UNIVERSITY OF SOUTHAMPTON

Artificial Intelligence Technologies in Complex Engineering Design

Yew Soon Ong

王友顺

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Computational Engineering and Design Center
School of Engineering Sciences

SEPTEMBER, 2002

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

COMPUTATIONAL ENGINEERING AND DESIGN CENTER

SCHOOL OF ENGINEERING SCIENCES

Doctor of Philosophy

Artificial Intelligence Technologies in Complex Engineering Design

by *Yew Soon Ong*

Engineering design optimization is an emerging technology whose application both tends to shorten design-cycle time and finds new designs that are not only feasible, but also nearer to optimum, based on specified design criteria. Its gain in attention in the field of complex designs is fuelled by advancing computing power now allowing increasingly accurate analysis codes to be deployed. Unfortunately, the optimization of complex engineering design problems remains a difficult task, due to the complexity of the cost surfaces and the human expertise necessary in order to achieve high quality results. This research is concerned with the effective use of past experiences and chronicled data from previous designs to mitigate some of the limitations of present engineering design optimization process. In particular, the present work leverages well established artificial intelligence technologies and extends recent theoretical and empirical advances, particularly in machine learning, adaptive hybrid evolutionary computation, surrogate modeling, radial basis functions and transductive inference, to mitigate the issues of i) choice of optimization methods and ii) dealing with expensive design problems. The resulting approaches are studied using commonly employed benchmark functions. Further demonstrations on realistic aerodynamic aircraft and ship design problems reveal that the proposed techniques not only generate robust design performance, they can also greatly decrease the cost of design space search and arrive at better designs as compared to conventional approaches.

Contents

Abstract ii

List of Figures vii

List of Tables xi

Acknowledgments xiii

Acronyms xiv

CHAPTER 1 INTRODUCTION 1

§1.1 Complex Engineering Design Optimization 1

§1.2 Optimization Techniques 2

§1.3 Limitations of Present Complex Engineering Design Optimization 3

§1.3.1 Choice of Optimization Methods 4

§1.3.2 Computationally Expensive Design Problems 4

§1.4 Research Focus and Scope 5

§1.5 Layout of the Thesis 7

CHAPTER 2 DOMAIN OPTIMIZATION METHOD ADVISOR 8

§2.1 Introduction 9

§2.2 Historical Design Data Sources 10

§2.3 Domain Knowledge Construction 12

§2.3.1 Strategy I - Best Performing Search Strategy 12

§2.3.2 Strategy II - Artificial Intelligence Selected Strategy 13

§2.4 Adaptability 18

§2.5 Domain Optimization Method Advisor Applied to Real World Problem Domains 19

§2.5.1 Chronicled Design Optimization Data 19

§2.5.2 Domain Knowledge Construction - Best Performing Search Strategy 20

§2.5.3 Domain Knowledge Construction - Artificial Intelligence Selected Strategy 25

§2.5.4 Performance Of Domain Optimization Method Advisor 30

§2.6 Conclusion 32

CHAPTER 3 META-LAMARCKIAN LEARNING 33

§3.1 Introduction 34

§3.2 Meta-Lamarckian Learning in GA-LS Hybrids 36

§3.2.1 Basic Meta-Lamarckian Learning Scheme, GA-B 37

§3.2.2 Adaptive Meta-Lamarckian Learning 37

§3.2.3 A Heuristic Approach, Sub-problem Decomposition - GA-S1 . . . 39

§3.2.4 A Stochastic Approach, Biased Roulette Wheel - GA-S2 41

§3.3 Numerical Studies on Benchmark Problems 41

§3.3.1 Effects of Local Method Choice on Search Performances 44

§3.3.2 Results of Meta-Lamarckian Learning 48

§3.3.3 Other Issues Considered 54

§3.4 Aerodynamic Aircraft Wing Design 58

§3.5 Conclusion 60

CHAPTER 4 COMPUTATIONALLY EXPENSIVE ENGINEERING DE-

SIGN PROBLEMS **61**

§4.1 Introduction 62

§4.2 Surrogate Modeling 67

§4.3 Present Framework 70

§4.4 Numerical Studies on Benchmark Problems 75

§4.5 Aerodynamic Aircraft Wing Design 81

§4.6 Conclusion 85

CHAPTER 5 CONCLUDING REMARKS **86**

§5.1 Research Contributions 86

§5.1.1 Domain Optimization Method Advisor 87

§5.1.2 Meta-Lamarckian Learning 87

§5.1.3 Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling 88

§5.2 Future Work 88

§5.2.1 Domain Optimization Method Advisor 89

§5.2.2 Meta-Lamarckian Learning 89

§5.2.3 Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling 90

Bibliography **92**

APPENDIX A DESIGN EXPLORATION SYSTEM **103**

APPENDIX B BENCHMARK TEST PROBLEMS **105**

§B.1 Sphere Problem 106

§B.2 Griewank Problem 107

§B.3 Rastrigin Problem 108

§B.4 Bump Problem 109

APPENDIX C REALISTIC INDUSTRIAL ENGINEERING DESIGN PROBLEMS 110

§C.1 Aircraft Wing Design Problem 110

§C.2 Ship Hull Form Design Problem 114

List of Figures

2.1 Relative Final Design Accuracy and Efficiency of OPTIONS DES Search Routines on Aircraft Design Domain. (Results presented are based on $\frac{2}{3}$ sampled wing design problems. Performances normalized to unity: larger values indicates faster searches or better designs.) 22

2.2 Relative Final Design Accuracy and Efficiency of OPTIONS DES Search Routines on Ship Design Domain. (Results presented are based on $\frac{2}{3}$ sampled ship hull-form design problems. Performances normalized to unity: larger values indicates faster searches or better designs.) 23

2.3 Relative Robustness of OPTIONS DES Search Routines on the Aircraft Design Domain. (Results presented are based on $\frac{2}{3}$ sampled wing design problems.) 24

2.4 Relative Robustness of OPTIONS DES Search Routines on the Ship Design Domain. (Results presented are based on $\frac{2}{3}$ sampled ship hull-form design problems.) 25

2.5 Knowledge on the speed performance of search routines on the Aircraft Wing Design Domain in the form of rules. These are derived by the C4.5 inductive algorithm using the Best-Speed datasets obtained from offiine simulations. 28

2.6 Estimated Mean Classification Accuracies and Standard Deviation of the Machine Learning techniques considered. 29

3.1 Pictorial Illustration and Pseudo Code of Sub-problem Decomposition Strategy, GA-S1. 40

3.2 Pictorial Illustration and Pseudo Code of Biased Roulette Wheel Strategy, GA-S2. 42

3.3 Proposed General Algorithm for Hybrid GA-LS With Meta-Lamarckian Learning. 43

3.4 Average convergence trends for minimizing 30D Sphere function using GA and various GA-LS hybrids with Lamarckian Learning. Note that the search traces terminate at the global optimum of the Sphere function at 0.0 or $\ln(0.0) = -\infty$. 46

3.5 Average convergence trends for minimizing 10D Griewank function using GA and various GA-LS hybrids with Lamarckian Learning. 47

3.6 Average convergence trends for maximizing 20D Bump function using GA and various GA-LS hybrids with Lamarckian Learning. 47

3.7 Average convergence trends for minimizing 30D Sphere function using strategies GA-B, GA-S1 and GA-S2 compared with standard GA and GA-AV. Also shown are the search traces for the GA-PO and GA-FL fixed hybrids. 48

3.8 Average convergence trends for minimizing 10D Griewank function using strategies GA-B, GA-S1 and GA-S2 compared with standard GA and GA-AV. Also shown are the search traces for the GA-PO and GA-DS fixed hybrids. 49

3.9 Average convergence trends for maximizing 20D Bump function using strategies GA-B, GA-S1 and GA-S2 compared with standard GA and GA-AV. Also shown are the search traces for the GA-NM and GA-DS fixed hybrids. 50

3.10 Average convergence trends for minimizing 10D Griewank function using GA-S2A and GA-S2B with the incorporation of Human Expert Knowledge compared with conventional GA and GA-S2. Also shown are the search traces GA-PO and GA-DS hybrids. 57

3.11 Design of a Transonic Civil Transport Aircraft Wing using Adaptive Meta-Lamarckian Learning hybrid GA-S2. GA-S2C differs from GA-S2 due to the incorporation of human knowledge (see text). Also shown are the search traces for GA, GA-BC, GA-FL, GA-DS, GA-CO, GA-AV and GA-B. 59

4.1 Proposed algorithm for integrating local surrogate models with hybrid EAs for optimization of computationally expensive problems. 71

4.2 Averaged convergence trends for m=100 and various values of k_{max} (3, 5 and 8) compared with standard GA and global surrogate modeling framework for the 20D Rastrigin function. 77

4.3 Averaged convergence trends for k_{max} = 3 and various values of m (100, 150 and 200) compared with standard GA for the 20D Rastrigin function. 78

4.4 Averaged convergence trends for k_{max} = 3 and various values of m (100, 150 and 200) compared with standard GA for the 20D Bump function. . 79

4.5 Algorithm based on global surrogate models proposed by Ratle [82]. . . . 80

4.6 Summary of minimum drag values using the VSAERO code and surrogate models. Results are average of 3 runs. 83

4.7 Comparison of best convergence trends as a function of wall-clock time for the Aerodynamic Wing Design Problem using the VSAERO code and surrogate models. 84

B.1 2-D Sphere function. 106

B.2 1-D slice of Griewank function. 107

B.3 2-D Rastrigin function. 108

B.4 2-D Bump Constrained function. 109

C.1 Geometric view of streamlines over a transport aircraft. 112

C.2 Wing Planform Geometry. 112

C.3 Geometric view of a frigate hull-form and its optimized version. 117

List of Tables

2.1 List of Machine Learning Techniques investigated. 17

2.2 List of search routines recommended in the design domains studied. Results presented are based on using the Best Performing Search Strategy of the advisor. 21

2.3 A portion of the processed Best-Speed dataset with search routine rankings and Best-Speed search technique classes/labels. (Refer to the Appendix A for details of the abbreviated search routines.) 27

2.4 Relative performance of the Domain Optimization Method Advisor in comparison with the 'Common Designers Strategies' CDS 1-5. Results are obtained for the various best schemes proposed, taken over the $\frac{1}{3}$ unseen design problems from the respective domains. Values closer to unity more nearly achieve the desired results. 31

3.1 Classes of General Constrained, Unimodal and Multimodal Benchmark test functions. *1: Epistasis, *2:Multimodality, *3:Discontinuity, *4:Constrained. 46

3.2 (a) Effects of changes in k on Search Quality for Sub-problem Decomposition Strategy, GA-S1. (b) Effects of changes in g on Search Quality for Sub-problem Decomposition Strategy, GA-S1. (c) Mean and Standard Deviation of Biased Roulette Wheel Strategy, GA- S2 on the benchmark problems. 52

3.3 Effects of Changes in LS Pool Size on the minimization of the Griewank function, mean function fitnesses are multiplied by 10^{04}. When the pool size is 25, the other methods used were taken from the OPTIONS DES. . 55

4.1 Radial Basis Kernels. 68

4.2 Summary of minimum drag values using the TADPOLE code and surrogate models. 82

A.1 The 31 optimization search routines employed from the OPTIONS DES. 104

C.1 Optimization conditions for wing design parameters, constraints and respective limits. 113

C.2 Ship Hull-form Parameter Definitions. 115

C.3 Optimization conditions for Hull-form design parameters, constraints and respective limits. 116

Acknowledgments

I sincerely thank my advisor Professor Andrew John Keane for his inspiring constant source of support, guidance and encouragement over the course of my research in the University of Southampton. I would also like to express my gratitude to Dr. Prasanth B. Nair for his encouragements and guidance in numerous ways, fulfilling the role of a co-advisor professionally.

A special thanks to Arindam Choudhury for the useful discussions on Statistical and Machine Learning and principally his earnest friendship. Further, I would like to thank all my colleagues at the Computational Engineering and Design Center (CEDC). In particular, I am grateful to Wen-Bin Song, Mamadou Bah, Nichola Petruzzelli, Cheng Hock Tan, Whui Liang Ng and András Sóbester for their unhesistating help and long-standing friendship.

I would like to gratefully acknowledge the financial support received from the Faculty of Engineering and Applied Sciences in the form of studentship. Besides, I am also grateful to the support displayed by BAE Systems and Rolls Royce both financially and technically, thus making the University Technology Partnership for Design a great success in CEDC.

Most importantly, special thanks to my parents, Chai Thuan Ong and Geok Mui Phua, and my lovely wife, Lee Ting Aw, for their invaluable support and encouragement, even though I could never thank them enough.

Acronyms

The commonly used acronyms in the dissertation are listed below. The notations for the optimization routines are not included here in this section, rather it is presented separately in Appendix A of the dissertation.

AISS	Artificial Intelligence Selected Strategy
BD	Best-Design Knowledge Model
BO	Balanced-Overall Knowledge Model
BPSS	Best Performing Search Strategy
BS	Best-Speed Knowledge Model
CDS	Common Designer Strategies
CFD	Computational Fluid Dynamic
DACE	Design and Analysis of Computer Experiments
DES	Design Exploration Systems
DOMA	Domain Optimization Method Advisor
DPD	Domain Problem Descriptor
EA	Evolutionary Algorithm
ES	Evolutionary Search
FE	Finite Element
FFSQP	Fortran Feasible Sequential Quadratic Programming

GA	Genetic Algorithm
GA-AV	Estimated performance one expect to achieve with selection of a fixed LS in a hybrid GA is made randomly
GA-B	Genetic Algorithm-Basic Meta-Lamarckian Learning Scheme
GA-LS	Genetic Algorithm-Local Search
GA-S1	Genetic Algorithm-Adaptive Meta-Lamarckian Learning Scheme 1, Sub-problem Decomposition
GA-S2	Genetic Algorithm-Adaptive Meta-Lamarckian Learning Scheme 2, Biased Roulette Wheel
GA-S2A	GA-S2 but with DS search routine biased with twice the chances of selection.
GA-S2B	GA-S2 but with use of six local methods (PO, NM, CO, BC, PD and DS) in the local method pool.
GA-S2C	GA-S2 but with CO search routine biased with twice the chances of selection.
LS	Local Search
ODE	Ordinary Differential Equations
PDE	Partial Differential Equations
SQP	Sequential Quadratic Programming

Chapter 1

Introduction

§1.1 Complex Engineering Design Optimization

PRESENTLY, most complex engineering design exploration is carried out manually. The design engineer uses computer aided design tools to make a modification to the design and evaluates this by numerical simulation. He then enters a design-evaluate-redesign process and stops when he thinks that the design is adequate based on his experience and knowledge of past designs. A numerical simulation may thus be used by hand to provide a design exploration process. Design Exploration Systems (DES) help in this process by attempting to automatically establish critical design parameters.

Today, several powerful DES such as OPTIONS [1] and iSight [2] have become available, most of which have the common characteristic of containing multiple sophisticated optimization routines for design-space search. The heart of a good DES is usually a design optimization facility, which is basically a design-space search tool that is employed to automatically find good solutions to some problem (e.g., by finding the minimum of a function) by generating a collection of potential solutions to the problem and then manipulating them. In this way, complex engineering design optimization helps reduce the cycle time of the design-evaluate-redesign iteration loops and finds better designs by computerizing parts of the iterative process.

In this dissertation, we shall limit our view of the engineering design process to continuous parametric design. In particular, we consider the general nonlinear programming problem of the form:

$$\begin{aligned} \textbf{Minimize:} \quad & f(\mathbf{x}) \\ \textbf{Subject to:} \quad & g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, p \qquad (1.1) \\ & h_i(\mathbf{x}) = 0, j = 1, 2, \ldots, l \qquad (1.2) \\ & \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u \end{aligned}$$

where $\mathbf{x} \in R^d$ is the vector of design variables representing the parametric description of the artifact being designed, and $\mathbf{x}_l$ and $\mathbf{x}_u$ are vectors of lower and upper bounds, respectively. $f(\mathbf{x})$ present a numerical property of the artifact that can be minimized and is often referred as the objective or fitness function. $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$ are the inequality constraint functions and equality constraint functions, respectively. The constraints acts as a form of quality control, ensuring that the design is realizable.

§1.2 Optimization Techniques

Optimization is a mature technology that has been studied extensively by researchers over the last half century. Over the years, optimization methods have evolved considerably and many algorithms and implementations are now available and used in the engineering optimization community. They can generally be classified into three broad categories: conventional numerical optimization methods (mostly gradient based), stochastic optimization methods and hybrid methods. Typical conventional numerical methods are steepest-descent methods, conjugate-gradient, quadratic programming, direct search methods and linear approximation methods [3, 4, 5]. These methods have the known advantage of their efficiency; however, they are very sensitive to starting point selection and are more likely to stop at non-global optima than modern stochastic algorithms. Several research efforts on these conventional numerical optimization methods have been applied with much success to some complex engineering design optimization problems such as aircraft design [6, 7, 8].

Stochastic techniques on the other hand produce new design points that do not use information about the local slope of the objective function and thus are not prone to stalling in false optima. They do tend to require more analysis effort, however. Among the modern stochastic optimizers are Genetic Algorithms; Simulated Annealing; Evolutionary Programming and Evolution Strategies [9, 10, 11, 12]. Modern stochastic techniques have also enjoyed success in several complex engineering design domains including aerodynamic design, communication network design, control system design, architectural and civil engineering design, and VLSI design [11, 12, 13, 14]. A view of modern stochastic techniques from the standpoint of an engineer, principally evolutionary computation in control system engineering and design can be found in [13].

The third category of optimization methods is hybrid methods, which are formed by combining different optimization methods or sometimes with the use of artificial intelligence. InterGEN [15] is a hybrid method that combined a Genetic Algorithm with a conventional numerical optimizer, and uses a rule based expert system to decide when to switch between the two. Gage [16] also combined Genetic Algorithm with Sequential Quadratic Programming for the design of aircraft wings. In addition, there exists a breed of hybrid Evolutionary Algorithm - Local Searches that implicitly involve the use of learning procedures within the population. Such hybrid methods are commonly referred to as Memetic Algorithms. [17, 18, 19].

§1.3 Limitations of Present Complex Engineering Design Optimization

Even though optimization has been studied and used in the engineering design community for many years, it has only been heavily used relatively recently for complex designs [20]. This take up is now happening because advances in computing technologies allow increasingly accurate analysis codes to be deployed in this way, see for example the work reported by Jameson in a special issue of Journal of Aircraft dedicated to optimization [21].

Unfortunately, the optimization of complex engineering design problems is a difficult task due to the complexity of the cost surfaces and the human expertise that are necessary in order to achieve high quality results. Two aspects that effect the performance of design searches significantly are: A) "Choice of Optimization Methods" and B) "Computationally Expensive Design Problems". The following two subsections set out these issues.

§1.3.1 Choice of Optimization Methods

In practice, design search tools often provide the advantage of multiple options of search routines for design engineers to choose from during a design search problem. This benefit, however, assumes the availability of design engineers with sufficient optimization experience on a particular problem domain to be able to make good method or routine choices. Generally, present design search tools tend to provide little help to ensure robust performance in the design search process. Intuitively, design engineers tend to rely on qualitative rules that are derived from their past design experiences of optimization methods and engineering domains to aid them in making decisions for new designs. The effect of this is that it tends to lead to inconsistent designs being produced at high cost due to the limited experiences of novice design engineers. Moreover, design engineers often stick to a very limited range of optimization techniques regardless of the design problem involved or the sophistication of the optimization suite. The lack of suitable support may also have a detrimental effect on design innovation by placing too much dependence on a single individual's past designs, which usually contain biases.

§1.3.2 Computationally Expensive Design Problems

Fuelled by advancing computing power, present complex engineering designs often involve the deployment of analysis codes or simulation models that are computationally very expensive. As a result, the overwhelming part of the total run time in design optimization is usually taken up by the evaluations of the analysis codes. A motivating example is aerodynamic design, where one function evaluation involving the solution of the Navier-Stokes equations may take many hours of supercomputer time. Such computationally expensive problems also arise in other areas such as electromagnetics, structural and photonic designs. The computational costs associated with the use of these high-fidelity simulation models often pose a serious impediment to the successful application of complex engineering design optimization.

§1.4 Research Focus and Scope

The term 'Artificial Intelligence (AI)' was coined by John McCarthy in 1956 [22]. Known as the father of artificial intelligence, John McCarthy defines 'Intelligence' as the computational part of the ability to achieve goals in the world. However, AI can mean many things to many people. Much confusion arises because the word 'intelligence' is ill-defined. The phrase is so broad that researchers in the AI community have found it useful to divide it into two classes: strong AI and weak AI. Strong AI makes the bold claim that computers can be made to think on a level (at least) equal to humans. Weak AI simply states that some "thinking-like" features can be added to computers to make them more useful tools and this has already started to happen (witness expert systems, unmanned vehicles and speech recognition software).

The scope of this research is on producing scientific assistants for complex engineering design, i.e., a form of "weak AI" research. In particular, the focus of this research is on leveraging artificial intelligence technologies for improving complex engineering design search performances via the effective use of chronicled design optimization data. Throughout the engineering design process, designer engineers often produce a great deal of data that is ignored or otherwise discarded. These historical design data often contain useful knowledge of the domain or process which may be extracted to aid future design optimization and search activities. The aim is to encourage computational discovery of new useful knowledge and design process automation. By doing so, one may attempt to mitigate the two limitations of present complex engineering design optimization process identified previously in section 1.3.

The first proposal in this dissertation seeks the use of pattern classification technologies for knowledge discovery from chronicled design optimization data. This is collected from previous search processes in the domain of interest, and via offline simulations. It aspires to reduce the influence of choice of optimization methods on engineering design search performance. A second proposal then presents an approach for reducing the influence of method choice in hybrid evolutionary design search performance. The approach removes the need to conduct offline simulations. In contrast to the first proposal, the latter uses adaptive strategies and makes use of optimization data chronicled during the lifetime of the search process.

Furthermore, to enable the successful application of evolutionary optimization techniques for solving computationally expensive design problems with general constraints, a new parallel metamodeling framework that leverages surrogate models constructed from chronicled design optimization data is proposed in this dissertation.

§1.5 Layout of the Thesis

This thesis is organized as follows:

Chapter 2 introduces a Domain Optimization Method Advisor [23, 24, 25] for assisting design engineers on the choice of optimization routines in design search. A knowledge-based approach proposed for reducing the influence of choice of optimization methods on engineering design search performance, when searching familiar design domains, is presented.

Chapter 3 investigates Meta-Lamarckian learning in Hybrid Genetic Algorithms [26] as an approach for reducing the influence of choice of local optimization methods on hybrid evolutionary design optimization performance. The proposed adaptive strategies facilitate a cooperative and competitive paradigm between different local optimization routines, working together to accomplish the shared goal.

Chapter 4 discusses metamodeling frameworks for tackling the computationally very expensive analysis codes commonly found in complex engineering design searches. Further, a general metamodeling framework proposed for evolutionary search of constrained or unconstrained problems that are computationally expensive is proposed [27, 28]. The framework employs well-established notions of hybrid evolutionary-gradient optimization, radial basis networks, transductive inference and trust-region methods.

Chapter 5 summarizes the contributions made in the present program of research, outlines directions for future research and provides conclusions.

Results from experimental studies on benchmark test problems and demonstrations on real world engineering design problems are presented in the respective Chapters.

Chapter 2

Domain Optimization Method Advisor

THIS Chapter describes a Domain Optimization Method Advisor (DOMA) [23, 24], a knowledge-based approach for reducing the influence of optimization routine choice on engineering design search performance. It is designed to work when searching familiar design domains. The approach makes use of historical design data collected from past optimization search processes on the same domain and from offline simulations. Knowledge on the merits and limitations of the available optimization search routines on these domains are derived from the chronicled design data to help facilitate intelligent search routine selection in future design searches. The approach employs two strategies: the Best Performing Search Strategy (BPSS) and the Artificial Intelligence Selected Strategy (AISS), to make recommendations of optimization routines for new design problem searches. These strategies are further combined to complement each other on predicting the most appropriate choice of optimization routines in future design searches. Demonstrations on two real world engineering problems, aircraft wing and ship hull-form design, show that the proposed advisor helps improve design optimization in terms of speed and solution accuracy. At the same time, it reduces reliance on human experts by ensuring that design engineers need only minimal knowledge of optimization routines.

§2.1 Introduction

Companies usually have limited diversity of trade and thus work-scope. For example, Airbus focuses mainly on aircraft design; Rolls Royce on engine design; while a ship building company may place its focus on the design of major ship elements. Depending on the complexity of a domain, some search routines that may have proven to be useful in one domain might not work so well in others. The same reasoning applies to individual design problems of similar domains. Therefore unless one knows which search routines in a DES most suits the design problem in hand, the optimization may not perform properly. It is thus necessary to support design engineers with search advisors that can assist them in their designing activities. Here we present an approach for supporting design engineers with an optimizer advisor that contains knowledge on the merits and limitations of different search routines on the design domains under study. The Domain Optimization Method Advisor helps reduce the influence of choice of optimization routines on engineering design search performance, when searching familiar design domains.

Much research on the choice of optimization methods has been focussed on attempts to identify those methods that will work well on rather limited ranges of design test problems [29, 30]. Additionally, Sandgren [6] applied 35 nonlinear optimization algorithms to 30 engineering design optimization problems and compared their performance. In complex engineering design, Bramlette and Cusic [7] compared the application and performance of different methods, including gradient-based numerical optimization, to the design and manufacture of aeronautical systems. The use of modern stochastic optimization methods on some artificial test problems was also explored by Keane [31]. On the whole, the general conclusion obtained from all these studies is that no single optimization search technique always performs well on all problems - a result that is sometimes referred to as the "no free lunch theorem" [32].

Few studies in the literature have directly addressed the issue of choice of optimization search routines in design. Even though interest in the problem was recently re-ignited by Fukunaga [33], very little progress has since been made. On the other hand, it is possible to relate the challenges considered here to those in the problem solving community. For example, the mapping of scientific software to various classes of problems that are represented by partial or ordinary differential equations (PDEs, ODEs) [35, 34, 36, 37]. Based on various characteristics of DE models, recommendations are made from the choices of numerous scientific software approaches available. In contrast, engineering design optimization cannot be easily set in these terms, as the design problems presented in the form of fitness functions are not often formulated as DEs. Even so, there is some useful experience that can be drawn on to reduce the influence of optimization routine choice on engineering design search performance.

This Chapter is organized in the following manner. Section 2.2 presents the forms of design data that have been chronicled here for performing knowledge discovery. Section 2.3 describes the strategies proposed for discovering knowledge on the merits and limitations of numerous search routines. Section 2.4 briefly discusses the need for adaptable knowledge bases, while section 2.5 demonstrates the proposed advisor in real world complex engineering designs. Finally, section 2.6 provides the main conclusions for this part of the work.

§2.2 Historical Design Data Sources

Throughout the engineering design process, design engineers often produce a great deal of data as a result of their design-evaluate-redesign actions. It is often possible to acquire knowledge about the merits and limitations of optimization search routines on a design domain from the mass of data that results. The data that may prove to be useful comprises the explored design space, objective fitness, expended evaluation count or time taken and forms of design space search violations. The latter consists of incomputable designs as well as bound and constraint violations. Incomputable designs occur possibly due to singularities within the objective function or geometrical inconsistencies. The gathered data may then be classified according to the key parameters that uniquely identified each design problem in the domain of interest. For example in the transonic civil aircraft wing design, these may include the cruise height, Mach number and fuel weight parameters. Note that here cruise height, Mach number and fuel weight are not design variables to be optimized but rather the static parameters of a design optimization problem which vary uniquely for different problems within the aircraft wing design domain. Such parameters will vary over some well-defined ranges and are here termed as Domain Problem Descriptors or DPDs. They set out the scope of interest of a design team.

In practice, simply relying on data chronicled from past optimization search processes is often insufficient to ensure the validity of any knowledge obtained, as the data is often incomplete. The other source of design data considered here is obtained via offline simulations conducted on sample problems from the domain under study. Here a design domain is sampled based on its DPDs and it is assumed that the DPDs vary over well-defined ranges identified by the design engineers. This sample set of design problems represents an approximate representation of the entire domain. To provide a good representation over the domain, design of experiments techniques such as Latin Hypercubes [38], Hammersley Sequence Sampling, Uniform Design or others may be used [39]. Offline simulations are then conducted over the set of sampled design problems by performing searches on them using the numerous search routines available in the DES or optimization suite. These simulation processes can be very time consuming if the underlying analyses used are of significant complexity or when the configurations of the search routines are taken into account. They do however provide the raw material that may be used to guide future search method selection.

Note also that even when a search routine has been selected, consideration must be given to control parameter settings. Consider an optimization routine with a control parameters, each with b possible values; there are a^b possible configurations to operate the routine. If it takes an average of n evaluations to arrive at a near-optimal design and t seconds to evaluate each of these candidate designs, then the expected time required to perform a simulation consisting of s sample problems and p search routines is $O(psnta^b)$. Given the enormous computational expense of the simulation, the proposed solution may prove to be intractable. Hence, in all the simulations conducted in this dissertation, the control parameters of the search routines are set up according to the DES's defaults, such that the computational expense at $O(psnt)$ becomes much more tractable when compared to the former. This adoption of a routine's default settings also signifies the analogue of design engineers relying on established settings in the initial stages of most design processes. Furthermore, the default settings hopefully represent the optimum operating conditions of the search routines recommended by its creator. Clearly in the future, a more sophisticated advisor might well also supply advice on the choice of search routine control parameters.

§2.3 Domain Knowledge Construction

The data gathered from past optimization search processes and offline simulations combine to form the total chronicled database available for use in the knowledge discovery process. Different modes of domain knowledge may be discovered from these data. Often the design engineer is trying to satisfy a number of conflicting desires and requirements when choosing an optimization routine. A common approach to such a predicament is to decide on some criteria that are considered the most important. An information equation may then be formulated to accommodate these criteria.

For instance, the performance quality q of a search routine on a design problem can be calculated as

$$q = \sum_{z=1}^{c} w_z \mu_z, \tag{2.1}$$

which is the weighted sum of all the criteria c, considered. w_z is the weight assigned to criteria z with quality of criteria z represented by μ_z in the equation. Appropriate weights are assigned accordingly to the importance of each criterion specified by the design engineer. Usually, non-zero fractional weights are used so that sum of all weights, $\sum_{z=1}^{m} w_z$, is equal to unity. Next, two strategies that make use of the domain knowledge acquired for predicting the best performing search routines are discussed.

§2.3.1 Strategy I - Best Performing Search Strategy

The first strategy considered here is the 'Best Performing Search' Strategy (BPSS), which attempts to recommend the optimization routine that is predicted to give the best search performance on the problem domain under study. It makes use of the information obtained via equation 2.1 to estimate the relative strengths of each search routine on the domain represented by the sampled design problems. In the present strategy, the best performing search routines (indexed by j^*) to be recommended for use on unseen design problems are determined as follows

$$j^* = \arg\max_j \sum_{i=1}^{s} q_i^j, \; j = 1, \dots, p \tag{2.2}$$

where each search routine denoted by r^j in the DES (indexed by $j = 1, \dots, p$) is run on s design sample problems (indexed $i = 1, \dots, s$). A Pareto front of optimization search routines r^{j^*} can be obtained using different values for the criterion weight factor w, and with the inclusion of various criteria in information equation 2.1. This Pareto front represents the set of search routines where each member of the set is optimal in some sense and is said to be non-dominated by all the others in the problem domain under study. Later in section 2.5.2, the strategy is further demonstrated on realistic problem domains in an attempt to provide a greater comprehension of the proposed BPS strategy and the motivations for including it in the advisor.

§2.3.2 Strategy II - Artificial Intelligence Selected Strategy

The second strategy proposed is the 'Artificial Intelligence Selected' Strategy (AISS). The main motivation towards using the strategy is based on the observation that a single search routine does not always emerge as the best method throughout the entire problem space of a design domain. This is in contrast to the BPSS, which generates domain knowledge models that recommend a single best search routine for a complete design domain. Here, we attempt to use artificial intelligence technologies to generate knowledge models that predict which search routine is best, given design problems of familiar domains and information about the new problem.

The approach commonly used by the problem solving community is generally based on "Learning from experience" techniques, where information extracted from past problem solving experiences is used to assist in solving future problems, especially those that are similar. Such an approach has also been used in some engineering design applications to improve the design process in various ways [15, 40, 41, 42]. A collection of "Learning from experience" approaches used in design applications can be found in the special issue of Engineering Applications of Artificial Intelligence Journal [43]. In general, "Learning from experience" techniques have been classified into two main categories by the artificial intelligence community. Expert systems, Case-Based reasoning, Explanation-based learning, Ontology based systems and others fall into the first group, which may be categorized as manual knowledge acquisition approaches. Many of the systems used in the problem solving community are instances of this category. The second category is automatic knowledge acquisition, and these usually employ machine learning techniques. Machine Learning is a recent approach to knowledge elicitation often referred to as "knowledge mining" or "knowledge discovery" [44]. Grounded on various artificial intelligence based techniques, the approach is automatic and acquires knowledge, extracts features or identifies patterns directly from examples or databases. Instances of the second category can be found in the work of Schwabacher, Wolberg, Reich and Rasheed [40, 45, 46, 47].

In the 'Artificial Intelligence Selected' Strategy presented here, machine learning is chosen to carry out the role of domain knowledge acquisition on data collected from past designs. It is particularly suitable in this work because it is able to automate the process of generalizing from historical design data on the applicability of search routines to different subsets of problems within a given domain. The choice of an automatic knowledge acquisition approach is also a consequence of the following factors:

- There are currently no standard models or techniques for predicting the most appropriate search routine in general design search activities. Here, the process is studied from a heuristic perspective with an automated knowledge capture approach for building such a model in the domain of interests. The knowledge is derived and then refined through the observation of new data.

- Knowledge on the applicability of a search routine to a particular domain or problem is normally based on the experience of human operators, which is difficult to extract, share and model. Moreover the capability of a given search algorithm differs even among multiple implementations of the same theory: this makes manual generalization from theory almost impossible.

- The main criticism of manual knowledge acquisition is the need for the availability of human domain experts who are willing to share their experiences and knowledge in a most unreserved manner. There is also the need for qualified knowledge engineers to perform the knowledge acquisition task, which is often difficult to model. Nonetheless, there is also a further need for the acquired domain theory to be relatively complete and consistent. These often pose a significant bottleneck for manual knowledge-based system development.

- Design engineers may well have some intuition and qualitative rules that give rise to biases (preferences for particular search techniques). These may result in non-optimum performance or even poor designs. It is also unlikely for one designer to be able to make decisions about all aspects of a design, which may possibly be simpler for a machine.

- A design related knowledge base has to be dynamic, so that results from new searches initiated by the design engineers generates new knowledge that can be updated into the knowledge base. The use of automatic knowledge acquisition enables this process to be accomplished effortlessly.

To be able to extract knowledge on the merits and limitations of optimization search routines in a design domain using machine learning, it is necessary to conduct further pre-processing of the data obtained in equation 2.1. This involves the conversion of the original data into table-like datasets, such that the sampled design problems are labeled and ranked according to the optimization search routine that performs the best. 'Best' here is taken to mean the search routine available in the DES (indexed $j = 1, \ldots, p$) emerging with the optimum value of performance quality q when run on i design sample problem (indexed $i = 1, \ldots, s$). Using the present AIS strategy, the best performing search routine (indexed by j^*) predicted for design problem i is found by

$$j^*{}_i = \arg\max_j \{q_i^j\}, \; j = 1, \ldots, p \tag{2.3}$$

The intention here is to construct a pattern classifier using pre-processed datasets. For this purpose, a class set $\mathbf{\Omega}$ is defined to include any search routine denoted by r^j that has been observed to have ever performed best on the sampled design problems, i.e., $r^j \in \mathbf{\Omega}$ for all $r^j = r^{j^*}$. The aim is to generalize or learn about the classes in $\mathbf{\Omega}$ from these datasets so as to accurately identify design problems of the sampled set belonging to the same class. In particular, the supervised learning problem of constructing a classification model using observational data is considered. Let $\{\mathbf{x}_i, y_i, i = 1, 2, \ldots, n\}$ denote the training dataset, where $\mathbf{x} \in R^d$ is the DPD input vector and $y \in \mathbf{\Omega}$ denotes the target output class to be predicted. If learning is successful, generalizing from these data would be possible to aid future design sessions by successfully inferring the most appropriate optimization routines to be recommended for searching on unseen design problems in the given domain.

A brief survey of machine learning techniques has been performed to identify a suitable learning or classification model for the present application [25, 48, 49, 50, 51, 52, 53]. Table 2.1 lists the variety of learning models investigated in this work. 1R [49] is one of the simplest classifier that makes a one-rule, i.e., a rule based on the value of a single attribute. Holte [49] shows that it is easy to get reasonable prediction accuracy on many commonly used datasets by simply looking at only one attribute. However, contrary to common claims and misinterpretations regarding Holte's results, the 1R inducer is often inferior to ID3 and C4.5 decision tree modeling techniques [53]. ID3 is a very basic decision tree algorithm. C4.5 is an extension of the original ID3 algorithm with pruning and unknown handling capabilities. Other learning models like Nearest-neighbour [48] are based on the idea that prediction may be effected through exploitation of similarity. They are also commonly known as lazy learners since they simply store all training dataset and postpone all efforts towards inductive generalization until prediction time. Naïve Bayes [52] computes conditional probabilities of the classes given the instance and picks the class with the highest posterior. Attributes are assumed to be independent, an assumption that is unlikely to be true, but the algorithm is nonetheless very robust to violations of this assumption. Latent semantic indexing technique has been effectively used for information retrieval in the text domain in a variety of tasks [25]. It presumes the existence of a latent structure in the textual data and treats unreliability in observed term-document association as a statistical problem to uncover this structure. The success of latent semantic indexing in information retrieval leads the author to contemplate its application here. Last but not least, the neural network [50] considered here learns to approximate the probability density function of the training dataset. When an instance is presented, the first layer computes the similarities between the instance and the training datasets. The second layer sum these contributions for each class of inputs to produce as its net output a vector of probabilities. Finally, the predicted class has the highest posterior.

Legends	**Machine Learning Techniques**
1R	Holte's Simple Classifier [49]
ID3	Decision Trees I [53]
C4.5	Decision Trees II [53]
IB	Nearest-Neighbour [48] (Instance Based)
NB	Probabilistic [52] (Naïve Bayes)
LSI	Information Retrieval [25, 51] (Latent Semantic Indexing)
PNN	Neural Network [50] (Probabilistic Neural Network)

Table 2.1: List of Machine Learning Techniques investigated.

In particular, the aim is to select a machine learning model $\mathbf{M}$, where $\mathbf{M} : R^d \to \mathbf{\Omega}$, that displays superiority in accuracy estimation, standard deviation and model transparency. On the basis of classification accuracy and standard deviation, the algorithms found to be most competitive in the present application are Probabilistic Neural Networks [50], Naïve Bayes [52] and C4.5 [53]. Even though most of the machine learning techniques considered here allow some form of manual tuning to obtain improved estimation performances, this has been considered to be too time-consuming and computationally expensive. Among the learning techniques considered, the use of the decision tree inductive learning algorithm C4.5 [53] is preferred because it produces reasonable classification accuracy at relatively low cost and, more importantly, because it possesses the ability to generate trees or rules that provide some of the transparency that design engineers seek. The empirical performances of the machine learning techniques when applied to aerodynamic aircraft wing design will be presented in section 2.5.3. Design engineers often lack great expertise in the use of optimization methods and therefore have little confidence that extensive computational runs will produce worthwhile results as opposed to just burning up compute cycles. Therefore, when recommending search routines for design activities, it is important for the decision-making process to provide the necessary transparency. Of the many machine learning techniques available, knowledge derived in the form of decision trees or rules seems to satisfy this human-centered criterion the most. Besides, human specialists can manually validate these machine-generated decision trees or rules and also use them to enhance the domain and optimization knowledge of less experienced design engineers.

From empirical studies, AISS often performs better than BPSS in the present application. Nevertheless, AISS tends to perform poorly under the provision of insufficient or incomplete chronicled data available for learning. These are instances where search routines are too similar in performance to each other, such that more design problem samples may be necessary to enable accurate differentiations of best, or under multi-class situations where too many search routines have been observed to work well on the problem domain under study (i.e., set $\boldsymbol{\Omega}$ is large). Under such events where generalization from the labeled datasets does not give robust performance, BPSS can fit in as a reliable alternative. Both strategies are therefore combined to complement each other in facilitating the appropriate choice of optimization technique. The recommendations given by the BPSS are employed if the BPSS displays estimated accuracy greater than 0.7 or when the estimations given by the AISS is under 0.7.

§2.4 Adaptability

It is important that any acquired knowledge is kept up-to-date. Here, once knowledge about the merits and weakness of the optimization routines on a domain has been successfully generated, an adaptability process is invoked repeatedly to ensure the continued usefulness of the knowledge base. As previously mentioned, throughout the design process, a great deal of data is produced as a result of design-evaluate-redesign actions conducted by designers. This data is archived and indexed according to the design problem being studied. At other times, scheduled batch jobs may be started to update the domain knowledge base using information from the archive to ensure good coverage. Such a process helps enable the optimization method advisor remain relevant to current design problem domains as well as to encompass new ones.

§2.5 Domain Optimization Method Advisor Applied to Real World Problem Domains

To demonstrate the applicability of the Domain Optimization Method Advisor on real world engineering design domains, the approach is used for the conceptual design of transonic civil transport aircraft wings [14] and ship hull-forms [54]. The objective of the aerodynamic problem considered here is minimization of wing drag D/q meters^2 as calculated by using an empirical drag estimation tool, TADPOLE [14], with target lift, wing weight, volume, pitch-up margin and root triangle layout chosen to be representative of a 220 seat wide body airliner. In the ship domain, the design of a frigate that has minimal resistance for a fixed displaced volume, block coefficient, waterline flare and depth to draught ratio is considered [54]. Both design problems have various design constraints that must be met. The Design Exploration System utilized is the one described by Keane, and known as OPTIONS [1]. OPTIONS is a design exploration and optimization package that contains a large range of optimization routines that may be used to search for the optimal design parameters. Further details of the OPTIONS DES package and the design problems are contained in Appendix A and C, respectively.

§2.5.1 Chronicled Design Optimization Data

To achieve a good representation of the design domains, offline simulations on the domain have been carried out. In the aircraft wing design domain, a set of 729 design problems representing the domain space are sampled using the Latin hypercube method [38], each defined by the DPDs: cruise height, Mach number and fuel weight fraction, bounded between 7500 – 12,000 meters, 0.1 – 0.85, and 0.2 – 0.5, respectively. The method provides good uniformity and flexibility on the size of the sample. The range for the design parameters was obtained from design engineers who are actively involved in aircraft wing design. The above comment also applies to the ship hull-form design domain. A set of 625 ship design problems was sampled across ranges of displaced volume, block coefficient, waterline flare at maximum beam and depth to draught ratio, bounded between 2000 – 4,000 meters^3, 0.43 – 0.50, 0.0 – 10.5 and 2 – 3, respectively. Offline simulations are then conducted on the sampled design problems using each of the 31 search routine available in the OPTIONS DES on each domain. Two-thirds of the sampled design problems form the training set while the remaining one-third form the validation sets, which are used after training to validate the advisor.

§2.5.2 Domain Knowledge Construction - Best Performing Search Strategy

As previously discussed, different modes of domain knowledge representing the intention of the design engineer can be learnt from the archived data sources. In this dissertation, design accuracy and the speed of search are considered here as the two most important criteria by the design engineers. This knowledge can be combined from the data sources of the design domain under consideration using

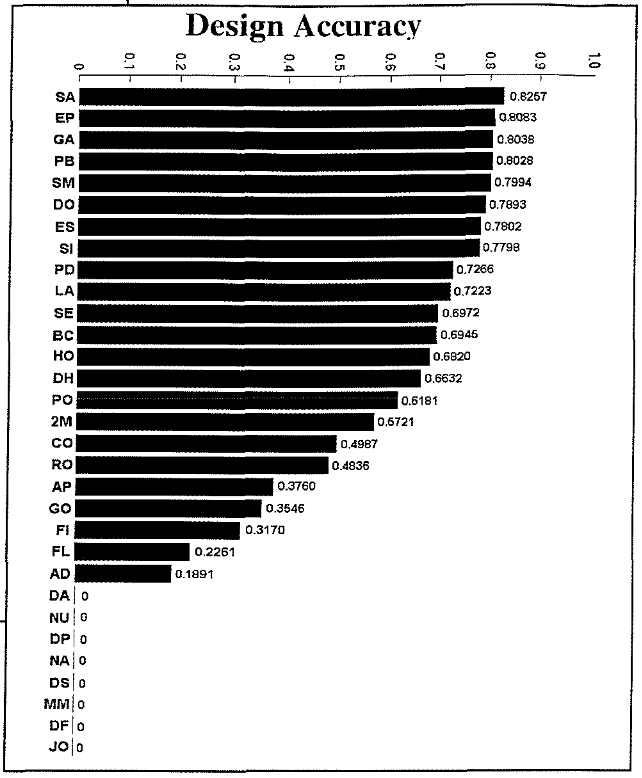
$$q = w * \mu_1 + (1 - w) * \mu_2 \tag{2.4}$$

The formulation in equation 2.4 is a specialized form of equation 2.1 for the two criteria instance. Only a single criterion weight factor, w, is necessary to define the balance for design speed and accuracy. μ_1 and μ_2 in equation 2.4 represent the evaluation count quality and objective fitness quality of a search routine, respectively. Using equation 2.4, a preferential list of optimization search routines can be obtained by varying w and subsequently used to recommend methods to design engineers for future search activities. This could be weighted depending on the importance of the two criteria to the design engineer; typically this might be to finish the design search as quickly as possible, achieve the best possible design or some compromise between them. This leads to three common domain knowledge models often desired by design engineers. By setting w in equation 2.4 to 0.5, the information necessary for a Balanced-Overall (BO) knowledge model can be obtained. The Best-Speed (BS) and Best-Design (BD) models can be obtained by having w set to 1.0 and 0.0, respectively.

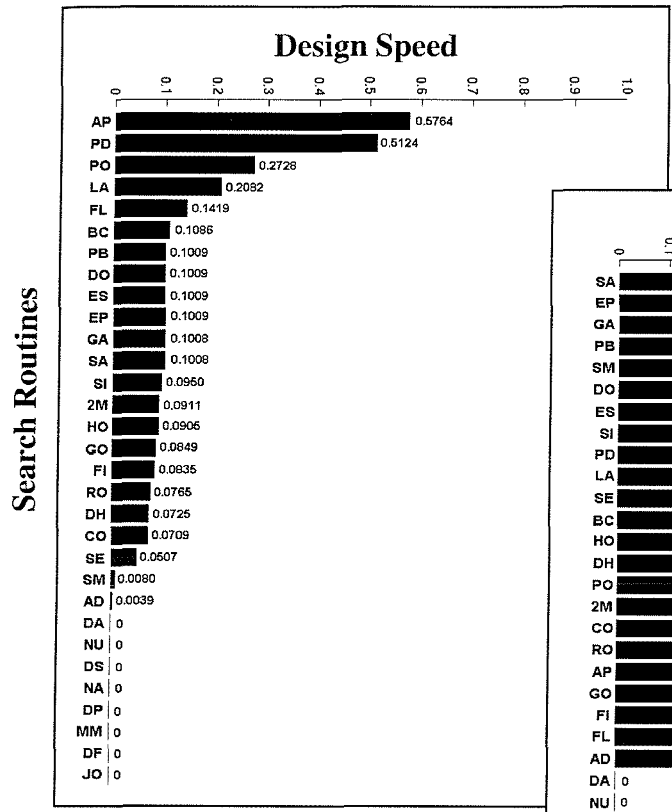
The final design accuracy and efficiency of the 31 optimization search routines on both real world design domains are summarized in Figures 2.1 and 2.2. Each abbreviation in the figures represents an optimization search routine available in the OPTIONS DES. All values in the figures are normalized to unity, where unity represents the best possible performance a search method among those considered here has achieved. The Pareto fronts of optimization routines for the two domains are also shown in Figures 2.1 and 2.2, represented by the dashed lines. From these fronts, the recommended optimization routines based on the use of Best Performing Search Strategy can be readily obtained. These are summarized in Table 2.2.

Knowledge Model	**Design Domain I: Transonic Aircraft Wing-Recommended Search Routine**	**Design Domain II: Ship Hull-Form-Recommended Search Routine**
Best-Speed	Successive Linear Approximation (AP)	Davidon-Fletcher-Powell Strategy (DF) by Schwefel
Best-Design	Simulated Annealing (SA)	Simplex Strategy of Nelder & Meade (SM) by Siddall
Balanced-Overall	Powell Direct Search Method (PD) by Siddall	Davidon-Fletcher-Powell Strategy (DF) by Schwefel

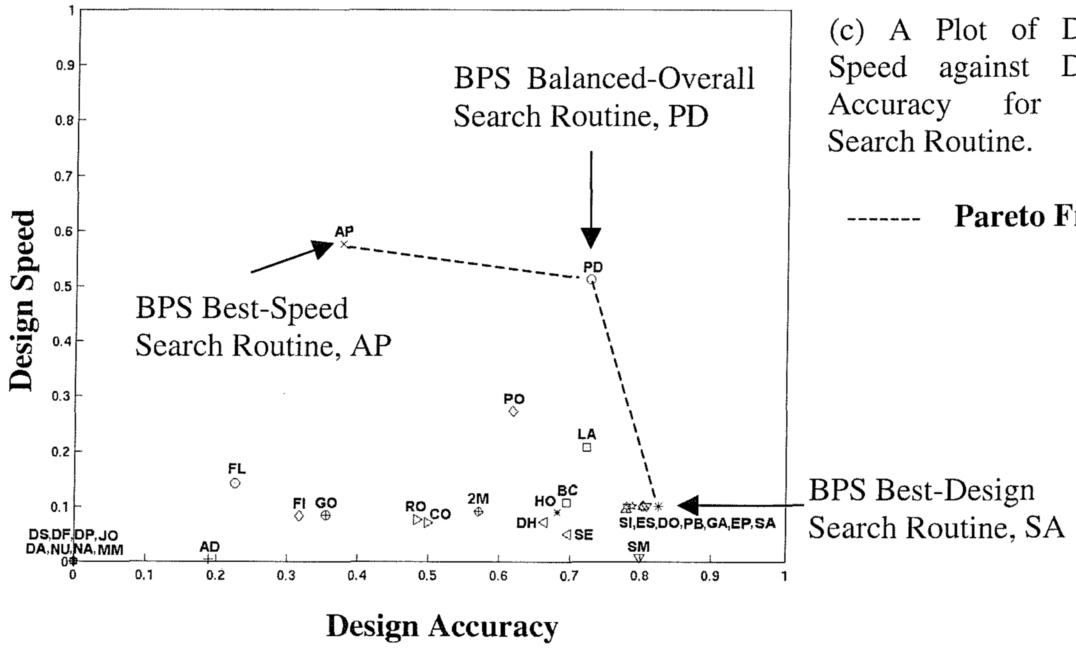
Table 2.2: List of search routines recommended in the design domains studied. Results presented are based on using the Best Performing Search Strategy of the advisor.



(a) Design Accuracy of each Search Routine with rankings in descending order.



(b) Design Speed of each Search Routine with rankings in descending order



(c) A Plot of Design Speed against Design Accuracy for each Search Routine.

------- **Pareto Front**

Figure 2.1: Relative Final Design Accuracy and Efficiency of OPTIONS DES Search Routines on Aircraft Design Domain. (Results presented are based on $\frac{2}{3}$ sampled wing design problems. Performances normalized to unity: larger values indicates faster searches or better designs.)

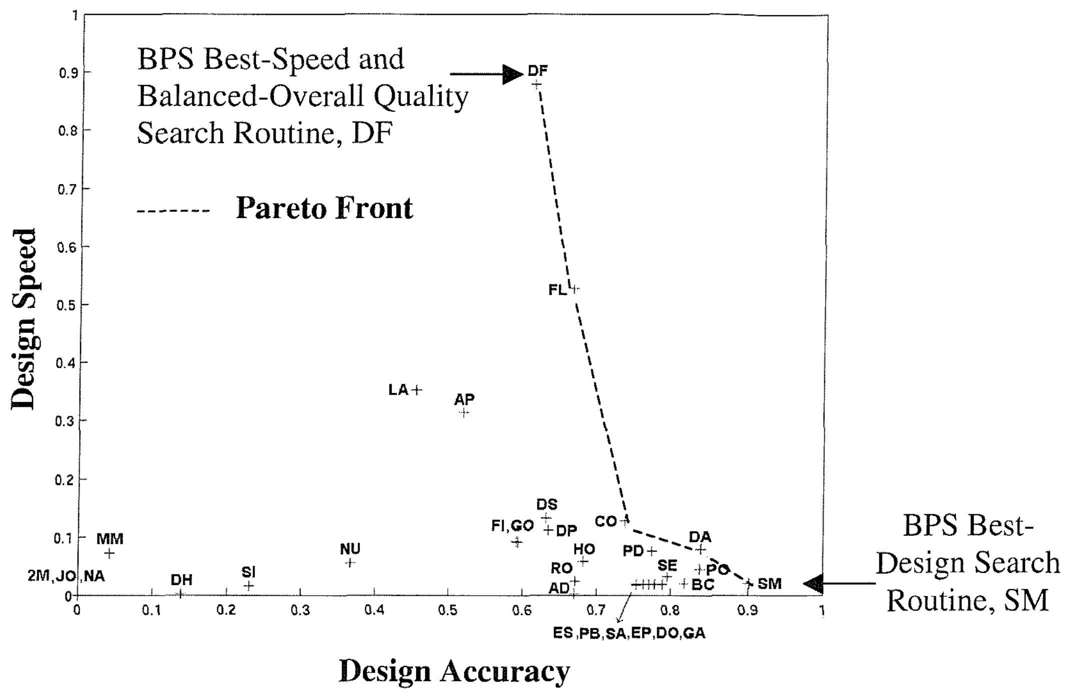


Figure 2.2: Relative Final Design Accuracy and Efficiency of OPTIONS DES Search Routines on Ship Design Domain. (Results presented are based on $\frac{2}{3}$ sampled ship hull-form design problems. Performances normalized to unity: larger values indicates faster searches or better designs.)

On the transonic aircraft wing design problem domain, the routine giving the best speed is identified to be Successive Linear Approximation [3] - the BS knowledge model emphasizes the importance of speed and therefore ignores the final result of the wing drag values as long as the final searched design parameters are feasible. Simulated Annealing [9] provides the best solution accuracies on average across the entire design domain. Finally the BO model balances speed and accuracy and the best overall routine is found to be the Powell Direct search routine [3]. In contrast, for the domain of ship hull-form design, the Balanced-Overall and Best-Speed models are found to yield the Davidon-Fletcher-Powell strategy by Schwefel [4] while the Simplex strategy of Nelder & Meade [3] produces the best design accuracy among the numerous search routines.

Besides design search accuracy and efficiency, other criteria may also be included into equation 2.1 so as to generate greater or more complex models that may better accord to the intentions of the design engineers. Alternatively, search routine robustness may be considered as the sole criterion required by the design engineers thus arriving at the results shown in Figures 2.3 and 2.4 for the two engineering design domains. This criterion represents the desire of the designers for routines that always improved the design and never produced infeasible final results. Summing up the results obtained in Figures 2.1 - 2.4, it is clear that these are consistent with the conclusions made in the earlier published works that no single search routine always performs best. As a matter of fact, it can be observed that the properties of each search routines; i.e., design accuracy, efficiency and robustness, differ significantly over the design domains.

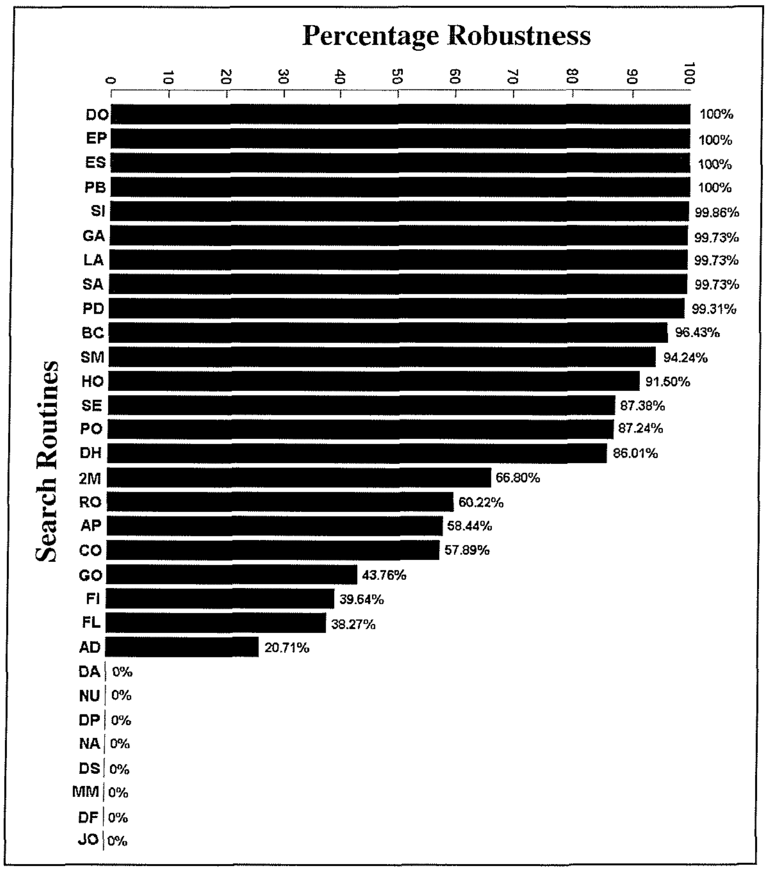


Figure 2.3: Relative Robustness of OPTIONS DES Search Routines on the Aircraft Design Domain. (Results presented are based on $\frac{2}{3}$ sampled wing design problems.)

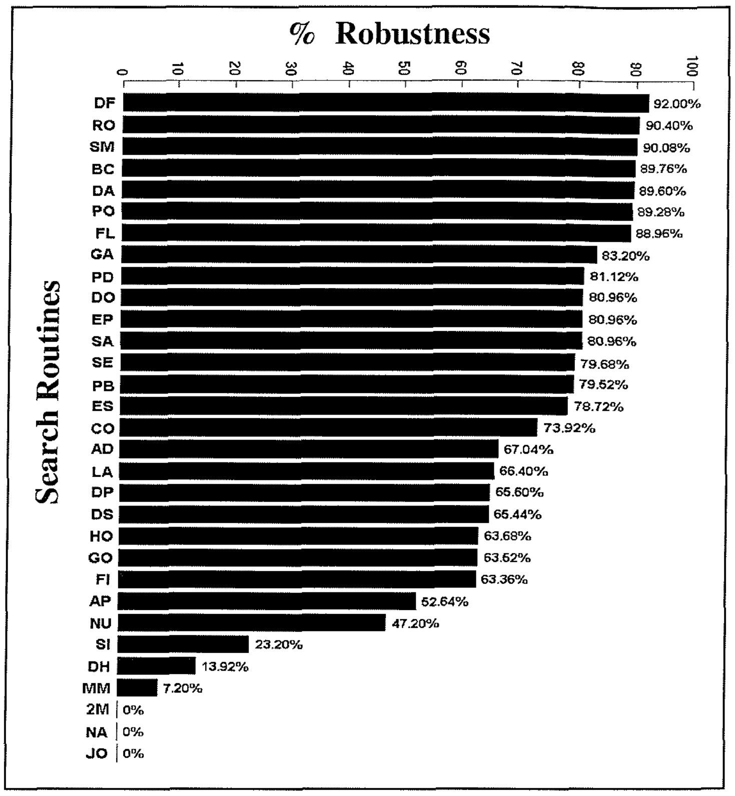


Figure 2.4: Relative Robustness of OPTIONS DES Search Routines on the Ship Design Domain. (Results presented are based on $\frac{2}{3}$ sampled ship hull-form design problems.)

§2.5.3 Domain Knowledge Construction - Artificial Intelligence Selected Strategy

Unlike the BPSS, any data available for learning in the Artificial Intelligence Selected Strategy undergoes further data processing to produce table-like datasets, before the knowledge models can be built. The sampled design searches are labeled and ranked according to the search routine that performs best. So for example, for the best speed model, 'best' represents the search routine that provides a feasible design within the shortest period of time. Table 2.3 shows a portion of the processed BS dataset.

Statistics on the best performing routine when gathered from the various datasets produced in the transonic civil aircraft wing design domain are found to yield the following information:

- Even though there are 31 optimization search routines available in the OPTIONS

DES, across the entire sampled wing design domain of 729 sets of parameters, only five search routines ever rank as 'Best-Speed'. These are AP, PD, PO, FL and LA and form 56.0%, 21.7%, 16.6%, 4.7% and 1.0% of the BS dataset. Class set $\boldsymbol{\Omega}$={AP, PD, PO, FL, LA}.

- Seven search routines rank as giving 'Best-Design' with class set $\boldsymbol{\Omega}$={SM, PD, SI, 2M, AD, CO, SA} and respective percentages of 28.2%, 18.4%, 15.1%, 12.5%, 9.2%, 8.9% and 7.7% in the BQ dataset.
- Finally six search routines rank as 'Balanced-Overall' with class set $\boldsymbol{\Omega}$={AP, PD, PO, FL, LA, 2M} and respective percentages of 49.1%, 28.4%, 14.1%, 4.7%, 2.2% and 1.5% in the BO dataset.

Using the C4.5 induction algorithm, the desired knowledge models are extracted from the processed datasets in the form of decision trees or rules. Figure 2.5 shows the BS knowledge rules obtained from the aircraft wing domain dataset. Further, the estimated classification accuracy and standard deviation of various machine learning techniques when applied on the aircraft wing problem datasets are summarized in Figure 2.6. It is evident from the figure that among the machine learning techniques considered, the C4.5, Naïve Bayes and Probabilistic Neural Network are most competitive in the present application. The knowledge models for the optimization routines generated using the present strategy may then be used to aid design engineers in future design search activities via recommendations of appropriate search routines.

Domain Problem Descriptors of Aircraft Wing Design Problem				
F-Mach (0.1 – 0.85)	***Fuel-Fraction (0.2 – 0.5)***	***(Height (7500 – 12,000)***	**Search Routine Rankings Based on Design Speed**	**Best-Speed (BS) Search Routine**
0.35	0.225	8625.0	AP, PD, LA, BC, EP, DO, ES, PB, SA, GA, SI, HO, SM, AD, CO, PO, RO, GO, FI, SE, FL, 2M, DH	AP
0.725	0.425	10125.0	PD, FL, LA, BC, DH, EP, DO, ES, PB, SA, GA, CO, PO, RO, SI, 2M, HO, FI, GO, SE, SM, AD, AP	PD
		...		
0.1	0.45	7500.0	PO, PD, LA, GO, FI, FL, BC, EP, DO, ES, PB, SA, GA, CO, RO, SI, 2M, HO, DH, SE, SM, AD, AP	PO
0.85	0.275	8250.0	FL, PD, FI, GO, 2M, LA, BC, EP, DO, ES, PB, SA, GA, CO, RO, SI, HO, DH, SE, SM, AD, PO, AP	FL
0.22	0.475	10875.0	LA, DH, PD, BC, EP, DO, ES, PB, SA, GA, PO, 2M, SI, SE, HO, SM, AD, CO, RO, GO, FI, AP, FL	LA

Table 2.3: A portion of the processed Best-Speed dataset with search routine rankings and Best-Speed search technique classes/labels. (Refer to the Appendix A for details of the abbreviated search routines.)

Rule 1: if

$\{(0.6 < \textit{F-Mach} \leq 0.6625)$ AND $(\textit{Fuel-Frac} \leq 0.3)$ AND $(9750 < \textit{Height} \leq 10500)\ \}$

$\longrightarrow$ **Class LA**

Rule 2: if

$\{\ (\textit{F-Mach} > 0.7875)$ AND $(\textit{Height} \leq 9375)\ \}$

$\longrightarrow$ **Class FL**

Rule 3: if

$\{\ (0.225 < \textit{F-Mach} \leq 0.6)\ \}$ OR

$\{\ (0.1625 < \textit{F-Mach} \leq 0.6)$ AND $(\textit{Fuel-Frac} \leq 0.45)$ AND $(\textit{Height} \leq 11250)\ \}$ OR

$\{\ (0.1625 < \textit{F-Mach} \leq 0.6)$ AND $(\textit{Height} \leq 9000)\ \}$ OR

$\{\ (\textit{F-Mach} > 0.7875)$ AND $(\textit{Height} \leq 9375)\ \}$

$\longrightarrow$ **Class AP**

Rule 4: if

$\{\ (\textit{F-Mach} \leq 0.1625)\}$ OR

$\{\ (\ \textit{F-Mach} \leq 0.225)$ AND $(\textit{Height} > 11250)\ \}$ OR

$\{\ (0.725 < \textit{F-Mach} \leq 0.7875)$ AND $(\textit{Fuel-Frac} > 0.375)$ AND $(9000 < \textit{Height} \leq 9750)\ \}$ OR

$\{\ (0.6 < \textit{F-Mach} \leq 0.6625)$ AND $(\textit{Fuel-Frac} \leq 0.3)$ AND $(\textit{Height} > 10500)\ \}$ OR

$\{\ (0.6625 < \textit{F-Mach} \leq 0.725)$ AND $(\textit{Fuel-Frac} \leq 0.275)$ AND $(9000 < \textit{Height} \leq 10500)\ \}$

$\longrightarrow$ **Class PO**

Rule 5: if

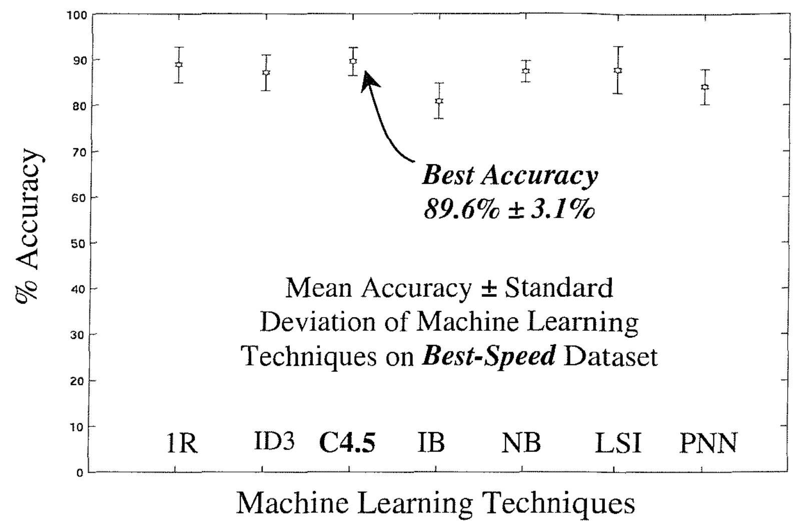
$\{\ (0.6 < \textit{F-Mach} \leq 0.\ 7875)\ \}$ OR

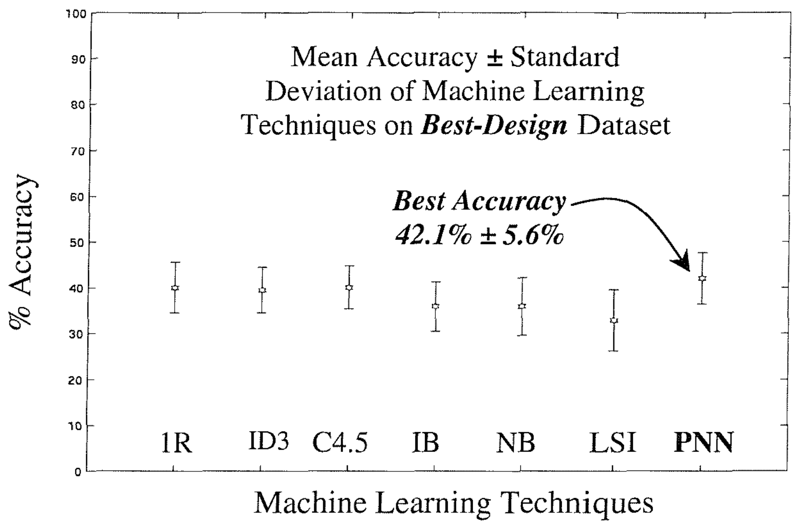
$\{\ (\textit{F-Mach} > 0.6)$ AND $(\textit{Height} > 49375)\ \}$ OR

$\{\ (0.1625 < \textit{F-Mach} \leq 0.\ 225)$ AND $(\textit{Fuel-Frac} > 0.45)$ AND $(9000 < \textit{Height} \leq 10500)\ \}$

$\longrightarrow$ **Class PD**

Figure 2.5: Knowledge on the speed performance of search routines on the Aircraft Wing Design Domain in the form of rules. These are derived by the C4.5 inductive algorithm using the Best-Speed datasets obtained from offline simulations.





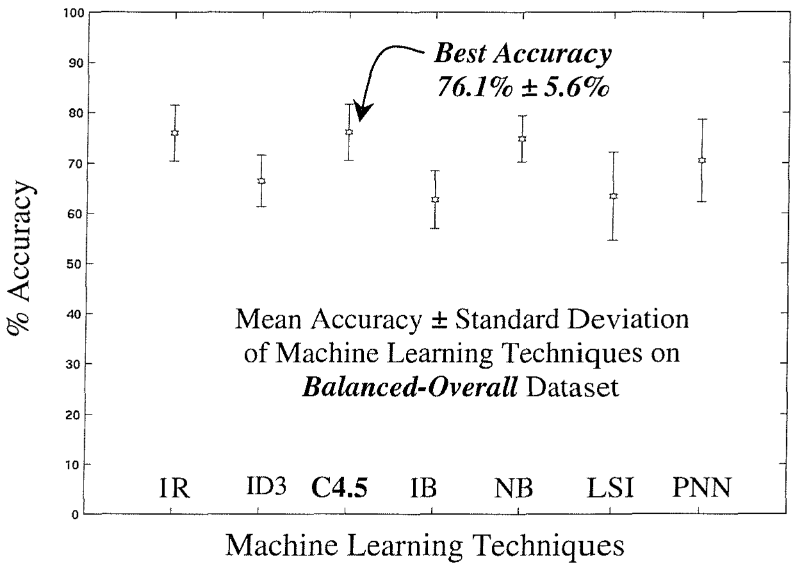


Figure 2.6: Estimated Mean Classification Accuracies and Standard Deviation of the Machine Learning techniques considered.

§2.5.4 Performance Of Domain Optimization Method Advisor

In assessing the proposed Domain Search Advisor, it is useful to develop standards for comparison. Some common procedures obtained from analogues to typical designers' behaviors have been derived for comparisons and are here termed 'Common Designer Strategies' (CDS). These involve tactics from both novices and optimization experts at work: five such conventional strategies have been identified as

- The simplest and most basic strategy adopted, often by novice design engineers is (CDS1) 'Random Guessing'. This represents a simple random choice of optimization routines from those available in the optimization engine each time a new design space search is carried out.
- The second strategy (CDS2) is the notion of selecting the optimization routine based on the method that was able to successfully generate a feasible design in its first attempt used in the domain under study (i.e., sticking with something that has worked before).
- The third strategy (CDS3) is to utilize an optimization method that is presumed to be most robust, e.g., the Evolutionary Programming (EP) optimization method is often regarded to be very robust and is thus chosen here.
- The fourth strategy (CDS4) is an analogue of the favoritism traits sometimes displayed by design engineers. Here the Genetic Algorithm (GA) is chosen to be the design engineers' favourite.
- The final strategy identified (CDS5), is utilizing an optimization method that has generally been accepted as having the ability to provide a design within the shortest period of time, e.g., Successive Linear Approximation (AP) is often regarded as the fastest available method in the OPTIONS DES.

The performances of the conventional approaches in comparison to the Domain Optimization Method Advisor for design search activities are tabulated in Table 2.4. The resultant performance statistics gathered were obtained based on searches conducted on the validation samples previously discussed. It can be observed that the advisor constantly attain performances that are better (much nearer to unity) when compared to any of the Common Designers Strategies on each respective scheme and on both design domains. Note that in the table, the nearer the value is to unity, the better is the performances of each strategy in conducting design search. This suggests that the strategies proposed in the advisor are capable of providing accurate predictions on choice of search routines, even for unseen design problems. The search advisor was also capable of providing significant improvement in the design search performances as compared to any of the conventional CDS. At the same time, it reduces reliance on human experts by ensuring that design engineers need only minimal knowledge of optimization routines.

Strategy for Choice of Optimization Search Routines	**Estimated Relative Search Performance on Aircraft Wing Design Domain**			**Estimated Relative Search Performance on Ship Hull-Form Design Domain**		
	BSS	BDS	BOS	BSS	BDS	BOS
CDS1	0.1007	0.4199	0.2496	0.0762	0.5385	0.2560
CDS2	0.1351	0.5166	0.3155	0.1056	0.5761	0.3353
CDS3	0.0981	0.6782	0.3882	0.0180	0.7017	0.3599
CDS4	0.0981	0.6740	0.3861	0.0192	0.7518	0.3855
CDS5	0.5908	0.2978	0.4443	0.3423	0.5513	0.4468
Domain Optimization Method Advisor	0.9447	0.7017	0.7191	0.9351	0.9614	0.7751

Table 2.4: Relative performance of the Domain Optimization Method Advisor in comparison with the 'Common Designers Strategies' CDS 1-5. Results are obtained for the various best schemes proposed, taken over the $\frac{1}{3}$ unseen design problems from the respective domains. Values closer to unity more nearly achieve the desired results.

§2.6 Conclusion

In this Chapter, a Domain Optimization Method Advisor has been presented. The Best Performing Search Strategy and Artificial Intelligence Selected Strategy have been combined to make effective use of chronicled design optimization data, thus providing an accurate inference of the most appropriate optimization method to use on unseen design problems. The work makes use of two realistic complex engineering design domains where significant improvement in search performances can be achieved if design searches are conducted using the proposed advisor. Besides improvement in design search performance, the system also helps reduce reliance on optimization domain experts by ensuring that minimum knowledge of optimization techniques is required by design engineers when performing design searches, and at the same time, eliminating any human biases that may exists.

The results presented also support the "no free lunch theorem" as well as the conclusions of many published papers that no single optimization method is best for all design problems, even including those within the same design domain. This further emphasizes the importance of the search advisor in supporting engineering design optimization process.

Nevertheless, a significant limitation of the Domain Optimization Method Advisor is the need for offline simulations to be conducted. Simulations are conducted to ensure the completeness of the historical design data and thus the acquired knowledge about the merits and limitations of the search methods on the domain under study. This process can be very time consuming if the underlying analysis codes used are of significant complexity. Fortunately, in most conceptual design activities this is not so. In addition, the derived knowledge could be easily affected by relatively common changes in the design problem statements, such as modification of constraint bounds and other parameters. In practice, if modifications on problem statements are expected to occur, evolutionary algorithms like the hybrid Genetic Algorithm are commonly made use of. Taking this cue from the engineering design community, the next Chapter presents a Meta-Lamarckian Learning methodology.

Chapter 3

Meta-Lamarckian Learning

IN Chapter 2, an approach for reducing the influence of inappropriate choice of search methods on complex engineering design optimization performance was discussed. In this Chapter, Meta-Lamarckian Learning in Hybrid Genetic Algorithm-Local Search [26] is presented. Meta-Lamarckian Learning presents an alternative approach to reducing the influence of local optimization method choice on design search performance. In contrast to the former approach, it is relatively unaffected by changes in design problem statements. Furthermore, since it makes use of optimization data chronicled during the lifetime of the design search process, there is no requirement for prior offline simulation. Two adaptive strategies proposed for local optimization routine selection in hybrid GA-LS are investigated and described in this Chapter. These strategies facilitate a cooperative and competitive environment for different local optimization routines, permitting the routines to collaborate and work together to accomplish the shared goal more effectively. Experimental studies and analysis of these strategies on benchmark test functions and the aerodynamic aircraft wing design problem already described are also presented. It is shown that the proposed approach aids design engineers working on complex engineering problems by reducing the possibilities of employing inappropriate local search methods in a hybrid GA-LS, while at the same time, yielding robust and improved design search performance.

§3.1 Introduction

Genetic Algorithms (GAs) are a powerful set of global search techniques that have been shown to produce very good results on a wide class of complex design problems. GAs are capable of exploring and exploiting promising regions of the search space. They can, however, take a relatively long time to locate the local optimum in the region of convergence and may sometimes not find the optimum with sufficient precision.

Torn and Zilinskas [55] in the section of their work entitled "Global search methods: exploration and exploitation", observe that two competing goals govern the design of global search methods: exploration is important to ensure global reliability; i.e., every part of the domain is searched enough to provide a reliable estimate of the global optimum; exploitation is also important since it concentrates the search effort around the best solutions found so far by searching their neighbourhoods to produce better solutions. Many search algorithms achieve these two goals using a combination of dedicated global and local searches. These are commonly known as hybrid methods. Hybrid Genetic Algorithm-Local Search methods, which incorporate local improvement procedures with traditional GAs may thus be used to improve the performance of GAs in search. Such hybrids have been used successfully to solve a wide variety of engineering design problems and experimental results show that GA-LS hybrids not only often find better solutions than simple GAs, but also that they may search more efficiently [11, 17, 18, 19, 55, 56]. Note that the term Local Search and the abbreviation LS are used interchangeably in this Chapter.

Davis [11] argues that hybridizing genetic algorithms with the most successful local search method for a particular problem gives one the best of both worlds: correctly implemented, these algorithms should do no worse than the traditional GA or LS alone. Clearly, what this implies is that unless one knows which local search method most suits the problem in hand (along with its correct parameters settings), a GA-LS hybrid method may not perform at its optimum or worse, it may perform less well than using the GA or the LS alone. Some recent studies on the choice of local search method employed have shown that this choice effects the efficiency of problem searches significantly. The influence of the local search method employed has been shown by Mitchell [57] and Hart [58] to have a major impact on the search performance of GA-LS hybrids. These experiments conducted on two different local methods, demonstrated that the performance obtained by GA-LS hybrids can indeed be worse than that obtained by the GA alone. The varied suitability of each local search method to different problems also helps explain why for the last 10-15 years, GA-LS hybrids have relied on the use of a variety of different methods as the local improvement procedure. The significance of local search method choice on the performance of GA-LS hybrids is therefore not a new observation. However, little work has been done to mitigate this problem. The greatest barrier to further progress is that, with so many local search methods available in the literature, it is almost impossible to know which is most relevant to a problem when one has only limited knowledge of the problem structure. In real situations, it is often not possible to have even a partial understanding of the problem structure that is being searched before one starts. Moreover, LS(s) by themselves are known to work very differently with different design problems, even among problems from the same design domain, as seen in Chapter 2. Depending on the complexity of a design problem, local search methods that may have proven to be successful in the past might not work so well, or at all, on others.

Given the restricted amount of well-established theoretical knowledge in this area and the limited progress on mitigating the effects of incorrect local search method choice in GA-LS hybrids, it is reasonable to ask whether the effects of this choice on performance might be reduced via some intelligent means while the GA-LS is running. Many adaptive systems already exist and have been shown to solve some problems very effectively. Spears [59] and Ko [60] have attempted to adapt the crossover operator of evolutionary algorithms based on locality and convergence or the contributions made by the solutions. Adaptation of the mutation operator has also been attempted by Schwefel [4], Fogel [61], Back [62], and Smith et al. [63]. A general survey of various adaptive systems in evolutionary computations and their influences can be found in the work of Hinterding [64]. Recently, some LS parameters in hybrid GA-LS were also adapted for three-matching [65] and identification problems [66]. Nevertheless, the operator that has perhaps the greatest influence on GA-LS hybrid performance, but one that has yet to be thoroughly explored for adaptation, is the choice of LS(s) that best matches the problem under study. In this Chapter, we focus on investigations into the adaptive choice of local search methods to ensure robustness in GA-LS hybrid search performance. Two adaptive strategies are presented for the selection of the local method from a pool of LS(s) while the design search progresses. The strategies make effective use of online chronicled design optimization data to aid in the choice of local search methods, thus reducing the influence of inappropriate choice on GA-LS hybrid design search performance.

Note that here we present work based only on Darwinian and Lamarckian evolution. The standard GA concept represents the Darwinian approach. Lamarckian learning forces the genotype to reflect the result of improvement through placing the locally improved individual back into the population to compete for reproductive opportunities [67]. The approach of using multiple local search methods during a hybrid GA-LS search in the spirit of Lamarckian learning is here termed Meta-Lamarckian learning.

This Chapter is organized in the following manner. Section 3.2 presents traditional Lamarckian learning, a basic Meta-Lamarckian learning scheme and two potential adaptive strategies inspired by social evolution. Section 3.3 summarizes some experimental studies on benchmark test functions, analyses the results and recommends the more competitive of the two adaptive Meta-Lamarckian learning strategies. Section 3.4 demonstrates the Meta-Lamarckian learning approach on the real world aerodynamic wing design problem. Finally, section 3.5 provides the main conclusions.

§3.2 Meta-Lamarckian Learning in GA-LS Hybrids

Traditionally, GA-LS hybrids with Lamarckian learning procedures are based on the use of the GA and a single local search method for local improvements. Little published work has dealt with GA hybrids using multiple local search methods to perform global optimization on a single problem. Every search algorithm, except for uniform random search, introduces some kind of bias into its search. Different local search methods have different biases. It is these biases that make a method efficient for some classes of problems but not for others [68]. This has been seen already in Chapter 2. Therefore the motivation for the use of multiple local search methods, and thus Meta-Lamarckian learning in hybrid GA-LS searches, is to reduce the influence of inappropriate choice of local search methods on design optimization performance and at the same time, to achieve improved search performance.

§3.2.1 Basic Meta-Lamarckian Learning Scheme, GA-B

The most basic Meta-Lamarckian learning scheme for LS selection is one in which the local search method is randomly selected at every iteration during the run - this does not adapt but has the advantage of at least giving all the available local methods a chance at all times. It is purely stochastic in nature: each local method has an equal probability of being chosen at all iterations. This approach forms a base-line algorithm with which other Meta-Lamarckian learning strategies may be compared.

§3.2.2 Adaptive Meta-Lamarckian Learning

Lamarckian learning in hybrid GA-LS search may be structured to promote cooperative, competitive, or individualistic efforts. The traditional approach in GA-LS where a single LS is used on the problem throughout the search is an example of individualistic effort. There has been a long history of research on these different kinds of effort in social evolution since the first study in 1898, where it was shown that cooperation and competition, as compared with individualistic efforts, typically result in higher achievement [69]. Inspired by these works, the adaptive strategies proposed here for Meta-Lamarckian learning in hybrid GA-LS are structured to promote cooperation and competition among the different LSs, working together to accomplish the shared optimization goal.

The idea behind the adaptive strategies is that as the search progresses, the effectiveness of each local search method in dealing with the current problem is learnt. Knowledge about the current population of solutions and each LS is thus built dynamically online, so identifying the strengths and weaknesses of the LSs for the problem currently being worked on, given its current state. To promote competition among the LSs, the local methods with higher fitness improvement measures are rewarded with greater chances of being chosen for subsequent chromosome optimizations. On the other hand, cooperation among the LSs in solving the problem is achieved via problem decomposition or diversity in the LS selection. In nature, fitness is determined by a creature's ability to overcome the dangers that may prevent it from reaching adulthood and reproducing. In GAs, a fitness function is used to evaluate the quality of the individuals in the population. This function takes the genotype of the individual as an argument, decodes it into a phenotype, and judges the quality of the phenotype using problem specific code. It is easy to reward the local methods according to their on-line performance on the basis of the fitness differences of the children and the parent after the application of the local method across the efforts is completed. In the Meta-Lamarckian learning approach considered here, the reward η assigned to a local method on a parent chromosome is calculated by

$$\eta = \frac{\lambda \mid pf - cf \mid}{\tau}, \tag{3.1}$$

where pf is the initial function fitness of a parent chromosome before local search and cf is the final function fitness of the child chromosome obtained after applying local search. Here, τ is the number of LS function evaluation calls made to reach the improved child chromosome or solution. Alternatively, the actual cpu time may be used in place of the number of LS function calls made on the parent chromosome. Further, we distinguished between absolute and relative reward of local search methods [70]. The term $\frac{\mid pf - cf \mid}{\tau}$ provides a simple measure of the rate at which the local search improves a design and is an obvious component of absolute reward measure; such a reward model was also used on an artificial problem by Lobo and Goldberg [71]. λ signifies the relative reward which scales the absolute reward in proportion to the method's ability to produce high quality genotypes when compared to the best global solution obtained so far. Here, λ is set as $\frac{bf}{cf}$ or $\frac{cf}{bf}$, for minimization and maximization problems, respectively. bf is the best solution encountered so far in the global search. We have tried a number of terms to provide this measure such as $\frac{1}{\mid pf - bf \mid}$, but find that the ratio of bf to cf gives the best performance in practice. The reward obtained by each LS on the chromosomes then influences which method is selected from the pool of available methods to proceed with the local improvement.

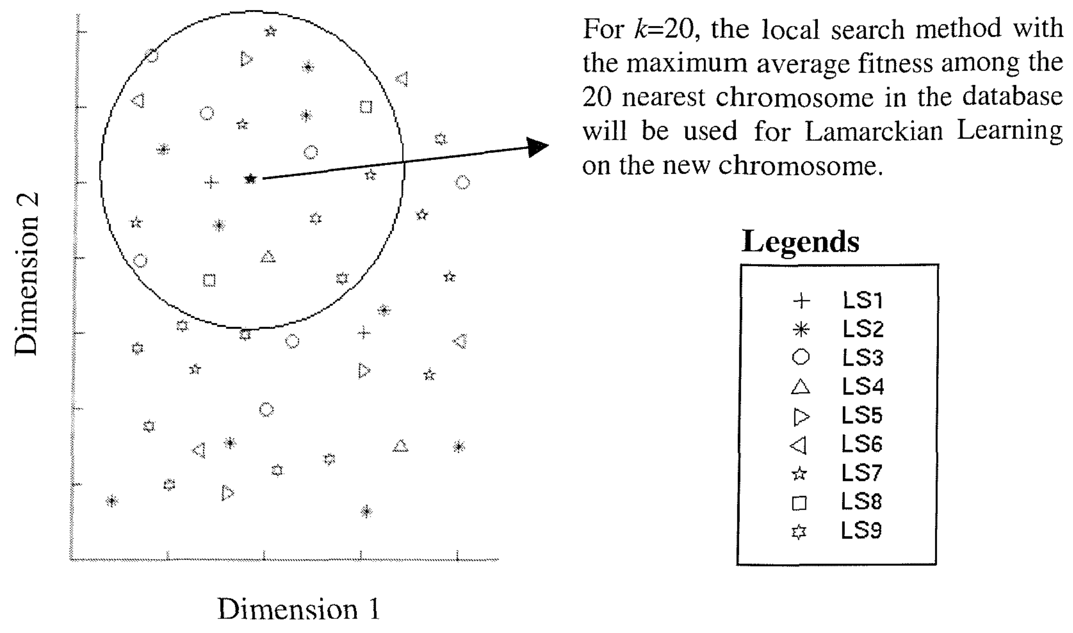
According to a survey conducted by Hinterding [64], adaptive systems are generally categorized as *deterministic, adaptive* or *self-adaptive*. In this dissertation, two adaptive Meta-Lamarckian learning strategies are proposed for study: 1) Sub-problem Decomposition - GA-S1 and 2) Biased Roulette Wheel - GA-S2.

§3.2.3 A Heuristic Approach, Sub-problem Decomposition - GA-S1

Sub-problem Decomposition, GA-S1, is a heuristic approach. At the start of the strategy, each LS is given an equal probability of being chosen as the local search method to be used. The reward of the chosen local method searching on a chromosome is measured using equation 3.1. The parent chromosome and selected local methods together with the reward obtained are archived in a database that is used later to guide future LS choice. The set of parent chromosome vectors archived in the database is denoted by $P \equiv \{p_j\}_{j=1}^{m}$, where m is the database size at any instant of search.

Next, after some pre-defined number of generations g has been completed, the mechanism of sub-problem decomposition takes over. For each unseen parent chromosome, denoted by $\hat{p}$, in the GA population to be searched, the strategy locates the k nearest neighbours from the archived database P, using a simple Euclidean distance metric. This subset of k chromosomes in P is denoted by P_k. The local search methods associated with P_k then form the local sub-pool of candidate LS's that will compete, based on their rewards, to decide on which method proceeds with the local improvement of $\hat{p}$. After local search, all $\hat{p}$ and the chosen LS, together with the reward obtained are updated into the database. See Figure 3.1 for a pictorial illustration and pseudo-code of the strategy.

With the choice of LS involving only the rewards for candidate LS's that are applicable in the neighbourhood of the chromosome to be improved, the strategy decomposes the original problem cost surface into many sub-partitions dynamically, and attempts to choose the most competitive local search method for each sub-partition. In the same manner, it creates opportunities for joint operations between different LSs in solving the problem as a whole, because the diverse LSs help improve the overall population based


For k=20, the local search method with the maximum average fitness among the 20 nearest chromosome in the database will be used for Lamarckian Learning on the new chromosome.
Legends
+ LS1
✳ LS2
○ LS3
△ LS4
▷ LS5
◁ LS6
☆ LS7
□ LS8
✡ LS9
Dimension 2
Dimension 1

GA-S1 Pseudo-Code

```
BEGIN
    If (GA Generation < g)
        * Generate a random number between 1 and LS pool size.
        * Select the LS method that the number indicates and apply.
        * Create/Update Database.
    Else
        * Locate $k$ nearest chromosomes in set $P$ to new chromosome $\hat{p}$ using Simple
          Euclidean Measures; i.e.,

              $P_k \Rightarrow \{ \; k \min_{p_j \in P} \| \hat{p} - p_j \| \; \}$        (3.2)

        * Find the average fitness of each member of the reduced LS pool based on
          $P_k$.
        * Select the LS method with the maximum average fitness and apply.
        * Update Database.
    End If
END
```

Figure 3.1: Pictorial Illustration and Pseudo Code of Sub-problem Decomposition Strategy, GA-S1.

on their areas of specialization. This Sub-problem Decomposition strategy thus promotes both cooperation and competition among the LS's during hybrid search.

§3.2.4 A Stochastic Approach, Biased Roulette Wheel - GA-S2

The Biased Roulette Wheel strategy, GA-S2, is a stochastic approach making use of knowledge gained on-line to form biases. During the training stage, each local method is first given a single opportunity to hybridize with the GA in optimizing a chromosome. After this, the probability that a local method will be chosen to work on any subsequent chromosome is biased according to its previous performance, which now changes dynamically as the overall search progresses. The measurement of each local method's fitness/reward on a chromosome is again based on equation 3.1, and a biased roulette wheel is used to pick the subsequent local search methods, based on the rewards taken over all previous local searches. See Figure 3.2 for a pictorial illustration and pseudo-code of the Biased Roulette Wheel strategy.

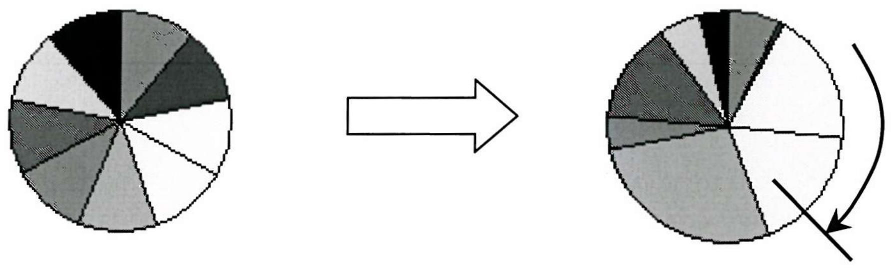
Since the choice of LS is biased according to the reward or fitness of each local search method, the Biased Roulette Wheel strategy is generally a competitive strategy. Likewise, the stochastic nature of the strategy guarantees diversity in the LS selection, hence restraining any LS from dominating throughout the search. By ensuring diverse LS methods participation in the problem search, the strategy promotes joint operation and hence cooperation between local search methods.

§3.3 Numerical Studies on Benchmark Problems

In this section, some experimental studies on benchmark problems obtained by implementing Meta-Lamarckian Learning within a standard genetic algorithm (GA) are presented. The basic steps of the hybrid GA-LS search with Meta-Lamarckian Learning are outlined in Figure 3.3.

In the first step, the GA population may be initialized randomly or using design of experiments techniques such as Latin Hypercube Sampling [38]. Subsequently, for each chromosome in the population, a local search is selected from the pool of multiple

Subsequently, spaces are allocated according to the LS's fitness.



Training Stage:
Local search methods have equal space on roulette wheel initially, thus equal probability.

A single spin of the roulette wheel will pick the local method.

GA-S2 Pseudo-Code

```
BEGIN
    If (Training Stage)
        * Ensure each LS is given one chance to participate in a random order.
        * Update LS's Global fitness.
    Else
        * Sum the fitness of each member of the LS pool.
        * Determine the normalized relative fitness of each member of the LS pool.
        * Assign space on roulette wheel proportional to local method's fitness.
        * Generate a random number between zero and unity, select the LS method
          where the random number falls within and apply.
        * Update LS's Global fitness.
    End If
END
```

Figure 3.2: Pictorial Illustration and Pseudo Code of Biased Roulette Wheel Strategy, GA-S2.

Meta-Lamarckian Learning Algorithm

```
BEGIN
    Initialize: Generate an initial GA population.
    While (Stopping condition are not satisfied)
        Evaluation of all individuals in the population.
        For each individual in the population.
          * Select LS using the Meta-Lamarckian Learning Strategy employed and
            proceed with local improvement. (Pseudo-code of GA-S1 or GA-S2
            comes in here.)
          * Replace the genotype in the population with the locally improved
            solution.
        End For
        Apply standard GA operators to create a new population; i.e., Selection,
        Mutation and Crossover.
    End While
END
```

Figure 3.3: Proposed General Algorithm for Hybrid GA-LS With Meta-Lamarckian Learning.

LS's based on the Meta-Lamarckian learning strategy in use and employed for local improvement. The fitness or reward given to the local method is updated and this is followed by replacement of the genotype in the population with the locally improved solution (in the spirit of Lamarckian learning). Standard GA operators are then used to generate the next population.

In this research work, a standard genetic algorithm with Baker's stochastic universal sampling selection algorithm, n-point crossover and mutation is employed. A pool of nine local search methods was also selected from the OPTIONS DES to be representatives of the many used in the previous experimental studies. These consist of various optimization methods from the Schwefel libraries, some briefly described by Siddall with a few others available in the literature [1, 3, 4]. The local search routines are implementations of constrained and unconstrained nonlinear methods commonly used in engineering design optimization. Those routines used here have search performances that lie between the best and worst performing GA-LS hybrids on these problems. The nine hybrid GA-

LSs used are GA-BC, GA-CO, GA-DS, GA-HO, GA-FL, GA-LA, GA-NM, GA-PD and GA-PO. The abbreviations have the following meanings:

GA:	Standard GA.
GA-BC:	GA with Bit Climbing Algorithm by Davis [1].
GA-CO:	GA with Complex Method of M.J.Box as implemented by Schwefel [4].
GA-DS:	GA with Davies, Swann and Campey Search with Gram-Schmidt orthogonalization as implemented by Schwefel [4].
GA-HO:	GA with Hooke and Jeeves Direct Search by Siddall [3].
GA-FL:	GA with Fletcher's 1972 method by Siddall [3].
GA-LA:	GA with Repeated Lagrangian Interpolation as implemented by Schwefel [4].
GA-NM:	GA with Simplex Method by Nelder and Meade [4].
GA-PD:	GA with Powell's Direct Search Method as produced by AERE Harwell [3].
GA-PO:	GA with Powell's Direct Search Method as implemented by Schwefel [4].

Three commonly used benchmark test problems already extensively discussed in the literature are used here to study the proposed Meta-Lamarckian learning in comparison with the conventional approaches. These consist of the thirty-dimensional Sphere (n=30), ten-dimensional Griewank (n=10) and twenty-dimensional Bump (n=20), representing classes of general constrained, unimodal and multimodal benchmark functions summarized in Table 3.1 [72, 55, 73]. For further details of the benchmark test function used here, the reader is referred to Appendix B.

§3.3.1 Effects of Local Method Choice on Search Performances

To see how the choice of the local search method employed effects the efficiency of problem searches, the nine different local search methods were used to form fixed GA-LS hybrids and used to search the benchmark problems. The averaged convergence trends obtained for the test problems as a function of the total number of function evaluations are shown in Figures 3.4 - 3.6.

All results presented are averages over twenty independent runs. Each run continues until the global optimum was found or a maximum of 40,000 trials (function evaluation calls) was reached, except for the Bump function where a maximum of up to 100,000 trials was used. The Bump constrained problem is a very hard problem and therefore requires greater effort. In each run, the control parameters for the hybrid GA-LS used in solving the benchmark problems were set as follows: Population size of 50, Mutation rate of 0.1%, 2-point Crossover with a rate of 60%, 10 bit binary encoding, and Maximum Local Search Length of 100 evaluations.

From the results of Figures 3.4 - 3.6, the effect of local method choice on the efficiency of GA-LS hybrids is clearly significant. For instance, GA-FL is seen to perform best on the Sphere function but very poorly on both Griewank and Bump. On the other hand, the majority of the nine fixed GA-LS(s) combinations do not show any improvement over the standard GA on the difficult Bump problem, with most having search capabilities closer to the least efficient GA-LS hybrid. In addition, the two different implementations of Powell's Direct Search included in the investigation illustrate that the capability of a given local search method may differ even among different implementations of the same basic algorithm. These characteristics make generalization in this field very difficult and also the a priori selection of a particular fixed GA-LS hybrid to fit an unseen or black-box optimization problem almost impossible.

The rationale behind the rapid optimum convergence of the GA-FL hybrid on the quadratic Sphere function can be explained by the quadratic convergence phenomenon. FL is a quasi-Newtonian method. It is a powerful optimization procedure that uses the Broyden, Fletcher, Shannon formula for updating the approximation of the Hessian matrix or the Davidon, Fletcher, Powell formula to approximate the inverse Hessian matrix. At some point along the FL search, on each successive iteration, a doubling (approximately) change in the Sphere function value occurs with each iteration. This doubling effect which is commonly referred to as quadratic convergence is responsible for the fast convergence, i.e., if ϵ^{ς} denotes the distance to the minimum at step ς, then quadratic convergence has the form $\epsilon^{\varsigma+1} \propto \epsilon^{2\varsigma}$.

Benchmark Test Functions	Range of x_i $[x_l, x_u]^n$	Characteristics				Function Minimum At
		Epi*1	Mul*2	Disc*3	Con*4	
$F_{Sphere} = \sum_{i=1}^{n} (x_i^2)$	$[-5.12, 5.12]^{30}$	none	none	none	no	0.0
$F_{Griewank} = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \left[\cos\left(x_i/\sqrt{i}\right)\right]$	$[-600, 600]^{10}$	weak	high	none	no	0.0
$F_{Bump} = \frac{abs\left(\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)\right)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}$ $\prod_{i=1}^{n} x_i > 0.75$ and $\sum_{i=1}^{n} x_i < 15n/2$	$[0, 10]^{20}$	high	high	none	Yes	Maximum at ~ 0.81

Table 3.1: Classes of General Constrained, Unimodal and Multimodal Benchmark test functions. *1: Epistasis, *2:Multimodality, *3:Discontinuity, *4:Constrained.

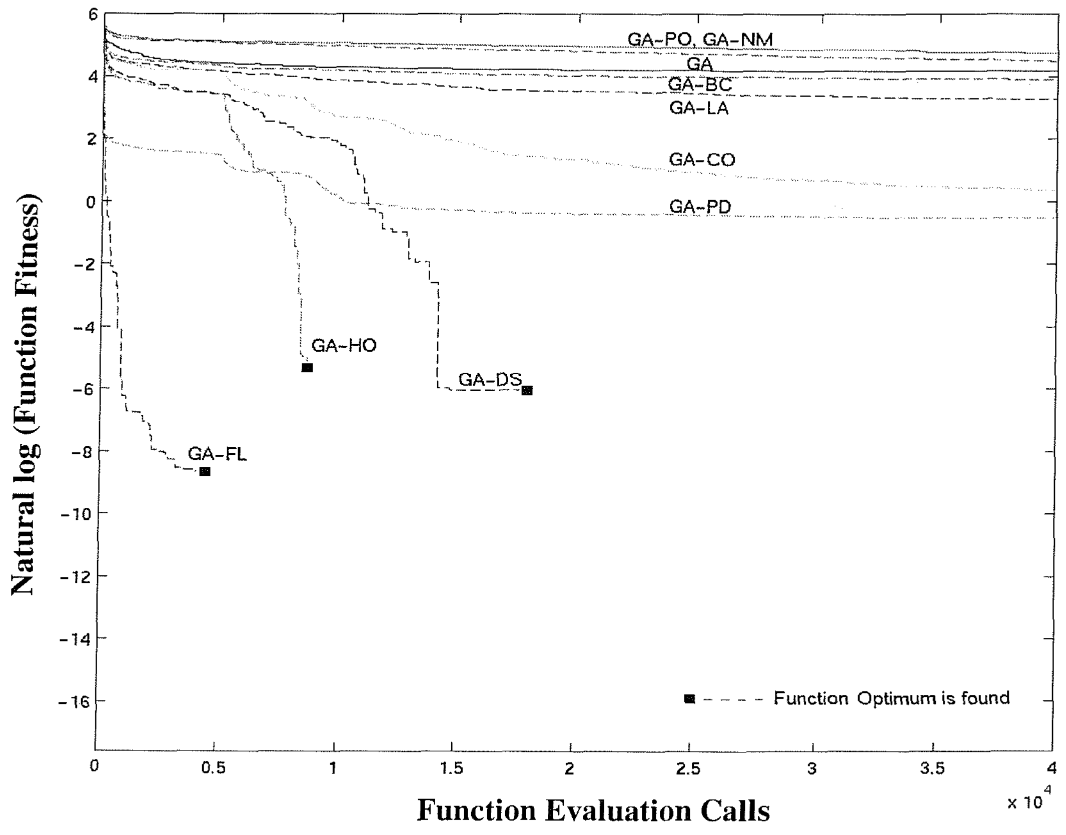


Figure 3.4: Average convergence trends for minimizing 30D Sphere function using GA and various GA-LS hybrids with Lamarckian Learning. Note that the search traces terminate at the global optimum of the Sphere function at 0.0 or $\ln(0.0) = -\infty$.

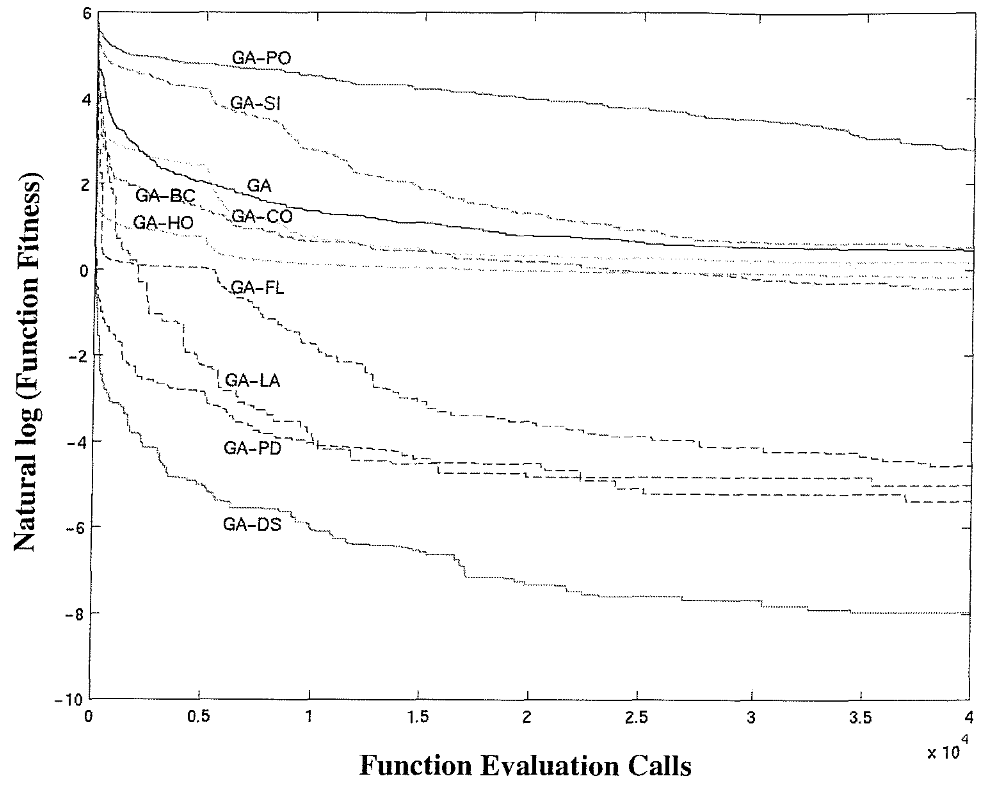


Figure 3.5: Average convergence trends for minimizing 10D Griewank function using GA and various GA-LS hybrids with Lamarckian Learning.

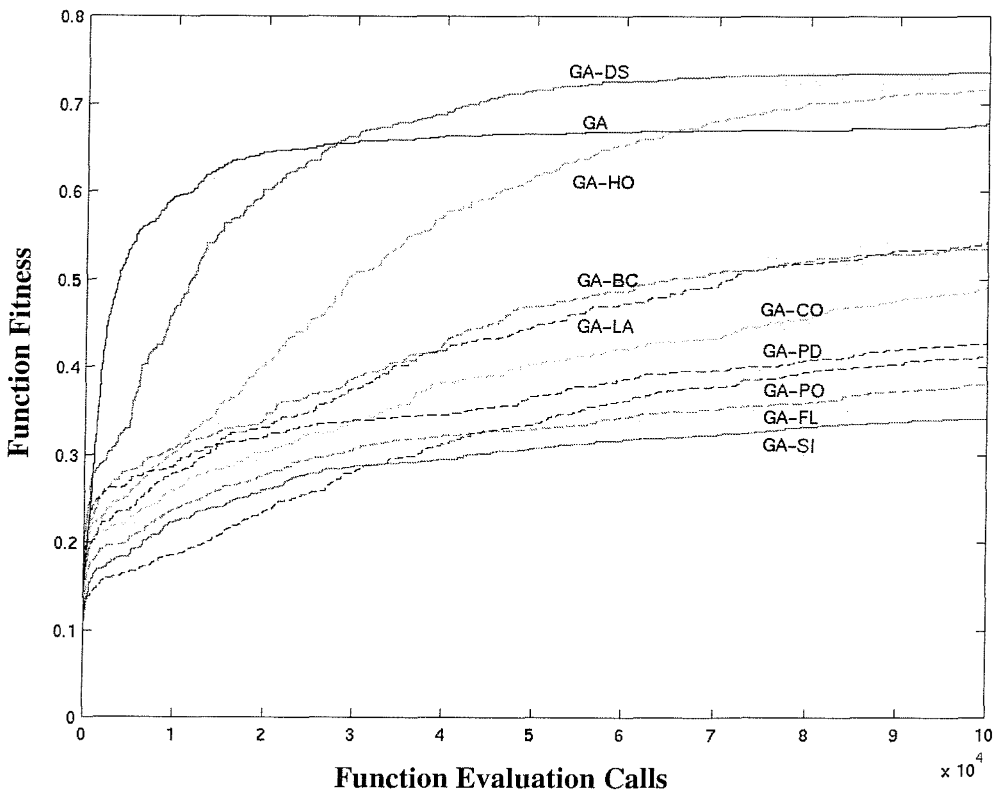


Figure 3.6: Average convergence trends for maximizing 20D Bump function using GA and various GA-LS hybrids with Lamarckian Learning.

§3.3.2 Results of Meta-Lamarckian Learning

To analyze the new approach proposed here, the three Meta-Lamarckian learning strategies discussed earlier were tested on the benchmark problems. In these studies, the control parameters, stopping criteria and other configurations were maintained at the same values as the previous experiments conducted in section 3.3.1. The averaged convergence trends obtained for the strategies are shown in Figures 3.7 - 3.9. Note, in all cases, results are plotted against the total number of function evaluations calls made by the combined GA and LS searches. The performance of the Meta-Lamarckian learning approach may be established by comparison with some of the fixed hybrids and also with GA-AV. GA-AV represents the estimated performance one might expect to get when the selection of a fixed LS in a hybrid GA is made randomly. This is assumed to be an average of the previous search traces in Figures 3.4 - 3.6 for the entire pool of nine fixed LS hybrids on each problem and is obtained from: $\text{GA-AV}_j = \frac{\sum_{i=1}^{L}(fitness_j^i)}{L}$, where L is the total number of LSs in the experimental studies. $fitness_j^i$ is the objective function fitness obtained from the fixed LS hybrid, GA-LS_i, at function evaluation call/count j.

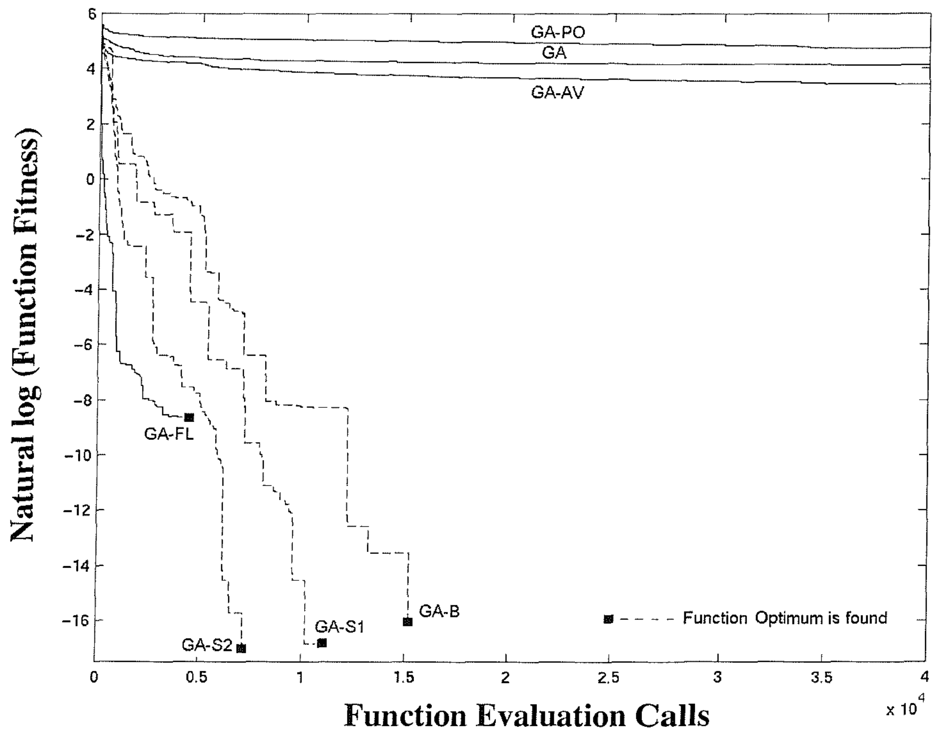


Figure 3.7: Average convergence trends for minimizing 30D Sphere function using strategies GA-B, GA-S1 and GA-S2 compared with standard GA and GA-AV. Also shown are the search traces for the GA-PO and GA-FL fixed hybrids.

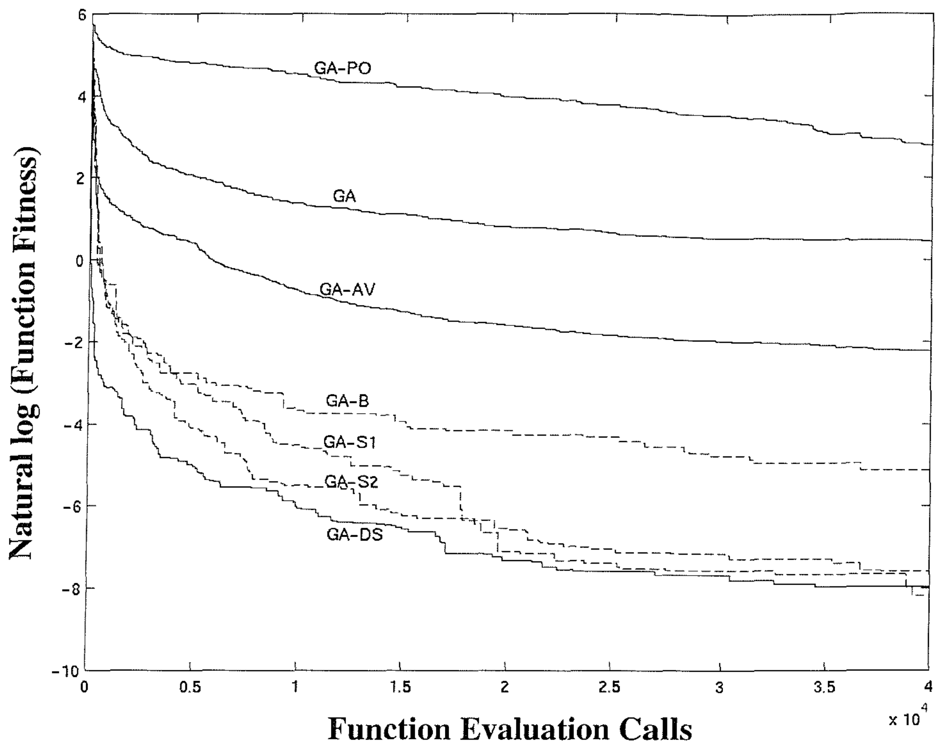


Figure 3.8: Average convergence trends for minimizing 10D Griewank function using strategies GA-B, GA-S1 and GA-S2 compared with standard GA and GA-AV. Also shown are the search traces for the GA-PO and GA-DS fixed hybrids.

From Figures 3.7 - 3.9, it is notable that although the basic Meta-Lamarckian learning scheme, represented by search trace GA-B, performs generally better than GA-AV, it still performs poorly when compared to the best LS hybrid on each problem, i.e., GA-FL for Sphere and GA-DS for Griewank and Bump. On the other hand, the adaptive strategies, GA-S1 and GA-S2, display search performances that are significantly better than GA, GA-AV and GA-B and also close to that of the best LS hybrid on each benchmark problem.

To develop a better understanding of the two adaptive strategies, GA-S1 and GA-S2, we further analyze them according to the following aspects:

- **Search Quality and Efficiency** - the capability of the strategy to provide high search quality and efficiency over different problems types.

- **Computational Cost** - the additional cpu time incurred over and above tradi-

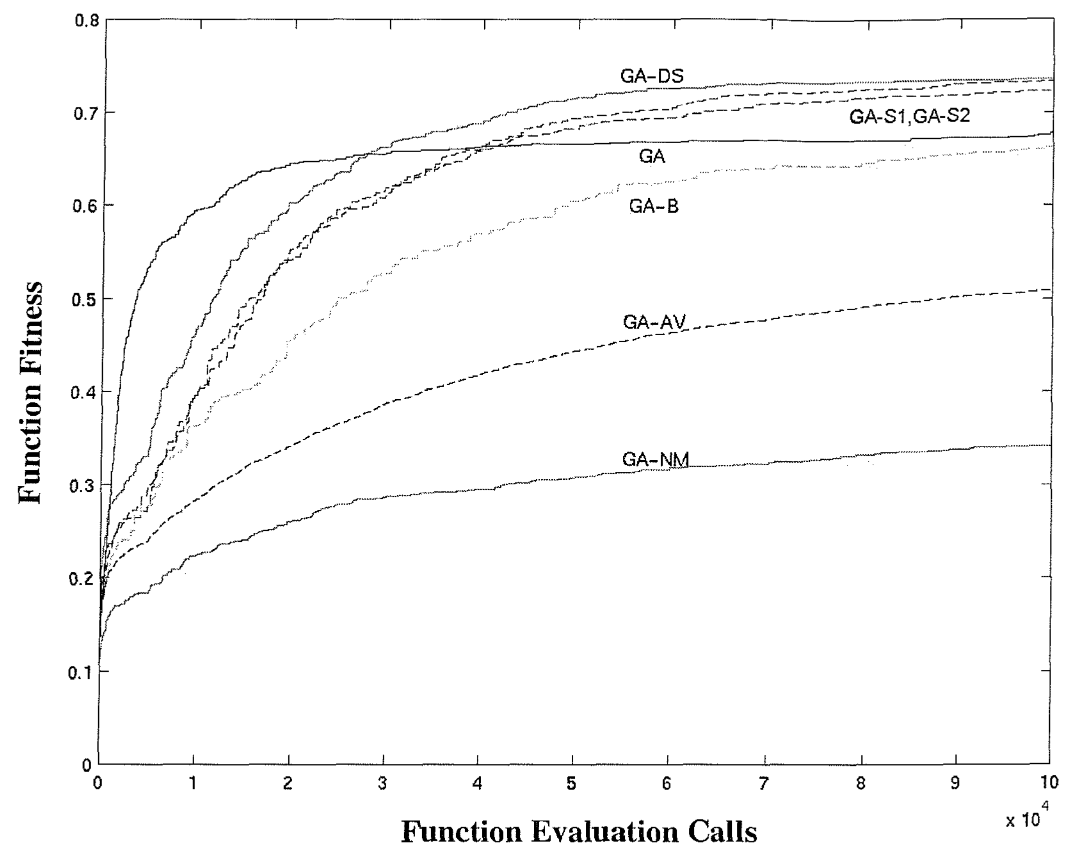


Figure 3.9: Average convergence trends for maximizing 20D Bump function using strategies GA-B, GA-S1 and GA-S2 compared with standard GA and GA-AV. Also shown are the search traces for the GA-NM and GA-DS fixed hybrids.

tional Lamarckian learning in fixed hybrid GA-LS.

- **Robustness** - the capability of the strategy to generate search performances that are close to the best LS hybrid (from among the pool tested), on different problems.
- **Simplicity** - ease of implementation. Simple strategies should require minimum effort to develop as well as a minimum numbers of additional control parameters that need to be managed.

(a) **Search Quality and Efficiency**

The search quality and efficiency performance of GA-S1 is dependent on the initial period, g, allocated for learning before the mechanism of sub-problem decomposition steps in. It is also strongly dependent on the nearest neighbour parameter, k, which defines the candidate LSs that will compete for selection. Shown in Table 3.2a, the mean search quality (i.e., function fitness value) and associated standard deviations for different values of k applied to the benchmark problems with g, set as 1 generation are shown. These include having k, set as 18 (i.e., twice the LS pool size, L of 9), $3L$, $4L$, $6L$ and $20L$. The effect of g on the convergence rate is next shown in Table 3.2b. Extensive studies on the variations of the g and k control parameters have been made; however, no fixed values are always found to generate the best search quality on the three test problems. Nevertheless, the search quality of GA-S1 is found to be generally more effective if parameters g and k are set within the ranges $1 \leq g \leq 3$ with $2L \leq k \leq 4L$, respectively. It may be seen in Table 3.2 that the search quality deteriorates when k and g gets larger as the sub-problem decomposition mechanism is gradually lost.

In contrast to GA-S1, the biased roulette wheel approach GA-S2 has no extra parameters to set. Its search performance on the three test functions are as shown in Table 3.2c. To determine if the search performances of the two Meta-Lamarckian strategies differ from each other in a significant way, the two-sample t test was employed. Using two-sample t test at a confidence levels of $\alpha = 0.05$ *or* 0.10, there is insufficient evidence to indicate that the mean search qualities of GA-S1 and GA-S2 (i.e., at specified stopping criteria) are significantly different when the control parameters of GA-S1 are maintained within the recommended range.

	Sphere Function (Minimum)	**Griewank Function (Minimum)**		**Bump Function (Maximum)**	
	Global Optimum found at Eval. Count of	Mean at 40,000	Standard Deviation	Mean at 100,000	Standard Deviation
GA-S1$_{k=2L}$	9833	5.08	5.58	73.8	5.00
GA-S1$_{k=3L}$	10177	9.07	22.9	72.1	5.93
GA-S1$_{k=4L}$	9997	8.56	10.2	72.7	7.31
GA-S1$_{k=6L}$	10833	15.4	39.3	69.3	8.89
GA-S1$_{k=20L}$	20133	37.3	66.1	66.9	11.7
Standard GA	Mean at 40,000 = 63.7; Standard Deviation =8.19	15700	2400	66.7	7.33

(a)

	Sphere Function (Minimum)	**Griewank Function (Minimum)**		**Bump Function (Maximum)**	
	Global Optimum found at Eval. Count of	Mean at 40,000	Standard Deviation	Mean at 100,000	Standard Deviation
GA-S1$_{g=1}$	10177	9.07	22.9	72.1	5.93
GA-S1$_{g=3}$	12297	55.3	21.8	72.2	6.46
GA-S1$_{g=8}$	23756	109.2	107.6	67.8	9.29

(b)

	Sphere Function (Minimum)	**Griewank Function (Minimum)**		**Bump Function (Maximum)**	
	Global Optimum found at Eval. Count of	Mean at 40,000	Standard Deviation	Mean at 100,000	Standard Deviation
GA-S2	7198	2.80	8.28	73.4	2.22

(c)

Table 3.2: (a) Effects of changes in k on Search Quality for Sub-problem Decomposition Strategy, GA-S1. (b) Effects of changes in g on Search Quality for Sub-problem Decomposition Strategy, GA-S1. (c) Mean and Standard Deviation of Biased Roulette Wheel Strategy, GA- S2 on the benchmark problems.

Griewank Function fitnesses are multiplied by 10^{04}, Bump Function fitnesses are multiplied by 10^{02}. Results are taken over 20 runs.

(b) **Computational Cost**

Of course, when searching across a domain where the algorithm function evaluations are expensive, such as aerodynamic wing design, all these adaptive strategies have negligible additional cost. For GA-S1 and GA-S2, the total computational costs incurred are of the order of $O(c^2 t_d + ct_e)$ and $O(ct_e)$, respectively, where c is the number of chromosomes evaluated so far or LS selections made, t_d is the time taken to perform the Euclidean distance measure between any two chromosomes and t_e is the time required to evaluate equation 3.1 and choose a LS from among the pool.

On a Pentium III processor, both t_e and t_d (for a 20D problem) are found to be around 2 *msec* and 7 *msec*, respectively. For a hybrid GA-LS search with stopping criteria of 100,000 maximum function evaluation calls, c is approximately 1400. The total time incurred by strategy GA-S1 and GA-S2 over traditional GA-LS search with Lamarckian learning at ($14002 * 7\ ms + 1400 * 2\ ms \approx 14\ seconds$) and ($1400 * 2\ ms = 2.8\ milliseconds$), respectively are negligible. Nonetheless, between the two strategies presented, GA-S1 is most costly.

(c) **Robustness and Simplicity**

Adopting the proposed adaptive strategies in a hybrid GA-LS search improves the robustness of the search performance greatly: this is one of the primary goals in this study. Both strategies are able to select a LS that matches the problem throughout the search, thus producing convergence trends that are close to the best fixed GA-LS hybrid on the benchmark problems. However, the Biased Roulette Wheel strategy GA-S2, is generally simpler to implement than GA-S1. It also has no control parameters requiring management, unlike the g and k parameters of GA-S1.

Based on these performance metrics, the Biased Roulette Wheel strategy, GA-S2, is considered the more competitive hybrid GA-LS search, especially when computational cost and simplicity are the main issues. Further discussion is thus restricted to this method in the dissertation. However, before demonstrating Adaptive Meta-Lamarckian learning on a real-world engineering design problem, some further issues are addressed. In the next subsection, the experimental studies to investigate the effect of these concerns on the convergence trends are presented.

§3.3.3 Other Issues Considered

The success of Meta-Lamarckian learning in hybrid GA-LS design optimization also involves taking into account the issues

- What is the effect of LS pool size on design search performance?
- What local search methods should be included in the LS pool?
- Can human expert knowledge be incorporated into the Meta-Lamarckian learning approach proposed (for instance, by the choice of LS pool members)?

(a) **Effects of LS Pool Size**

The chances of obtaining robust or better design search performance from a hybrid GA-LS generally increase by using multiple LS during the search, especially when adaptive strategies are used to control the choice of LS. The effect of different sizes of LS pool (i.e., 2, 4, 9 and 25) on GA-S2 is presented for the Griewank benchmark problem in Table 3.3. From the table, it may be seen that the use of a smaller LS pool size is often associated with quicker improvements during earlier stages, as less evaluations are needed to acquire sufficient knowledge about the LS before the learning strategy begins to take effect. So, although it is advantageous to include a large pool of LS to maintain wide ranging robustness, one concern is the number of evaluations required before the LS decision space is sufficiently explored. From extensive studies conducted on a range of benchmark problems and LS methods, the GA-S2 strategy is found to remain generally effective even with a pool size of up to 25 different local search methods. However, a pool size of around 10 local search methods would be more easily accessible in a real world engineering design environment and is thus recommended.

LS Methods within the Pool	**Mean Search Fitness at Function Evaluation Count of**				
	1000	5000	10000	30000	40000
Pool Size = 2 DS, PO	536	59.3	23.9	6.74	5.34
Pool Size = 4 CO, DS, NM, PO	973	62.0	23.2	7.05	5.81
Pool Size = 9	2770	165	41.1	5.17	2.80
Pool Size = 25	9140	287	96.6	20.9	8.40
Best fixed LS hybrid GA-DS	443	65.3	23.1	4.63	3.45

Table 3.3: Effects of Changes in LS Pool Size on the minimization of the Griewank function, mean function fitnesses are multiplied by 10^{04}. When the pool size is 25, the other methods used were taken from the OPTIONS DES.

(b) Choice of Local Search Methods in the Pool

The choice of which local search methods to include within the pool is another issue that is considered critical, especially when one has no prior knowledge of the problem in hand. Nevertheless, as a rule of thumb, the recommended LS pool should contain both derivative based and non-derivative local methods. Typical derivative based LS include steepest descent, conjugate-gradient and quasi-Newton methods [3, 5]. As previously discussed, the FL quasi-Newton method, which possesses the ability to converge rapidly to the local optimum on quadratic functions was shown to contribute as the best fixed LS hybrid GA on the Sphere problem. For non-derivatives methods, direct search methods, linear approximation methods, and local evolutionary search are all good choices [4, 74, 75]. DS, Evolutionary Strategy (ES), Evolutionary Programming (EP) and Simulated Annealing (SA) are direct search or local evolutionary methods that makes a good combination with FL in the same pool as they can handle problems where derivative based LS may fare badly. Other considerations on choice of LS pool would include the abilities of the LS to handle general non-linear constrained problems.

(c) **Incorporation of Human Expert Knowledge**

The proposed Meta-Lamarckian learning approach also permits the incorporation of a designer's intuition, experience and knowledge during design activities. For example, assume that a design specialist has knowledge about the DS local search method, i.e., DS often performs relatively well on most design problems of interest to him/her. However, not being totally sure about its suitability for a new problem, he/she may not wish to commit to a fixed hybrid and instead incorporate this expert knowledge into the adaptive strategies by biasing the DS local method with a greater probability of being selected. Trace GA-S2A in Figure 3.10 illustrates the case where the DS method is biased with twice the chances of selection, as compared to the other local methods in the same pool. This combination clearly works well. The Domain Optimization Method Advisor described in Chapter 2 may be used to give such advice if an historical database of methods is available.

Shown also in Figure 3.10 is the trace GA-S2B where the designer chooses to use six local methods (PO, NM, CO, BC, PD and DS) as the pool to perform a search on the Griewank function. From these results, it is seen that superior search performances are obtained when human expert knowledge is incorporated into the Meta-Lamarckian learning process. Improvements can also be found for the other benchmark problems when knowledge is incorporated in this way. Every search algorithm, except for uniform random search, introduces some unique form of bias, suitable for some classes of problems but not for others. Therefore, any fixed GA-LS hybrid will include biases. Since a priori knowledge about problems is often scarce, this makes selection of the appropriate LS for use in such fixed schemes difficult. The great advantage of the proposed adaptive GA-LS approaches is that they are able to address this fundamental problem by allowing a range of local searches to cooperate and compete in finding good solutions. With the incorporation of human designer knowledge, further improvement of search performance may be attained.

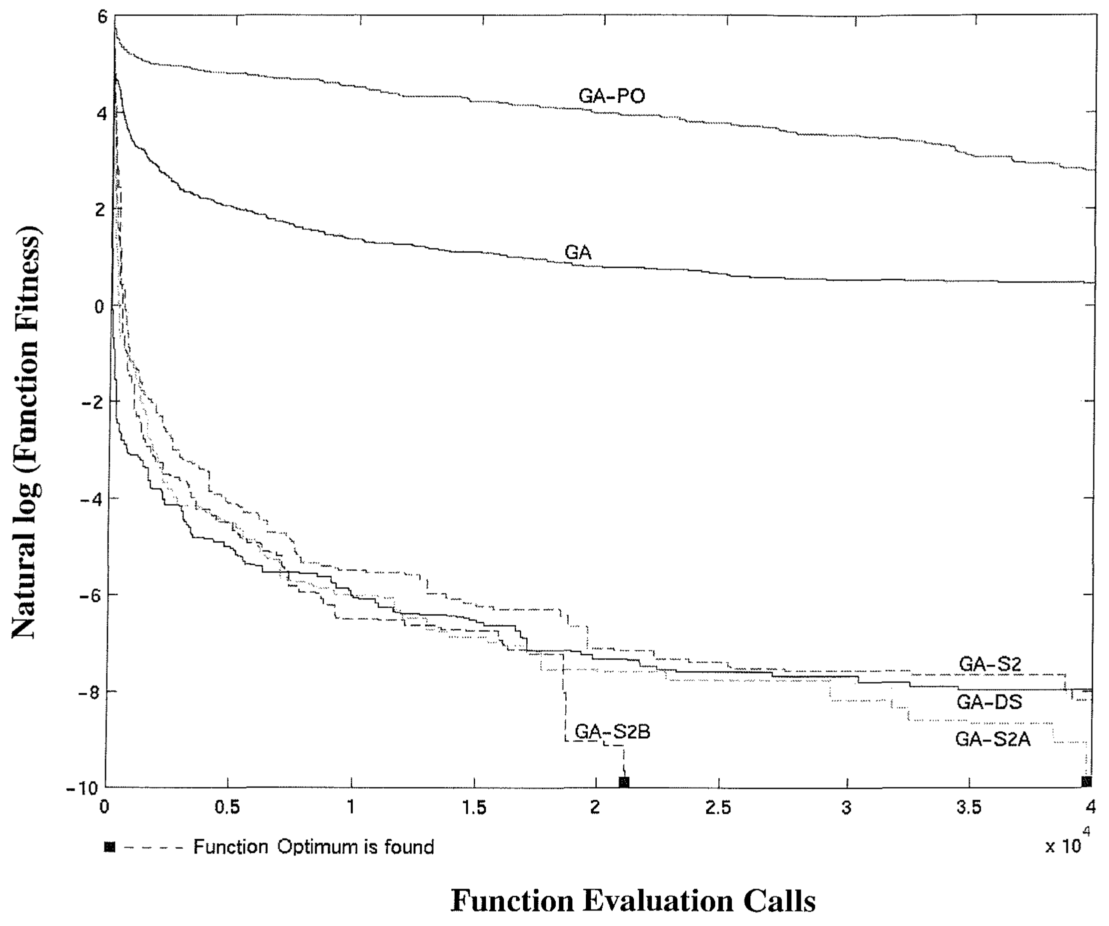


Figure 3.10: Average convergence trends for minimizing 10D Griewank function using GA-S2A and GA-S2B with the incorporation of Human Expert Knowledge compared with conventional GA and GA-S2. Also shown are the search traces GA-PO and GA-DS hybrids.

§3.4 Aerodynamic Aircraft Wing Design

In aerospace companies, design teams are often required to work on design problems that are new to them and accompanied by tight deadlines. To demonstrate the practicality of the Adaptive Meta-Lamarckian learning approach, strategy GA-S2 is next applied to a real world engineering design problem, the design of transonic civil transport aircraft wing considered in Chapter 2 [14]. The objective is again set as the minimization of wing drag D/q meters^2 as calculated by using TADPOLE, with target lift, wing weight, volume, pitch-up margin and root triangle layout chosen to be representative of a 220 seat wide body airliner. The reader is referred to Appendix C for further details of the aerodynamic wing design problem considered.

Here, the worst and best fixed hybrid GA-LS on this problem were found to be GA-BC and GA-CO, respectively (see Figure 3.11). In addition, both GA-FL and GA-DS fared very poorly compared to others in the LS pool of nine. Once again, GA-S2 was able to generate design search performance that is as good as the best fixed hybrid on this realistic problem (GA-CO). With the incorporation of human expert knowledge, superior design search performance may be attained, as shown by the GA-S2C search trace in Figure 3.11 where the CO local method is biased with greater probability of being selected. Such results encourage the use of multiple local methods, rather than relying simply on one fixed, and possibly poor choice. Adaptive Meta-Lamarckian learning clearly offers a high quality and robust approach for engineers working on engineering design problems, regardless of whether a priori knowledge about the problem is available or not.

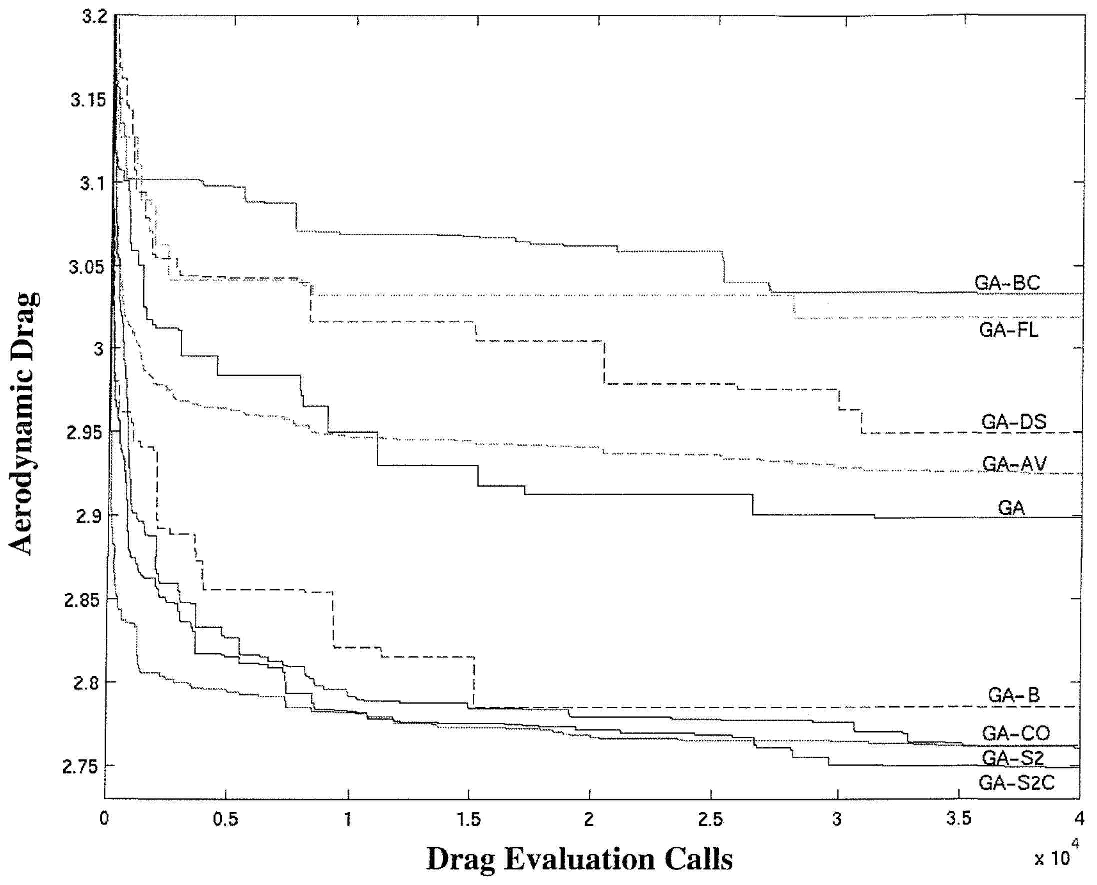


Figure 3.11: Design of a Transonic Civil Transport Aircraft Wing using Adaptive Meta-Lamarckian Learning hybrid GA-S2. GA-S2C differs from GA-S2 due to the incorporation of human knowledge (see text). Also shown are the search traces for GA, GA-BC, GA-FL, GA-DS, GA-CO, GA-AV and GA-B.

§3.5 Conclusion

The major impact of local optimization search routine choice in hybrid Genetic Algorithm-Local Searches has been discussed and illustrated in this Chapter. Meta-Lamarckian learning in hybrid GA-LS search has been proposed to mitigate the problem. Two new adaptive strategies for Meta-Lamarckian learning have been described and analyzed. Experimental studies on three representative classes of benchmark problems have shown that the strategies proposed are effective in reducing the influence of local method choice on design search performance. Overall, the Biased Roulette Wheel approach to method selection is the most promising adaptive strategy considered here. It is shown to be capable of reducing the influence of inappropriate choice of search methods on hybrid GA-LS searches, and also attaining robust, high quality and efficient performance on both benchmark problems as well as a real-world engineering design problem. Further, the approach reduces reliance on optimization domain experts, hence ensuring that design engineers in hybrid evolutionary design search require minimum knowledge of local optimization methods.

Chapter 4

Computationally Expensive Engineering Design Problems

IN Chapters 2 and 3, methodologies for reducing the influence of inappropriate choice of search methods on complex engineering design optimization performance were presented. As previously noted in Chapter 1, besides a high reliance on human expertise, present engineering design processes must also deal with the complexity of cost surfaces. For this reason, a study on handling computationally expensive analysis codes in engineering design optimization under limited computational resources has been conducted and the findings are presented in the Chapter.

Most studies in the literature have addressed this problem using frameworks that incorporate approximation models in design procedures using gradient based optimization algorithms, with rather less efforts placed on Evolutionary Search. In this Chapter, a novel parallel evolutionary optimization algorithm that leverages approximation models for solving computationally expensive design problems with general constraints, on a limited computational resource [27] is proposed and investigated. The essential backbone of the proposed framework is an evolutionary algorithm coupled with a feasible sequential quadratic programming solver in the spirit of Lamarckian learning. Further, a trust-region approach is employed for interleaving the use of exact models for the objective and constraint functions with computationally cheap surrogate models during local search. In contrast to earlier work, the local surrogate models are constructed from data chronicled online using radial basis functions in the spirit of transductive inference. In addition, the present approach retains the intrinsic parallelism of evolutionary algorithms and can hence be readily implemented on grid computing infrastructures. Experimental results are presented for some benchmark functions and on the aerodynamic wing design problem to demonstrate that the present algorithm converges to good designs on a limited computational resource.

§4.1 Introduction

Complex engineering design problems commonly have large design spaces. In such design spaces, typically thousands of function evaluations are required to locate a near optimal solution. Often in aerospace and ship detailed design processes, variable-fidelity analysis codes are employed to strike a balance between design cost, time and estimation accuracy. Nevertheless, in design optimization processes where high-fidelity analysis code are used, each function evaluation may require a finite element or computational fluid dynamics simulation costing hours of supercomputer time. Therefore, the overwhelming part of the total run time in complex engineering design optimization is usually taken up by the evaluations of the analysis code. This often poses a serious impediment to the practical application of optimization to complex engineering design problems.

To reduce the cost of each function evaluation, surrogate models, metamodels or approximation models, which are often described as a "model of the model", are often constructed from historical design optimization data and then used in lieu of the actual expensive analysis models. Hence, for many large–scale design problems where computationally expensive high–fidelity models are used for predicting design improvements, a popular and widely followed practice for optimal design is to make use of a gradient based optimization module linked to an approximate analysis routine, which is continuously updated at each design cycle based on the results of exact model analysis. This practice leads to a computationally efficient search procedure, and hence, the solution of expensive large design space problems is made possible in a tractable amount of time. Since line search procedures are utilized in gradient based optimization algorithms, the issue of range of validity of the approximation models or the control of approximation errors can be directly addressed by using *ad hoc* move limits or a trust region framework in the line searches. Similarly, frameworks for approximation model management based on metamodelling strategies for pattern search algorithms have also been proposed in the literature [76].

A variety of approximation model management frameworks for gradient based optimization algorithms exist in the literature. However, a detailed analysis of the literature reveals that few studies have addressed the issue of incorporating approximation models in design procedures using Evolutionary Search (ES). The reason being that since ESs make use of probabilistic recombination operators, controlling the step size of design changes (to control the accuracy of approximate fitness predictions) is not straightforward as it is in gradient based optimization algorithms. This difficulty becomes particularly severe when local approximation models are employed during search. In principle, global models can be employed to circumvent this problem. However, in practice, due to the *curse of dimensionality*, such models become increasingly difficult to construct for problems with large number of variables.

In order to successfully apply evolutionary algorithms (EAs) to engineering design problems with high-fidelity simulation models, further research for new approaches to handle the problem is necessary. Most of the research work related to ES has mostly involved the use of problem specific knowledge to increase computational efficiency [77, 78, 79, 80]. Other attempts have been described in Chapters 2 and 3. Although it has been shown that such approaches can be effectively used to achieve performance improvements, there are finite limits to the improvements achievable when dealing with computationally expensive problems. The history of theoretical developments and applications of gradient based optimization techniques to design indicates that the most influential factor for their wide spread use has been the ease with which approximation models can be incorporated to achieve substantial savings in the computational cost. The development of faster and more efficient optimization algorithms alone would not have sufficed to make this possible. Taking this cue from the evolution of classical design optimization procedures, the question of how to integrate approximation models with evolutionary search procedures needs to be addressed in order to study their practical applicability for design problems, where computational cost is often a critical issue. This requirement for studies focusing on the extent to which approximation concepts can be combined with stochastic evolutionary search was also noted in a recent survey of the state of the art in multidisciplinary design optimization methodology by Sobieszczanski-Sobieski and Haftka [81].

Robinson and Keane presented a case for employing variable-fidelity analysis models and approximation techniques to improve the efficiency of evolutionary optimization for complex design tasks, for instance aeronautical design [79]. A computational framework for integrating a class of single-point approximation models with Genetic Algorithm was also proposed by Nair and Keane [80]. However, such frameworks are restricted to a special class of approximation models that are domain specific. For more general approximation models, Ratle [82] examined a strategy for integrating evolutionary search with Kriging models. This problem was revisited by El-Beltagy et al. [83] where it is argued that the issue of balancing the concerns of optimization with that of design of experiments should be addressed. Numerical studies were presented for certain pathological cases to show that the idea of constructing an accurate global surrogate model may be fundamentally flawed due to the curse of dimensionality. Liang et al. [84] attempted to couple hybrid evolutionary search (i.e., Evolutionary Search + Local search) instead of traditional ES with quadratic response surface methods. The basic idea of Liang is to increase the exploitation factor of traditional ES via the use of local search. It does so by first mapping the approximated solution to its exact solution, which is then followed by concentrating search efforts around the neighbourhoods of the mapped exact solution to produce better solutions using the exact analysis codes. However the use of the computationally expensive exact analysis codes to perform local searches is inefficient. Moreover by discarding the interacting term between design variables of the standard quadratic polynomial approximation model when working with high dimensional and multi-modal problems, the accuracy of the quadratic model on realistic problems may become questionable. A framework for coupling EAs and neural network-based surrogate models is also presented by Jin et al. [85]. This approach uses both the expensive and approximate models throughout the search, with an empirical criterion to decide the frequency at which each model should be evaluated.

In spite of extensive work on this topic, existing strategies for integrating approximation models with EAs have met with limited success in applications to real-world problems. Some of the key factors responsible for this are:

- The curse of dimensionality that causes significant difficulties in constructing a global surrogate model which is capable of accurately predicting fitness improvements during the search.
- The inability of most frameworks to handle problems with general nonlinear inequality and equality constraints.
- Most of the proposed strategies for managing the interplay between the exact and approximate models tend to compromise on the intrinsic parallelism of traditional EAs.

Based on the survey conducted, a general framework or algorithm for integrating surrogate models with EAs, which addresses the limitations of existing strategies outlined above is proposed. The proposed algorithm leverages well-established notions in the literature on hybrid evolutionary optimization techniques, radial basis functions, and trust-region frameworks. The essential backbone of the present approach is an EA hybridized with a feasible sequential quadratic programming solver. Each individual in an EA generation is used as an initial guess for local search in the spirit of Lamarckian learning. It uses a trust-region framework to manage the interplay between the original objective and constraint functions and computationally cheap surrogate models during local search, thereby maintaining the accuracy of the surrogate model.

Furthermore, the idea of employing local surrogate models that are constructed using data points that lie in the vicinity of an initial guess is proposed. This idea of constructing local models is similar in spirit to the multipoint approximation technique proposed by Toropov et al. [86]. Here, the local learning technique is viewed and discussed as an instance of the transductive inference paradigm, which has been the focus of recent research in statistical learning theory. Traditionally, global surrogate models are constructed using inductive inference, which involves using a training dataset to estimate a functional dependency and then using the computed model to predict the outputs at the points of interest. However, when constructing surrogate models for optimization, the specific interest is to ensure that the models predict the objective and constraint function values accurately at the sequence of iterations generated during the search - how well the model performs at other points in the parameter space is of no concern in this specific context. Transductive inference offers a solution to this problem by directly estimating the functional dependency and outputs at the point of interest; the reader is referred to [87, 88] and Vapnik's text [89] (see in particular Chapter 8) for a detailed theoretical analysis of its superior generalization capabilities over standard inductive inference.

In the present work, transduction is implemented by constructing radial basis networks using data points in the local neighbourhood of an optimization iteration. In other words, instead of constructing a global surrogate model, a local model is created on the fly whenever the objective and constraint functions must be estimated at a design point during local search. The localized training data can be readily selected from an online historical database containing previous iterations, which is continuously updated as the search progresses. Further, the proposed algorithm is shown to be readily and efficiently parallelized on grid computing architectures. Extensive numerical studies are presented for some benchmark test functions and the real world aerodynamic wing design problem. It is shown that the present framework allows for the possibility of converging to good designs on a limited computational resource.

This Chapter is organized as follows: Section 4.2 outlines surrogate model construction using radial basis functions, and section 4.3 presents the proposed algorithm for integrating local surrogate models and trust-region methods with EAs. The grid infrastructure employed to achieve parallelism is also briefly discussed. In sections 4.4 and 4.5, the experimental studies on some benchmark test functions and application of the proposed framework for aerodynamic wing design are presented. Finally, section 4.6 summarizes the main conclusions.

§4.2 Surrogate Modeling

Surrogate models or metamodels are (statistical) models that are built to approximate computationally expensive simulation codes. Surrogate models are orders of magnitude cheaper to run, and can be used in lieu of exact analysis during evolutionary search. Further, the surrogate model may also yield insights into the functional relationship between the input $\mathbf{x}$ and the output y. If the true nature of a computer analysis code is represented as

$$y = f(\mathbf{x}), \tag{4.1}$$

then a surrogate model is an approximation of the form

$$\hat{y} = \hat{f}(\mathbf{x}), \tag{4.2}$$

such that $y = \hat{y} + \epsilon$, where ϵ represents the approximation error.

There exist a variety of techniques for constructing surrogate models; see, for example, the texts by Vapnik [89] and Bishop [90] for an exposition of learning theory. One popular approach in the design optimization literature is least-squares regression using low-order polynomials, often known as response surface methods. A statistically sound alternative for constructing surrogate models of a deterministic computer model is Kriging, which is sometimes referred to as a form of design and analysis of computer experiments (DACE) model in the statistics literature [91] and Gaussian process regression in the neural network literature [92]. A comparison of some surrogate modeling techniques has been presented by Giunta and Watson [93] and Jin et al [94].

As mentioned earlier, in the present study it is proposed to use local surrogate models in the spirit of transductive inference. In particular, a surrogate model is built on the fly when the objective and constraint functions at an optimization iteration are to be estimated. This local model is built using only a small set of data points that lie in the local neighbourhood of the design point of interest. Since surrogate models will probably be built thousands of times during the search in this fashion, computational efficiency is a major concern. This consideration motivates the use of radial basis function networks, which can be efficiently applied to approximate multiple-input multiple-output data, particularly when a few hundred data points are used for training.

Let $\{\mathbf{x}_i, y_i, i = 1, 2, \ldots, n\}$ denote the training dataset, where $\mathbf{x} \in R^d$ is the input vector and $y \in R$ is the output. Since interest lies in cases where the training data is generated by running deterministic computer models, we assume the data to be noise-free. Hence, the focus is on interpolating radial basis function models of the form

$$\hat{y} = \sum_{i=1}^{n} \alpha_i K(||\mathbf{x} - \mathbf{x}_i||), \tag{4.3}$$

where $K(||\mathbf{x} - \mathbf{x}_i||) : R^d \to R$ is a radial basis kernel and $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \in R^n$ denotes the vector of weights.

Typical choices for the kernel include linear splines, cubic splines, multiquadrics, thin-plate splines, and Gaussian functions [90]. The structure of some commonly used radial basis kernels and their parameterizations are shown in Table 4.1.

Nomenclature	$k_i(x, \beta)$
Linear Splines	$\|x - c_i\|$
Thin Plate Splines	$\|x - c_i\| \ln \|x - c_i\|$
Cubic Splines	$\|x - c_i\|^3$
Gaussian	$e^{-\frac{\|x - c_i\|^2}{\beta_i}}$
Multiquadrics	$\sqrt{1 + \frac{\|x - c_i\|^2}{\beta^i}}$

Table 4.1: Radial Basis Kernels.

Given a suitable kernel, the weight vector can be computed by solving the linear algebraic system of equations $\mathbf{K}\boldsymbol{\alpha} = \mathbf{y}$, where $\mathbf{y} = \{y_1, y_2, \dots, y_n\} \in R^n$ denotes the vector of outputs and $\mathbf{K} \in R^{n\times n}$ denotes the Gram matrix formed using the training inputs (i.e., the ijth element of $\mathbf{K}$ is computed as $K(||\mathbf{x}_i - \mathbf{x}_j||)$).

Micchelli [95] proved that non-singularity of the Gram matrix $\mathbf{K}$ can be theoretically guaranteed for a class of kernels only when the set of input vectors in the training dataset are distinct. In many papers in the radial basis function literature, a polynomial term P is often appended to equation 4.3 along with some constraints. In other words, if K is a conditionally positive definite function of order q, then to ensure a unique solution for the weight vector, equation 4.3 is rewritten as

$$\hat{y} = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + P_q(\mathbf{x}), \tag{4.4}$$

where P_q is a polynomial of order $q-1$ and the following homogeneous constraint equations are imposed

$$\sum_{i=1}^{n} \alpha_i P_k(\mathbf{x}_i) = 0, \quad 1 \leq k \leq q \tag{4.5}$$

Then the weight vector can be computed by solving a linear algebraic system of equations of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} \mathbf{K} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix}, \mathbf{x} = \{\boldsymbol{\alpha}, \boldsymbol{\theta}\}^T, \text{ and } \mathbf{b} = \{\mathbf{y}, \mathbf{0}\}^T, \tag{4.6}$$

where $\mathbf{P}$ is a matrix which arises by substituting the input vectors in the training dataset into the polynomial term P, and $\boldsymbol{\theta}$ is a vector composed of the undetermined coefficients of P. In practice, good approximations can be obtained by using a constant instead of a full-order polynomial. Here, the coefficient matrix $\mathbf{A}$ becomes

$$\mathbf{A} = \begin{bmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \in R^{(n+1)\times(n+1)}, \tag{4.7}$$

where $\mathbf{1} \in R^n$ is a vector of ones.

For problems with multiple outputs, the weight vector can be efficiently computed for all the outputs of interest once the matrix $\mathbf{K}$ is decomposed. For a typical dataset with 500 training points, 20 inputs, and five outputs, surrogate model construction using linear splines generally takes a fraction of a second on one processor of an SGI Power Challenge. When dealing with computationally expensive problems that cost more than a few minutes of cpu time per function evaluation, this training cost is generally negligible.

In the present study, we used linear splines to construct surrogate models since experimental studies in the literature [94] suggest that this kernel is capable of providing models with good generalization capability at a low computational cost. Next, an algorithm that integrates a local version of such surrogates in hybrid evolutionary search is presented.

§4.3 Present Framework

In this section, the essential ingredients of the proposed local surrogate modeling algorithm for parallel evolutionary optimization of computationally expensive problems are presented. Consider a general nonlinear programming problem of the form:

$$\begin{aligned} \textbf{Minimize}: \quad & f(\mathbf{x}) \\ \textbf{Subject to}: \quad & g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, p \\ & \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u \end{aligned} \tag{4.8}$$

where $\mathbf{x} \in R^d$ is the vector of design variables, and $\mathbf{x}_l$ and $\mathbf{x}_u$ are vectors of lower and upper bounds, respectively.

Here, we are interested in cases where the evaluation of $f(\mathbf{x})$ and $g(\mathbf{x})$ is computationally expensive, and it desired to obtain a near optimal solution on a limited computational resource. The basic steps of the proposed algorithm are outlined in Figure 4.1.

```
BEGIN
    Initialize: Generate a database containing a population of designs.
    (Optional: Upload a historical database if one exists, i.e., from Chapter 2 and 3.)
    While (computational resource not exhausted)
        Evaluate all individuals in the population using the exact models (if solutions are not available).
        For each non-duplicated individual in the population
            * Apply trust-region enabled feasible SQP solver to each individual in
              the population by interleaving the exact and local surrogate models
              for the objective and constraint functions.
            * Update the database with any new design points generated during
              the trust-region iterations and their exact objective and constraint
              function values.
            * Replace the individuals in the population with the locally improved
              solution in the spirit of Lamarckian learning.
        End For
        Apply standard EA operators to create a new population.
    End While
END
```

Figure 4.1: Proposed algorithm for integrating local surrogate models with hybrid EAs for optimization of computationally expensive problems.

In the first step, a database using a population of designs, either randomly or using design of experiments techniques such as Latin hypercube sampling [38] is initialized. All the design points thus generated and the associated exact values of the objective and constraint functions are archived in the database that will be used later for constructing local surrogate models. Alternatively, one could use a database containing the results of a previous search on the problem or domain, such as that in Chapters 2 and 3 or a combination of them.

Subsequently, a hybrid EA is employed, where for each non-duplicated design point or chromosome in the population, a local search is conducted using surrogates. The local strategy used here embeds the feasible SQP optimizer within a trust-region framework [96, 97]. A rationale behind using a feasible SQP solver is to exploit its well-known ability to efficiently locate the local optima of optimization problems *with general constraints.* Besides, since local surrogate models are in use, there is little difficulty in computing the derivatives (gradient or Hessian), making a derivative-based method such as the SQP solver an excellent choice.

Further, instead of adopting an augmented Lagrangian approach, we handle the objective and constraint functions separately in the spirit of Giunta and Eldred [98]. More specifically, during local search for each chromosome in an EA generation, a sequence of trust-region subproblems of the following form is solved.

$$\begin{aligned} \textbf{Minimize}: &\quad \hat{f}^k(\mathrm{x} + \mathrm{x}_c^k) \\ \textbf{Subject to}: &\quad \hat{g}_i^k(\mathrm{x} + \mathrm{x}_c^k) \le 0, i = 1, 2, \ldots, p \\ &\quad ||\mathrm{x}|| \le \Delta^k \end{aligned} \tag{4.9}$$

where $k = 0, 1, 2, \ldots, k_{max}$, x_c^k and Δ^k are the initial guess and the trust region radius used for local search at iteration k, respectively. In practice, the L_∞ norm can be employed to impose the second constraint in equation 4.9. Hence, this constraint can be transformed into appropriate bounds on the design variables, which are updated at each trust-region iteration based on the value of Δ^k.

For each subproblem (or during each trust-region iteration), surrogate models of the objective and constraint functions, viz. $\hat{f}^k(\mathbf{x})$ and $\hat{g}_i^k(\mathbf{x})$, are created dynamically. The m nearest neighbours of the initial guess, $\mathbf{x}_c^k$, are first extracted from the archived database of design points evaluated so far using the exact analysis code. The criterion used to determine the similarity between design points is the simple Euclidean distance metric. These points are then used to construct local surrogate models of the objective and constraint functions. It is worth noting here that care has to be taken to ensure that repetitions do not occur in the training dataset, since this may lead to a singular Gram matrix $\mathbf{K}$.

The surrogate models thus created are used to facilitate the necessary objective and constraint function estimations in the local searches. After each iteration, the trust region radius Δ^k is updated based on a measure which indicates the accuracy of the surrogate model at the kth local optimum, $\mathbf{x}_{lo}^k$. After computing the exact values of the objective and constraint functions at this point, the figure of merit, ρ^k, is calculated as

$$\rho^k = \min(\rho_f^k, \rho_{g_i}^k), \text{ for } i = 1, 2, \ldots, p \tag{4.10}$$

where

$$\rho_f^k = \frac{f(\mathbf{x}_c^k) - f(\mathbf{x}_{lo}^k)}{\hat{f}(\mathbf{x}_c^k) - \hat{f}(\mathbf{x}_{lo}^k)} \text{ and } \rho_{g_i}^k = \frac{g_i(\mathbf{x}_c^k) - g_i(\mathbf{x}_{lo}^k)}{\hat{g}_i(\mathbf{x}_c^k) - \hat{g}_i(\mathbf{x}_{lo}^k)} \tag{4.11}$$

The above equations provide a measure of the actual versus predicted change in the objective and constraint function values at the kth local optimum. The value of ρ^k is then used to update the trust region radius as follows [96]:

$$\begin{aligned} \Delta^{k+1} &= 0.25\Delta^k, && \text{if } \rho^k \leq 0.25, \\ &= \Delta^k, && \text{if } 0.25 < \rho^k \leq 0.75, \\ &= \xi\Delta^k, && \text{if } \rho^k \geq 0.75, \end{aligned} \tag{4.12}$$

where $\xi = 2$, if $||\mathbf{x}_{lo}^k - \mathbf{x}_c^k||_\infty = \Delta^k$ or $\xi = 1$, if $||\mathbf{x}_{lo}^k - \mathbf{x}_c^k||_\infty < \Delta^k$.

The trust region radius, Δ^k, is reduced if the accuracy of the surrogate, measured by ρ^k is low. Δ^k is doubled if the surrogate is found to be accurate and the kth local optimum, $\mathbf{x}_{lo}^k$, lies on the trust region bounds. Otherwise the trust region radius remains unchanged.

The exact values of the objective and constraint functions at the best solution of the kth subproblem are combined with the m nearest neighbouring design points to generate a new surrogate model for the next iteration. In addition, the initial guess for the $k+1$th iteration within each local search is determined by

$$\begin{aligned} \mathbf{x}_c^{k+1} &= \mathbf{x}_{lo}^k, \quad \text{if } \rho^k > 0 \\ &= \mathbf{x}_c^k, \quad \text{if } \rho^k \leq 0. \end{aligned} \tag{4.13}$$

The trust region iterations (for each chromosome) are terminated when $k \geq k_{max}$, where k_{max} is the maximum number of trust-region iterations that is set *a priori* by the user. At the end of k_{max} trust-region iterations for a chromosome, the exact fitness of the locally optimized design point is determined. If it is found to be better than that of the initial guess then Lamarckian learning proceeds. Lamarckian learning forces the genotype to reflect the result of improvement by placing the locally improved individual back into the population to compete for reproductive opportunities. In addition, the optimized design point and its corresponding objective and constraint function values are added to the database. This process of hybrid EA search is continued until the computational resource is exhausted or a user specified termination criterion is met.

Note that apart from the parameters used in standard EAs, the proposed algorithm has two additional user-specified parameters, k_{max} and m. In Section 4.4, the experimental studies used to investigate the effect of these parameters on the convergence trends are presented.

Further, in engineering design optimization, evaluation of the objective and constraint functions takes up the overwhelming bulk of the computation. Therefore, a sub-linear improvement in design search efficiency can be achieved via global parallelism, where all design points within a single population are evaluated simultaneously across multiple machines. Parallelism is thus considered a desirable feature of any framework for optimization of computationally expensive problems. In the present algorithm, it is relatively straightforward to achieve parallelism, since local search for each individual in an EA generation can be conducted independently. To ensure load balancing, it is only necessary to specify that the number of trust region iterations be kept the same for each individual.

In the present implementation of the proposed algorithm, NetSolve [99] is employed. Netsolve is a computational platform which facilitates grid-based heterogeneous computing [100] in a transparent and efficient manner. Parallelism is achieved by wrapping the local search and surrogate modeling routines on a NetSolve server. The analysis codes are also wrapped on NetSolve servers, which can be invoked by the local search and the client routines. Hence, using the farming client application programming interface, local search for each chromosome in an EA generation can be readily conducted in parallel on remote servers. Even though we used a centralized database, it should be noted that NetSolve has capabilities for distributed data storage on remote servers. Note that this approach does not parallelize the SQP steps and thus is only suitable where the SQP updates can be efficiently offered as serial processes.

§4.4 Numerical Studies on Benchmark Problems

In this section, the numerical results obtained by implementing the proposed approach within a standard binary coded genetic algorithm (GA) are presented. In the experimental studies, 10 bit binary encoding, uniform crossover and mutation operators at probabilities 0.6 and 0.01, respectively, are employed for the standard GA with a population size of 50. A linear ranking algorithm is used for selection. The codes implementing the objective and constraint functions were wrapped on NetSolve servers running Red

Hat Linux on Pentium III processors. The GA code is linked to the NetSolve client library so that the objective function and constraint evaluation modules, and the local search routines can be invoked remotely.

The feasible SQP implementation that is used here is the "Fortran code for Feasible Sequential Quadratic Programming" code developed by Lawrence and Tits [101]. Feasible Quadratic Programming is a quasi-Newton method that solves a nonlinear constrained optimization problem by fitting a sequence of quadratic programming problems to it, and then solving each of these problems using a quadratic programming method. When started from an infeasible point (one that violates at least one of the linear or nonlinear constraints), FFSQP first carries out an optimization in which it minimizes the maximum of the constraint violations. This optimization continues until a feasible point is found. Then it minimizes the maximum of the objective functions, while keeping the nonlinear equality and inequality constraints, linear equality and inequality constraints, and simple bounds on the variables satisfied. In another words, when presented with an infeasible chromosome in the population, FFSQP attempts to repair it, optimize it locally and return a feasible chromosome into the population via the Lamarckian learning procedure. Hence, one main motivation of using FFSQP in this work is its chromosome repairing capabilities.

Two benchmark problems commonly used in the literature are adopted here for testing the proposed algorithm. They represent classes of unconstrained and constrained multimodal test problems. These problems make it possible to study whether the proposed approach would bring any increase in efficiency or computational cost reduction when used on complex problems. The problems considered are the minimization of the twenty-dimensional Rastrigin and maximization of the Bump functions described in Appendix B. These functions have very rugged landscapes and are often difficult to optimize. The averaged convergence trends obtained by applying the present algorithm to the benchmark test problems as a function of the total number of function evaluations are shown in Figures 4.2 - 4.4.

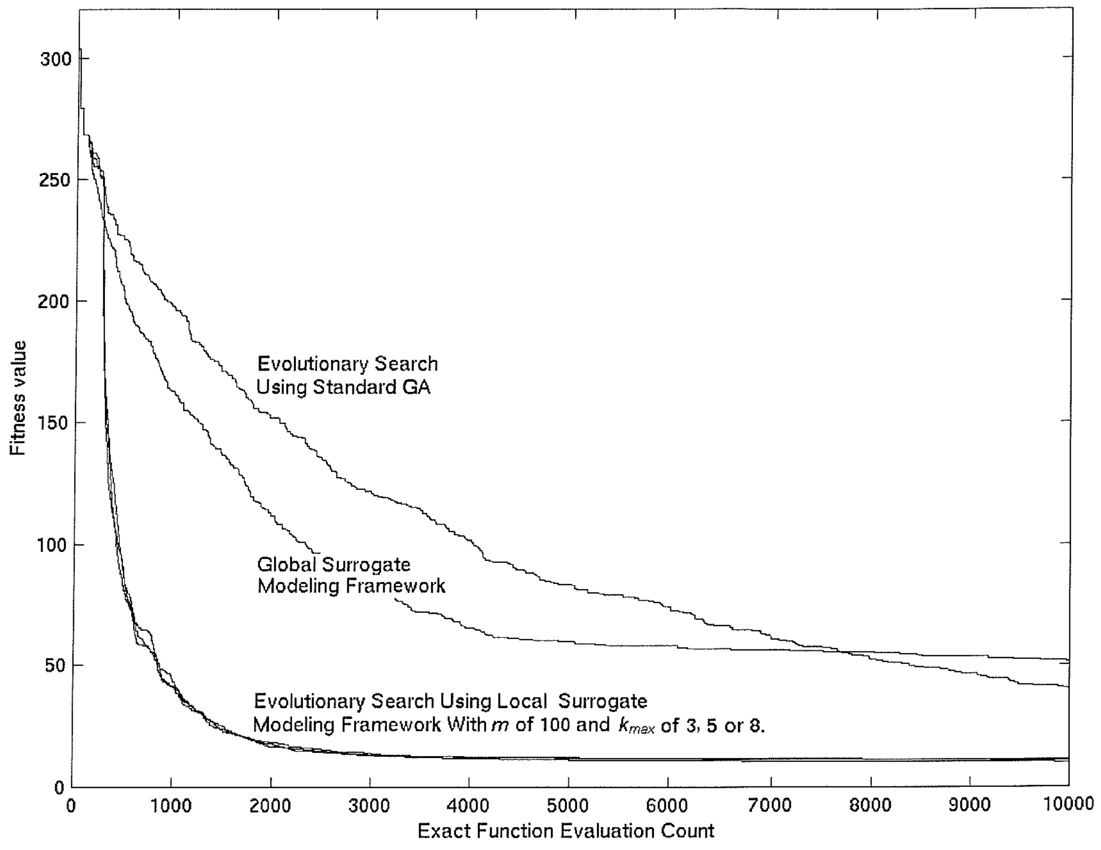


Figure 4.2: Averaged convergence trends for m=100 and various values of k_{max} (3, 5 and 8) compared with standard GA and global surrogate modeling framework for the 20D Rastrigin function.

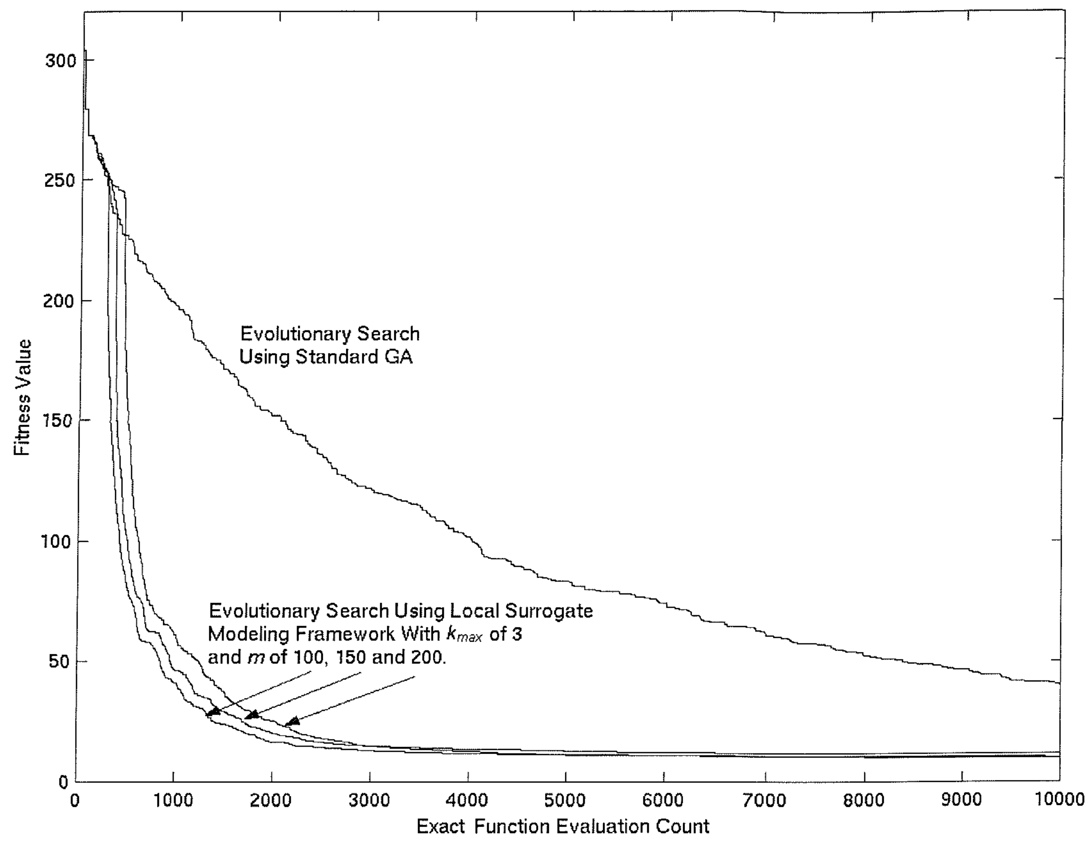


Figure 4.3: Averaged convergence trends for $k_{max} = 3$ and various values of m (100, 150 and 200) compared with standard GA for the 20D Rastrigin function.

The results presented here were averaged over 20 runs for each test function. Also shown in the figures are averaged convergence trends obtained using a standard GA and a global surrogate modeling strategy. The algorithm based on global surrogate models employed in the present numerical studies is based on the approach proposed by Ratle [82]; see Figure 4.5 for an outline of the steps involved in this algorithm.

The results obtained for the test functions show that the global surrogate framework displays early sign of stalling. This is consistent with previous studies in the literature [82, 83, 85] which suggest that when global surrogate models are applied to high-dimensional and multimodal test functions, the search generally tends to stall early on. Such an effect is a result of the curse of dimensionality, which often leads to early convergence at false global optima of the surrogate model. In contrast, the results obtained using the proposed algorithm clearly demonstrate that solutions close to the global optima

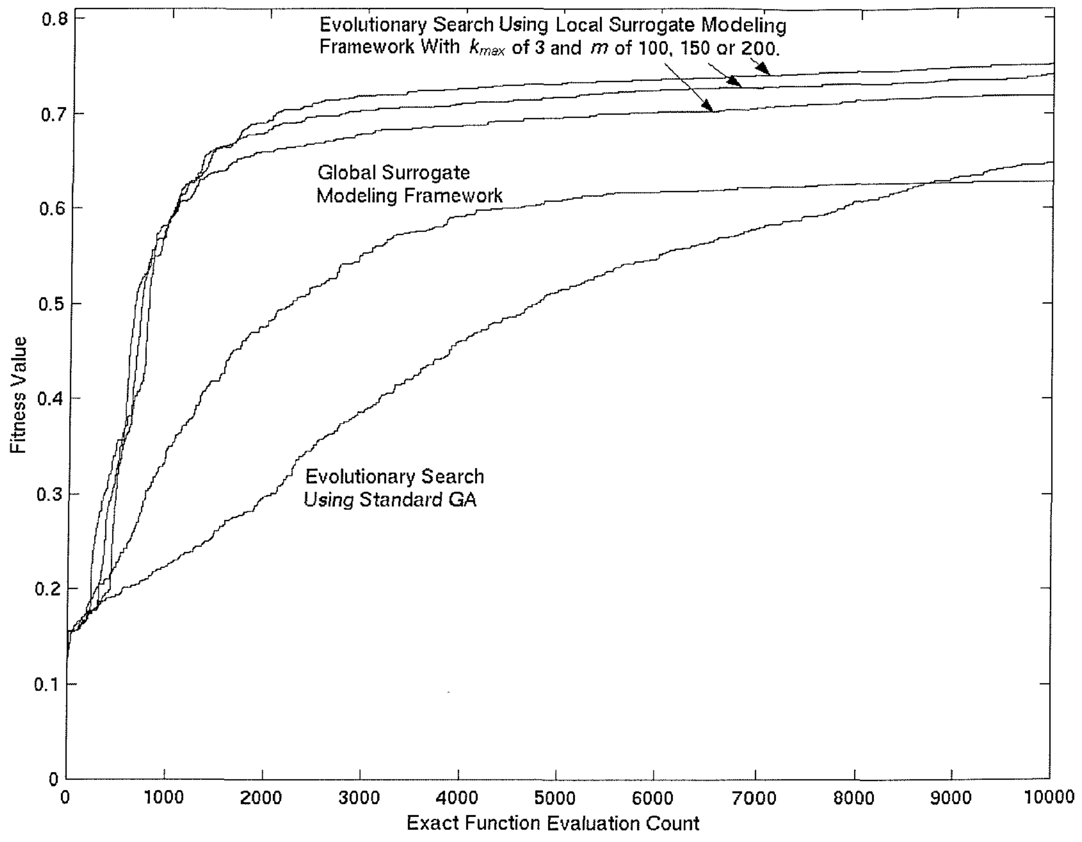


Figure 4.4: Averaged convergence trends for $k_{max} = 3$ and various values of m (100, 150 and 200) compared with standard GA for the 20D Bump function.

can be obtained on a limited computational resource. As surrogates are used only for local searches, i.e., as the exact model is used for all analysis conducted at the EA level, the chances for convergence to false global optima are greatly reduced. In addition, the use of the trust-region framework maintains convergence close to the local optima of the original problem during the SQP steps.

For these benchmark problems, the effect of increasing the maximum number of trust-region iterations and the number of nearest neighbours (employed to construct the local surrogate model) on the convergence behavior is also studied; see Figures 4.2 - 4.4. A number of observations can be made from the convergence trends. First, it appears that there is not much to be gained by increasing k_{max} beyond three. Secondly, it appears that smaller values of m generally lead to faster convergence during the early stages of search, but there is a tendency to stall at later stages. The converse is true for increases in m.

```
BEGIN
Initialize: Generate a database containing a population of designs.
(Optional: Upload a historical database if one exists)
Construct surrogate model using all available design points in the database.
Set fitness function := Surrogate model
While (computational resource not exhausted)
    Evaluate all individuals in the population using the fitness function.
    Apply standard EA operators to create a new population.
    If (fitness function := Exact model)
        Update database with any new designs generated using the exact model.
        Update surrogate model using all designs in the database.
    End If
    If (convergence over surrogate model)
        fitness function := Exact model
    Else
        fitness function := Surrogate model
    End If
End While
END
```

Figure 4.5: Algorithm based on global surrogate models proposed by Ratle [82].

This suggests the possibility of adaptively selecting m during the search. The following simple strategy is proposed:

$$m = (m_{min} + m_{max})\frac{t_c}{t_t} \tag{4.14}$$

where m_{min} = population size, $m_{max} = 400$, while t_c and t_t are the current time spent and the amount of affordable computational resource specified by the user, respectively.

§4.5 Aerodynamic Aircraft Wing Design

In this section, the application of the proposed algorithm to the transonic civil transport aircraft wing design problem considered by Keane and Petruzzelli [14] and previously in Chapters 2 and 3 is presented. Again, the objective is set as the minimization of wing drag D/q meters^2 with target lift, wing weight, volume, pitch-up margin and root triangle layout chosen to be representative of a 220 seat wide body airliner.

Aerodynamic wing design is an extremely complex task, which is why it is normally undertaken over an extended time period and at different levels of complexity. Here, the wing drags may be predicted using the empirical drag estimation tool, TADPOLE, or the the linearized potential method, VSAERO, which is a full Computational Fluid Dynamic code. TADPOLE takes only some 6 seconds to run, and returns drag values based on curve fits to previously analyzed wings. VSAERO is more computational expensive than TADPOLE and requires approximately 11 minutes of compute time per drag evaluation. However, it has the advantage of providing more accurate drag predictions provided Mach numbers are not too high, i.e. ≤ 0.9. The reader is referred to Appendix C for further details of the aerodynamic wing design problem considered.

A summary of the search results obtained for the aerodynamic wing design problem is shown in Table 4.2, and Figures 4.6 and 4.7.

	Mean Wing Drag Values (D/q, meters^2) at Evaluation Counts of					
m	**250**	**500**	**600**	**800**	**1200**	**1500**
100	2.823	2.781	2.788	2.788	2.788	2.788
200	3.218	2.788	2.775	2.771	2.768	2.768
300	3.235	3.049	3.049	2.771	2.764	2.761
400	3.265	3.064	3.049	3.020	2.772	2.758
Adaptive m	*2.919*	*2.771*	*2.771*	*2.768*	*2.760*	*2.758*
Standard GA	3.213	3.004	2.983	2.978	2.974	2.961

Table 4.2: Summary of minimum drag values using the TADPOLE code and surrogate models.

Table 4.2 shows the search results when using TADPOLE for drag estimation. Note that there are at least m exact design points available in the database before local search using surrogates steps in. Once again, it is observed from these results that smaller values of m lead to faster convergence of the wing drag values during the early stages of search, but result in stalling at later stages, and the converse is true for increases in m. These results are in line with those observed earlier for the test functions. Hence, for the computationally expensive VSAERO code, m is chosen adaptively using equation 4.18. The maximum number of trust region iterations (k_{max}) is also set to three. Note that during local search, the surrogate models for the objective function and the four inequality constraints are constructed.

For this problem, it has been found using the standard GA as well as observed from previous studies by Keane and Petruzzelli [14] that the near optimal value of 2.758 using the TADPOLE code can be obtained after 10,000 evaluations. In comparison, using the proposed approach, we are able to converge to this solution on an average after 1,500 exact evaluations, when m is chosen adaptively during the search.

In Figure 4.6, the averaged convergence trends using VSAERO for wing drag esti-

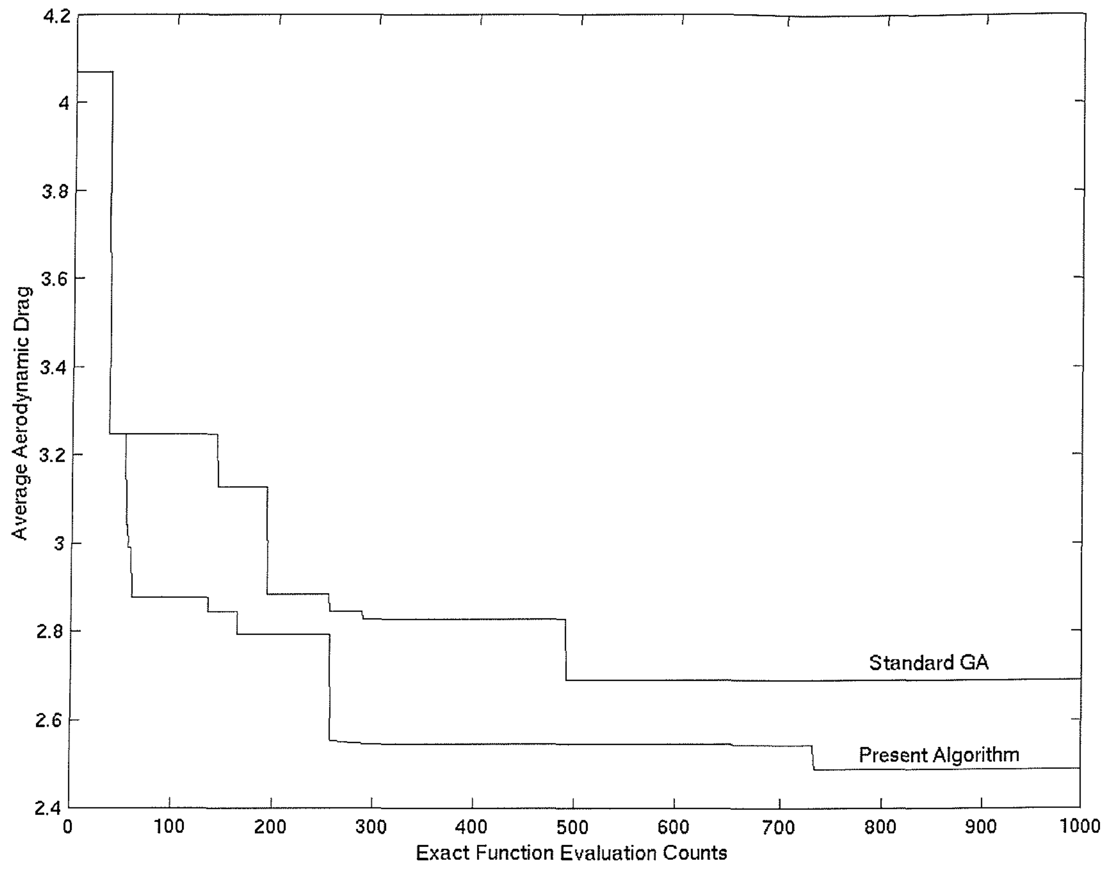


Figure 4.6: Summary of minimum drag values using the VSAERO code and surrogate models. Results are average of 3 runs.

mation also clearly illustrates the ability of the proposed algorithm to arrive at good designs on a limited computational resource. The convergence trends of the best run for the aerodynamic wing design problem using VSAERO are also plotted as a function of wall-clock time in Figure 4.7. The wall-clock time refers to the actual time or the total amount of elapsed time that the program takes to run or to execute the allocated search task(s). Due to the availability of only eight licenses for the VSAERO code, the timing plot obtained in Figure 4.7 is based on a total of eight processors being used for parallel computations. Previous studies using the VSAERO code for wing drag estimation have revealed that the best design that can be obtained using the standard GA with only the exact analysis model has a drag value of 2.63 after 1,800 evaluations. In comparison, using the present approach, this solution was obtain on average after 250 exact evaluations.

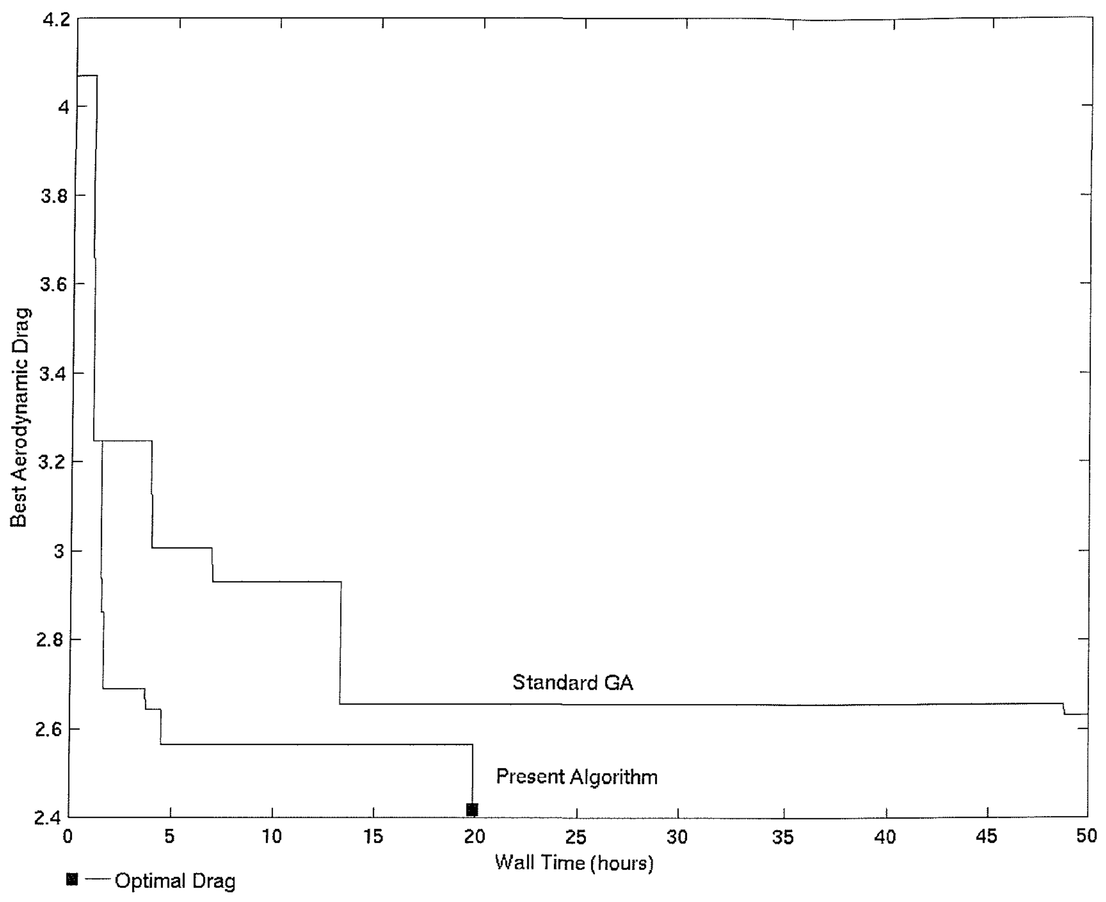


Figure 4.7: Comparison of best convergence trends as a function of wall-clock time for the Aerodynamic Wing Design Problem using the VSAERO code and surrogate models.

After 733 exact evaluations, the best design obtained using the proposed algorithm had a drag value of 2.404, which is the lowest value obtained to date for this problem using various optimization algorithms. Note that the best solutions reported here satisfied all the four inequality constraints.

§4.6 Conclusion

In this Chapter, a general framework that leverages surrogate models created from historical data for evolutionary optimization of computationally expensive constrained design problems is proposed and presented. It is argued that for such complex design problems, constructing an accurate global surrogate model suffers from fundamental difficulties due to the curse of dimensionality. A local learning approach in the spirit of transductive inference is employed to construct surrogate models. Such local approximation models are shown to be easily incorporated into hybrid evolutionary-gradient optimization algorithms. The proposed local search strategy employs a trust-region framework to interleave the exact and approximate models. Further, it is shown that such an approach retains the intrinsic parallelism of traditional EAs.

Extensive numerical studies on some benchmark test functions to demonstrate the effectiveness of the proposed algorithm are presented. The results were compared with those obtained using a standard genetic algorithm and a global surrogate modeling strategy. Experimental results are also presented for the real world aerodynamic wing design problem using a full 3–d CFD code. These studies clearly indicate that the present approach allows for the possibility of arriving at good solutions on a limited computational resource.

Chapter 5

Concluding Remarks

THIS Chapter concludes the dissertation with a brief synopsis of the contributions of the present research. Areas for future work and a program of future research are also outlined.

§5.1 Research Contributions

The focus of this research was to develop methodologies that leverage Artificial Intelligence techniques to achieve advancement in complex engineering design search. In particular, the present methodologies integrate machine learning techniques along with an existing Design Exploration System and search routines to enhance the chances of arriving at good designs and at the same time, improving search efficiency. This was achieved via effective use of design optimization data chronicled from previous design search processes and online searches. The proposed approaches have concentrated on mitigating two limitations found in most current complex engineering design optimization processes, identified as

- Choice of Optimization Methods.
- Computationally Expensive Design Problems.

The primary contributions of the present program of research are summarized below.

§5.1.1 Domain Optimization Method Advisor

In Chapter 2, a Domain Optimization Method Advisor [23, 24, 25] was presented. The proposed advisor seeks to reduce the influence of choice of optimization methods on engineering design search performance, when searching familiar design domains. Domain-specific knowledge is used to support the use of a DES by engineers, using data collected from previous optimization search processes in the same domain and from offline simulations. The methodology described is simple, automatic and is applicable to many engineering design process as it is neither tightly coupled to domains nor particular DES. A machine learning induction technique based on a decision tree algorithm is identified as being most suitable in this application as it provides accurate predictions and transparency in making optimization method recommendations for design problems. Furthermore, the advisor helps reduce the reliance on human experts and caters to the multiple needs of design engineers of different abilities. It is also shown to achieve better design search performances in both efficiency and design quality when compared to the conventional approaches commonly adopted by general design engineers.

§5.1.2 Meta-Lamarckian Learning

In Chapter 3, Meta-Lamarckian Learning in Hybrid Genetic Algorithm-Local Search (GA-LS) [26] was presented. The proposed learning methodology aims to assist in the choice of local optimization method in hybrid evolutionary design search. The methodology does not require offline simulations to be conducted but instead makes effective use of the optimization search data amassed while the design problem is searched online. It is applicable to general engineering design problems and adapts to changes on design problems statements. Two adaptive strategies have been investigated and the better strategy recommended for use. Besides assisting design engineers in the choice of local optimization methods when working with general complex engineering design problems, the proposed approach was shown to yield robust and improved design search performance. Furthermore the methodology also permits the incorporation of human expert knowledge to achieve improved designs.

§5.1.3 Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling

In Chapter 4, a study of the literature on handling computationally expensive analysis codes or simulation models in complex engineering design optimization was presented. From this survey, the need for a general framework for evolutionary optimization of computationally expensive constrained problems which use surrogate models is established. A novel algorithm or framework that interleaves computationally expensive analysis models for the objective and general constraint functions with cheap surrogate models created from online chronicled data is then proposed and the results from several investigations were presented. The proposed approach employs a trust-region framework, hybrid evolutionary-gradient optimization, with radial basis networks and transductive inference. Further, it is shown that the proposed framework improves design search efficiency and arrives at superior designs when compared to the more conventional approaches [27, 28].

§5.2 Future Work

The methodologies proposed in this dissertation provide some of the groundwork for an intelligent optimization system in complex engineering design by making effective use of experiences or chronicled optimization data from past designs. Computational and Artificial Intelligence technologies have already assisted and will continue to advance this field in numerous ways. In future research, the two traditions of computational and artificial intelligence technologies will continue to help in dealing with complex engineering designs. One aims at understanding the human creative scientific process (strong AI), the other at producing scientific assistants (weak AI). Both aim at the computational discovery of new scientific knowledge. A summary of potential future work is outlined below.

§5.2.1 Domain Optimization Method Advisor

The Domain Optimization Method Advisor presented symbolizes an initial endeavor to predicting the most appropriate search routines on general optimization problems. Therefore, above and beyond the acquisitions of domain knowledge, it is important that future research focuses on the derivation of domain-independent knowledge on optimization methods. In that case, the advisor may assist design engineers working on general black-box design problems, in contrast to restricted design domains. Clearly, the design of a more sophisticated advisor that might supply advice on the choice of search routine control parameters would also be appealing. Several artificial intelligence technologies that may prove to be valuable in this part of the work are Agents, Expert Systems, Ontology-Based Systems and Reinforcement Learning.

§5.2.2 Meta-Lamarckian Learning

In Meta-Lamarckian Learning, we would like to foster novel adaptive strategies and credit assignment metrics that better define the rewards assigned to optimization routines. Studies on the dynamics of Meta-Lamarckian Learning and extension to Meta-Baldwin Learning in hybrid evolutionary search are also worthy of note. In addition, fuelled by advancing computing power, further efforts may provide a parallel form of Meta-Lamarckian Learning in hybrid evolutionary techniques which would greatly benefit the engineering design community. The work presented on Meta-Lamarckian Learning epitomizes the adaptive methodology for automating the appropriate choice of local optimizers in hybrid evolutionary search. The proposed methodology was then validated via extensive empirical studies. Given the restricted amount of well-established theoretical knowledge in the choice of local optimizer, research efforts should therefore be placed on providing some theoretical background in the area.

§5.2.3 Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling

(a) Global Convergence of Proposed Framework

An intriguing future activity would be to attempt to prove the global convergence of the general metamodeling framework proposed for evolutionary search of computationally expensive problems discussed previously in Chapter 4. Global convergence is defined in the optimization literature as the mathematical assurance that the iterations produced by an algorithm, started from an arbitrary initial guess, will converge to a stationary point or local optima for the original high-fidelity expensive analysis code. One approach based on the classical trust region idea from nonlinear programming is shown by Alexandrov et. al. [96] to be provably convergent to a local optima of the original problem.

Global convergence results for Evolutionary Algorithms that make use of approximation models in the search do not seem to have appeared in the literature so far. Nevertheless, it may be possible to design EAs that inherit the existing global convergence properties of classical algorithms. The work by Hart [102] has shown one such possibility where a provably convergent evolutionary pattern search algorithm was proposed that inherits the existing theory for classical pattern search.

In order to prove global convergence for trust-region frameworks that embed surrogate models in the local search, Alexandrov et al. showed that zero-order and first-order consistency conditions have to be imposed at the initial guess, i.e.,

$$\hat{f}(\mathbf{x}_c^k) = f(\mathbf{x}_c^k) \tag{5.1}$$

$$\nabla\hat{f}(\mathbf{x}_c^k) = \nabla f(\mathbf{x}_c^k) \tag{5.2}$$

Since we use an interpolating surrogate model in the present approach, only the zero-order consistency condition is satisfied at the initial guess. To satisfy equation 5.2, the exact sensitivities of the objective and constraint functions are also required, which would be computationally prohibitive for many problems.

Clearly, the framework presented has some chance at having provable global convergence properties, therefore it is worth noting here that future efforts might be placed in this area of research.

(b) Surrogate-assisted Coevolutionary Search on Problems with General Constraint Functions

It may also be worth exploring the employment of coevolutionary search [103] for solving computationally expensive optimization problems. The motivation for this study arises from the fact that since coevolutionary search is based on the divide-and-conquer paradigm, it may also be possible to circumvent the curse of dimensionality inherent in surrogate modeling techniques such as radial basis networks. The efficacy of surrogate-assisted coevolutionary search on general bound constrained problems was reported in [28]. In particular, the use of coevolutionary search with surrogates to solve computationally expensive optimization problems under limited computational budget was investigated via studies on bound constrained benchmark test functions and a realistic two-dimensional cantilevered space structure design problem. Y. S. Ong et al. showed that by employing approximate models for the fitness, it becomes possible to converge to good solutions even for bound constraint functions with a high degree of epistasis. In continuation of the work developed on surrogate-assisted coevolutionary search, it would be of great interests to determine if the deduction drawn in [28] may also apply to computationally expensive optimization problems with general nonlinear equality and inequality constraints.

(c) Other future work

Further research efforts placed on evolutionary optimization of computationally expensive problems via surrogate modeling would nonetheless involve spawning novel solutions in the direction discussed earlier in section 4.1 of this dissertation, i.e., engaging the factors responsible for limiting the success of approximation models+EA frameworks to real world problems.

Bibliography

[1] A. J. Keane, 1995, "The OPTIONS Design Exploration System", Available at: http://www.soton.ac.uk/~ajk/options /welcome.html.

[2] Isight, 2000, "Engineous Software", Available at: http://www.engineous.com/isight.html.

[3] J. N. Siddall, 1982, "Optimal Engineering Design: Principles and Applications", Marcel Dekker Inc., New York.

[4] H. P. Schwefel, 1995, "Evolution and Optimum Seeking", John Wiley&Sons.

[5] C. T. Lawrence and A. L. Tits, 1996, "Nonlinear Equality Constraints in Feasible Sequential Quadratic Programming", Optimization Methods and Software, Vol. 6, pp. 265–282.

[6] E. Sandgren, 1977, "The utility of nonlinear programming algorithms", Technical report, Purdue University, Ph.D Thesis.

[7] M. Bramlette and R. Cusic, 1989, "A comparative evaluation of search methods applied to parametric design of aircraft", Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, Morgan Kaufmann, pp. 213-218.

[8] G. N. Vanderplaats, 1984, "Numerical Optimization Techniques for Engineering Design: With Applications", McGrawHill, New York.

[9] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, 1983, "Optimization by simulated annealing", Science, Vol. 220, No. 4598, pp. 671–680.

[10] T. Back, F. Hoffmeister and H. Schwefel, 1991, "Survey of evolution strategies", editors: R. Belew and L. Booker, Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 2-9.

[11] L. Davis, 1991, "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York.

[12] Z. Michalewicz, 1994, "Genetic Algorithms + Data Structures = Evolutionary Programs", Springer–Verlang, New York, 2nd edition.

[13] P. J. Fleming and R. C. Purshouse 2001, "Genetic Algorithms in Control Systems Engineering", Technical Report No. 789, Department of Automatic Control and Systems Engineering, University of Sheffield, UK, May.

[14] A. J. Keane and N. Petruzzelli, 2000, "Aircraft wing design using GA–based multi–level strategies", Proceedings of the 8th AIAA/USAF/NASSA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA–2000–4937, Long Beach, pp. A00–40171.

[15] S. S. Tong, D. Powell and S. Goel, 1992, "Integration of artificial intelligence and numerical optimization techniques for the design of complex aerospace systems", Aerospace Design Conference, Irvine, CA, February. AIAA-92–1189.

[16] P. Gage, I. Kroo, and I. Sobieski, 1995, "A Variable–Complexity Genetic Algorithm for Topological Design", AIAA Journal, Vol. 33, No. 11, pp. 2212–2217.

[17] H. Bersini and B. Renders, 1994, "Hybridizing genetic algorithms with hill–climbing methods for global optimization: Two possible ways", IEEE International Symposium Evolutionary Computation, pp. 312–317, Orlando, Fl.

[18] A. Vicini and D. Quagliarella, 1999, "Airfoil and wing design using hybrid optimization strategies", AIAA Journal, Vol.37, No.5, pp. 634–641.

[19] J. A. Joines, R. E. King and M. G. Kay, 2000, "Utilizing a hybrid genetic search for manufacturing cell design", Journal of the Chinese Institute of Industrial Engineers, Special Issue on Soft computing in Industrial Engineering, Vol. 17, No. 5, pp. 549–564.

[20] A. J. Keane and P. B. Nair, 2001, "Problem Solving Environments in Aerospace Design", Advances in Engineering Software, Vol. 32, pp. 477–487.

[21] A. Jameson, 1999, "Re–Engineering the Design Process Through Computation", Journal of Aircraft, Vol. 36, No. 1, pp. 36–50.

[22] J. McCarthy, 1958, "Artificial Intelligence Project", Progress Report No. 4 of the Research and Educational Activities in Machine Computation by the Cooperating Colleges of New England, Dec.

[23] Y. S. Ong and A. J. Keane, 2002, "A domain knowledge based search advisor for design problem solving environments", Engineering Applications of Artificial Intelligence, Vol. 15, No. 1, pp. 105-116.

[24] Y. S. Ong and A. J. Keane, July 2002, "An automated optimization system for aircraft wing design", Seventh International Conference on Artificial Intelligence in Design, Cambridge, UK.

[25] A.Choudhury, Y. S. Ong and A.J. Keane, 2002, "Extracting Latent Structures in Numerical Classification: An investigation using two factor models", 9th International Conference on Neural Information Processing, Singapore, November.

[26] Y. S. Ong and A. J. Keane, 2002, "Meta–Lamarckian in Hybrid Memetic Algorithms", IEEE Transactions On Evolutionary Computation, in revision.

[27] Y. S. Ong, P. B. Nair, and A. J. Keane, 2002, "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling", AIAA Journal, in press.

[28] Y. S. Ong, A.J. Keane and P.B. Nair, 2002, "Surrogate–Assisted Coevolutionary Search", 9th International Conference on Neural Information Processing, Special Issue on Trends in Global Optimization, Singapore, November.

[29] M. J. Box, 1966. "A comparison of several current optimization methods and the use of transformations in constrained problems", Computer Journal, Vol. 9, May, pp. 67–77.

[30] Colville, 1968. "A Comparative Study on Nonlinear Programming Codes", IBM Scientific Center Report 320–2949, No.6.

[31] A. J. Keane, 1996, "A Brief Comparison of Some Evolutionary Optimization Methods", Modern Heuristic Search Methods, editors: V. Rayward–Smith, I. Osman, C. Reeves and G. D. Smith, J. Wiley, pp. 255–272.

[32] D. H. Wolpert and W. G. Macready, 1997, "No free lunch theorems for optimization", IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 67–82.

[33] A. S. Fukunaga, S. Chien, D. Mutz, R. Sherwood and A. Stechert, 1997, "Automating the Process of Optimization in Spacecraft Design", Proceeding IEEE Aerospace Conference, Snowmass CO, Vol. 4, pp. 411–428.

[34] M. S. Kamel, K. S. Ma and W. H. Enright, 1993, "ODEXPERT: An expert system to select numerical solvers for initial value ode systems", ACM Transactions on Mathematical Software, Vol. 19, pp. 44–62.

[35] W. R. Dyksen and C.R. Gritter, 1992, "Scientific computing and the algorithm selection problem", Intelligent Mathematical Software Systems, editors: E. N. Houstis, J. R. Rice and R. Vichnevetsky, North–Holland, pp. 19–31.

[36] E. N. Houstis, J. R. Rice, S. Weerawarana, A. C. Catlin, M. Gaitatzes, P. Papachiou, and K. Wang, 1998, "Parallel ELLPACK: a problem solving environment for PDE based applications on multicomputer platforms", ACM Trans. Math. Soft., Vol. 24, No. 1, pp. 30–73.

[37] E. N. Houstis, A. C. Catlin, J. R. Rice, V. S. Verykios, N. Ramakrishnan and C. E. Houstis, 2001, "PYTHIA–II, A Knowledge/Database System for Managing Performance Data and Recommending Scientific Software", Available at: http://www.cs.purdue.edu/homes/acc/pythiaPaper.html.

[38] M. D. McKay, R. J. Beckman and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code", Technometrics, Vol. 21, No. 2, pp.239-245, 1979.

[39] G. E. Box and N. R. Draper, "Empirical Model Building and Response Surfaces", John Wiley, New York, NY, 1987.

[40] K. Rasheed and H. Hirsh, 1997, "Using Case–Based learning to improve genetic–algorithm–based design optimization", Proceedings of the Seventh International Conference on Genetic Algorithms. Morgan Kaufmann.

[41] J. Surma and B. Braunschweig, 1996, "Case–Base Retrieval in Process Engineering: Supporting Design by Reusing Flowsheets", Engineering Application Artificial Intelligent, Vol. 9, No. 4, pp. 385–391.

[42] KBSI 1999, "An Intelligent Assistant for Optimization Modeling", Available at: http://www.kbsi.com/Research/PlanningScheduling/oma.html.

[43] EAAI, 1996, Special Section on AI in Design Applications, Engineering Application Artificial Intelligent, Vol. 9, No. 4.

[44] W. J. Frawley, G. Pietetsky–Shapiro and C. J. Matheus, 1991, "Knowledge discovery in databases: an overview", Knowledge Discovery in Database, MIT Press, Cambridge, MA, pp. 1–30.

[45] W. H. Wolberg, W. N. Street and O. L. Mangasarian, 1994, "Machine learning techniques to diagnose breast cancer from fine–needle aspirates", Cancer Letters 77, pp. 163–171.

[46] M. Schwabacher, 1996, "The use of artificial intelligence to improve the numerical optimization of complex engineering designs", Technical Report HPCDTR–45,

Department of Computer Science, Rutgers University, Ph.D. Thesis, Available at: http://www.cs.rutgers.edu/~schwabac/thesis.html.

[47] Y. Reich, 1996, "Modeling engineering information with machine learning", Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, Vol. 10, No. 2, pp. 171–174.

[48] T. M. Mitchell, 1997, "Machine Learning", McGraw–Hill International Editions.

[49] R. C. Holte, 1993, "Very simple classification rules perform well on most commonly used datasets", Machine Learning, No. 11, pp. 63-90.

[50] G. Paass, 1992, "Probabilistic Reasoning and Probabilistic Neural Networks", International Journal of Intelligent Systems, Vol.7, pp. 47-59.

[51] S. Deerwester, S. T. Dumais, G . W. Furnas, T . K. Landauer and R. Harshman, 1990, "Indexing by Latent Semantic Analysis", Journal of the American Society for Information Science, Vol. 41, No. 6, pp. 391–407.

[52] P. Langley and S. Sage, 1994, "Induction of selective bayesian classifiers", Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, Inc., Seattle, WA, pp. 399–406.

[53] J. R. Quinlan, 1993, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, Inc., Los Altos, California.

[54] A. J. Keane, W. G. Price and R. D. Schachter, 1990, "Optimization Techniques in Ship Concept Design", The Royal Institution Naval Architects, Spring Meetings.

[55] A. Torn and A. Zilinskas, 1989, "Global Optimization", Volume 350 of Lecture Notes in Computer Science. Springer–Verlag.

[56] P. Merz, 2001, "On the Performance of Memetic Algorithms in Combinatorial Optimization", 2nd Workshop on Memetic Algorithms, GECCO, San Francisco, CA, USA.

[57] M. Mitchell, S. Forrest and J. H. Holland, 1991, "The royal road for genetic algorithms: Fitness landscape and GA performance", VB92, pp. 245–254.

[58] W. E. Hart, 1994, "Adaptive Global Optimization with Local Search", Ph.D. Thesis, University of California, San Diego, May.

[59] W. M. Spears, 1995, "Adapting crossover in a genetic algorithm", Proceedings of the 5th Conference on Evolutionary Programming, editors: J. R. McDonnell, R. G. Reynolds and D. B. Fogel, Cambridge: MIT Press, pp. 367–384.

[60] M. Ko, T. Kang, and C. Hwang, 1996, "Adaptive crossover operator based on locality and convergence", Proceedings of the IEEE International Joint Symposia on Intelligence and Systems (IJSIS).

[61] L. J. Fogel, P. J. Angeline, and D. B. Fogel, 1995, "A Preliminary Investigation on Extending Evolutionary Programming to Include Self–adaptation on Finite State Machines", Informatica, Vol. 18, pp. 387–398.

[62] T. Back and M. Schutz, 1996, "Intelligent mutation rate control in canonical genetic algorithms", Proceedings of the 9th International Symposium, ISMIS 96, June, Springer–Verlag, Berlin (Germany), pp. 158–167.

[63] J. E. Smith and T. C. Fogarty, 1997, "Operator and parameter adaptation in genetic algorithms", Soft Computing, Vol. 1, No. 2, pp. 81–87.

[64] R. Hinterding, Z. Michalewicz, and A. E. Eiben, 1997, "Adaptation in evolutionary computation: A survey", Proceedings of the Fourth International Conference on Evolutionary Computation (ICEC 97), IEEE Press, Piscataway (NJ), pp. 65–69.

[65] G. Magyar, M. Johnsson and O. Nevalainen, 2000, "An adaptive hybrid genetic algorithm for the three–matching problem", IEEE Transactions on Evolutionary Computation, July, Vol. 4, No. 2, pp. 135–146.

[66] I. K. Jeong and J. J. Lee, 1996, "Adaptive simulated annealing genetic algorithm for system identification", Engineering Application of Artificial Intelligence, Vol. 9, No. 5, pp. 523–532.

[67] C. Houck, J. Joines, and M. Kay, 1996, "Utilizing Lamarckian Evolution and the Baldwin Effect in Hybrid Genetic Algorithms", NCSU–IE Technical Report 96–

01, Meta–Heuristic Research and Applications Group, Department of Industrial Engineering, North Carolina State University.

[68] T. Kido, K. Takagi and M. Nakanishi, 1994, "Analyses and Comparisons of Genetic Algorithm, Simulated Annealing, TABU Search, and evolutional combination algorithm", International Journal of Informatica, Vol.18, pp. 399–410.

[69] D. W. Johnson and R. T. Johnson, 1989, "Cooperation and competition: Theory and research", Edina, MN: Interaction Book Company.

[70] P. Cowling, G. Kendall and E. Soubeiga, 2000, "A Hyperheuristic Approach to Scheduling a Sales Summit", Lecture Notes In Computer Science, Springer, Third International Conference on the Practice and Theory of Automated Timetabling.

[71] F. G. Lobo, and D. E. Goldberg, 1997, "Decision making in a hybrid genetic algorithm", IEEE International Conference on Evolutionary Computation, , Piscataway, NJ: IEEE, pp. 122–125.

[72] De Jong, 1975, "An analysis of the behaviour of a class of genetic adaptive systems", Ph.D. Thesis, University of Michigan.

[73] A. J. Keane, 1995, "Genetic algorithm optimization of multi–peak problems: studies in convergence and robustness", Artificial Intelligence in Engineering, Vol. 9, No. 2, pp. 75–83.

[74] M. Powell, 1996, "A direct search optimization method that models the objective and constraint functions by linear interpolation", SIAM conference, Virginia.

[75] H. Voigt and J. Lange, 1998, "Local evolutionary search enhancement by random memorizing", Proceedings of IEEE International Conference on Evolutionary Computation, Piscataway, NJ. IEEE Press, pp. 547–552.

[76] D. B. Serafini, 1998, "A Framework for Managing Models in Nonlinear Optimization of Computationally Expensive Functions", Ph.D. Thesis, Rice University.

[77] H. Furuya and R. T. Haftka, 1996, "Combining Genetic and Deterministic Algorithms for Locating Actuators on Space Structures", Journal of Spacecraft and Rockets, Vol. 33, No. 3, pp. 422-427.

[78] S. Nagendra, R. T. Haftka, Z. Gurdal, and L. T. Watson, 1996, "Improved Genetic Algorithms for the Design of Stiffened Composite Panels", Computers and Structures, Vol. 58, No. 3, pp. 543-555.

[79] G. M. Robinson and A. J. Keane, 1999, "A Case for Multi–level Optimization in Aeronautical Design", Aeronautical Journal, Vol. 103, No. 1028, pp. 481–485.

[80] P. B. Nair and A. J. Keane, 2001, "Passive Vibration Suppression of Flexible Space Structures via Optimal Geometric Redesign", AIAA Journal, Vol. 39, No. 7, pp. 1338–1346.

[81] J. Sobieszczanski-Sobieski and R. T. Haftka, 1996, "Multidisciplinary aerospace design optimization: Survey of recent developments", 34th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada. AIAA-96-0711.

[82] A. Ratle, 1998, "Accelerating the convergence of evolutionary algorithms by fitness landscape approximation", Parallel Problem Solving from Nature V, pp. 87–96.

[83] M. A. El–Beltagy, P. B. Nair, and A. J. Keane, 1999, "Metamodelling Techniques For Evolutionary Optimization of Computationally Expensive Problems: Promises and Limitations", Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99), Morgan Kaufman, pp. 196–203.

[84] K. H. Liang, X. Yao and C. Newton, 2000, "Evolutionary search of approximated N–dimensional landscapes", International Journal of Knowledge–Based Intelligent Engineering Systems, Vol. 4, No. 3, pp. 172–183.

[85] Y. Jin, M. Olhofer and B. Sendhoff, 2000, "A Framework for Evolutionary Optimization with Approximate Fitness Functions", IEEE Transactions on Evolutionary Computation, in press.

[86] V. V. Toropov, A. A. Filatov, A. A. Polynkin, 1993, "Multiparameter structural optimization using FEM and multipoint explicit approximations", Structural Optimization, Vol. 6, No. 1, pp. 7–14.

[87] P. Bottou and V. Vapnik, 1992, "Local learning algorithms", Neural Computation, Vol. 4, No. 6, pp. 888–900.

[88] O. Chapelle, V. Vapnik, and J. Weston, 1999, "Transductive Inference for Estimating Values of Functions", Advances in Neural Information Processing Systems, Vol. 12.

[89] V. Vapnik, 1998, "Statistical Learning Theory", John Wiley and Sons, New York.

[90] C. Bishop, "Neural Networks for Pattern Recognition", Oxford University Press, 1995.

[91] J. Sacks, W. J. Welch, T. J. Mitchell and H. P. Wynn, 1989, "Designs and analysis of computer experiments", Statistical Science, Vol. 4, No. 4, pp. 409–423.

[92] C. K. I. Williams and C. E. Rasmussen, 1996, "Gaussian Processes for Regression", Advances in Neural Information Processing Systems 8, editors: D. S. Touretzky, M. C. Mozer and M. E. Hasselmo, MIT Press.

[93] A. A. Giunta and L. T. Watson, 1998, "A Comparison of Approximation Modeling Techniques: Polynomial versus Interpolating Models", AIAA–98–4758.

[94] R. Jin, W. Chen and T. W. Simpson, 2001, "Comparative Studies of Metamodeling Techniques under Multiple Modeling Criteria", Journal of Structural Optimization, Vol. 23, No. 1, pp. 1–13.

[95] C. A. Micchelli, 1986, "Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions", Constructive Approximation, Vol. 2, No. 1, pp. 11–22.

[96] N. Alexandrov, J. E. Dennis, R. M. Lewis and V. Torczon, 1998, "A trust region framework for managing the use of approximation models in optimization", Structural Optimization, Vol. 15, No. 1, pp. 16–23.

[97] J. F. Rodrguez, J. E. Renaud and L. T. Watson, 1998, "Convergence of Trust Region Augmented Lagrangian Methods Using Variable Fidelity Approximation Data", Structural Optimization, Vol. 15, pp. 141–156.

[98] A. A. Giunta and M. S. Eldred, 2000, "Implementation of a Trust Region Model Management Strategy in the DAKOTA Optimization Toolkit", Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.

[99] H. Casanova and J. Dongarra, 1998, "NetSolve: A Network Enabled Server, Examples and Users", Proceedings of the Heterogeneous Computing Workshop, Orlando, Florida.

[100] C. Foster and C. Kesselman, 1999, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann.

[101] C. T. Lawrence and A. L. Tits, 2001, "A Computationally Efficient Feasible Sequential Quadratic Programming Algorithm", SIAM Journal on Optimization, Vol. 11, No. 4, pp. 1092–1118.

[102] W. E. Hart, 2000, "A Convergence Analysis of Unconstrained and Bound Constrained Evolutionary Pattern Search", Evolutionary Computation, Vol. 9, No.1, pp. 1–23.

[103] M. Potter, 1997, "The Design and Analysis of a Computational Model of Cooperative Coevolution", Ph.D. Thesis, George Mason University, Fairfax, VA.

[104] N. Petruzzelli and A. J. Keane, 2001, "Wave drag estimation for use with panel codes", Journal of Aircraft, Vol. 38, No. 4, pp. 778–782.

Appendix A

Design Exploration System

Design exploration systems (DESs) address the needs of engineers responsible for establishing critical design parameters, usually during the early stages of the design process. Such systems aim to provide a controlled framework for studying the effects of parameter choices on the traditional programs available to the designer for dealing with geometric decisions, stress analysis, performance estimation, etc. They have started to become available over recent years with the advent of relatively cheap and powerful computational facilities and sophisticated optimization and exploration techniques. The increasing cost of professional manpower and pressure to gain maximal performance from designs makes it reasonable to expect that DESs will grow in use and influence.

The Design Exploration system used in this research work is known as OPTIONS [1]. OPTIONS is a design exploration and optimization package that may be used to study design problems. The user provides routines describing his or her problem plus entries in a problem-specific database. It is then possible to manipulate the design manually, systematically map out the effects of design changes, or, having specified design variables, constraints and an objective function, invoke one of the many optimizers within the package. Among the many different methods in OPTIONS, some are from standard libraries by Siddall and Schwefel [3, 4], while others have been specially developed for the suite, based on ideas culled from the literature. The 31 optimization search routines of the OPTIONS DES together with the abbreviations employed in this dissertation are listed in Table A.1.

Abbreviations	31 Optimization Search Routines From OPTIONS DES
AP	Method of successive linear approximation by Siddall
AD	Adaptive random search by Siddall
BC	Bit climbing algorithm by Davis
CO	Complex strategy of Box by Schwefel
DA	Davidon-Fletcher-Powell Strategy by Siddall
DF	Davidon-Fletcher-Powell strategy by Schwefel
DH	Dynamic hill-climbing algorithm
DO	Design of experiments based optimizer
DP	Davis, Swan and Campey with Palmer orthogonalizational by Schwefel
DS	Davis, Swan and Campey, with Gram-Schmidt orthogonalization by Schwefel
EP	Evolutionary programming
ES	Evolution strategy
FI	Repeated one-dimensional Fibonacci search by Schwefel
FL	Fletcher's 1972 method by Siddall
GA	Genetic algorithm based on clustering and sharing
GO	Repeated one-dimensional Golden section search by Schwefel
HO	Hooke and Jeeves direct search by Schwefel
JO	Jacobson and Oksman Method by Siddall
LA	Repeated one-dimensional Lagrangian interpolation search by Schwefel
MM	Schwefel's multi-membered evolution strategy by Schwefel
NA	E04UCF improved general purpose routine found in NAg maths library
NU	Powell routine in the Numerical Recipes cookbook
PB	Population-based incremental learning algorithm
PD	Powell direct search method by Siddall
PO	Powell's strategy of conjugate directions by Schwefel
RO	Rosenbrock's rotating co-ordinated search by Schwefel
SA	Simulated annealing
SE	Hooke and Jeeves direct search by Siddall
SI	Simplex strategy of Nelder & Meade by Schwefel
SM	Simplex strategy of Nelder & Meade by Siddall
2M	Schwefel's two-membered evolution strategy by Schwefel

Table A.1: The 31 optimization search routines employed from the OPTIONS DES.

Appendix B

Benchmark Test Problems

Some commonly used benchmark test problems already extensively discussed in the literature are used here in this research work. They represent classes of general constrained, unimodal and multimodal continuous parametric test problems. The first is the bound constrained unimodal Sphere function [72]. The second and third are the bound constrained Griewank [55] and Rastrigin functions, while the last is the bump or Keane function [31], which is subject to two nonlinear inequality constraint functions. Griewank, Rastrigin and Bump are all highly multimodal problems and considered as difficult for most search methods.

§B.1 Sphere Problem

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

Minimize:

$$F_{Sphere} = \sum_{i=1}^{n} x_i^2 \tag{B.1}$$

$$\textbf{Subject to}: -5.12 \leq x_i \leq 5.12, \quad i = 1, \dots, n$$

where $\mathbf{x} \in R^n$ is the vector of design variables, and n is the variables size, respectively. The surface of the Sphere function for $n = 2$ is shown in Figure B.1.

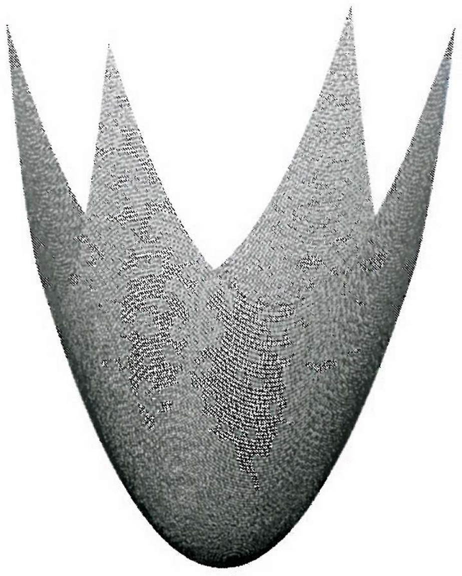


Figure B.1: 2-D Sphere function.

§B.2 Griewank Problem

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

Minimize:

$$F_{Griewank} = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} [\cos(x_i/\sqrt{i})] \tag{B.2}$$

$$\textbf{Subject to}: -600 \leq x_i \leq 600, \quad i = 1, \dots, n$$

where $\mathbf{x} \in R^n$ is the vector of design variables, and n is the variables size, respectively. Figure B.2 shows a one-dimensional slice of this function for $[-200, 200]^{10}$.

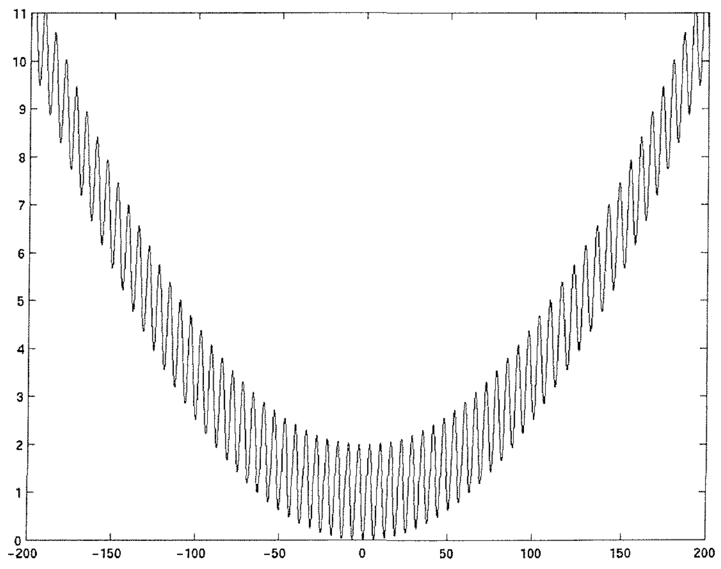


Figure B.2: 1-D slice of Griewank function.

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

§B.3 Rastrigin Problem

Like the Griewank function, the Rastrigin test problem is also a high dimensional multimodal function with many local minima and a global minimum located at (0, ..., 0). There are many other local minima surrounding the global minimum. It also has a very rugged landscape and is difficult to optimize. The bound constrained Rastrigin function was proposed in Rastrigin, 1974 and its generalized version by Muhlenbein is defined as

Minimize:

$$F_{Rastrigin} = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i)) \tag{B.3}$$

$$\textbf{Subject to}: -5.12 \leq x_i \leq 5.12, \quad i = 1, \ldots, n$$

where $\mathbf{x} \in R^n$ is the vector of design variables, and n is the variables size, respectively. The surface of the Rastrigin function for $n = 2$ is shown in Figure B.3.

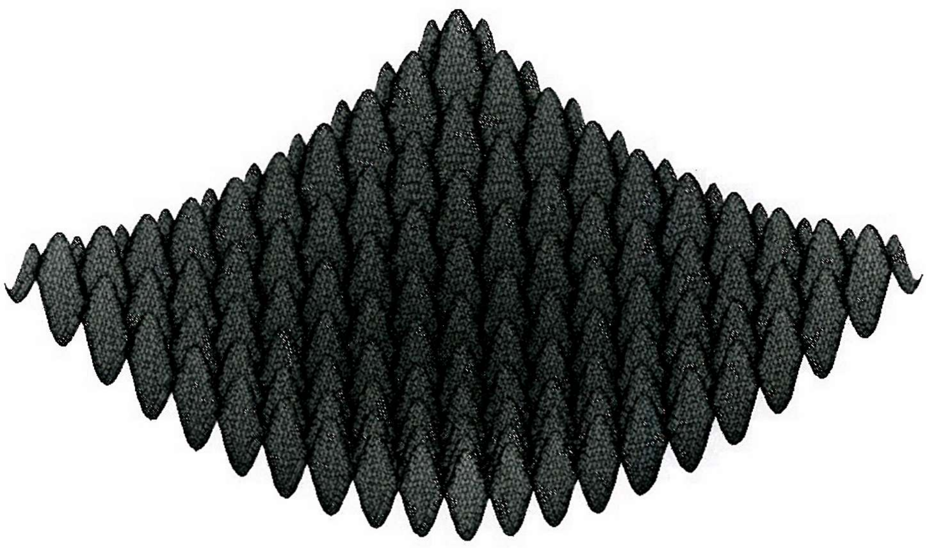


Figure B.3: 2-D Rastrigin function.

§B.4 Bump Problem

The Bump test problem was developed by Keane. It is the most difficult for optimizers to deal with among those considered in this dissertation. This function gives a highly bumpy surface where the true global optimum is usually defined by the product constraint. It is quite smooth but contains many peaks, all of similar heights and has strong inter-parameter linkage. Its main purpose is to test how methods cope with optima that occur hard up against the constraint boundaries commonly found in engineering design. These properties of bump make it suitable for the study of GA performance as well as in adaptive control of evolutionary optimization methods. The function is defined as

Maximize:

$$F_{Bump} = \frac{abs\ [\ \sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)\ ]}{\sqrt{\sum_{i=1}^{n}(ix_i^2)}} \tag{B.4}$$

$$\textbf{Subject to}: \prod_{i=1}^{n} x_i > 0.75 \ and \ \sum_{i=1}^{n} x_i < \tfrac{15n}{2},\ and$$

$$0.0 \leq x_i \leq 10.0, \quad i = 1, \ldots, n$$

where $\mathbf{x} \in R^n$ is the vector of design variables, and n is the variables size, respectively. The surface of the Bump function for $n = 2$ is shown in Figure B.4.

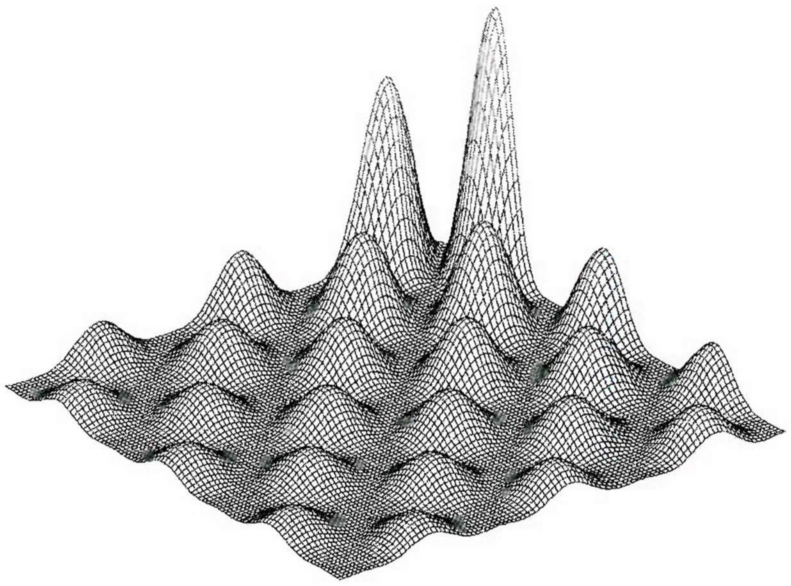


Figure B.4: 2-D Bump Constrained function.

Appendix C

Realistic Industrial Engineering Design Problems

The two realistic industrial engineering design problems have been used in the present program of research and are described here.

§C.1 Aircraft Wing Design Problem

The design of the wings for a transonic civil transport aircraft is an extremely complex task. It is normally undertaken over an extended time period and at a variety of levels of complexity. Typically, simple empirical models are use at the earliest stages of concept design, followed by ever more complex methods as the design process proceeds towards the final detailed stages.

In this dissertation, the design optimization of a civil transport aircraft wing for operation at Mach 0.785 and a Reynolds number of 7.3 million is considered. The objective is to design a wing with minimal wing drag D/q meters^2 as calculated by using tools with variety of levels of complexity, with target lift, wing weight, volume, pitch-up margin and root triangle layout chosen to be representative of a 220 seat wide body airliner. Figure C.1 shows a geometric view of streamlines over the transonic civil transport aircraft. The planform geometry is also shown in Figure C.2, while the definitions of the wing design variable, nonlinear inequality constraints and optimization conditions considered are given in Table C.1. The common parameters used to describe the wing design problem considered here consist of the free-stream velocity and viscosity and coefficient of lift of the wing together with a small number of overall wing geometry variables. The wing geometry is characterized by the planform shape of the wing together with several span-wise functions such as twist and thickness to chord ratio. Here, a wing design is represented by eleven parameters (i.e., eleven optimization design variables). In order to prevent the optimizer from driving the designs to unworkable extremes, several constraints are placed on the wings designed. These are the under-carriage bay length (which must be accommodated within the root to kink section of the wing), the fuel tank volume (which must be accommodated between the main spars within the wing), the wing weight and the pitch-up margin.

It is assumed that drag predictions/estimations accuracy increases with the complexity level. One of the principle tools used by British Aerospace in this area is the TADPOLE program, which is based on empirical models by Cousin and Metcalfe. Here, the wing drags predicted by this code are supplemented by the linearised potential method VSAERO. The VSAERO code is available as a commercial package. Both codes return the total drag coefficient defined by the wave drag due to the presence of shocks, viscous wake or profile drag due to the boundary layer and vortex or induced drag due to the tip vortex of the 3-d wing. A common approach to drag recovery is also implemented in the two codes.

In terms of computational cost, TADPOLE is the fastest code, taking only some 6 seconds to run, and simply returns drag values based on curve fits to previously analyzed wings making assumptions about the kinds of roof-top pressure profiles now commonly achieved in transonic wing design. VSAERO is a linearised potential code with coupled viscous boundary layer and as employed here, with added correction for compressibility [104]. It is computationally more expensive than TADPOLE and requires approximately 11 minutes of compute time per drag evaluation. However, it has the advantage of providing more accurate drag predictions provided Mach numbers are not too high.

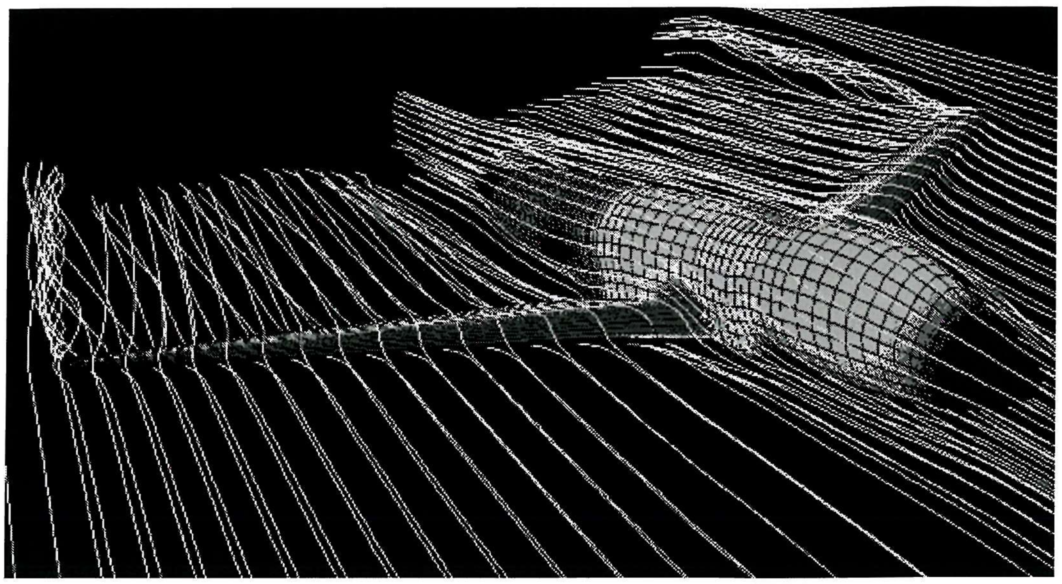


Figure C.1: Geometric view of streamlines over a transport aircraft.

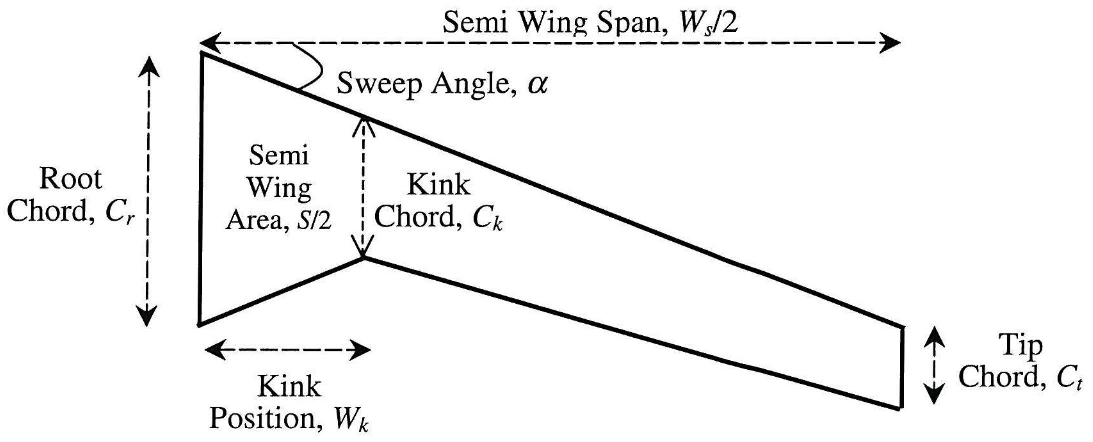


Figure C.2: Wing Planform Geometry.

11 Wing Design Variable Definitions		
Lower Limit	**Upper Limit**	**Quantity (units)**
100	250	Wing Area $(m^2), S$
6	12	Aspect Ratio, W_s^2/S
0.2	0.45	Kink position, $2W_k/W_s$
25	45	Sweep angle (degrees), α
0.4	0.7	Inboard taper ratio, C_k/C_r
0.2	0.6	Outboard taper ratio, C_t/C_r
0.1	0.18	Root thickness/chord, T_r/C_r
0.06	0.14	Kink thickness/chord, T_k/C_k
0.06	0.14	Tip thickness/chord, T_t/C_t
4.0	5.0	Tip wash (degrees)
0.65	0.85	Kink washout fraction
Four Design Constraint Functions		
2.5		Under-Carriage bay length
	135000	Wing weight (N)
40.0		Wing volume (m^3)
	5.4	Pitch-up margin

(c)

Table C.1: Optimization conditions for wing design parameters, constraints and respective limits.

§C.2 Ship Hull Form Design Problem

The preliminary design of a frigate hull is also considered in this work. Here, a frigate of some 3,300 tonnes deep displacement is designed so that its hull-form displays minimal resistance at a design speed of 30 knots in the deep condition, together with a payload consisting of a fuel dead-weight of 600 t at 0.8 m above the keel plus equipment totaling 175 t at a height which varies according to the hull and super-structure particulars.

A hull-form is defined by nineteen parameters listed in Table C.2. The design example and code (a statistically based hullform resistance code) used here are based on that described in Keane et al. [54]. With the routines used by Keane et al., only the 1) length to displacement ratio, 2) beam to draught ratio, 3) breadth to draught ratio, 4) non-dimensional position of maximum beam, 5) waterline flare at maximum beam and 6) ship type are required to define the hull form. From these primary parameters, all the nineteen parameters listed are generated or based on defaults. The defaults provide convenience by using type ship data to provide a useful starting point for design.

The objective of the optimization is to design a hull-form that has the smallest total resistance at design speed 30 knots (through altering the design variables) for a particular displaced volume, block coefficient, waterline flare at maximum beam and depth to draught ratio while meeting the design variable boundaries and design constraints. The constraints and design variables selected for this problem are summarized in Table C.3. The constraints adopted were the six on roll stability, an upper limit on the length to depth ratio to ensure against longitudinal strength problems and a minimum enclosed volume to guarantee payload capacity. Additionally waterline length, waterline beam, draught, depth, and maximum section coefficient were constrained, but within very wide limits. The design variables chosen were those strongly affect both resistance and stability. These were length to volume ratio, beam to draught ratio, prismatic coefficient and non-dimensional position of maximum beam. Resistance is calculated based on Holtrop and Mannen's power prediction method and is fully in accordance with that method, except that it does not include the propulsion analysis (instead a default or designer chosen propulsion coefficient is used). The method calculates the total resistance from the addition of frictional resistance, appendages resistance, wave resistance, additional pressure resistance of a bulbous bow near the surface, addition pressure resistance due to transom immersion and model ship correlation allowance, ignoring any interactions between them. Figure C.4 gives an example of a frigate hull-form displaying the geometry of its hullform before and after an optimization search for minimum resistance. The optimized design is shown to be considerably longer and thinner than the original design.

Ship Hull form Parameter Definitions	
Waterline Length	Forward Rake Angle
Waterline Beam	Aft Rake Angle
Draught	Rise of Floor Angle
Depth	Waterline Flare at Maximum Beam
Block Coefficient	Waterline Flare the Aft Perpendicular
Maximum Section Coefficient	Half Angle of Water-plane Entrance
Overall Beam at the position of Maximum Section	Upper Deck Half Angle of Water-plane Entrance
Non-dimensional Position of the Maximum Beam	Ratio of Waterline Transom Width to Waterline Beam
Length of the Parallel Middle Body	Ratio of Overall Transom Width to Overall Beam
Length of Keel	

Table C.2: Ship Hull-form Parameter Definitions.

Four Hull-form Design Variable Definitions		
Lower Limit	**Upper Limit**	**Quantity (units)**
5.0	12.0	Length to volume ratio
1.0	6.0	Beam to draught ratio
0.55	0.6	Prismatic coefficient
0.14	0.6	Non-dimensional position of maximum beam
13 Design Constraint Functions		
0.7		Initial GM (*m*)
0.09		Area below GZ curve from 0^0 to 40^0 (*m rad*)
0.03		Area below GZ curve from 30^0 to 40^0 (*m rad*)
0.055		Area below GZ curve from 0^0 to 30^0 (*m rad*)
30		Angle of Maximum GZ (*degrees*)
0.2		Maximum GZ (*m*)
	14.0	Length to depth ratio
	12000	Minimum enclosed volume
0.0	1000.0	Waterline Length
0.0	1000.0	Waterline Beam
0.0	1000.0	Draught
0.0	1000.0	Depth
0.4	1.0	Maximum Section Coefficient

Table C.3: Optimization conditions for Hull-form design parameters, constraints and respective limits.

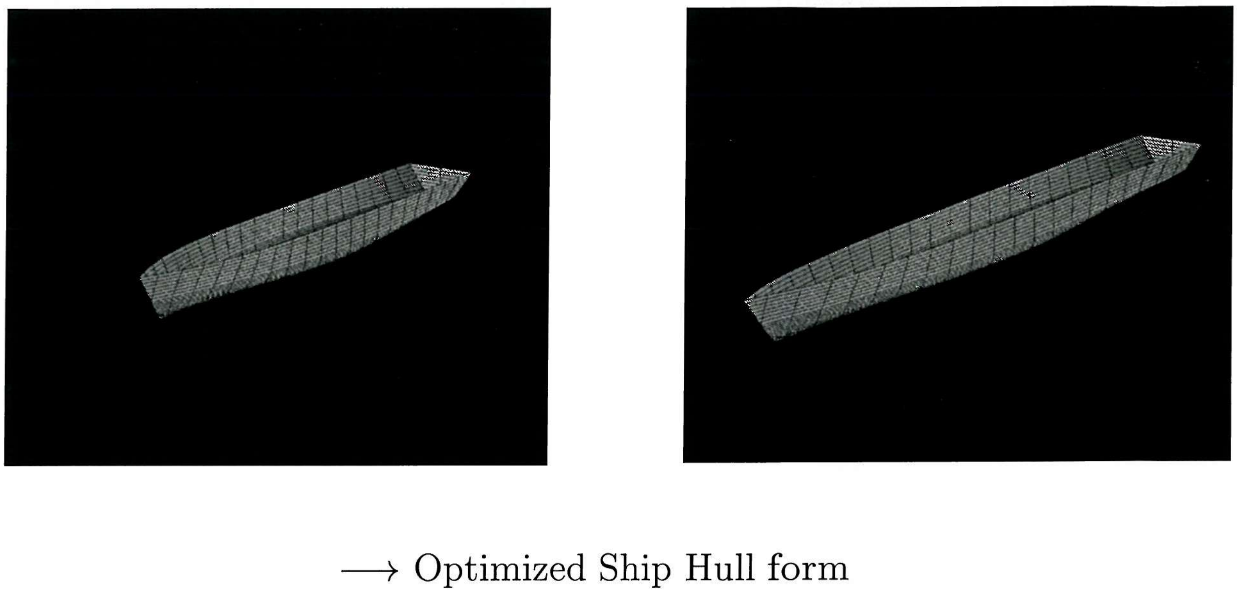


Figure C.3: Geometric view of a frigate hull-form and its optimized version.