that much progress have been made. In fact, the recent years have seen an increasing interest in optimization approaches suitable for dynamically changing problems (Nguyen et al., 2012). Hence, the performance of evolutionary algorithms on dynamic problems has been improved; the improvement strategies include:
i. convergence avoidance strategies;

ii. memory based approaches;

iii. multi-populations strategies; and

iv. robust optimization policies.

The following sections present a chronological review of some landmark investigations in dynamic optimization literature using EAs.

### 7.3.1 Convergence Avoidance strategies

These strategies typically include dynamically changing the mutation and/or crossover rates. The methods aim to preserve diversity in an evolutionary search pool by completely avoiding population convergence during the entire optimization period.

An early study on dynamic problems by Cobb (1990) has led to deployment of a mechanism based on triggered hypermutation to detect and adaptively increase the mutation rate whenever a change is detected. Enjoying the benefit of hypermutation, this technique has a strong tendency to revitalise the diversity necessary to aid further exploration of the new landscape. However, this original model of triggered hypermutation was designed to detect dynamism only when there is a change in the fitness of the best solution found so far. Thus, as demonstrated by Grefenstette (1992), there are a great many scenarios at which the technique fails to detect dynamic changes; and with no triggered hypermutation, the evolution eventually converges to a suboptimal region.

Consequently, Grefenstette (1992) proposed a remedy by augmenting a genetic algorithm with a partial hypermutation step. The method replaces a percentage (called replacement rate) of the population with randomly generated individuals in every generation. While the addition of the replacement rate parameter adds to the challenges in parameter tuning, a sensitivity test was shown to support replacing 30% of the total pool size. However, the investigations in Grefenstette (1992) remained silent on what individuals should be replaced for optimum performance. This method was later named as random immigrants (Cobb and Grefenstette, 1993). The random immigrants was shown to outperform (based on tracking ability) the standard GA, a triggered hypermutation GA (Cobb and Grefenstette, 1993), and memory based GA (Trojanowski and Michalewicz, 1999).
Further theoretical analysis on the use of mutation only EA to adapt and search a time-dependent environment can be found in Ronnewinkel et al. (2001).

It is important to note that the successive introduction of the random immigrants may significantly slow down the optimization efficiency. This is because it essentially adds uninformed diversity into the search pool at every generation. Another challenge lies in establishing optimum size and frequency of introducing the random immigrants. This may further complicate parameter tuning since it is, in general, problem dependent.

### 7.3.2 Memory-Based strategies

Memory-based strategies preserve and revert to a previously seen state when the problem state space cycles in a periodic manner. Thus, they are particularly useful for dynamic environments that exhibit periodicity. The memory structure can be either implicit or explicit. Implicit memory strategies utilise redundant representation (Goldberg and Smith, 1987) to tag the visited high quality states, but explicit memory methods use buffer (Branke, 1999) to store their previous states.

Goldberg and Smith (1987) investigated the capabilities of genetic algorithms to harness redundant information stored via the use of genetic diploidy with dominance. Goldberg and Smith (1987) believe that at least for some classes of dynamic problems having two state response surfaces\(^3\), genetic algorithms with the diploidy strategy can effectively handle certain forms of dynamic changes. Although in this proposal information is only implicitly stored and retrieved, it turns out to be one of the early breakthroughs in the design of memory-based EAs. It is evident from their investigations (Goldberg and Smith, 1987) that diploidy and/or multiploidy representations can be quite effective on dynamic environments exhibiting periodicity among few states, but the suitability of these strategies to problems without periodicity or those with more than a few re-occurring states is debatable.

Elsewhere, Branke (1999) proposed using a buffer to serve as an explicit memory for direct storage and retrieval of previously seen good solutions. Evaluation results on an “oscillating peaks” benchmark have shown improved performance over the standard EC model, albeit this is limited to a small set of problems

\(^3\)A two state response surface model is an optimization problem that switches between two optimal states.
where the optimum repeatedly returns to previously visited locations. Note that the proposal in Branke (1999) has not explicitly suggested an optimum size for the memory and how long the stored information should be retained.

A recent investigation by Barlow and Smith (2008) yields some new insight into the use of memory to enhance EA performance on dynamic scheduling problems. They found that for such problems having both a dynamic fitness landscape and time-dependent constraints that shift the feasible region of the search space, a memory should be designed to store some indirect representation of jobs in terms of their properties to allow mapping to similar solutions in future scheduling states. Based on their evaluation results, Barlow and Smith (2008) concluded that at least for dynamic job shop scheduling problems, this classifier-based memory-enhanced EA is more effective than approaches which aim squarely at maintaining population diversity.

Notably, the ability of memory-based approaches to recall previously seen states means that they could be effective on specific class of dynamic problems. However, in addition to the challenges in seeking optimum memory size, updates and replacement strategies, their application could be highly limited since most dynamic problems are non-periodic – exhibiting random and chaotic behaviours. Therefore, the diversity-based methods could have wider applicability provided that they improve robustness without compromising efficiency.

### 7.3.3 Multi-population strategies

In this case, the main population is divided into sub-pools which are set to search and closely track the high quality sub-regions as they dynamically drift and/or alter their fitness levels over generations. As these are introduced in Section 5.3.2, this section would only emphasise on the multi-population strategy proposed by Branke et al. (2000). Branke et al. (2000) proposes a multi-pool strategy that exclusively searches the sub-regions of its search space by using one of several sub-populations called self-organising scouts (SOS). The method could be seen as an opposite of the shifting balance GA (Oppacher and Wineberg, 1999). This is because while the main (parent) population continuously searches for new promising regions (peaks), the child populations concentrate on exploiting the previously detected promising areas. Empirical experiments have shown that SOS outperforms the standard EC under both changes in the frequency of change (dynamism) and increasing severity of change. It also performs better than the conventional
memory-based dynamic optimization methods which perform poorly when the change severity increases.

Note that multi-population based approaches enhance the exploration capabilities of EAs. This is because they stimulate a speedy exploration of different regions of the search space through niching. They also improve diversity maintenance since even when sub-populations individually converge; they still preserve diversity across them. However, as earlier discussed in Section 5.3.2, managing more than one population requires introduction of additional parameters, and majority of such methods are prone to communication challenges among the sub-populations; but these are the main limitations addressed by the newly proposed dual-pool EC model (Algorithm 5.2).

7.3.4 Robust optimization policies

These approaches favour searching for not only the solutions with high fitness levels, but also those that can survive minor environmental changes with little degradation, see Branke (1998); Paenke et al. (2006). The aim is to minimise frequent changes to an optimized system, which could be costly. Thus, in anticipation of slight variations in the search domain or the optimization goal, robust optimization policies enforce prioritising robust solutions over their highly fit but fragile counterparts. Note, however, that this approaches lack the capability to evolve and adapt to a newly formed landscape that follows a dynamic change. In other words, robust optimization policies do not adapt, they instead search for secure solutions that may survive minor changes, see Branke (1998); Tsutsui and Ghosh (1997) for more on such approaches.

7.4 Evaluation of the Dual-Pool HESA model on Dynamic Optimization Benchmarks

Firstly, this section presents a detailed description of the dynamic rotation peak (DRP) benchmark. Then, the test problem is utilised as a case study to evaluate the performance of the proposed dual-pool HESA model (Section 7.1) on a wide range of dynamic changes.
CHAPTER 7. ENABLING CONTINUOUS OPTIMIZATION

7.4.1 The Dynamic Rotation Peak (DRP) Benchmark

The case study to be used for evaluating the proposed dual-pool HESA model on dynamic optimization problems is the IEEE CEC\textsuperscript{4} 2009 dynamic rotation peaks (DRP) benchmark. Note that the benchmark information provided in this section is an excerpt from the complete benchmark specification in Li et al. (2008). As illustrated in the 3-D plot of a 2-dimensional DRP benchmark in Figure 7.2, the DRP benchmark is a global, asymmetric and highly multimodal problem; it features various degrees of dynamism ranging from small, large, random, chaotic, recurrent and recurrent noisy step changes. The test problem was originally proposed by Branke (1999) but was later improved by Li et al. (2008) and then recognised as a state-of-the-art platform for evaluating dynamic optimization algorithms. It was then used in the IEEE CEC 2009 competition on dynamic optimization; see Brest et al. (2009); de França and Von Zuben (2009); Korošec and Šilc (2009); Li and Yang (2009); Yu and Suganthan (2009).

\textsuperscript{4}IEEE Congress on Evolutionary Computation, 2009.
Table 7.1: Properties of the Dynamic Rotation Peaks Benchmark (cf. Figure 7.2)

<table>
<thead>
<tr>
<th>Property</th>
<th>Symbol</th>
<th>Values/Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensionality</td>
<td>$n$</td>
<td>variable</td>
</tr>
<tr>
<td>Complexity</td>
<td>$-$</td>
<td>global, scalable, asymmetric, rotated</td>
</tr>
<tr>
<td>Search Domain</td>
<td>$\mathcal{D}$</td>
<td>$D : x_i \in [-5.0, 5.0], i \in \mathbb{R}^n$</td>
</tr>
<tr>
<td>Number of Local Optima (Peaks)</td>
<td>$m$</td>
<td>10 (adjustable)</td>
</tr>
<tr>
<td>Local Optima Height range</td>
<td>$H$</td>
<td>$H_j \in [10, 100] : j \in \mathbb{R}^m$</td>
</tr>
<tr>
<td>Local Optima Width range</td>
<td>$W$</td>
<td>$W_j \in [1, 10] : j \in \mathbb{R}^m$</td>
</tr>
<tr>
<td>Global Optimum</td>
<td>$x^*(t)$</td>
<td>$\max(f(x, \phi, t)) : H_i(t) = \max_{i=1}^m H_i$</td>
</tr>
</tbody>
</table>

Problem Definition:

Let $\phi = (\vec{H}, \vec{W}, \vec{X})$, the DRP benchmark seeks to maximise the function $f(x, \phi, t)$, which is defined as

$$f(x, \phi, t) = \max_{i=1}^{m} \left( \frac{\vec{H}_i(t)}{1 + \vec{W}_i(t) \sqrt{\sum_{j=1}^{n} \left( \frac{x_j - \vec{X}_i(t)}{n} \right)^2}} \right), \quad (7.2)$$

where $\vec{H}$, $\vec{W}$, and $\vec{X}$ are vectors of peak height, width and position respectively; the size of each of these vectors is equal to the number of peaks $m$. These parameters change dynamically according to the dynamic rotation algorithm proposed by Salomon (1996). Details of the remaining parameters of this benchmark function are summarised in Table 7.1.

7.4.2 Evaluation of Dual-Pool HESA on DRP Benchmark

Table 7.2 summarises the parameter settings for the proposed HESA model and unless otherwise stated, these settings will be adopted throughout the dynamic optimization experiments. Tables 7.3 to 7.5 show the details of the parameters investigated during these experiments. The proposed model is evaluated on the rotation peaks benchmarks under six ($T_1$ to $T_6$) different types of dynamic changes (see Table 7.3).

The relative performance measures described in Table 7.5 signify the following:

- **Offline Performance ($r_{offline}$):** This is the ratio of the best solution found
Table 7.2: Dual-pool HESA model Parameter settings for Dynamic Optimization Experiment

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Values/Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Sizes</td>
<td>$N$</td>
<td>[10, 50, 100, 500, 1000]</td>
</tr>
<tr>
<td>Encoding</td>
<td>–</td>
<td>Real-valued</td>
</tr>
<tr>
<td>Selection Scheme</td>
<td>–</td>
<td>Binary Tournament</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>$P_c$</td>
<td>1.0</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>$P_m$</td>
<td>0.01</td>
</tr>
<tr>
<td>Replacement Scheme</td>
<td>–</td>
<td>Generational–Elitist</td>
</tr>
<tr>
<td>Initial Population</td>
<td>–</td>
<td>SSP quasi-random heuristic, §5.4.5</td>
</tr>
<tr>
<td>Termination Criteria</td>
<td>Max-FEs</td>
<td>Maximum Evaluations ($10^4 \times n$)</td>
</tr>
</tbody>
</table>

$x_{best}$ to the true optimal solution $x^*$ at the end of a prescribed time period. It indicates the tracking performance of the evaluated model.

- **Online Performance** ($r_{online}$): This reports the relative value of the best solution found $x_{best}$ to the true optimal solution $x^*$ at every epoch or sampling period. It assesses the tracking performance of the model through continual monitoring of its exploitation of the search space during the course of the optimization. See De Jong (1975); Grefenstette (1986, 1992) for further details on these measures.

Accuracy is measured from the relative performance analysis, i.e. the characteristics of the $r_{offline}/r_{online}$ curves. Robustness is assessed from the absolute error ($E^{last}$) analysis and also the relative performance analysis. And finally, efficiency assessment is based on the characteristics of the convergence curves derived from the median performances on the six different types of dynamic changes.

Having introduced detailed test problem parameters in Table 7.1 and detail description for the experimental evaluation parameters in Tables 7.3 to 7.5, the following sections present and analyse the results obtained during the optimization of the DRP benchmark.

### 7.5 Results – Analysis and Interpretation

The objectives of these experiments are: (i) to conduct parameter sensitivity test on the proposed HESA model, and (ii) to evaluate its performance with emphasis...
Table 7.3: Evaluation Parameters for the DRP Benchmark

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Values/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>$n$</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_1$: Small step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_2$: Large step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_3$: Random step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_4$: Chaotic step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_5$: Recurrent step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_6$: Recurrent Noisy step</td>
</tr>
<tr>
<td>Dynamic change instances</td>
<td>$T$</td>
<td>Small step</td>
</tr>
<tr>
<td>or types</td>
<td></td>
<td>Large step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Random step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chaotic step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recurrent step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recurrent Noisy step</td>
</tr>
<tr>
<td>Number of Changes/type</td>
<td>num-change</td>
<td>60</td>
</tr>
<tr>
<td>Function Evaluations/change</td>
<td>FEs/change</td>
<td>$10,000 \times n$</td>
</tr>
<tr>
<td>Max. Function Evaluations</td>
<td>Max-FEs</td>
<td>$10,000 \times n \times$ num-change</td>
</tr>
<tr>
<td>Current Best Solution</td>
<td>$x_{best}(t)$</td>
<td>The position of the highest peak at generation $t$; i.e., $x_{best}(t) = x_i(t)$ : $H_i(t) = \max^m(H_i)$</td>
</tr>
<tr>
<td>Sampling Frequency</td>
<td>$s_f$</td>
<td>The rate at which $x_{best}(t)$ is recorded: $s_f = 100 \times$ FEs</td>
</tr>
<tr>
<td>Sampling Period</td>
<td>$s$</td>
<td>The period after which $x_{best}(t)$ is recorded: $s = \text{FEs}/\text{change}/s_f$</td>
</tr>
<tr>
<td>Termination Criteria</td>
<td>Max-FEs</td>
<td>Maximum Function Evaluations</td>
</tr>
</tbody>
</table>

on accuracy, robustness and convergence efficiency. Table D.1 in Appendix D summarises the complete results obtained during these experiments. These results are averages of 20 independent runs for the HESA algorithm on the $T_1$ to $T_6$ dynamic change types (Table 7.3).

7.5.1 Pool size Sensitivity Analysis

This section investigates the influence of varying the parameter $N$ (pool size) on the offline and online performances ($r_{\text{offline}}/r_{\text{online}}$) of the proposed dual-pool HESA model. To facilitate results analysis, the plots in Figure 7.3(a and b) graphically demonstrate the offline/online performances of the proposed HESA model showing its sensitivity to pool size variations of 10 to 1000 samples. As expected, the plots in Figure 7.3(a and b) show that the average offline performances (Figure 7.3a) across all the six dynamic change types always exceed the corresponding online performances (Figure 7.3b). This is because the HESA
Table 7.4: Absolute Error ($E_{\text{last}}$) Analysis – analysed over 20 independent runs

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Runs</td>
<td>$\text{runs}$</td>
<td>Independent repetitions: $\text{runs} = 20$</td>
</tr>
<tr>
<td>Absolute Error Value</td>
<td>$E_{\text{last}}$</td>
<td>This records the absolute error between the current best solution $x_{\text{best}}$ and the true optimal solution $x^*$ at the end of every change instance $t = \text{FEs/change}$ (see Table 7.3). $</td>
</tr>
<tr>
<td>Average of best $E_{\text{last}}$</td>
<td>$\text{Avg-min-}E_{\text{last}}$</td>
<td>This parameter records the average of the minimum values of $E_{\text{last}}$ for all runs. $\sum_{i=1}^{\text{runs}} \min_{j=1}^{\text{num-change}} E_{\text{last}}^{(i,j)} / \text{runs}$</td>
</tr>
<tr>
<td>Average of mean $E_{\text{last}}$</td>
<td>$\text{Avg-mean-}E_{\text{last}}$</td>
<td>This parameter records the average of the means of $E_{\text{last}}$ for all runs. $\sum_{i=1}^{\text{runs}} \sum_{j=1}^{\text{num-change}} E_{\text{last}}^{(i,j)} / (\text{runs} \times \text{num-change})$</td>
</tr>
<tr>
<td>Average of worst $E_{\text{last}}$</td>
<td>$\text{Avg-max-}E_{\text{last}}$</td>
<td>This parameter records the average of the maximum values of $E_{\text{last}}$ for all runs. $\sum_{i=1}^{\text{runs}} \max_{j=1}^{\text{num-change}} E_{\text{last}}^{(i,j)} / \text{runs}$</td>
</tr>
<tr>
<td>STD of $E_{\text{last}}$</td>
<td>$\text{STD-}E_{\text{last}}$</td>
<td>This measures the spread (std. deviation) of $E_{\text{last}}$ for all runs. It is defined as: $\sqrt{\sum_{i=1}^{\text{runs}} \sum_{j=1}^{\text{num-change}} \left( E_{\text{last}}^{(i,j)} - \text{Avg-mean-}E_{\text{last}} \right)^2 / (\text{runs} \times \text{num-change} - 1)}$</td>
</tr>
</tbody>
</table>

The model is an elitist algorithm that monotonically improves on its previously found best solution. Thus, at the end of any given run, the final solution\(^5\) is always the best found so far.

On one hand, it is important to note from Figure 7.3(a and b) that highest performances for both the offline and online measures are obtained when a pool size of 50 is utilised; this is followed by a pool size of 100. On the other hand, these plots show that the larger sample sizes (greater than 100 to 1000) improve neither offline nor online performance. This is an interesting finding since the idea of utilising large pool sizes is generally thought to be a viable diversity control policy; although this outcome is rather unexpected, these 3-D plots validate that over all the $T_1$ to $T_6$ dynamic changes, both the offline and online performances deteriorate when a large pool size is utilised. Hence, it could be deduced from these sensitivity results that larger sample sizes tend to severely retard fitness

\(^5\)Note that the best solution found at the end every run is used to measure the algorithm’s offline performance.
Table 7.5: Relative Performance \((r(t))\) Analysis. Results are expressed as percentages of the true optimal solution \(x^*\)

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative performance (r(t))</td>
<td>(r^t(t))</td>
<td>The ratio of the fitness values of (x_{best}) to (x^<em>) at a given generation (t). It is defined as: (f(x_{best}(t))/f(x^</em>(t)))</td>
</tr>
<tr>
<td>(r(t)) per sampling period (s)</td>
<td>(r^s(t))</td>
<td>This records (r(t)) at the end of every sampling period for a given change (at (\text{FEs}=s)).</td>
</tr>
<tr>
<td>(r(t)) per change</td>
<td>(r^{last}(t))</td>
<td>This records (r(t)) at the end of every generation (t) for a given change.</td>
</tr>
<tr>
<td>Cumulative (r(t)) per change</td>
<td>(r^{cum})</td>
<td>This is the running average for (r(t)) at every sampling period (s) for a given change. (r^{last}/(1 + \sum_{s=1}^s(1 - r^s)/s))</td>
</tr>
<tr>
<td>Offline performance</td>
<td>(r^{offline})</td>
<td>This is the mean of (r^{last}) for all changes in a given run. (\sum_{j=1}^{\text{num-change}} r^{last}_j/\text{num-change})</td>
</tr>
<tr>
<td>Online performance</td>
<td>(r^{online})</td>
<td>This is the mean of (r^{cum}) for all changes in a given run. (\sum_{j=1}^{\text{num-change}} r^{cum}_j/\text{num-change})</td>
</tr>
</tbody>
</table>

progress during the evolutionary search. As validated here, the acceleration in fitness progress enjoyed by small to medium sized pools (50 to 100 samples) is vital for successful optimization in dynamically changing environments.

Importantly, these findings support a recent theoretical analysis that provided new insights into the effect of using large sample sizes in EAs by Chen et al. (2012). Their analysis (Chen et al., 2012) suggests that using large sample sizes, on at least some classes of global optimization problems, does not guarantee optimal performance.

### 7.5.2 Relative Performances and Convergence Efficiency

The above sensitivity analysis reveals that the HESA algorithm consistently yields better performance with \(N = 50\) samples. Thus, this section assesses the relative performances (Table 7.5) of the HESA algorithm on the DRP benchmark when a pool of 50 samples is utilised (see Figure 7.4). Notice from Figure 7.4 that the dynamic change type \(T_1\) (small step) appears to be the easiest, whereas \(T_6\) (recurrent noisy) seems to be most difficult for the HESA model. In fact, both
Figure 7.3: Sensitivity of the proposed HESA model under varying pool sizes and across the six different dynamic change types: (a) Shows the offline performance ($r_{\text{offline}}$), and (b) Shows the online performance ($r_{\text{online}}$). All results are averages of 20 independent runs.

The offline and online performances degrade as the dynamic change types get harder, i.e. from $T_1$ to $T_6$. Notice however, that with the exception of $T_6$, the HESA model seems to maintain at least 90/80% of offline/online performances across the entire range of dynamic change types. This generally shows a tracking consistency of at least 80% for $T_1$ to $T_5$ dynamic changes types.
Figure 7.4: The offline and online performances ($r_{\text{offline}} / r_{\text{online}}$) of the Dual-pool HESA model averaged over 20 independent runs. A pool of size 50 is utilised. The error bars show the standard errors of the mean.

Figure 7.5: Convergence graph showing the median offline performance characteristics of the HESA model on the six different dynamic change types over 20 independent runs.

To understand the convergence characteristics of the proposed HESA model on the DRP benchmark, Figure 7.5 compares the medians of the offline performances for the six different dynamic changes ($T_1$ to $T_6$). This figure demonstrates
CHAPTER 7. ENABLING CONTINUOUS OPTIMIZATION

Figure 7.6: Absolute Error Analysis for the Dual-pool HESA model on $T_1$ to $T_6$ dynamic change types of the DRP benchmark. (a): Compares the mean $E^{last}$ under varying pool sizes. (b): Shows the mean $E^{last}$ with standard error bars for the spread of the mean absolute error when a pool size $N = 50$ is used.

7.5.3 Absolute Error Analysis

This section investigates the absolute error ($E^{last}$) statistic (Table 7.4) of the proposed HESA model. A complete summary for the resulting absolute error in these experiments is presented in Table D.2 in Appendix D.

To gain better insight into the absolute error characteristics of the HESA model, the complete results is visualised in Figure 7.6. In particular, Figure 7.6a compares the mean absolute error ($E^{last}$) obtained for the six different dynamic change types of the DRP benchmark. In fact, Figure 7.6a also depicts the sensitivity of the HESA model to varying pool sizes; notice that a log scale is utilised.
on the pool size axis of this figure. Importantly, it is observed from Figure 7.6a that over all the six different dynamic change types, the least absolute error is witnessed when the HESA model is run with a pool size of 50 samples. In fact the absolute error appears to worsen both when a too small pool size (e.g. 10) and when a too large pool size (e.g. 1000) is utilised. Nevertheless, the characteristic shapes of the absolute error curves, over $T_1$ to $T_4$ dynamic change types, show that the sensitivity of the HESA model to varying pool sizes on such dynamic changes is rather mild (see the four curves at the bottom part of Figure 7.6a).

For clarity, Figure 7.6b illustrates the resulting absolute error for the sample size of $N = 50$. It provides a closer look into the spread of the mean $E_{\text{last}}$ across the six different dynamic change types. It could be concluded from this figure that from at least the $T_1$ to $T_5$ dynamic change types, the amount of absolute error sustained by the HESA model is no more than $\pm 20$.

### 7.6 Parameter Adaptation for the hybrid EA

The optimum setting for a set of evolutionary algorithm’s parameters does not only vary from problem to problem, but also across the stages of the evolution itself (Bäck, 1993; Eiben et al., 1999; Harik and Lobo, 1999; Spears, 1995). This phenomenon applies not only to the conventional EC frameworks, but also to the hybrids and memetic methodologies (Chen et al., 2011; Queiroz and Lyra, 2009). Thus, in the EC community, there is a general consensus on the need to involve a systematic adaptation of the crucial evolution parameters, such as mutation and recombination step sizes (Hansen et al., 1995; Herrera and Lozano, 2000) and/or probabilities (Dai et al., 2010; Hong et al., 2002; Niehaus and Banzhaf, 2001; Queiroz and Lyra, 2009; Yang, 2003), selection pressure (McGinley et al., 2011), etc. The objective has been to potentially improve exploration of promising regions of the search space, and most importantly, to minimise reliance on manual parameter tuning which affects the overall solution quality.

Figure 7.7 demonstrates the commonly used approaches to parameter adaptation in EC. Like in most stochastic search methods, parameter setting in EC fundamentally accounts for: what parameters are changed, how they are changed and the extent of the change. Thus, parameter setting can generally be divided into parameter tuning, and parameter adaptation (cf. Figure 7.7). Parameter tuning involves manual setting of the evolutionary parameters prior to running
the evolutionary search. The alternative approach involves controlling parameters dynamically during the optimization. This mode of tuning can be sub-divided into deterministic, adaptive and self-adaptive parameter control. In deterministic parameter control, the adaptation is generally aligned to some deterministic rules, such as a temporal schedule; thus, a parameter is adjusted without any feedback from the search output. In the adaptive parameter control however, some form of feedback from the search is used to dictate the extent to which a parameter is modified. Finally, self-adaptation\(^6\) entails incorporating the parameters into the encoding of the solution points with the hope that improved parameters are evolved over generations.

The objective of parameter control has been to minimise user influence. Thus, the hypothesis guiding the following proposals is:

\[ \#H: \text{Adaptation of EC parameters via heuristic incorporation of useful information from the current state of the evolutionary search process should provide robust (stable) performance with less user input.} \]

As a caveat to this, the \textit{useful information} in \#H excludes features that simply return the mere temporal state of the search; it specifically refers to crucial features that reflect the instantaneous state of the evolutionary search pool (such as pool’s diversity and population evolvability). The following sections propose adaptation models for both the mutation and recombination operators.

### 7.6.1 Adapting the Mutation operator

Recall that the proposed HESA model (Section 7.1) utilises real-valued encoding, and it adopts a mutation strategy that is based on the Breeder GA (BGA) mutation algorithm. The BGA (Mühlenbein and Schlierkamp-Voosen, 1993) mutation operator is an advanced version of the well-known Gaussian mutation method which mutates a sample solution by adding some Gaussian random noise. Thus, from a search pool of size \( N \), any sample solution \( x_k : k \in [1, N] \) has a chance to mutate any of its \( n \)-dimensional variables \( x^i_k : i \in [1, n] \). A sample mutates with a probability \( P_M \) by taking a custom step size, \( \sigma_k^i \), such that

\[ x^i_k = x^i_k + \sigma_k^i, \quad (7.3) \]

\(^6\)Self adaptation often requires some form of tailored/problem dependent encoding. Since this study aims to propose a generalised model for evolutionary optimization in continuous domain, self-adaptation is beyond the scope of this thesis.
where $\sigma_k^i$ is the mutation step size which is defined as

$$\sigma_k^i = \zeta_i \cdot r_i \cdot a_i. \quad (7.4)$$

The parameter $\zeta_i$ is the mutation directivity; it defines the sign such that $\zeta_i = \pm 1$ (uniform at random). $r_i$ is the mutation range, chosen as 10% of the width of the variable’s bound\(^7\), such that

$$r_i = 0.1(x^i_u - x^i_l).$$

Then, $a_i$ is the mutation step size parameter. To facilitate mutation with smooth (short) step sizes, Mühlenbein and Schlierkamp-Voosen (1993) suggest deriving the step size parameter, $a_i$, from a uniform distribution that favours values around

---

\(^7\)The mutation range $r_i = 10\%$ was suggested by Mühlenbein and Schlierkamp-Voosen (1993) to ensure that the resulting mutated variable remains within the feasible search space. The motivation was to minimise the likelihood of generating infeasible solutions (i.e., solutions outside the feasible domain) – thus avoiding the need for repair.
its lower-tail, such that

\[ a_i = 2^{-\mu_i^M \tau}. \] (7.5)

In (7.5), \( \tau \in [2, 16] \) is the step size precision parameter. For the EC models in this study, \( \tau = 16 \) is adopted to allow for higher precision, which means that smoother steps can be taken for better refinements of high quality solutions. \( \mu_i^M \) is the effective precision parameter, \( \mu_i^M \sim \mathcal{U}(0, 1) \).

Having chosen a constant value for the mutation precision parameter \( \tau \), the effective precision can mainly be adapted by varying \( \mu_i^M \) from 0 to 1 depending on the instantaneous state of the search pool and/or the temporal progress of the evolutionary search. Although, in the original model for the BGA operator, the effective precision parameter \( \mu_i^M \) is chosen purely at random, the following section presents the proposed adaptation strategy in this thesis.

A. The Proposed closed-loop Adaptive Mutation

This section proposes a closed-loop based strategy\(^8\) for the proposed hybrid EC model. As depicted in Figure 7.8, the strategy allows heuristic determination of an appropriate mutation step size by using the instantaneous state of the spatial diversity \( \tilde{C}_{Div} \) and evolvability \( \sigma_{Xover} \) of the search pool\(^9\); this output information (current diversity state of search pool) is then fed back to the input (mutation controller unit) at every generation. Subsequently, the control unit of the closed-loop system determines the appropriate mutation size; it achieves that by regulating the boundaries of the distribution within which the mutation step size is randomly sampled from.

Recall from the original BGA mutation (7.5) that \( a_i = 2^{-\mu_i^M \tau} \) with \( \mu_i^M \sim \mathcal{U}(0, 1) \). In this proposal, the controller splits the boundary \([0, 1]\) within which the effective mutation step size parameter \( \mu_i^M \) is generated in accordance to the following rule:

\[ \mu_i^M = \begin{cases} [0.5, 1] & \text{if } \tilde{C}_{Div} > 0.25 \text{ and } \sigma_{Xover} > 0.25, \\ [0, 0.5] & \text{otherwise}. \end{cases} \] (7.6)

The intuition here is that, since the values of the diversity and evolvability

---

\(^8\)The closed-loop model here is a sort of observer-controller model (cf. Figure 7.8); it monitors the output (pool’s diversity) and feed it back to the input (mutation controller) to guide setting an appropriate mutation step size.

\(^9\)The spatial diversity and evolvability measures are detailed in Sections 4.2 and 4.4.
Figure 7.8: A closed-loop adaptive mutation operator. \( Q_r(t) \) is the search pool after recombination and \( Q_m(t) \) is the search pool after mutation; the mutation step-size \( \mu_i^M \) is determined by the adaptive mutation controller based on the instantaneous diversity \( \hat{C}_{Div} \) and evolvability \( \sigma_{Xover} \) of the search pool at generation \( t \). This adaptive mutation block (shaded in grey) replaces the mutation block in the standard EC model of Figure 2.2 in Chapter 2.

measures \((\hat{C}_{Div}, \sigma_{Xover})\) are normalised and range from \([0, 1]\) (Section 4.5), we consider any value larger than 0.25 to mean that a search pool has a sufficient diversity (i.e., at least 25%)\(^{10}\); in this case the mutation controller samples the step size parameter from its upper-tail, i.e., \( \mu_i^M \in [0.5, 1] \). Otherwise, it is sampled from the lower-tail \( \mu_i^M \in [0, 0.5] \).

The above definition for \( \mu_i^M \) has the following implications:

i. It stimulates further exploration of the search space by allowing individuals to mutate with larger step sizes when the current spatial diversity and evolvability in the search pool are high, i.e., when both \( \hat{C}_{Div} \) and \( \sigma_{Xover} \) are higher than 25%; and

ii. It allows the mutation operator to yield smoother refinements of the already found high quality solutions when the search pool sufficiently draws to the highest quality region found so far.

Thus, beyond introducing occasional disruption into the evolutionary search process, with this adaptive model (7.6) the mutation operator ensures that:

(i) whenever the HESA model is in its exploratory-phase, the mutation operator aids further exploration; and

---

\(^{10}\)The 25% baseline for the pool’s diversity and evolvability is decided following empirical experimentations.
(ii) when it is in its exploitative-phase, the operator facilitates smoother refine-
ments by constricting the sub-region of the randomly generated mutation step sizes.

Moreover, it is worth noting that in spite of this controlled adaptation mecha-
nism, the overall mutation operation remains largely stochastic since $\mu_i^M$ (7.6) is
still sampled from a uniform random distribution. Crucially, from the perspective
of simulated evolution (Atmar, 1994; Fogel et al., 1966), retaining randomness in
the mutation operation is generally considered more plausible to the biological
principle of mutation itself.

7.6.2 Adapting the Recombination operator

In a typical evolutionary paradigm, the role of the recombination (crossover)
operator is to facilitate transfer via exchange of high quality traits from parent
individuals to their offspring in a process that mimics a form of inheritance. The
focus in this section is on the recombination operator for real-valued encoding.
Thus, this section firstly presents the intermediate recombination operator and
then proposes its adaptive version for the hybrid EC model.

A. Intermediate Recombination Operator

This operator is mainly applicable to real-valued encoding and it is an extension
of the simple line recombination operator which generates offspring only along
the straight line connecting the parent. With intermediate recombination, a new
offspring variable $\hat{x}_i$ is generated via linear combination – a sort of averaging
process – of the corresponding variables of the parent individuals. Hence, the
resulting offspring variable may lie within or outside the hypercube defined by
the parent.

For any two parent individuals $X$ and $Y$ in a problem of dimensionality $n$,

$$X = (x_1, x_2, \ldots, x_n)$$
$$Y = (y_1, y_2, \ldots, y_n),$$

(7.7)

the offspring variables $\hat{X}$ and $\hat{Y}$ produced with the intermediate recombination
operator are:

$$\hat{x}_i = \alpha_i^C x_i + (1 - \alpha_i^C) y_i$$
$$\hat{y}_i = (1 - \alpha_i^C) x_i + \alpha_i^C y_i,$$

(7.8)
where $\alpha_i^C$ is the linear recombination parameter which has a unique value for each variable. In essence, $\alpha_i^C$ weights the contribution of each parent to the generated offspring. Note that if a single value for $\alpha_i^C$ is utilised for all the variables, the intermediate operator reverts to a simple line recombination operator.

For most intermediate recombination designs (Mühlenbein and Schlierkamp-Voosen, 1993), the value of $\alpha_i^C$ is either:

(i) set to 0.5 to enforce an unbiased averaging of the contributions from the parent to their offspring, such that every parent contributes 50% share; or

(ii) is randomly generated using a uniform distribution within the interval $[0, 1]$.

The latter is the most commonly used approach as it allows a stochastic process to guide the recombination operation which would eventually decide the fate of the evolutionary search.

To allow for the generation of offspring in an area within and outside (interpolation and extrapolation) the hypercube defined by the vertices of the two parents (cf. Figure 7.9), $\alpha_i^C$ is often randomly chosen from a slightly larger interval (Mühlenbein and Schlierkamp-Voosen, 1993), such that

$$\alpha_i^C \in [-\alpha^C, 1 + \alpha^C]$$  \hspace{1cm} (7.9)

where $\alpha^C$ defines the degree to which the parent hypercube gets extrapolated for generation of new offspring individuals. Typically, $\alpha^C$ is chosen from an interval $\alpha^C \in [0.25, 0.75]$ (Mühlenbein and Schlierkamp-Voosen, 1993).

As argued by Mühlenbein and Schlierkamp-Voosen (1993), intermediate recombination generates most of its offspring only within, but not around, the boundaries of the hypercube defined by the parent. As a result, the specified area for the generation of possible offspring inevitably shrinks over generations\(^{11}\). Empirical investigation in Mühlenbein and Schlierkamp-Voosen (1993) suggests, statistically, that an $\alpha^C \in [0.25, 0.75]$ is required to counteract the shrinking problem. It however remains difficult to pre-specify an optimum value for $\alpha^C$. This is because a suitable value is largely problem dependent and may vary across different stages of the optimization.

\(^{11}\)The collapse in the offspring hypercube often leads to premature convergence of the search pool.
Figure 7.9: A 3-dimensional illustration of Offspring Hypercube for Recombination operator. This shows two parents (□ squares) with their possible offspring (○ circles) produced within the hypercube formed by the two parents. Notice that the generation of the offspring via this recombination process is analogous to interpolation between the two parent samples.

B. The Proposed Adaptive Intermediate Recombination Operator

To dynamically adjust the size of the hypercube within which an offspring is generated following a recombination of two parent individuals, this section proposes adapting the recombination parameter ($\alpha^C$) in (7.9). Similar to the above adaptive mutation operator (Section 7.6.1), $\alpha^C$ is adjusted with the aid of a closed-loop system that monitors the current state of the spatial diversity $\tilde{C}_{Div}$ in the search pool. However, this model adapts the recombination weighting parameter ($\alpha^C$) based on an exponential function (7.10). The model ensures that while $\alpha^C$ grows exponentially with decreasing spatial diversity $\tilde{C}_{Div}$ in the search pool, it remains bounded within the following interval:

\[
\alpha^C = \begin{cases} 
  e^{-0.8} & \text{when } \tilde{C}_{Div} \geq 0.8 \\
  e^{-0.2} & \text{when } \tilde{C}_{Div} \leq 0.2 \\
  e^{-C_{Div}} & \text{else.}
\end{cases} 
\] (7.10)

This will mean that depending on the current diversity ($\tilde{C}_{Div}$) in the search pool, this proposal partitions the interval (7.9) for the recombination parameter $\alpha^C$ into
CHAPTER 7. ENABLING CONTINUOUS OPTIMIZATION

three:

- Firstly, during high diversity situations (i.e., with $\hat{C}_{\text{Div}}$ of at least 80%), $\alpha^C$ is fixed at $e^{-0.8} \approx 0.45$; this narrows the sampling interval (7.9) to $\alpha^C_i \in [-0.45, 1.45]$, which is the smallest.

- Secondly, for limited diversity (i.e. $\hat{C}_{\text{Div}}$ of at most 20%), $\alpha^C$ is also fixed, but now at $e^{-0.2} \approx 0.82$; this widens the sampling interval (7.9) to $\alpha^C \in [-0.82, 1.82]$, which is the largest.

- Finally, during moderate diversity levels (i.e, $0.2 < \hat{C}_{\text{Div}} < 0.8$), $\alpha^C$ is set to dynamically vary with the exponential function, such that $\alpha^C = e^{-\hat{C}_{\text{Div}}}$; this sets the sampling interval to $\alpha^C_i \sim U(-e^{-\hat{C}_{\text{Div}}}, 1 + e^{-\hat{C}_{\text{Div}}})$, which is moderate and dynamic.

Note that, similar to the above adaptive mutation model, pool’s diversity $\hat{C}_{\text{Div}}$ of at least 80% is considered high, lower than 20% is considered low and anything in between is moderate. These ranges are decided through repeated empirical experiments, and they are used to guide the partitioning process of the recombination parameter interval in (7.9).

For clarity and to demonstrate the dynamics governing the proposed adaptive model (7.10), Figure 7.10 shows the feasible space where an offspring is generated within and/or outside the hypercube (top) of the two parent individuals via interpolation and/or extrapolation in an adaptive manner. Notice that the characteristic curve (bottom) of the adapted range follows the adaptive function described by (7.10). Therefore, the space within which the two parents generate their offspring (i.e., the offspring hypercube) is bounded and grows exponentially with decreasing diversity. Figure 7.10 reveals the following for the proposed adaptive model.

i. When the level of diversity in the search pool is high (i.e. at least 80%), then, $\alpha^C_i \in [-0.45, 1.45]$. This means that the parent individuals undergoing recombination generate offspring, with high probability, within their neighbourhood (niche) and with low probability away from their niche. Inspired by the biological process of niching\textsuperscript{12} (Darwen and Yao, 1996; Goldberg and

\textsuperscript{12}Individuals within a niche (group) are similar to each other, while individuals from different groups share little or no similarities. Niching encourages maintenance of population diversity, and thus better exploration of search spaces.
Deb, 1991), this phenomenon could stimulate formation of dynamic clusters around the basins of attraction in the fitness landscape. The rationale behind this niching effect is to enhance the ability of the search pool to gather information about the search domain by quickly exploiting their neighbourhoods during the early stages of the evolutionary search (i.e. when the diversity is high). This therefore aims to improve robust exploration of complex and rugged problems possessing high levels of nonlinearities and multimodalities. The idea conforms to preservation of multimodal diversity via the use of some form of dynamic clustering as suggested in Kubalšk et al. (2005).

Furthermore, recombination with low levels of $\alpha^C$, when there is already high diversity in the pool, could minimise the chances of producing infeasible offspring which lie outside the search domain boundaries.
CHAPTER 7. ENABLING CONTINUOUS OPTIMIZATION

ii. As the evolutionary search progresses, it is natural for the diversity to eventually shrink. To tackle this, the proposed adaptive model dynamically increases the size of the offspring hypercube. This is achieved by exponentially enlarging the recombination weighting parameter interval, such that $\alpha^C_i \in [-e^{-\tilde{C}_{Div}}, 1 + e^{-\tilde{C}_{Div}}]$ as the diversity falls (see the adaptive range curve at the bottom of Figure 7.10). This ultimately helps to offset any chances of premature loss of diversity in the search pool by triggering further recombination among the previously formed niches/clusters. This model is expected to eventually guide the search towards the high quality regions of the search space for further exploitation.

7.7 Comparison of the Adaptive and Non-Adaptive HESA Algorithm

This section investigates the effect of the adaptation introduced into the mutation and recombination operators of the dual pool HESA model (Algorithm 7.1). The effectiveness of the proposed adaptation strategies is analysed by comparing its performance against that of the previously evaluated non-adaptive HESA algorithm on the DRP benchmarks. Recall that the prior sensitivity analyses (Section 7.4) on the non-adaptive HESA model reveal that the algorithm attained its best performance when a pool size of 50 samples is utilised. Similar investigations (not shown here for brevity) on the newly proposed adaptive HESA algorithm have led to the same conclusion. Therefore, the analyses presented herein are limited to the cases having pool size of 50 samples.

7.7.1 Experimental Goals

Recall that, in addition to potential performance improvement, the adaptation introduced into the original HESA model principally aims to relieve the user of the burden associated with manual parameter tuning, which is often a time consuming trial and error task. Therefore, this section verifies the hypothesis made in Section 7.6 by comparing the adaptive and non-adaptive HESA models with respect to their:

- Relative accuracy – using the offline and online performance measures;
• Robustness – via the absolute errors analysis; and
• Efficiency – using convergence characteristics.

A. Accuracy Comparison

The complete evaluation results for the adaptive HESA model are summarised in Table E.1 Appendix E. From the results obtained during the evaluation of the non-adaptive HESA model in Section 7.5, Figure 7.11(a and b) compares the online and offline performances of the two versions of the HESA model.

An obvious similarity with regard to the dynamics of the two models under various instances of the dynamic change types can be seen in the characteristic shapes of the relative performance curves in Figure 7.11(a and b). While the improvements in relative accuracies (offline and online) achieved by the adaptive HESA model remain rather mild, Figure 7.11(a and b) reveal that the characteristics curves of the adaptive model appear flatter. This means that, regardless of the nature of the dynamic changes, the use of adaptive parameterisation assures consistent and high performance with less user input.

B. Robustness Comparison

This section compares the absolute errors sustained by the non-adaptive/adaptive HESA models as shown in Figure 7.12. Notice the sizeable amount of standard errors suffered by the non-adaptive HESA model especially on the $T_6$ dynamic change instance of the DRP benchmark, which further illustrates the high degree of complexity and change severity in this instance of the DRP benchmark. Importantly, while the reductions in the absolute errors sustained by the adaptive HESA algorithm across most of the dynamic change types remain mild, these improvements (i.e., reduced levels of absolute error) are significant on the dynamic change types having higher levels of change severity, see $T_6$ for example. Such types of dynamic changes generally pose greater tracking difficulties to manually parameterised models.

C. Efficiency Comparison

The two models are compared based on their convergence characteristics on the $T_1$ to $T_6$ dynamic change types; Figure 7.13 shows the comparison results. The convergence characteristics curves in this figure reveal that for the dynamic change
types exhibiting milder change severity and randomness such as $T_1$ and $T_2$, the two models compete head-to-head, both yielding exceptionally good convergence efficiencies. On the other hand, when the dynamic change severity and randomness increase, i.e. for $T_3$ to $T_6$, the non-adaptive HESA model exhibits a much weaker convergence speed. This means that the convergence efficiency...
of the non-adaptive HESA is significantly affected by the increased complexity of the dynamic problem. In contrast, the adaptive HESA model turns out with fairly smoother characteristics – indicating a robust change detection and efficient tracking capability. It is noteworthy that the above three features (accuracy, robustness and convergence efficiency) are required attributes for ensuring optimal performance in dynamically changing environment.

### 7.7.2 Analysis and Interpretation of Adaptation effect

The above findings reveal that the adaptive HESA model demonstrates improved robustness and accuracy over its non-adaptive counterpart. However, in some specific test cases, the non-adaptive HESA model seems to perform best. For instance, the Figures 7.11(a and b) show that under the small step dynamic change type ($T_1$), the non-adaptive model yields slightly better offline/online performances. This discrepancy could be due to the fact that the dynamic change associated with $T_1$ is relatively mild both in severity and randomness. Thus, compared to other dynamic change types ($T_2$ to $T_6$), it seems easier for the manually parameterised model to achieve a near optimum performance on $T_1$. On the other hand, the adaptive HESA model would require some extra effort to learn and gather sufficient knowledge about the complexity of the problem’s landscape.
before coming up with a suitable parameter set for that particular test problem.

Thus, although the adaptive model is more consistent with regards to uniformity in its levels of accuracy and robustness, it could be slightly slower on less complex dynamic problems. A similar behaviour is observed from the convergence characteristics curve in Figure 7.13 for the $T_1$ dynamic change type (where the adaptive algorithm slightly trails its non-adaptive counterpart).

Importantly, it could be said that there is a high tendency for the non-adaptive HESA model to overfit a specific problem instance (see Figure 7.11 where the non-adaptive HESA yields fairly good offline/online performances on $T_1$ but poor on $T_6$). This is an obvious consequence of relying on user pre-defined parameter settings, which risks poor generalisation ability. This effect could result in significant performance deterioration even on similar problem instances. Conversely, it is observed from Figure 7.11 to Figure 7.13 that the adaptive HESA model features higher sense of generalisation over its non-adaptive version (exhibiting fairly uniform performance across all the six dynamic change types).

It could be inferred from the above findings that the adaptive HESA model yields a more robust and stable performance. This suggests that it is more suitable for optimization of wider range of highly multimodal and dynamic problems. The findings also support the earlier hypothesis (Section 7.6) that an adept parameter adaptation technique that utilises the current state of pool’s diversity as feedback can generally improve the performance of a hybrid evolutionary algorithm.

Finally, since the results of this experiment agreed with the well-known facts about the benefits of adaptive parameter control in the EC literature (Eiben et al., 1999), the objective set out earlier in Section 7.6 has been satisfied. Of course further comparative experiments with other adaptation strategies (such as the commonly used fitness-based adaptation) could yield additional information on the impact of the proposed method, but such investigations are considered as future work since they are outside the goal set out in this proposal, which has already been met.
Figure 7.14: Theory Research Relevance Tree: A complete roadmap for the design and development of the Adaptive Dual-pool HESA model for global optimization in stationary and dynamic environments.
7.8 Theory Research Relevance Tree for the Proposed HESA Model

The framework that guides the design and development of the proposed evolutionary computation model is as depicted by the theoretic research relevance tree (TRRT) in Figure 7.14. Highlighting where the proposed method stands in the wealth of EC literature, the TRRT (Figure 7.14) delineates the fundamental research domains, various sub-approaches, and the proposed techniques with their respective data structures – which led to the design of the mathematical model for the proposed adaptive HESA algorithm. In particular, it describes the linkages between the series of investigations, designs and implementations involving: (i) the new convergence analysis techniques (Chapter 4), (ii) various diversity control strategies (Chapter 5), (iii) hybridization methodologies (Chapters 6 to 7), and finally (iv) the adaptation model proposed above (Section 7.6).

While the approaches, components and models outlined in this TRRT (Figure 7.14) are by no means exhaustive of the many considerations for designing EC models, this TRRT paradigm has thus far made possible the design of the various mathematical models proposed in this thesis. Importantly, it lays a solid framework for further research and more theoretical investigations into the design of effective evolutionary-based models for general global optimization purposes.

7.9 Contribution and Remarks

The key contributions in this chapter are:

Firstly, a novel framework for continuous optimization (Figure 7.1) that hybridizes a dual-pool EC model with an improved SQP local search algorithm – forming the newly proposed dual-pool HESA algorithm (Algorithm 7.1). Experiments were designed to evaluate the performance of the proposed dual pool HESA model under dynamically changing environment; various performance metrics such as relative performance measures, absolute errors and convergence efficiency measures (median performances), were utilised. Through this, the chapter presented a rigorous analysis of the behaviour of the HESA model on the dynamic optimization benchmarks. The sensitivity analyses presented in this Section 7.5.1 have provided vital insights into the effect of large sample sizes in evolutionary optimization. Importantly, these are new insights from the perspective of solving
dynamic optimization problems with hybrid EAs.

The second contribution is in the integration of the instantaneous population diversity and evolvability measures to facilitate parameter adaptation. This results in the proposed closed-loop based adaptive models for the mutation and recombination operators in continuous domain. Importantly, experiments show that in contrast to its non-adaptive version,

(i) the proposed adaptive HESA model has minimal need for manual parameterisation; thus, it avoids the risk of falling into a vicious cycle of reparameterisation across various problem types; and

(ii) the closed-loop based adaptation strategy demonstrated significant improvements in robustness with regards to change detection, tracking and convergence efficiency on the complex dynamic optimization test problems; but it mainly maintains acceptable levels of efficiency on the less complex ones.

Therefore, in response to the hypothesis in Section 7.6 and the research question $Q_5$ in Section 1.3.2, for the examined benchmark problems, the proposed adaptation method has greater impact on robust and stable performance than it has on the overall search efficiency.

Furthermore, as is presented later in a comparison test with several other state-of-the-art dynamic optimization algorithms (Section 8.4.2), the proposed HESA model demonstrated competitive performance on the various dynamic optimization test cases. Thus, the comparison results in Section 8.4.2 further validate the significance of the performance improvements in this model.
Chapter 8

Investigating Extended Hybrid Evolutionary Algorithms

Having analysed the effect of adaptation on the proposed hybrid evolutionary model (HESA) in Chapter 7, this final research chapter aims to investigate the effect of extending the proposed hybrid model. In particular, the role of incorporating a derivative-free algorithm into the HESA model will be examined. Of course, the preceding investigations in this thesis have so far demonstrated the seamless cooperation of the global evolutionary computation (EC) (Algorithm 7.1) and local sequential quadratic programming (SQP) (Algorithm 3.1) algorithms on solving both stationary and dynamic optimization problems. However, it is vital to reiterate the existence of a fundamental difference in the modus operandi of the two algorithms. While both algorithms (EC and SQP) run successfully on optimization problems which are precisely defined with a smooth mathematical model, the SQP algorithm does not work in the absence of such mathematical model.

Thus, the motivation here is to extend the proposed hybrid model to solve optimization problems without necessarily evaluating their derivatives. The goals are twofold:

(i) to minimise the cost of evaluating derivatives for differentiable problems with expensive cost functions; and

(ii) to accommodate such problems that lack precise mathematical representations (i.e., black-box problems).

In Section 8.1 this chapter introduces a new derivative-free optimization method
that is based on a *stochastic coordinate ascent* (SCA) algorithm. The proposed derivative-free method then complements the local search component of the previously proposed *adaptive* HESA model; this forms a new extended hybrid model (ext-HESA) that comprises the three different algorithms (EC, SQP and SCA) (Section 8.3). Following the evaluation of the ext-HESA in Section 8.4, Section 8.5 proposes a generalised framework for extended hybrid EAs. The chapter concludes with a summary of the key contributions and a remark in Sections 8.6 and 8.7 respectively.

### 8.1 The stochastic Coordinate Ascent (SCA) Derivative-Free Algorithm

As a derivative-free optimization approach, the coordinate ascent (CA) algorithm is an inexpensive classical multidimensional optimization method; it has its origins in the field of numerical programming, see Brent (1973); Schwefel (1993) for historical details\(^1\). Fundamentally, the CA algorithm works by decomposing an \(n\)-dimensional optimization problem into \(n\) one-dimensional sub-problems. Then, the algorithm – which was shown to have linear time complexity (Loshchilov et al., 2011) – cycles through the different coordinate directions during the search. At every iteration, the resulting one-dimensional optimization sub-problem is solved using any suitable one-dimensional optimization algorithm such as a line search.

Before we delve into the detail proposal in this section, it is imperative to reiterate the following two points. Firstly, this proposal is not a departure from the scope of this thesis (Section 1.4), which is focused on optimizing continuous (non-)linear problems that are at least twice differentiable. In fact, one of the key objectives of this chapter is to enable local optimization of such problems but without necessarily evaluating their derivatives. Therefore, although the traditional coordinate ascent algorithm is known to have convergence problems with non-smooth (discontinuous) problems (Bezdek et al., 1987; Loshchilov et al., 2011)\(^2\), this is not a matter of concern for the method proposed in this section. Secondly, while this method is traditionally known as coordinate descent

---

\(^1\)CA is also called *coordinate strategy* in the EC community (Schwefel, 1993).

\(^2\)Coordinate ascent/descent algorithm is said to have convergence problems with non-smooth functions, hence the proposal of its variants including the so-called adaptive coordinate descent algorithm, see Acar and Rais-Rohani (2009).
algorithm (for minimisation), the proposal here is named coordinate ascent because the theme in this thesis is centred on optimization from the perspective of maximisation. Otherwise, this proposal applies to both min and max.

Given a general optimization problem (9.1) in continuous domain:

$$\text{maximise: } y = f(x) : x \in \mathbb{R}^n,$$  \hspace{1cm} (8.1)

where $f(x)$ is a linear/nonlinear function of $n$-dimensional variables, the proposed stochastic coordinate ascent (SCA) algorithm is as outlined in Algorithm 8.1. Unlike the traditional CA method, the SCA algorithm locally optimises (8.1) by randomly\(^3\) searching a set of coordinates around the current solution point $x_k$ in every iteration. As can be seen from Algorithm 8.1 (lines 1-6), SCA operates only within a defined neighbourhood ($\mathcal{D}$) of the initial search point $x_t$. The initial neighbourhood size, defined by $\delta^i_R | i = [1,n]$, is set to 1% of the width of the problem’s search domain across all $n$ dimensions (line 5). Note that the 1% initial radius is empirically decided so as to constrain the SCA algorithm to focus around a limited area of the starting point $x_t$. This restricted form of local search also minimises the risk of creating infeasible solutions (i.e. points outside the feasible bounds) by the SCA algorithm. During the optimization process, the size of this neighbourhood ($\mathcal{D}$) is expected to continuously shrink over iterations (line 12). Therefore, as a stopping criterion, a minimum threshold for the size of the search domain $\delta_{\text{min}}$ is empirically set to $10^{-8}$ (lines 6-7).

Having set its initial parameters, the main SCA algorithm (Algorithm 8.1) iterates through the steps in lines (7-14) until the stopping condition is satisfied. In line 8, the SCA Algorithm invokes the $\text{GenerateAndEvaluateSearchPoints}$ sub-function (Algorithm 8.2) with the current search point $x_t$, the neighbourhood radius $\delta_R$, and the problem dimension $n$ as parameters. As will be described shortly, this sub-function generates and evaluates the fitness of a new set of (fixed-size) search points $\mathcal{X}_D, f(\mathcal{X}_D)$ within the neighbourhood ($\mathcal{D}$) of the current point $x_t$. Now, if the fitness of the best sample in the set $\mathcal{X}_D$ is greater than that of $x_t$, then the best sample becomes the next solution point $x_{t+1}$ (lines 9-10). Otherwise the radius of the neighbourhood is halved (lines 11-12).

\(^3\)Random here means selection of a set of dimensions in the state space in a stochastic manner.
Algorithm 8.1 Stochastic Coordinate Ascent Algorithm (SCA)

1: Problem’s dimension \( n \);
2: Problem’s search bounds \([\bar{x}_i, \underline{x}_i] : i = 1 \) to \( n \);
3: Starting point \( x_t \);
4: Search neighbourhood \( \mathcal{D} \);
5: Search neighbourhood radius \( \delta_R \leftarrow 0.01 \times (\bar{x}_i - \underline{x}_i) : i = 1 \) to \( n \);
6: Minimum neighbourhood radius \( \delta_{\text{min}} \leftarrow 10^{-8} \);
7: repeat until termination condition: \( \delta_R < \delta_{\text{min}} \)
8: \([X_D, f(X_D)] \leftarrow \text{GenerateAndEvaluateSearchPoints}(x_t, \delta_R, n) \);
9: if \( \max(f(X_D)) > f(x_t) \) then // when a better solution is found
10: \( x_{t+1} \leftarrow x_D^j | f(x_D^j) = \max(f(X_D)) \); // replace \( x_t \) with its best neighbour
11: else // if \( x_t \) is better than all its neighbours
12: \( \delta_R \leftarrow \frac{1}{2} \delta_R, \forall i = 1 : n \); // shrink the neighbourhood radius
13: end
14: end (repeat)
15: return \((x_t, f(x_t))\)

Note that maximisation problem is assumed.

The above procedure (lines 7-14) is repeated until the size of the neighbourhood falls below the user defined threshold \( \delta_{\text{min}} \) (line 7). Finally, the SCA algorithm returns a pair \((x_t, f(x_t))\) (i.e., a solution point with its fitness value) which is the locally optimum solution in the region of the initial starting point (line 15). Note that the iterative shrinking of the search neighbourhood ensures that the SCA algorithm always eventually terminates.

### 8.2 Samples Generation for the SCA Algorithm

As mentioned above, the sub-function \((\text{GenerateAndEvaluateSearchPoints})\) invoked by Algorithm 8.1 (line 8) is responsible for generation and evaluation of the appropriate search points in the neighbourhood \( \mathcal{D} \). The listing for this sub-function is outlined in Algorithm 8.2. The illustration in Figure 8.1 graphically describes the process of generating the sample search points. The neighbourhood of the current search point \( x_t \) is sampled in a circular (or spherical) manner; and the number of the sample points depends on the dimensionality \( n \) of the search domain (see Algorithm 8.2, lines 3 and 7). From Figure 8.1, note that the set of the sample points \((X_D)\) in the neighbourhood of \( x_t \) consists of two points for a 1-dimensional problem, four points for a 2-dimensional problem and six points for an \( n \)-dimensional problem given that \( n > 2 \). Accordingly, the cardinality of
Algorithm 8.2 Neighbourhood Samples Generation Function

1: \textbf{GenerateAndEvaluateSearchPoints}(x_t, \delta_R, n)
2: \hspace{1em} \textbf{if} (problem dimension } n \leq 3 \textbf{ then}
3: \hspace{1em} \mathcal{X}_D \leftarrow \{ [x_i - \delta_R, x_i + \delta_R], \forall i = 1 : n \}; \quad \text{// generate } 2n \text{ samples at } \delta_R \text{ from } x_t
4: \hspace{1em} f(\mathcal{X}_D) \leftarrow \text{evaluate the fitness of samples in } \mathcal{X}_D;
5: \hspace{1em} \textbf{else}
6: \hspace{2em} k \leftarrow \text{rand}(n, 3); \quad \text{// chose any } 3 \text{ out of the } n\text{-dimensions;}
7: \hspace{2em} \mathcal{X}_D \leftarrow \{ [x_i - \delta_R, x_i + \delta_R], \forall i = 1 : k \}; \quad \text{// generate } 2k \text{ samples at } \delta_R \text{ from } x_t
8: \hspace{2em} f(\mathcal{X}_D) \leftarrow \text{evaluate the fitness of samples in } \mathcal{X}_D;
9: \hspace{1em} \textbf{end}
10: \textbf{return } \mathcal{X}_D, f(\mathcal{X}_D)

the set of the sample points, \( \mathcal{X}_D \), is in general given by

\[
|\mathcal{X}_D| = \left\{ \begin{array}{ll}
2n, & \forall n \leq 3, \\
6, & \text{otherwise.}
\end{array} \right. \tag{8.2}
\]

It is noteworthy that for higher dimensions \( (n > 3) \) the proposed SCA algorithm essentially conducts a block coordinate ascent search (Richtárik and Takáč, 2012), that is, the algorithm searches the multi-dimensional neighbourhood by successively optimizing any three randomly chosen dimensions at every iteration – hence the naming convention stochastic coordinate ascent.

8.3 The Proposed Extended HESA model

This section integrates the proposed SCA derivative-free algorithm into the local optimization stage\(^4\) of the hybrid model. The flow diagram in Figure 8.2 describes the newly proposed hybridization framework. The overall hybrid model now comprises a global (EC) model coupled in a task-switching manner to a set of local (SCA and SQP) algorithms.

Notice from Figure 8.2 that this proposal resembles the hybrid framework earlier described in Section 7.1; however, it precedes the SQP algorithm with the newly proposed derivative-free SCA algorithm. The motivation behind this proposal stems from the following key points\(^5\).

i. \textit{Computational complexity:} Recall that the proposed SQP is a 2nd order

\(^4\)Local optimization stage of the adaptive HESA model previously analysed in Chapter 7.

\(^5\)The motivation also follows some additional insights gained during the series of experimentations – especially through the Evolution in Action (EiA) tool introduced in Section 1.7.
CHAPTER 8. INVESTIGATING EXTENDED HYBRID EAS

Figure 8.1: The search neighbourhood for the SCA algorithm for an \( n \)-dimensional search domain. The current search point \( x_t \) represented by a black (red in colour) circle is at the centre of the linear (for \( n = 1 \)), circular (for \( n = 2 \)) or a spherical (for \( n > 2 \)) search domain. Situated at a neighbourhood radius \( \delta_R \), the new search points represented by grey circles (green in colour) are 2, 4 or 6 depending on the dimensionality of the search domain.

algorithm and although it has a Quadratic rate of convergence (see Section 3.4), the SCA algorithm which is linear remains computationally cheaper.

ii. Pre-processing: Recall also that, to assure robust convergence, it is essential to initialise the SQP in the vicinity of the optimum solution point. Therefore, preceding it by the SCA algorithm helps to pre-process the solution \( x^* \) (Figure 8.2) returned by the global EC algorithm; this then prepares it for a more intense derivative-guided local optimization by the SQP algorithm.

Thus, with the SCA and SQP algorithms collaborating as local optimizers, this extended framework is expected to boost the overall efficiency and effectiveness of the local optimization stage of the newly proposed hybrid framework.

8.4 Evaluation and Analysis of the Proposed Extended HESA Algorithm

This section evaluates the proposed ext-HESA model (Section 8.3) on the dynamic optimization benchmark suite introduced in Section 7.4. The experiments
Figure 8.2: A hybrid model for effective and continuous optimization. Titled ext-HESA, this combines a global EC algorithm with a set of derivative-free (SCA) and exact-derivative (SQP) local optimization algorithms. Stopping requires user limit on function evaluations. The convergence check for the global EC algorithm is as detailed in Section 4.5.
evaluate two different perspectives. Firstly, the performance of the ext-HESA model is compared against that of the original HESA model previously evaluated in Section 7.7. Secondly, both algorithms are compared to the best performing algorithms in a dynamic optimization competition.

8.4.1 Comparison of the HESA and ext-HESA models

This section utilises similar evaluation criteria and parameterisations that are introduced in Section 7.7. The experiments compare the performance of the newly proposed ext-HESA model with that of the original HESA with respect to:

i) accuracy levels – using relative performances;

ii) robustness – using absolute errors; and

iii) tracking efficiency – using convergence graphs for median performances.

For subsequent analysis, the comparison results of the relative performances are as shown in Figure 8.3. Recall from Section 7.4 that the relative performances describe the overall optimization convergence accuracies of the compared algorithms. Both plots in Figure 8.3(a and b) show that, in comparison to the HESA algorithm, the ext-HESA model has significantly improved the offline (Figure 8.3a) and the online (Figure 8.3b) performances. Specifically, the offline performance ($r_{\text{offline}}$) of the new algorithm now remains within 95-100% accuracy levels; whereas the online performance ($r_{\text{online}}$), which actually describes the real-time optimization accuracy, swings within 90-100% across all the six different dynamic change types ($T_1$ to $T_6$).\(^6\)

On both plots in Figure 8.3, the characteristic curves for the ext-HESA across $T_3$ (random) to $T_6$ (recurrent-noisy) dynamic change types reveal noticeable improvements. This is important since the dynamic change types $T_3$ to $T_6$ exhibit higher levels of change severity and randomness – making them more challenging to dynamic optimization methods. It is sufficient to say that the incorporation of the SCA derivative-free local optimization algorithm into the HESA model (Figure 8.2) has upgraded its global and dynamic optimization abilities to within a minimum of 90% accuracy levels.

\(^6\)A complete description of these performance measures and the six dynamic change types ($T_1$ to $T_6$) has been provided in Section 7.4, Table 7.3.
Figure 8.3: Comparison of the relative performances of the ext-HESA and HESA Algorithms. Results are averages of 20 independent runs. (a): Compares the Offline performances ($r_{\text{offline}}$). (b): Compares the Online performances ($r_{\text{online}}$).

On the other hand, Figure 8.4 shows that the reduction in absolute errors (earlier detailed in Table 7.4) suffered by the ext-HESA algorithm is quite mild. This is not hugely surprising since the introduction of the SCA local optimization algorithm into the HESA model is mainly expected to improve its overall
search efficiency. Besides, both the ext-HESA and HESA models enjoy the same benefit of having adaptive evolutionary operators (see Section 7.6 for details on the adaptation principle) – which is the main aspect that minimises convergence errors. Nevertheless, compared to HESA model, the ext-HESA model witnessed some improvement in this regard. This is especially true on challenging dynamic problems like $T_3$ to $T_6$ (Figure 8.4). Ultimately, the observed improvement in accuracy (Figure 8.3) and the mild reduction in absolute error (Figure 8.4) corroborate the improved tracking (convergence) efficiency (see Figure 8.5) exhibited by the ext-HESA model as compared to its exact-derivative counterpart.

8.4.2 Comparing the HESA models with Dynamic Optimization Competition Algorithms

The dynamic optimization competition in the 2009th congress on evolutionary computation (CEC2009) features a variety of global optimization algorithms designed to cope with non-stationary optimization landscapes. This section compares the performances of the proposed HESA algorithms\(^7\) with that of the five

---

\(^7\)The performance comparison involves both the ext-HESA and HESA models.
Table 8.1: Comparison of the Congress on Evolutionary Computation (CEC2009) Dynamic Optimization Competition Algorithms with the HESA algorithms on the DRP benchmarks.

<table>
<thead>
<tr>
<th>No.</th>
<th>Acronym</th>
<th>Algorithm Details</th>
<th>Reference</th>
<th>Total score</th>
<th>Percentage score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CEC:DASA</td>
<td>Differential Ant-Stigmergy Algorithm</td>
<td>Korošec and Šilc (2009)</td>
<td><strong>567</strong></td>
<td>94.5</td>
</tr>
<tr>
<td>2</td>
<td>ext-HESA</td>
<td>Extended adaptive HESA Algorithm</td>
<td>see Section 8.3</td>
<td><strong>561</strong></td>
<td>93.5</td>
</tr>
<tr>
<td>3</td>
<td>CEC:jDE</td>
<td>Self-Adaptive Differential Evolution</td>
<td>Brest et al. (2009)</td>
<td>552</td>
<td>92.0</td>
</tr>
<tr>
<td>4</td>
<td>CEC:CPSO</td>
<td>Clustering Particle Swarm Optimizer</td>
<td>Li and Yang (2009)</td>
<td>545</td>
<td>90.8</td>
</tr>
<tr>
<td>5</td>
<td>HESA</td>
<td>Adaptive HESA Algorithm</td>
<td>see Section 7.1</td>
<td><strong>507</strong></td>
<td>84.5</td>
</tr>
<tr>
<td>6</td>
<td>CEC:EP</td>
<td>Evolutionary Programming with Ensemble of Explicit Memories</td>
<td>Yu and Suganthan (2009)</td>
<td>498</td>
<td>83.0</td>
</tr>
<tr>
<td>7</td>
<td>CEC:DAI</td>
<td>Dynamic Artificial Immune Algorithm</td>
<td>de França and Von Zuben (2009)</td>
<td>458</td>
<td>76.3</td>
</tr>
</tbody>
</table>

Note: The HESA algorithms did not participated in the CEC2009 competition. The algorithms are sorted in order of their performances with the total score of the winning algorithm highlighted in **Bold**. The maximum total score (600) is obtained by summing the scores of each algorithm across the six instances ($T_1$-$T_6$) of the DRP benchmark. The two hybrid evolutionary models (HESA) proposed, designed and evaluated in this thesis are highlighted in grey backgrounds and use gradient information.

best performing algorithms on the dynamic rotation peaks (DRP) benchmark. Again, note that the detail description of the DRP benchmark and the evaluation criteria are given in Section 7.4.

Table 8.1 details the participating algorithms for the CEC2009 dynamic optimization competition. Note that the comparison results in this table are obtained from the overall online performances for the six different dynamic change types ($T_1$ to $T_6$) on the DRP benchmark (F1) function. As referenced in Table 8.1, the results for each of these algorithms are published in their respective accompanying articles in the proceedings of the CEC2009 conference. For clarity and ease

---

8Note that the DRP benchmark is labelled F1 in the CEC2009 competition. Other benchmarks evaluated during this competition (not considered here) include various hybrid composition functions.
of analysis, the comparison results for the seven algorithms listed in Table 8.1 are graphically illustrated in Figures 8.6 and 8.7.

It is vital to note that while the proposed HESA models utilise gradient information, all the other contesting algorithms in the CEC2009 competition also have their unique design characteristics which make them able to cope with the multimodal changing landscapes. In fact, the criteria guiding this dynamic optimization competition (Li et al., 2008) require that no algorithm should be informed when a change has occurred, but it does not preclude using any heuristics (such as hybridization, ensemble, adaptation, clustering or gradient information where applicable) that could enhance the performance of the constituent algorithms. As can be noticed from Table 8.1, the contesting algorithms come from various stochastic evolutionary computation domains and feature a variety of heuristics. Therefore, this comparison test is fair to all the participating algorithms.

Generally, the results in Figure 8.6a reveal the remarkable performances exhibited by the seven evolutionary-based global optimization algorithms on the various dynamic landscapes. It is observed that (Figure 8.6a) all the competing algorithms exceed a minimum online performance score of 65%. In particular, the least performing algorithm (CEC:DAI), see Figure 8.6b, has an overall percentage score of 76.3% (Table 8.1) and reports a minimum online performance of 67.4% on the $T_6$ dynamic change type (see Figure 8.6a). A careful examination of Figure 8.6a reveals that amongst all the seven algorithms, the CEC:DAI algorithm which uses artificial immune system heuristics, exclusively suffers a piecewise monotonic performance degradation across the $T_1$ to $T_6$ dynamic change types. Nevertheless, this is not entirely unexpected since it was established earlier (Section 7.5) that complexity increases from $T_1$ to $T_6$. In fact, the observed trend in Figure 8.6a further justifies that $T_1$ is the least challenging test case to the majority of the competing algorithms, whereas $T_6$ poses a high degree of change severity and randomness – making it one of the most challenging problems in this regard.

From the side of the best performing algorithms, Figure 8.6b shows that the Ant colony-based CEC:DASA algorithm\(^9\) comes first with an overall percentage

\(^9\)CSA:DASA is an advanced Ant-colony method that uses differential ant stigmergy technique to refine solutions over generations. Stigmergy is a derived Greek word which refers to the “Stimulation of workers by the performance they have achieved” (Bonabeau, 1999). It is based on the intuition that an agent’s actions leave signs in the environment, which it and other agents can sense to determine their next-line of actions. The technique seems to have shown great promise in avoiding premature convergence; this could provide a justification for the observed effectiveness of the CSA:DASA algorithm in solving the DRP dynamic benchmark.
Comparing Online Performances ($r_{online}$) on DRP Benchmark

(a) CEC2009: Performance Comparison

Online Performance Summary for Dynamic Changes T1-T6

(b) CEC2009: Cumulative Score

Figure 8.6: Comparison of HESA models with the IEEE CEC2009 Dynamic Optimization Algorithms on Rotation Peaks Benchmark: (a): Compares individual online performances of the seven algorithms for the six different types of dynamic change types $T_1$-$T_6$. (b) Summarises the cumulative online score across the $T_1$-$T_6$ dynamic changes. The higher the cumulative score the better – total maximum possible score is 600%.
Comparing Online Performances ($r^{online}$) on DRP Benchmark

Figure 8.7: Exclusive performance comparison of the two HESA models with the CEC:DSA Algorithm on the $T_1$ to $T_6$ dynamic change types of the DRP benchmark.

...
Figure 8.6b (or Table 8.1) shows that the original HESA model outperforms only two of the CEC2009 competition algorithms on this benchmark.

Finally, Figure 8.7 exclusively compares the CEC:DASA algorithm with the ext-HESA and HESA models proposed in this thesis. Notice that, for the two best performing algorithms in this evaluation (ext-HESA and the CEC:DASA), the performance characteristics in Figure 8.7 revealed some subtle differences between the two models. The characteristic curves of the two algorithms (Figure 8.7) remain intertwined across the $T_1$ to $T_6$ dynamic change types. This means that where one of them fails, the other picks up. Hence, it could be said that, on this benchmark, none of the two algorithms outperforms the other in all respects (see no free lunch and benchmarks (Duéñez Guzmán and Vose, 2013; Ho and Pepyne, 2002)). But, importantly, the vital insights drawn from these findings could guide the selection of appropriate model for optimization of any given non-stationary environment.

### 8.5 A Generalised Extended Hybrid Framework

Thus far, the performance analyses of the various hybrid models introduced in this thesis have highlighted the need for a flexible framework that combines the global EC algorithm with the two (exact-derivative (SQP) and derivative-free (SCA)) local search algorithms. As such, this section proposes a generalised framework (Figure 8.8) that features three different execution paths of hybrid models. This proposal aims to allow optimizing moderate and complex problems with or without directly evaluating their derivatives; and importantly, to permit solving such complex optimization tasks which may lack explicit functional representations.

Therefore, given any optimization task (in continuous domain) the controller in Figure 8.8 selects an appropriate execution path (path 1, 2, or 3) depending on the following scenarios:

i. availability of precise mathematical representation;

ii. differentiability of the mathematical model; and

iii. computational complexity of the optimization task.

For the first scenario, if the optimization task lacks precise mathematical representations.
Figure 8.8: A Generalised Framework for Extended Hybrid Algorithm with three execution paths for the local refinement of the global algorithm’s solution. Depending on the problem at hand, the controller dynamically chooses a suitable execution path.

representation\textsuperscript{10}, then, the controller takes the second execution path. This second path conducts a global search to acquire a global picture of the problem domain using an evolutionary algorithm; it then refines the resulting solution with a

\textsuperscript{10}Lack of precise mathematical model is typical in poorly understood problems or such requiring some heuristic or even human evaluator like in \textit{interactive evolutionary computation}, see Inoue and Takagi (2008); Kowaliw et al. (2012); Mizutani et al. (1995); Takagi (2001).
derivative-free SCA local search algorithm. As introduced earlier (Section 8.1), the SCA algorithm is a variant of the coordinate descent algorithm which has linear time complexity, and like the EC, it is derivative-free.

For the second scenario, if the problem to be optimized has a smooth (at least twice differentiable) functional representation, then the controller chooses the first execution path. Similarly, this path starts with a global search using the EC algorithm, but then refines the resulting solution with a gradient-guided SQP algorithm. The SQP algorithm (Section 3.5) is a 2nd order local search method that ensures quadratic convergence rate if initialised in the vicinity of the optimum solution point.

For the last scenario, the controller goes for the third execution path if and only if the optimization task has a precise mathematical representation, the mathematical model is differentiable, and the cost of function and derivative evaluations is not prohibitively expensive. While such problems are not necessarily the norm in all fields, they actually are commonplace in nowadays complex engineering applications (see Chai et al. (2013); Giannakoglou et al. (2006) on computational fluid dynamics problems for example).

At this point it is natural to ask whether there is any trade-off in choosing one execution path (Figure 8.8) over the others. Unfortunately, the answer is yes. When an optimization task has an explicit mathematical model and is smooth, then there is certainly an efficiency trade-off between the first and the third execution paths. To understand this trade-off, it is sufficient to reflect on the outcomes of the various experiments presented in this thesis:

The investigations in Chapters 6 to 7 are manifestations of the effectiveness of the first execution path depicted by Figure 8.8. The results in those chapters show that, for smooth (differentiable) problems, a hybrid of the global EC and the local SQP algorithm provides sufficiently good optimum solutions (outperforming the standard EC algorithm). This holds for global optimization tasks in both stationary (Section 6.4) and dynamic (Sections 7.4 and 7.7) environments. On the other hand, after comparing the first execution path with the third execution path (i.e., the HESA vs. ext-HESA) on the DRP benchmark in Section 8.4, the results (Figure 8.7) show a clear difference in their performance characteristics. The analysis in Section 8.4 revealed that the third execution path is the most preferred and should be used whenever possible. This is because the combination of the SCA and SQP algorithms in the third execution path forms an extended
1: if no mathematical model then  // User provides this information
2:     go to path 2;
3: else
4:     if model not differentiable then
5:         go to path 2;
6:     else if derivative cost acceptable then  // User may set the value for $\delta_{\text{cost}}$
7:         go to path 3;
8:     else
9:         go to path 1;
10: end
11: end

Figure 8.9: Controller rules for the Generalised Extended Hybrid Framework. See Figure 8.8 for the algorithmic makeup of execution paths 1, 2, and 3. Note that each path begins with the global EC algorithm.

local search framework that enhances both convergence efficiency and robustness.

From a first look at the composition of this proposal (Figure 8.8), the superiority of the third execution path over the others may seem counter-intuitive. This is because, in theory, the incorporation of the SCA algorithm adds to the complexity of the overall hybrid system. However, the performance improvement is not entirely unexpected because the SCA algorithm further refines the best solution returned by the EC algorithm, thereby easing the task left for the gradient-guided SQP algorithm. The SQP is then in a better position to converge to a better optimum solution in much fewer iterations.

Finally, the control settings in Figure 8.9 suggest a setting for the controller used in Figure 8.8. Note that based on this proposal, the generalised extended hybrid framework would require the user to explicitly supply the information about the availability of a mathematical model for the problem at hand (line 1). In addition, the user would be allowed to recommend a threshold for the cost (in execution time) of the derivative evaluations ($\delta_{\text{cost}}$). This parameter – derivative cost ratio (8.3) – signifies the percentage with which the cost of evaluating the derivatives outweighs that of function evaluation. Therefore,

$$\delta_{\text{cost}} = \frac{\text{cost of function evaluation}}{\text{cost of derivative evaluation}} \mid \delta_{\text{cost}} \in [0, 1].$$ (8.3)

Notice from (8.3) that for low cost differentiable functions, $\delta_{\text{cost}} \to 1$, otherwise $\delta_{\text{cost}} \to 0$. Therefore, if no such information is available to the user, the controller could default to a pre-defined threshold of, say $\delta_{\text{cost}} = 0.75$; and based on this
(Algorithm 8.9, line 7), the controller may automatically choose either the first or the third execution path.

8.6 Contributions

Thus far, the various proposals and subsequent investigations in this chapter have provided some vital new insights in the following ways:

i. The chapter proposed a stochastic coordinate ascent (SCA) derivative-free algorithm (Algorithm 8.1) for low cost local searching. The SCA algorithm is a novel stochastic block coordinate search method that solves local optimization tasks with/without explicit mathematical model for the problem. By focusing the search to a limited local area, SCA assures local convergence. In other words, the method gradually shrinks its search area to maximise focus on the search neighbourhood of interest.

ii. This chapter also proposed a new hybrid architecture for effective optimization (ext-HESA). The newly proposed model (Figure 8.2) combines the earlier proposed HESA model with the SCA algorithm. Performance comparisons reveal that the proposed model is only second to the best algorithm (CEC:DASA) in the CEC2009 dynamic optimization competition. Importantly, from the experimental results on various instances of the DRP benchmark, it was established that neither the ext-HESA nor the CEC:DASA is better than its counterpart in all respects.

iii. Finally, this chapter developed a novel extended hybrid framework that features three separate execution paths. The proposed hybrid system is suitable for optimizing differentiable and non-differentiable problems in continuous domain regardless of the availability of problem representation. Importantly, this generalised framework (Figure 8.8) is reusable as it can easily be tailored to several other optimization domains. For instance, it can be applied to discrete problems by simply replacing the constituent algorithms with their discrete counterparts.
8.7 Conclusion

This chapter proposed a derivative-free local optimization method called stochastic coordinate ascent algorithm. In a series of empirical experiments, the chapter highlighted the effectiveness of extended hybrid systems which combine global and local search methods. From empirical evaluations on various dynamic optimization case studies, it was found that extending the hybrid framework of a global EC and local SQP algorithm (HESA) with a computationally cheap algorithm has a profound impact on the overall search efficiency and robustness. The chapter therefore suggested a generalised framework that hybridizes multiple optimization paradigms. While the specific design aspects of this framework are still in their infancy and open to further refinements, it is believed that the suggested controller settings would facilitate dynamic selection of appropriate execution path depending on the nature and complexity of the problem at hand.

The following chapter concludes this thesis with a presentation of future investigations on the underlying evolutionary optimization principles; it also proposes a number of future research avenues and highlights some expected application areas for the various optimization models examined thus far.
Chapter 9

Conclusion

To conclude this study of evolutionary optimization from hybrid perspective, this chapter summarises the contents across the two parts of this thesis. In response to the research questions behind this study, this chapter discusses the new insights and contributions gained thus far and their implication in this domain. The chapter then highlights the limitations in the scope, theory and experimentations of this study. Finally, it proposes a series of recommendations for further research from both theoretical and design perspectives. But first, for clarity, the following section recaps the motivation that prompted the several investigations and proposals in this research.

9.1 Development of Ideas

As a framework that is composed of collections of various heuristics, an extensive understanding of hybrid optimization systems requires a thorough investigation into their constituent algorithms. The motivation and concept evolution behind these is discussed in the following:

*Genetic algorithm:* Out of the plethora of stochastic global search methods, many of which are evolutionary-based heuristics, *genetic algorithm* (GA) is chosen in this study. This was motivated partly by the suitability and applicability of GAs to various domains\(^1\), and partly by their ease of implementation, robustness, and relatively fair computational complexity.

*Sequential quadratic programming:* Beside the stochastic methods, there are

\(^1\)GAs are well-understood adaptable heuristics that can be easily tailored to solve global optimization tasks of either continuous or discrete/combinatorial types.
several numerical programming methods that could provide local refinement to a hybrid framework. However, the realisation that efficient convergence\(^2\) is attainable for some specific categories of Newton-based methods has motivated the adoption of the *sequential quadratic programming* (SQP) algorithm in this study. As described in Chapter 3, with a few enhancements, the SQP algorithm can sufficiently provide the local refinement required by a hybrid framework. In fact, the proposal for the vectorised automatic differentiation method in Section 3.7 was an attempt to provide the SQP with accurate derivatives, at low-cost, to reassure its local convergence.

*Collaborative hybrids with task-switching:* Whilst the general concept of hybridization is quite vast, it also lacks precise formal definition. A survey of the recent hybrid frameworks revealed that some hybrid approaches that aim to guarantee robust convergence do it at the expense of search efficiency (Integrative Hybrids); and on the other hand, other approaches tend to favour search efficiency over robust convergence guarantees (Collaborative Hybrids). This immediately led to the investigations in Chapter 6 which showed that a trade-off between these crucial performance requirements is not entirely inevitable. The proposal of the collaborative (task-switching) hybrid model in this study (Section 6.3) was motivated by the need to ensure improved search efficiency while preserving global convergence of the resulting system. Of course such collaborative frameworks need sound switching criteria; and this is where the extensive investigation (Chapter 4) on convergence detection mechanisms comes in.

*Hybrid framework, extended & flexible hybrids:* It was also noticed that to further minimise computational cost, such as in the conventional evolutionary optimization methods, hybrid models also need to be adaptable and should accommodate (dis)similar problems in respect to the available resources. This led to the extension of the initial hybrid framework (Chapter 6) to a flexible, extended hybrid system as described in Chapters 7 to 8 of this thesis.

### 9.2 Summary

It is common in evolutionary optimization literature to combine approaches from both *stochastic* and *deterministic* methods for global optimization. But hybrids of evolutionary algorithms (EAs) with gradient-based approaches are often deemed

\(^2\)Example, the 2nd order rate of convergence exhibited by the SQP algorithm (Section 3.4)
to suffer computational mismatch; this is due to the need for derivative evaluations in the majority of gradient-guided methodologies which is not a requirement in the EAs. As summarised below, each of the two related research areas (Part I and Part II) comprised in this thesis has uncovered some exciting research avenues in this regard.

**Part I:** The central focus of the first part of this thesis has been to understand optimization methodologies from both the *stochastic* and *deterministic* perspectives. The aim has been to search for opportunities from the wealth of challenges surrounding these domains.

From stochastic optimization side, Part I highlighted the challenges and opportunities for improving essential evolutionary optimization phenomena, including the crucial aspects of *parameterisation* (Chapter 2), *convergence analysis* (Chapter 4) and *diversity control* (Chapter 5). On the other hand, from the deterministic optimization side, Part I showed that a cost effective provision of accurate derivatives to certain classes of gradient-based algorithms (e.g. SQP) solidifies their roles as suitable candidates for hybridization with EAs (Chapter 3).

**Part II:** The second part of this thesis has mainly focussed on analysing evolutionary optimization from a hybridization perspective.

Chapter 6 has surveyed the current state of hybridization with EAs by providing a taxonomy of the hybrid framework. It then proposed a basic collaborative (task-switching) hybrid of global (EC) and local (SQP) algorithms (HESA) that boosts overall search efficiency without compromising global convergence guarantees. To facilitate optimization in both stationary and non-stationary environments, the hybrid algorithm has been extended into a framework for continuous optimization (Chapter 7). While the outcome of evaluations of this HESA model on a set of dynamic rotation peaks benchmarks yielded promising results, it was found to suffer significant tracking difficulties in severe dynamic conditions. Therefore, further investigation on the effect of adaptation on EAs in Chapter 7 has led to the proposal of a *closed-loop adaptive parameterisation* for the mutation and recombination operators. This hybrid model was then developed into a flexible model that hybridized the global (EC) algorithm with a diverse set of local refinement methods (Chapter 8). With an extended applicability, the new framework is suitable to both differentiable and non-differentiable problems.
Overall, Part II revealed that parameter adaptation is essential even on hybrid evolutionary systems. But most importantly, Part II showed that collaborative hybrids that coupled global and local algorithms via a robust convergence detection method need not to trade convergence guarantees for search efficiency.

9.3 What insights are learned from this thesis?

This section revisits the key questions asked by this study by delineating what have been learned from the findings and contributions in this research. It also discusses their implications where applicable.

Understanding evolutionary spaces and parameterisation
In Chapter 2, the survey and investigations on the fundamentals of evolutionary computation have yielded a number of new insights.

Firstly, Chapter 2 has analysed the differences between the two simulated evolutionary spaces – genotype and phenotype – via a simple visualisation (Figure 2.3). This has helped illustrate the processes in which evolutionary operators map sample solutions from one evolutionary space to another. But importantly, the insight has helped to elaborate on the debate of biological plausibility of the present simulated evolution processes. While further research on simulated evolution will undoubtedly enhance the current understandings of the natural evolutionary phenomena, it is thought that the incorporation of any aspect of natural evolution into simulated evolution should be for the benefit of optimization. The aim of simulated evolution should remain on devising effective optimization systems that benefit from the adaptive capabilities inherent in natural systems – but not necessarily mimicking the biological evolution itself. Nevertheless, since it is difficult to predict which aspect of biological evolution may be beneficial for optimization a priori, this issue would continue to remain an ongoing debate for the foreseeable future.

A new role for the evolutionary variation operators
Variation operators in EC are mainly seen as drivers of evolution. By design, such operators sought new information about the problem landscape by evolving new (and diverse) samples across the search space. In addition to this, Chapter 4 has revealed new insights into their impact on understanding convergence dynamics
in EC. By utilising the Price’s theorem as described in equations (4.7), (4.13) and (4.14), Chapter 4 demonstrated how the contributions of every evolutionary operator (e.g. selection, crossover, mutation, inversion, etc.) to the fitness growth can be monitored in real-time. Most importantly, the dynamical effect of the crossover operator was found to reflect, directly, the instantaneous convergence status of the evolutionary search pool.

Empirical investigations in Chapter 4 revealed that, unlike with the mutation operator, the effect of crossover operator fades out as convergence sets in; this insight was essential to devising the new convergence detection method that relies on the concept of evolvability measure. The strategy permits monitoring the cause of convergence (crossover as a driver of evolution) rather than trying to detect it after it had already happened. Thus, the proposed method took a step ahead of the conventional convergence detection methods which mainly rely on spatial diversity or similarity measures.

Diversity control – challenges and opportunities

The investigations in Chapter 5 have revealed how failure of diversity control may hinder evolvability in a search pool. In the EC literature, lack of diversity maintenance is often linked to premature convergence, which leads to stagnation of the evolutionary search on sub-optimal solution. Thus, Chapter 5 has suggested a new diversity control strategy which harnesses:

i) the information derived from the taxonomy of diversity control policies in the literature (Section 5.3);

ii) the insight derived from visualising diversity dynamics (Section 5.5); and

iii) the insights from the analysis of population evolvability in Chapter 4.

The new method separates the search pool into a dedicated evolution pool and an occasionally created diversity pool. In order to reinforce diversity, newly generated diverse samples (created via search space partitioning heuristic initialisation) are dynamically merged with the evolution pool based on the convergence status of the evolution pool. By sustaining useful levels of diversity, the proposed method facilitates responding to optimization tasks in uncertain and non-stationary environments as analysed in Chapters 7 to 8.
Could a hybrid framework balance the exploration-exploitation trade-off?

This is perhaps the most fundamental question that this thesis attempts to address. It is well known that hybrids of evolutionary algorithms benefit from synergy resulting from coupling a global (stochastic) algorithm with a local (stochastic or deterministic) one. Chapter 6 of this thesis examined hybrid systems from various perspectives and delineated a series of design considerations for successful hybrids. Generally, this study found hybrid architectures to be either integrative – having a local algorithm sandwiched into a global one to enable it serve as a sub-function; or collaborative – with the global and local algorithms operating independently in a task-switching manner.

It was found that majority of hybrid models that have integrative architecture enjoy robust convergence characteristics. This is because integrative architectures apply local refinement to almost all sample solutions at every iteration; this allows intense exploration of the search space, but it impedes overall search efficiency. On the other hand, the collaborative architectures apply local refinement to only a selected solution(s) and only at some defined stages of the evolution. Such architectures favour search efficiency but risk poor convergence guarantees. Thus, there is a huge challenge in balancing search efficiency and robust convergence guarantees when either of such methodologies is used. Therefore, a topical question here is – why use the hybrid systems? The fact is while the stochastic methods mostly allow effective global exploration (robust searching), it is the deterministic methodologies that permit good local exploitation (efficient searching); thus, hybridization is probably the best way to achieve both – without trading one for the other.

Therefore, to alleviate the huge computational cost inherent in integrative hybrid architectures, Chapter 6 proposed a novel collaborative hybrid system that combines the EC and SQP algorithms through a robust convergence detection mechanism. Importantly, this proposal differs from the traditional collaborative hybrid methodologies wherein the local refinement is generally applied at pre-defined stages (based on iterations or function evaluations) of the evolution. Herein, our proposed method explores the global search space with the EC and then applies the SQP to the best solution returned by the EC algorithm only when sufficient convergence is detected. As compared to a standard EC model and the standard CMA-ES algorithm on various global optimization benchmarks,
this proposal has shown promise in minimising the number of function evaluations required to converge to the optimal solution (Section 6.4). Although on the CEC2013 benchmarks (Section 6.5) the proposed hybrid framework did not outperform any of the two (known) best-performing hybrid EAs in the recent literature, the comparison results reveal that the proposed hybrid framework has good convergence characteristics.

Hence, while the integrative hybrids may always have to trade search efficiency for robust convergence, it could be said that, with good convergence detection mechanism, collaborative hybrids can simultaneously improve both exploration and exploitation.

*What performance features of hybrid EC benefit from parameter adaptation?*

The majority of adaptive mechanisms applied to optimization methods are set to relieve users from the burden of parameter tuning; this is because effective parameter tuning needs sound understanding of the problem domain besides the metaheuristic itself. But in addition, adaptive systems are also thought to improve overall search performance. To examine this on hybrid EAs, Chapter 7 proposed a closed-loop adaptive mechanism that controls the step sizes of evolutionary variation operators. The proposed method dynamically adjusts the mutation and recombination step sizes based on the current state of diversity in the search pool, i.e., by continuously monitoring the output side of the evolutionary process. This closed-loop architecture (Section 7.6.1) differs from the conventional methods that mostly control evolutionary parameters based on some pre-defined temporal information.

Empirical comparison with its non-adaptive version in Section 7.7 showed that the proposed adaptive mechanism facilitates robust convergence characteristics by minimising convergence errors. However, the adaptation was found (Section 7.7) to have little impact on the overall search efficiency – this is especially on less complex problems. This could be due to the fact that such adaptive systems learn – in real-time – the nature of the problem landscape, and therefore incur some computational cost to generate suitable parameters. Nevertheless, from the outcomes of the empirical investigations in Section 7.7, it could be said that by minimising stochastic variability in the EC dynamics, adaptive parameterisation has a positive impact on the overall performance of hybrid EC models.
Note that, when the key optimization goal is the search efficiency, then, realising the true benefits of such adaptive mechanisms is to some extent problem dependent. This is because while such mechanisms remain highly beneficial on highly complex and noisy problems, on simple problems, their cost (risk of reduced efficiency) may outweigh the benefits (i.e. convergence guarantees).

Extended hybrid systems

Although the last couple of decades have seen a growing interest in hybrid evolutionary models that combine two or more optimization algorithms, the goal has mainly been limited to improving optimization performance. Therefore, Chapter 8 of this thesis proposed an extended hybrid system (Section 8.5) which is designed to handle a variety of optimization problems by dynamically establishing the most effective hybrid model from its constituent algorithms.

To achieve this, Chapter 8 proposed in the first place a new derivative-free stochastic coordinate ascent (SCA) algorithm (Section 8.1) that facilitates local optimization without explicit evaluation of derivatives. The SCA matches the EC model since they are both derivative-free. This means that an EC-SCA hybrid can optimize problems with or without a well-defined mathematical model – for example, such hybrid model can directly be used on interactive evolutionary computation tasks. Then, in a series of experiments, Section 8.4 demonstrated how an extended hybrid system consisting of EC, SCA and SQP algorithms outperforms several other optimization frameworks on a set of dynamic optimization benchmarks. Finally, this study demonstrated how the various algorithms can be combined in a flexible extended-hybrid framework (Section 8.5); this novel proposal improves optimization performance, and most importantly extends applicability to other problem domains.

9.4 Limitations of this study

As mentioned in the Introduction to this thesis (Section 1.4), this study has a number of limitations, and this section attempts to recast them into three main groups. Note that some of these issues would also be discussed as recommended future studies in the subsequent section.
Scope: In its scope, this study is limited to only optimization problems in continuous domain, that is, such problems having real-valued representation. While there are many engineering problems – especially in industrial control applications – that are continuous in nature, there are equally many discrete optimization problems (e.g. in industrial scheduling) which require similar attention from the hybridization perspective. It is noteworthy that, while most of the theoretical principles of the evolutionary optimization presented in Part I apply to both discrete and continuous domains, the design considerations (such as the selected data structures, and chosen operators) and experimentations herein have mainly focused on problems in continuous domain. In addition, the gradient-based local optimization method (SQP) introduced in Chapter 3 is also a numerical method that is exclusively tailored to continuous optimization domain. Nevertheless, the derivative-free SCA algorithm proposed in Chapter 8 can be applied to both domains; and importantly, the various hybrid architectures proposed in this thesis (Sections 6.3, 7.1 and 8.5), are adaptable to various optimization domains.

Experimentations: The strategy adopted in this study was largely experimental; and the empirical investigations have generally utilised test cases (i.e., benchmarks) that have explicit mathematical models. Of course this is necessary to satisfy the SQP algorithm, that is, as a gradient-based 2nd order method, the SQP not only requires the presence of a functional model, but also the model has to be smooth and twice differentiable.

Case Studies: This study has utilised several global optimization benchmarks from across the stationary to dynamic optimization domains. But additional insights could be gained by extending the investigations to other forms of problems, for instance, the real-world case studies from the fields of engineering to bio-techniques. For a typical engineering case study, see our referred previous work (Ximing et al., 2010) in Section 1.7.

9.5 Future Research Directions

Beside the abovementioned contributions, there are several other aspects of this study that have the potential of been advanced in some interesting ways. Thus, this section suggests and discusses some future research questions in this regard.
EC Theory

Diversity vs. Evolvability – what more could be learned? Traditionally, diversity measures are used to assess evolutionary convergence, but this study suggested an evolvability measure (Section 4.4) and uses it (Section 4.5), empirically, to infer convergence of evolutionary pool. On predicting the convergence dynamics of the search pool, the evolvability measure seems to provide more upfront information than the conventional diversity measures. Therefore, it is vitally important to further assess how the population evolvability theoretically relates to diversity and hence convergence. It is thought that applying information theoretic metrics (Abu-Mostafa, 1986; Pincus, 1991; Shannon, 1948), like the entropy and mutual information, to analyse the information content in each of these measures (i.e., Diversity and Evolvability) would provide new insights into their relationships with regard to convergence. Moreover, such analysis would certainly yield additional information by providing more theoretical justifications about the observed advantage of combining the evolvability and diversity measures in this work.

Could hybrid frameworks benefit multi-objective and discrete problems? Firstly, multi-objective optimization (MO) methods are specifically used for solving optimization problems with multiple conflicting objectives. However, the challenges surrounding single objective optimization (e.g. slow convergence, lack of efficient termination criteria, etc.) are also present in the multi-objective frameworks (Sindhya et al., 2013). In fact, the hybrid solutions proposed in this thesis could equally scale in multi-objective optimization. Therefore, as the literature is beginning to see growing interest in this direction (Ahn et al., 2010; Denysiuk et al., 2013; Sindhya et al., 2013; Tang and Wang, 2013; Zhou et al., 2008), extending the proposed flexible hybrid framework to MO could facilitate further understanding of the hybrid multi-objective frameworks in general. Notice that the proposed flexible hybrid framework (Sections 8.3 and 8.5) would mainly require replacing the standard EC model with a state-of-the-art MO model like the NSGA-II (Deb et al., 2002) to build a hybrid multi-objective solver. Thus, it would be interesting to investigate how the new hybrid framework would fare on various multi-objective test-beds. Even though multi-objective optimization is in itself a vast field, the reusability of the proposed hybrid framework will certainly ease conducting such investigation.

Secondly, in a similar manner, the newly proposed hybrid framework can be
extended with additional execution paths consisting of hybrids of global and local
discrete optimization algorithms. Importantly, such extension could maintain the
same architectural construction of the proposal in Chapter 8 but utilise a set of
discrete optimization algorithms instead. This can also be evaluated on several
scheduling problems, for example job scheduling.

How does self-adaptation compare to closed-loop adaptive mechanism? It is be-
lieved that evolution in itself is guided by three main processes, namely: (self-)adaptation, cooperation and competition. In this study, Chapter 7 has demon-
strated the role of adaptation in hybrid EAs with emphasis on adapting the
step sizes of the evolutionary variation operators. Experiments have shown that
the proposed closed-loop adaptation successfully minimises convergence errors in
the hybrid EC model; however, the results have also shown that such adaptive
method yields no significant improvement on the overall search efficiency. There-
fore, it would be vital to investigate how other adaptation methodologies (e.g.
self-adaptation) perform on such hybrid architectures.

Implementing self-adaptation would require restructuring the data structure
of the sample solutions; this would allow parameters like operator probabilities
and step sizes to be evolved during the optimization. Thus, with self-adaptation,
both operator step size and rate (probability) of application can be adapted at
once. However, self-adaptation may not necessarily guarantee efficiency improve-
ments, and as argued by Glickman and Sycara (2000), self-adapting mutation
rates risks premature convergence. The objective here is to investigate whether
self-adapting the step sizes of both mutation and recombination operators can
provide additional information into the general behaviour of adaptive mecha-
nisms on hybrid models.

**EC Design Considerations**

*Parallelisation:* With the advent of efficient graphic processing units (GPUs),
and cheap multi-core and distributed computing facilities, it is pertinent to re-
structure the design of some of the models proposed in this thesis to benefit
from these resources. In particular, with the exception of the selection process,
the population-based EC model is in general amenable to parallelisation. In ad-
dition, the vectorised Automatic differentiation method presented in Chapter 3
simultaneously evaluates the function values, gradient and Hessian using a computational graph. The design of the computational graphs, as illustrated in Chapter 3, could allow evaluation of several graph vertices concurrently. Of course there would be some dependencies that must be resolved especially around the roots of the computational graphs (some of these dependencies are autonomously resolved by the compiler). Nevertheless, including parallelisation frameworks (such as the OpenMP shared memory multiprocessing platform (Foster, 1995)) is expected to boost the computational efficiency of the individual algorithms and also the overall hybrid architecture.

**Dynamic precision control:** Because the evolutionary optimization models employed for global searching in this thesis are population-based in nature, at every generation during the search, every sample solution has its fitness evaluated and stored. This could become an issue – especially when large sized pools are needed – since the precision (granularity) with which every sample solution is represented and evaluated contributes hugely to the overall computational cost. One possible way to tackle this problem is through varying precision during the course of evolution. In fact, low level of precision can suitably be used during the early stages of evolution without much information loss.

Neville (1997) and Neville et al. (2000) showed that by using quantised weights in a Sigma-pi neural network (site-values, $S_m = \{-64, \ldots, 64\}$ stored in 8-bit binary code and quantised activations $a = \{-128, \ldots, 128\}$ stored in 9-bit binary code, it is possible to map real-valued functions to an accuracy of 1%. While the quantisation is important, a more crucial finding is that pre-calculation of large blocks of the computations in a mathematical algorithm are only possible if the parameters are quantised. The impact of this has been empirically demonstrated by Ferguson (1995) on hardware implementation of real-valued hyper-net, but the insight has not as yet been extended to the domains of massively parallel processing (MPP), or hybrid systems – and may lead to fruitful research.

It is therefore thought that a dynamic granularity control that, during early generations, evaluates sample solutions with a limited precision, and gradually increases the precision as optimization nears its end is likely to impact the overall computational efficiency of hybrid evolutionary-based optimization systems.
Appendix A

Vectorised Automatic Differentiation Technique

This appendix elaborates on the automatic differentiation (AD) approach presented in Chapter 3. First, Section A.1 presents the process of overloading some arithmetic operators and functions. Then using a simple example Section A.2 demonstrates how AD evaluates the value, first and second derivatives of a given function in a single run.

A.1 Overloaded Operators & functions for AD Objects

Having constructed the AD objects with their property fields initialised, it is possible to execute all arithmetic operations on them so long as the built-in operators and functions (i.e., the standard real arithmetic operators and other mathematical functions) are properly overloaded to handle objects of their type. This is done by overloading the following default arithmetic operators and mathematical functions in Matlab.

- Arithmetic operators – both the binary and unary versions
- Logarithmic and exponential operators
- Trigonometric functions
- Norm (ABS), etc.

The following sections demonstrate the modelling process of operator overloading for a small set of operators and functions. Note that for all the arithmetic operators, basic chain rule of differentiation (3.21) is applied, but for any arbitrary differentiable functions such as trigonometric, logarithmic, exponential etc., a more general formulation (A.1) is used.

\[ F(X) = (f(x), \nabla f(x), Hf(x)) \equiv (f(x), \dot{x}f'(x), \ddot{x}f''(x)). \quad (A.1) \]
A. Binary Addition/Subtraction

Suppose that the function of interest is a 2-dimensional function \( f \), such that

\[
f(x_1, x_2) = x_1 \pm x_2.
\]  

(A.2)

Then, the following simple graph with root vertices corresponding to each of the two independent (input) variables and a top vertex corresponding to the dependent (output) variable represents the evaluation process graphically.

Let the vertices be: \( v_{-1}, v_0 \) and \( v_1 \), then, applying equation (A.1) yields

\[
v_{-1} = \left( x_1, \frac{dx_1}{dx_1}, \frac{d^2x_1}{dx_1^2} \right) = (x_1, \dot{x}_1, \ddot{x}_1),
\]

and

\[
v_0 = \left( x_2, \frac{dx_2}{dx_2}, \frac{d^2x_2}{dx_2^2} \right) = (x_2, \dot{x}_2, \ddot{x}_2).
\]

Since the proposed implementation seeks to vectorise the definition for each of these vertices\(^1\), their gradient and Hessian fields are initialised based on the dimensionality (\( n \)) of the function under consideration, such that

\[
\dot{x}_1 = \begin{bmatrix} \frac{\partial x_1}{\partial x_1} & \frac{\partial x_1}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \dot{x}_1 & 0 \end{bmatrix},
\]

(A.3)

and

\[
\ddot{x}_1 = \begin{bmatrix} \frac{\partial x_1}{\partial x_1} & \frac{\partial x_1}{\partial x_2} \\ \frac{\partial x_1}{\partial x_2} & \frac{\partial x_1}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \ddot{x}_1 & 0 & 0 \\ 0 & 0 \end{bmatrix}.
\]

(A.4)

Similarly,

\[
\dot{x}_2 = \begin{bmatrix} \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 0 & \dot{x}_2 \end{bmatrix},
\]

(A.5)

and

\[
\ddot{x}_2 = \begin{bmatrix} \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial x_2} \\ \frac{\partial x_2}{\partial x_2} & \frac{\partial x_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \ddot{x}_2 \end{bmatrix}.
\]

(A.6)

\(^1\)This is the way to ensure evaluation of the complete derivatives in a single sweep of the forward accumulation.
APPENDIX A. VECTORISED AUTOMATIC DIFFERENTIATION

Therefore, the input vertices \( v_{-1} \) and \( v_0 \) are redefined as:

\[
v_{-1} = \left( x_1, [\dot{x}_1 \ 0], \begin{bmatrix} \ddot{x}_1 & 0 \\ 0 & 0 \end{bmatrix} \right)
\] (A.7)

\[
v_0 = \left( x_2, [0 \ \dot{x}_2], \begin{bmatrix} 0 & 0 \\ 0 & \ddot{x}_2 \end{bmatrix} \right).
\] (A.8)

Then, the output vertex \( v_1 \) is:

\[
v_1 = (v_{-1} \pm v_0)
\]

\[
= \left( (x_1, [\dot{x}_1 \ 0], \begin{bmatrix} \ddot{x}_1 & 0 \\ 0 & 0 \end{bmatrix}) \pm (x_2, [0 \ \dot{x}_2], \begin{bmatrix} 0 & 0 \\ 0 & \ddot{x}_2 \end{bmatrix}) \right)
\]

\[
= \left( x_1 \pm x_2, [\dot{x}_1 \ \pm \dot{x}_2], \begin{bmatrix} \ddot{x}_1 & 0 \\ 0 & \pm \ddot{x}_2 \end{bmatrix} \right)
\]

\[
= (f, \nabla f, \nabla^2 f).
\]

Notice that an addition/subtraction operation on the two AD objects \( v_{-1} \) and \( v_0 \) yields not only the value, but also the derivative and Hessian of the function \( f \) (A.2) as components of the output AD object \( v_1 \). Such single sweep execution demonstrates the power of the vectorised forward accumulation of derivatives. The same rule applies for multiplication and division operators.

B. The Sine Function

Now consider the following trigonometric function (sine):

\[
f(x) = \sin x.
\] (A.9)

The root vertex \( v_0 \) and the top vertex \( v_1 \) as shown in the graph are defined as:

\[
v_0 = (x, \dot{x}, \ddot{x}).
\]
Therefore, the AD variables are then defined as:

\[
\begin{align*}
    v_1 & = \sin v_0 \\
    & = (\sin x, \dot{x} \cos x, \dot{x} \cos x - \ddot{x} \sin x) \\
    & = (f, \nabla f, \nabla^2 f).
\end{align*}
\]

All other operators and functions are overloaded based on the above principles. In the following, a concrete example is presented to further highlight the key aspects which differentiate the vectorised forward mode AD technique from the traditional symbolic method of differentiation.

A.2 Symbolic vs. vectorised Forward-mode AD

In order to examine the difference in computational approach between the symbolic differentiation method and the forward mode AD technique, let the 2-dimensional function (A.10) be used as a case study. In this example, the value and derivatives (1st and 2nd) of this function are evaluated at \( x = (\pi, \pi/2) \), first via the traditional symbolic method, and then the vectorised forward AD approach described above.

\[
f(x_1, x_2) = (x_1 x_2 + \sin x_1 + 4). \tag{A.10}
\]

A. Symbolic Differentiation

This entails direct substitution of the solution point \( x_k = (x_1, x_2) = (\pi, \pi/2) \) after evaluating the formula for the value and the derivatives using the chain rule.

Value: For the problem under consideration (A.10),

\[
f(x_1, x_2) \bigg|_{x_k} = (\pi \pi/2 + \sin \pi + 4) = \frac{\pi^2 + 8}{2}.
\]

Gradient: The gradient is evaluated in two stages: first, the formula is derived by applying chain rule (3.21) on the partial derivatives of function (A.10). Second, the solution point is substituted into the obtained formula to get the gradient

\[
f'(x_1, x_2) = \begin{bmatrix} \frac{\partial f(x_1, x_2)}{\partial x_1} & \frac{\partial f(x_1, x_2)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_2 + \cos x_1 & x_1 \end{bmatrix}.
\]

Therefore,

\[
f'(x_1, x_2) \bigg|_{x_k} = \begin{bmatrix} \frac{\pi^2}{2} + \cos \pi & \pi \end{bmatrix} = \begin{bmatrix} \frac{\pi^2}{2} & \pi \end{bmatrix}.
\]
**APPENDIX A. VECTORISED AUTOMATIC DIFFERENTIATION**

**Hessian:** Similarly, the Hessian expression is obtained by differentiating the gradient expression as follows:

\[
\begin{bmatrix}
\frac{\partial^2 f(x_1,x_2)}{\partial x_1 \partial x_1} & \frac{\partial^2 f(x_1,x_2)}{\partial x_1 \partial x_2} \\
\frac{\partial^2 f(x_1,x_2)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x_1,x_2)}{\partial x_2 \partial x_2}
\end{bmatrix}
= \begin{bmatrix}
-\sin x_1 & 1 \\
1 & 0
\end{bmatrix},
\]

therefore,

\[
\left. f''(x_1, x_2) \right|_{x_k} = \begin{bmatrix}
-\sin \pi \\
1
\end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix}.
\]

Notice that the above symbolic approach requires the computer to explicitly evaluate and store the formula before substituting the values and solving via the basic real arithmetic. In the following, a vectorised forward mode AD is presented.

**B. Vectorised Forward mode AD**

First, the problem variables (both dependent and independent) are initialised as AD objects. Then, the differentiation arithmetic described above is used to concurrently evaluate the function value and derivatives algorithmically. It is interesting to note that this is done by only evaluating problem (A.10) with the newly defined AD variables using overloaded operators. The problem \((x_1 x_2 + \sin x_1 + 4)\) is decomposed into the following three input variables (vertices):

\[v_{-2} = x_1; \quad v_{-1} = x_2; \quad v_0 = 4.\]

Then, the vertices are initialised as AD objects based on the solution point \(x_k = (x_1, x_2) = (\pi, \pi/2)\), as follows:

\[v_{-2} = \left( \pi, [1 \ 0], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right); \quad v_{-1} = \left( \frac{\pi}{2}, [0 \ 1], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right); \quad v_0 = \left( 4, [0 \ 0], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right).\]

Hence, problem (A.10) is now

\[v_1 = v_{-2} v_{-1} + \sin v_{-1} + v_0.\quad (A.11)\]

Therefore, the value and derivatives of problem (A.10) are then obtained by evaluating equation (A.11) based on the differentiation arithmetic (i.e. the AD approach).

\[v_1 = \left( \pi, [1 \ 0], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right) \left( \frac{\pi}{2}, [0 \ 1], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right) + \left( \sin \pi, [\cos \pi \ 0], \begin{bmatrix} -\sin \pi & 0 \\ 0 & 0 \end{bmatrix} \right) + \left( 4, [0 \ 0], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right).\]
On further simplifications:

\[ v_1 = \left( \frac{\pi^2}{2}, \left[ \frac{\pi}{2} \right], \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right) + \left( 0, [-1 \ 0], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right) + \left( 4, [0 \ 0], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right) \]
\[ = \left( \frac{\pi^2 + 8}{2}, \left[ \frac{\pi^2 - 2}{2} \right], \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right) \]
\[ = (f(x_k), f'(x_k), f''(x_k)) , \]

which is similar to the solution obtained via the traditional symbolic method above. Now, from the \( v_1 \), which is an AD object, one can extract the function value, derivative and Hessian as follows:

Function Value = \( v_1.funcValue = \frac{\pi^2 + 8}{2} \),

Function Derivative = \( v_1.funcDerivative = \left[ \frac{\pi^2 - 2}{2} \pi \right], \) and

Function Hessian = \( v_1.funcHessian = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \).

The elegance of this approach is in its suitability for algorithmic computation in computer. Notice that the final solution yields the exact results for the function value, gradient and the Hessian. A key advantage of the vectorised forward AD method is twofold; it is both accurate and computationally inexpensive.
Appendix B

Benchmark Test Case Studies

Table B.1 outlines the formulations, domain specifications and the respective universal tags for the global optimization benchmarks used across the various chapters of this thesis.

### Table B.1: Global Benchmark (Basic) Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Benchmark Function</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimodal</td>
<td>$f_0(x) = x_1 e^{-(x_1^2 + x_2^2)}$</td>
<td>$[-2.0, 2.0]$</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>$f_1(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i))$; $n = 100$.</td>
<td>$[-5.12, 5.12]$</td>
</tr>
<tr>
<td>Schwefel</td>
<td>$f_2(x) = \sum_{i=1}^{n} x_i \sin\left(\sqrt{</td>
<td>x_i</td>
</tr>
<tr>
<td>Easom</td>
<td>$f_3(x) = \cos(x_1) \cos(x_2) e^{-(x_1-\pi)^2-(x_2-\pi)^2}$</td>
<td>$[-100, 100]$</td>
</tr>
<tr>
<td>Sphere</td>
<td>$f_4(x) = \sum_{i=1}^{n} x_i^2$; $n = 2$.</td>
<td>$[-100, 100]$</td>
</tr>
<tr>
<td>Weierstrass</td>
<td>$f_5(x) = \sum_{i=1}^{n} \left( \sum_{k=0}^{k_{\text{max}}} a^k \cos(2\pi b^k(x_i + 0.5)) \right)$</td>
<td>$[-0.5, 0.5]$</td>
</tr>
<tr>
<td></td>
<td>$-n \sum_{k=0}^{k_{\text{max}}} \left[ a^k \cos(\pi b^k) \right]$; $a = 0.5$, $b = 3$, $k_{\text{max}} = 20$, $n = 2$.</td>
<td></td>
</tr>
<tr>
<td>Griewank</td>
<td>$f_6(x) = \frac{1}{100} \sum_{i=1}^{n} (x_i)^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{7}} \right) + 1$; $n = 2$.</td>
<td>$[-100, 100]$</td>
</tr>
<tr>
<td>Ackley</td>
<td>$f_7(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + c$; $n = 2$.</td>
<td>$[-32, 32]$</td>
</tr>
</tbody>
</table>
Appendix C

Result Summary

This presents the complete data recorded during the evaluation process of the Dual-pool EC model proposed in Chapter 5. Table C.1 shows the results of comparing the proposed method against the standard EC algorithm.
Table C.1: Computational cost in terms of function evaluations required by the Dual-pool and Standard EC algorithms to converge to a 0.1% accuracy level of the global optimal solution for the six different global optimization benchmarks. The table shows sensitivities of the two algorithms to varying population sizes. All results are averages of 100 independent runs.

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Traditional Benchmark Test Problems</th>
<th>Modified Benchmark Test Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rastrigin</td>
<td>Schwefel</td>
</tr>
<tr>
<td>20</td>
<td>1.86e3</td>
<td><strong>930.8</strong></td>
</tr>
<tr>
<td>50</td>
<td>2.07e3</td>
<td><strong>1.07e3</strong></td>
</tr>
<tr>
<td>100</td>
<td><strong>1.54e3</strong></td>
<td>1.81e3</td>
</tr>
<tr>
<td>200</td>
<td><strong>2.47e3</strong></td>
<td>2.98e3</td>
</tr>
<tr>
<td>500</td>
<td>5.59e3</td>
<td>6.45e3</td>
</tr>
<tr>
<td>1000</td>
<td><strong>9.84e3</strong></td>
<td>1.10e4</td>
</tr>
<tr>
<td>Avg. Cost</td>
<td><strong>3.90e3</strong></td>
<td>4.04e3</td>
</tr>
</tbody>
</table>

Notation: DP-EC = Dual-Pool EC algorithm, Std-EC = Standard EC algorithm, Avg. Cost = Average computational cost in terms of number of function evaluations. The **bold face** items indicate where an algorithm outperforms its counterpart.
Appendix D

Evaluation Results for the Dual-Pool HESA Model

For the series of experiments on the proposed Dual-pool HESA model presented in Chapter 7, Table D.1 summarises the complete results obtained for its sensitivity analysis to various pool sizes on the different categories of the dynamic changes ($T_1$ to $T_6$); and Table D.2 presents the results for its absolute error characteristics. All results are averages of 20 independent runs.

Table D.1: The offline/online performances ($r_{\text{offline}}/r_{\text{online}}$) of the Dual-pool HESA model under varying pool sizes on the six different types of dynamic changes.

<table>
<thead>
<tr>
<th>Pool Sizes</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>91.62/83.18</td>
<td>86.64/76.12</td>
<td>80.91/68.90</td>
<td>87.01/76.58</td>
<td>68.38/51.13</td>
<td>56.81/41.80</td>
</tr>
<tr>
<td>50</td>
<td>99.03/96.61</td>
<td>93.27/88.35</td>
<td>88.17/80.33</td>
<td>94.81/89.38</td>
<td>92.80/84.36</td>
<td>75.22/64.41</td>
</tr>
<tr>
<td>100</td>
<td>97.95/94.10</td>
<td>93.17/87.75</td>
<td>84.32/75.06</td>
<td>94.44/88.40</td>
<td>90.50/81.68</td>
<td>73.52/62.18</td>
</tr>
<tr>
<td>500</td>
<td>95.32/85.83</td>
<td>91.33/81.36</td>
<td>85.77/74.24</td>
<td>93.17/82.55</td>
<td>69.26/55.71</td>
<td>70.50/57.11</td>
</tr>
<tr>
<td>1,000</td>
<td>92.53/78.17</td>
<td>90.05/76.14</td>
<td>85.76/71.77</td>
<td>92.01/76.53</td>
<td>61.09/46.27</td>
<td>65.54/51.00</td>
</tr>
</tbody>
</table>

1Detail descriptions of these dynamic change instances are presented in Table 7.3
### APPENDIX D. EVALUATION RESULTS FOR THE DUAL-POOL HESA

Table D.2: A statistical summary for the absolute error ($E_{last}$) encountered by the proposed dual-pool HESA model on optimization of the DRP benchmark under varying pool sizes.

<table>
<thead>
<tr>
<th>Pool Sizes</th>
<th>Absolute Error ($E_{last}$)</th>
<th>Dynamic Change Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_1$</td>
<td>$T_2$</td>
</tr>
<tr>
<td>10</td>
<td>Avg-min-$E_{last}$</td>
<td>0.0120</td>
</tr>
<tr>
<td></td>
<td>Avg-mean-$E_{last}$</td>
<td>7.8879</td>
</tr>
<tr>
<td></td>
<td>Avg-max-$E_{last}$</td>
<td>37.4360</td>
</tr>
<tr>
<td>50</td>
<td>Avg-min-$E_{last}$</td>
<td>0.0337</td>
</tr>
<tr>
<td></td>
<td>Avg-mean-$E_{last}$</td>
<td>0.9036</td>
</tr>
<tr>
<td></td>
<td>Avg-max-$E_{last}$</td>
<td>12.4980</td>
</tr>
<tr>
<td></td>
<td>STD-$E_{last}$</td>
<td>2.7542</td>
</tr>
<tr>
<td>100</td>
<td>Avg-min-$E_{last}$</td>
<td>0.0433</td>
</tr>
<tr>
<td></td>
<td>Avg-mean-$E_{last}$</td>
<td>1.9120</td>
</tr>
<tr>
<td></td>
<td>Avg-max-$E_{last}$</td>
<td>17.9332</td>
</tr>
<tr>
<td></td>
<td>STD-$E_{last}$</td>
<td>4.3733</td>
</tr>
<tr>
<td>500</td>
<td>Avg-min-$E_{last}$</td>
<td>0.0578</td>
</tr>
<tr>
<td></td>
<td>Avg-mean-$E_{last}$</td>
<td>4.2796</td>
</tr>
<tr>
<td></td>
<td>Avg-max-$E_{last}$</td>
<td>27.6999</td>
</tr>
<tr>
<td></td>
<td>STD-$E_{last}$</td>
<td>7.3423</td>
</tr>
<tr>
<td>1,000</td>
<td>Avg-min-$E_{last}$</td>
<td>0.1701</td>
</tr>
<tr>
<td></td>
<td>Avg-mean-$E_{last}$</td>
<td>6.9518</td>
</tr>
<tr>
<td></td>
<td>Avg-max-$E_{last}$</td>
<td>41.0491</td>
</tr>
<tr>
<td></td>
<td>STD-$E_{last}$</td>
<td>10.1599</td>
</tr>
</tbody>
</table>
Appendix E

Evaluation Results for the Adaptive HESA Model

This appendix presents (Table E.1) a complete summary of the evaluation results for the Adaptive HESA Algorithm introduced and evaluated in Section 7.6.

Table E.1: A statistical summary for the evaluation results for the Adaptive HESA model on the DRP benchmarks. Results include Relative performances and Absolute errors. All results are *averages* of 20 independent runs.

<table>
<thead>
<tr>
<th>Evaluation Parameters</th>
<th>Dynamic Change Types</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline ± std. error</td>
<td>$T_1$</td>
<td>98.72 ± 0.0015</td>
<td>96.14 ± 0.0027</td>
<td>90.44 ± 0.0011</td>
<td>96.03 ± 0.0011</td>
<td>95.00 ± 0.0016</td>
<td>90.03 ± 0.0033</td>
</tr>
<tr>
<td>Online ± std. error</td>
<td>$T_2$</td>
<td>90.94 ± 0.0031</td>
<td>87.59 ± 0.0039</td>
<td>79.54 ± 0.0016</td>
<td>86.23 ± 0.0016</td>
<td>84.44 ± 0.0023</td>
<td>78.08 ± 0.0048</td>
</tr>
<tr>
<td>Avg-min-$E_{last}$</td>
<td>$T_3$</td>
<td>0.1610</td>
<td>0.1025</td>
<td>0.0976</td>
<td>0.2040</td>
<td>0.4657</td>
<td>0.4993</td>
</tr>
<tr>
<td>Avg-mean-$E_{last}$</td>
<td>$T_4$</td>
<td>1.1626</td>
<td>3.7883</td>
<td>8.1382</td>
<td>3.6357</td>
<td>4.9212</td>
<td>9.7917</td>
</tr>
<tr>
<td>Avg-max-$E_{last}$</td>
<td>$T_5$</td>
<td>11.0640</td>
<td>39.6530</td>
<td>44.2541</td>
<td>18.2540</td>
<td>36.0650</td>
<td>64.1480</td>
</tr>
<tr>
<td>STD-$E_{last}$</td>
<td>$T_6$</td>
<td>2.2756</td>
<td>8.6548</td>
<td>11.9966</td>
<td>4.3681</td>
<td>7.5803</td>
<td>14.7220</td>
</tr>
</tbody>
</table>
Bibliography


Bibliography


function optimization. In Proceedings of the Second International Conference on Genetic
Algorithms on Genetic algorithms and their application. L. Erlbaum Associates Inc.

gorithm with dominance and diploidy. In Proceedings of the Second International Conference
on Genetic Algorithms on Genetic algorithms and their application. L. Erlbaum Associates
Inc.

Gomes, C. and Selman, B. (1997). Algorithm portfolio design: Theory vs. practice. In Geiger,
D. and Shenoy, P., editors, Proceedings of the 13th Conference on Uncertainty in Artificial


Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. Systems,

Solving from Nature 2, pages 137–144.

of Algorithmic Differentiation. SIAM - Society for Industrial and Applied Mathematics,

Guo, P., Wang, X., and Han, Y. (2010). The enhanced genetic algorithms for the optimization
design. In 3rd International Conference on Biomedical Engineering and Informatics (BMEI),

Hancock, P. J. B. (1994). An empirical comparison of selection methods in evolutionary al-
gorithms. In Selected Papers from AISB Workshop on Evolutionary Computing. Springer-
Verlag.


Functions, volume 3242 of Lecture Notes in Computer Science, pages 282–291. Springer
Berlin / Heidelberg.

mutation distributions in evolution strategies: The generating set adaptation. In Proceedings
of the 6th International Conference on Genetic Algorithms. Morgan Kaufmann Publishers
Inc.

on Evolutionary Computation, pages 523–528.

He, J. and Yao, X. (2004). A study of drift analysis for estimating computation time of
evolutionary algorithms. Natural Computing, Kluwer Academic Publishers, Netherlands,


