

1999

Reducing the Computational Effort Associated with Evolutionary Optimisation in Single Component Design

VEKERIA, HARISH DHANJI

<http://hdl.handle.net/10026.1/2390>

<http://dx.doi.org/10.24382/3811>

University of Plymouth

All content in PEARL is protected by copyright law. Author manuscripts are made available in accordance with publisher policies. Please cite only the published version using the details provided on the item record or document. In the absence of an open licence (e.g. Creative Commons), permissions for further reuse of content should be sought from the publisher or author.

Reducing the Computational Effort Associated with Evolutionary Optimisation in Single Component Design

by

HARISH DHANJI VEKERIA

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

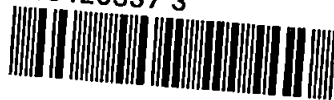
Plymouth Engineering Design Centre
School of Civil and Structural Engineering
Faculty of Technology

December 1999

UNIVERSITY OF PLYMOUTH	
Item No.	900 4263373
Date	18 MAY 2000 7
Class No.	T 620.0042
Contl. No.	X7040 68639
LIBRARY SERVICES	

VEK

900426337 3



REFERENCE ONLY

LIBRARY STORE

Reducing the Computational Effort Associated with Evolutionary Optimisation in Single Component Design

by

Harish Dhanji Vekeria

ABSTRACT

The dissertation presents innovative Evolutionary Search (ES) methods for the reduction in computational expense associated with the optimisation of highly dimensional design spaces. The objective is to develop a semi-automated system which successfully negotiates complex search spaces. Such a system would be highly desirable to a human designer by providing optimised design solutions in realistic time.

The design domain represents a real-world industrial problem concerning the optimal material distribution on the underside of a flat roof tile with varying load and support conditions. The designs utilise a large number of design variables (circa 400). Due to the high computational expense associated with analysis such as finite element for detailed evaluation, in order to produce “good” design solutions within an acceptable period of time, the number of calls to the evaluation model must be kept to a minimum. The objective therefore is to minimise the number of calls required to the analysis tool whilst also achieving an optimal design solution.

To minimise the number of model evaluations for detailed shape optimisation several evolutionary algorithms are investigated. The better performing algorithms are combined with multi-level search techniques which have been developed to further reduce the number of evaluations and improve quality of design solutions. Multi-level techniques utilise a number of levels of design representation. The solutions of the coarse representations are injected into the more detailed designs for fine grained refinement. The techniques developed include Dynamic Shape Refinement (DSR), Modified Injection Island Genetic Algorithm (MiiGA) and Dynamic Injection Island Genetic Algorithm (DiiGA). The multi-level techniques are able to handle large numbers of design variables (i.e. >100). Based on the performance characteristics of the individual algorithms and multi-level search techniques, distributed search techniques are proposed. These techniques utilise different evolutionary strategies in a multi-level environment and were developed as a way of further reducing computational expense and improve design solutions.

The results indicate a considerable potential for a significant reduction in the number of evaluation calls during evolutionary search. In general this allows a more efficient integration with computationally intensive analytical techniques during detailed design and contribute significantly to those preliminary stages of the design process where a greater degree of analysis is required to validate results from more simplistic preliminary design models.

List of Contents

1. INTRODUCTION	1
1.1 Design	1
1.1.1 Importance of Engineering Design	2
1.1.2 Current Design Methods	5
1.1.3 Optimum design	6
1.2 Evolutionary Optimisation	7
1.3 Objectives of the Work	9
1.4 Overview of Thesis	11
2. EVOLUTIONARY ALGORITHMS	14
2.1 Introduction	14
2.2 Evolutionary Algorithms	16
2.2.1 The Simple Genetic Algorithm (SGA)	19
2.2.2 Evolutionary Programming	23
2.2.3 Evolution Strategies	24
2.2.4 Genetic Programming	26
2.3 Exploration and Exploitation	27
2.4 Variations of the Evolutionary Algorithms	28
2.4.1 The CHC Adaptive Search Algorithm	30
2.4.2 Population-Based Incremental Learning (PBIL)	38
2.4.3 The Breeder Genetic Algorithm (BGA)	41
2.5 Summary	42
3. INTEGRATION OF EVOLUTIONARY ALGORITHMS WITH MATHEMATICAL MODELS	43
3.1 Structural Optimisation	43

3.2	Development of Evolutionary Software for Single Component Design.....	50
3.2.1	Genetic Representation of the Plate.....	52
3.2.2	Current Design Practice at the Company.....	54
3.2.3	The Evaluation Model (Complex Stress)	55
3.2.4	Multi-Criteria Optimisation	57
3.2.5	Two Dimensional Crossover	59
3.2.6	Development of Automated Design Tool	60
3.2.7	Selection of Design Variables.....	61
3.2.8	NISA FEA Software	63
3.3	Summary.....	66
4.	APPLICATION OF EVOLUTIONARY / ADAPTIVE ALGORITHMS.....	68
4.1	Application of Evolutionary Algorithms to the Plate Problem.....	68
4.2	Operator Settings for the Evolutionary Algorithms.....	70
4.2.1	Results for the Flat Plate Problems Utilising the Complex Stress Model	71
4.2.2	Results for the Flat Plate Problems Utilising FEA	86
4.2.3	Multiple Load Case Problem	92
4.3	Amended PBIL with a given population size (PBIL_POP).....	95
4.3.1	Comparison of Results for PBIL_POP Utilising the Complex Stress Model..	98
4.4	Other Techniques.....	100
4.5	Summary.....	101
5.	MULTI - LEVEL SEARCH STRATEGIES.....	104
5.1	Dynamic Shape Refinement (DSR).....	105
5.1.1	Mapping of Encoding	108
5.1.2	Discussion and Results for the DSR Technique	109
5.2	Parallel Genetic Algorithms.....	116
5.2.1	Micro-grain GA (mgGA).....	118

5.2.2	Fine-Grain GA's (fgGA's).....	118
5.2.3	The Distributed Genetic Algorithm (DGA).....	119
5.2.4	Cooperative Co-evolutionary Optimisation.....	120
5.3	The Injection Island GA (iiGA).....	121
5.3.1	Application of MiiGA's on the Plate Problem	123
5.4	Dynamic Injection Island GA (DiiGA).....	131
5.4.1	Application of Dynamic Injection Island GA to the Plate Problem utilising the Complex Stress Model.....	134
5.4.2	Application of Dynamic Injection Island GA on the Plate Problem utilising FEA	138
5.5	Summary.....	145
6.	MULTI AGENT SEARCH TECHNIQUES	147
6.1	Hybrid Search Techniques.....	147
6.2	Multi-Search Techniques.....	149
6.2.1	Application of Multi-Search Techniques to the Plate Problem	150
6.3	Distributed Search Techniques	154
6.3.1	Application of Distributed Search Techniques to the Plate Problem	154
6.4	Variable Complexity Modelling	160
6.5	Summary.....	162
7.	CONCLUSIONS.....	163
7.1	Conclusions.....	163
	References.....	170
	Publications	178

List of Figures

Figure 1-1 Asimow's three-phase design model [Balachandran 1993]	2
Figure 1-1: An example of a 3 dimensional landscape.....	15
Figure 2-2: Pseudo code for an EA.....	18
Figure 2-3: Types of evolutionary algorithms	18
Figure 2-4: One point crossover	20
Figure 2-5: Pseudo code for the canonical GA.....	22
Figure 2-6: Pseudo code for EP	24
Figure 2-7: Uniform Crossover applied to a real coded string	25
Figure 2-8: Example of a tree structure	26
Figure 2-9: Crossing over two parent trees by swapping sub-trees.	27
Figure 2-10: Pseudocode for CHC.....	32
Figure 2-11: Disruptive Crossover	34
Figure 2-12: the PBIL algorithm for a binary alphabet.....	40
Figure 2-13: Breeder Genetic Algorithm	42
Figure 3-1: Decoding process used by the plate problem	53
Figure 3-2: Representation of plate elements	55
Figure 3-3: Discretisation of element depth variation	56
Figure 3-4 : 2 dimensional crossover representation	60
Figure 3-5: A diagram to show element tapers.....	64
Figure 3-6: A diagram to show node id's	64
Figure 3-7: Explanation of the different types of variables	65
Figure 4-1: Simply supported plate with a central load	72
Figure 4-2: Performance comparison of the various search techniques (1 load case)	73
Figure 4-3: Comparison of PBIL and CHC GA (1 load case) (20 x 20 plate)	
max =24mm	75

Figure 4-4: Comparison of PBIL and CHC GA (1 load case) (20 x 20 plate)	
max =18mm	76
Figure 4-5: Comparison of PBIL and CHC GA (1 load cases) (24 x 24 plate)	
max =24mm	76
Figure 4-6: Comparison of PBIL and CHC GA (1 load case) (24 x 24 plate)	
max =18mm	77
Figure 4-7: Comparison in performance of different learning rates (1 load case).....	77
Figure 4-8: Simply supported plate with three load cases	79
Figure 4-9: Performance comparison of the various search techniques (three load cases)	
(20x20), dmax = 24mm, dmin = 8mm.....	82
Figure 4-10: Evolution of PBIL and CHC (20x20), dmax = 24, dmin = 8,	
load cases = 3	83
Figure 4-11: Evolution of PBIL and CHC (20x20), dmax = 18, dmin = 8,	
load cases = 3	83
Figure 4-12: Evolution of PBIL and CHC (24x24), dmax = 24, dmin = 8,	
load cases = 3	84
Figure 4-13: Evolution of PBIL and CHC (24x24), dmax = 18, dmin = 8,	
load cases = 3	84
Figure 4-14: 3 load case problem utilising different population sizes for the CHC algorithm	
(20000 evaluations) max 24, min 8.	85
Figure 4-15 : Simply supported plate with a central load (utilising FEA).....	87
Figure 4-16 : Best Average Fitness utilising FEA (1 load case, 48 variables)	89
Figure 4-17 : Best Average Weight utilising FEA (1 load cases, 48 variables)	89
Figure 4-18 : Best Average Fitness utilising FEA (1 load case, 200 variables)	91
Figure 4-19 : Best Average Weight utilising FEA (1 load case, 200 variables).....	91
Figure 4-20 : Simply supported plate with three load cases (utilising FEA).....	92

Figure 4-21 : Best Average Fitness utilising FEA (3 load cases, 48 variables).....	94
Figure 4-22 : Best Average Weight utilising FEA (3 load cases, 48 variables)	95
Figure 4-23: The amended PBIL algorithm (PBIL_POP)	97
Figure 4-24: Effect of different population sizes on amended PBIL algorithm for a single load case.....	98
Figure 4-25 : Effect of different population sizes on amended PBIL algorithm for single load case (learning rate =1.0).....	99
Figure 5-1: An Example of the Dynamic Shape Refinement Technique utilising Three Levels of Representation.....	106
Figure 5-2: The migration of encoding.....	109
Figure 5-3: Performance of stand-alone CHC, CHC DSR and PBIL for 20x20 plate (1 load cases). Max = 24mm.....	111
Figure 5-4: Performance of stand-alone CHC, DSR and PBIL for 24x24 plate (1 load case). max 24mm	113
Figure 5-5: Performance of stand-alone CHC, DSR and PBIL for 20x20 plate (3 load cases) max 24mm	114
Figure 5-6: Performance of stand-alone CHC and DSR for 20x20 plate (3 load cases) max 18mm.	115
Figure 5-7: Performance of stand-alone CHC and DSR for 20x20 plate (3 load cases) max 18mm during Latter stages of search	115
Figure 5-8 : Two examples of PGA topologies	120
Figure 5-9 : Michigan Injection Island Topology	122
Figure 5-10: Migration between subpopulations.....	125
Figure 5-11: Performance of stand-alone CHC GA and MiiGA CHC (20 x20 1 load case) max 24mm	127

Figure 5-12: Performance of stand-alone CHC GA and MiiGA CHC (20 x20 1 load case)	
max 24mm	127
Figure 5-13: Performance of stand-alone CHC GA and MiiGA CHC (24 x24 1 load case)	
max 18mm	128
Figure 5-14: Performance of stand-alone CHC GA and MiiGA CHC (20 x20 3 load cases,	
max. upper limit = 24mm)	129
Figure 5-15: Performance of stand-alone CHC GA and MiiGA CHC (24 x24 3 load cases,	
max upper limit =18 mm).....	130
Figure 5-16 : Grid Representation for the DiiGA.....	133
Figure 5-17: Performance of DiiGA against other techniques for 20x20 plate (3 load cases)	
max 24mm min 8mm.....	135
Figure 5-18: Performance of DiiGA against other techniques for 20x20 plate (3 load cases)	
max 18mm min 8mm.....	135
Figure 5-19: Performance of DiiGA against PBIL and CHC for 24x24 plate (3 load cases)	
max 24mm min 8mm.....	136
Figure 5-20: Performance of DiiGA against PBIL and CHC for 24x24 plate (3 load cases)	
max 18mm min 8mm.....	137
Figure 5-21 : Best Average Fitness utilising FEA (1 load cases, 48 variables).....	140
Figure 5-22 : Best Average Fitness utilising FEA (1 load cases, 48 variables).....	140
Figure 5-23 : Best Average Weight utilising FEA (1 load cases, 48 variables)	141
Figure 5-24 : Best Average Fitness utilising FEA (3 load cases, 48 variables).....	142
Figure 5-25 : Best Average Weight utilising FEA (3 load cases, 48 variables)	143
Figure 5-26 : Best Average Fitness utilising FEA (1 load case, 200 variables)	144
Figure 5-27 : Best Average Weight utilising FEA (1 load case, 200 variables).....	144
Figure 6-1 : Graph to show a comparison of the early stages of the search process for	
different search methods -(1 load case 24x24 plate max. 18mm, min. 8mm).....	151

Figure 6-2 : Graph to show a comparison of latter stages of the search process for different search methods -(1 load case 24x24 plate max. 18mm, min. 8mm)..... 151

Figure 6-3 : Graph to show a comparison of different search methods -(3 load cases 24x24 plate max. 18mm, min. 8mm)..... 152

Figure 6-4 : Grid Representation for the chc-pbil-pbil (c-p-p) configuration..... 155

Figure 6-5 : Graph to show a comparison of the early stages of the search process for different search methods -(1 load cases 24x24 plate max. 18mm, min. 8mm) 156

Figure 6-6 : Graph to show a comparison of latter stages of the search process for different search methods -(1 load cases 24x24 plate max. 18mm, min. 8mm)..... 157

Figure 6-7 : Graph to show a comparison of different search methods -(3 load cases 24x24 plate max. 18mm, min. 8mm)..... 158

Figure 6-8: Grid Representation for the pbil-chc-chc (p-c-c) configuration..... 159

List of Tables

Table 4-1 : Operator Settings for the Various Algorithms	70
Table 4-2 : Results for CHC for various problem cases utilising a single load case (no. of runs = 10)	74
Table 4-3 : Results for PBIL for various problem cases utilising a single load case (no. of runs = 10)	74
Table 4-4 : Results for CHC for various problem cases utilising three load cases (no. of runs = 10)	80
Table 4-5 : Results for PBIL for various problem cases utilising three load cases (no. of runs = 10)	80
Table 4-6 : Computational expense for individual FE evaluations utilising a single load case.....	88
Table 4-7 : Results for 48 variables 1 load case problem	88
Table 4-8 : Results for 200 variables 1 load case problem	90
Table 4-9 : Computational expense for individual FE evaluations utilising three load cases	93
Table 4-10 : Results for 48 variables 3 load case problem	93
Table 4-11: Amended PBIL (with population) (PBIL1) Population Size = 20	100
Table 4-12: Amended PBIL (with population) (PBIL1) (3 load cases) learning rate = 1.0	100
Table 5-1: Results for the DSR technique utilising various problem cases (no. of runs = 10)	112
Table 5-2 : Results for MiiGA utilising various problem cases (no. of runs = 10)	126
Table 5-3: Results for DiiGA utilising various problem cases (no. of runs = 10)	134
Table 5-4 : Computational expense for individual FE evaluations	139
Table 5-5 : Results for 48 and 200 variable problems	139

Table 6-6: Results for CHC_PBIL utilising various problem cases (no. of runs = 10) 150

Table 6-7: Results for PBIL_CHC utilising various problem cases (no. of runs = 10) 150

Table 6-8: Results for CHC-PBIL-PBIL utilising various problem cases
(no. of runs = 10) 157

Table 6-9: Results for PBIL-CHC-CHC utilising various problem cases
(no. of runs = 10) 159

ACKNOWLEDGEMENTS

I would like to thank the following people:

My director of studies Dr Ian Parmee whose help, encouragement and influence has made this thesis possible. I am deeply indebted to him for his guidance, insight and enthusiasm, which have provided the framework for the development of new ideas and approaches.

My supervisors Professor Geoff Bullock, Dr Yacob Rafik and Mr David Easterbrook for their constructive help and advise.

Mr Jim Pearce and numerous members of the Teaching Company Centre for their help and guidance.

Martin Beck, Chris Bonham, Dragan Cvetkovic, Carlos Coello Coello, George Bilchev, Andy Watson and Joanne Levers, all of whom have helped me in various ways during my life as a research assistant.

My family and friends for their encouragement and unwavering support.

My wife Amrat, for her moral support, love and motivation and my daughter Kavina who has been a welcome distraction during the writing up of this thesis.

AUTHOR'S DECLARATION

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

The first two years of the study was financed jointly by the EPSRC and DTI as part of a Teaching Company Programme. The remainder of the study was financed with the aid of a research assistantship from the EPSRC and carried out at Plymouth Engineering Design Centre.

Relevant scientific seminars and conferences were regularly attended, and a number of papers prepared for publication. In addition work was also presented at workshops at Rutherford Appleton Laboratories and British Telecommunications Laboratories. Close links were maintained with Industry and the Genetic Algorithms Research and Applications Group (GARAGe) at Michigan State University.

Publications :

1. Vekeria H.D, Parmee I.C. "Structural Shape Optimisation for Highly Dimensional Problems". In M. Polkinghorne (editor), Applications of Artificial Intelligence for Technological and Business Processes. Technology Transfer Series. Volume 2, University of Plymouth, 1996. ISBN 0 905227 57 3.
2. Vekeria H.D, Parmee I.C. "The Use of a Co-operative Multi-Level CHC GA for Structural Shape Optimisation", Proceedings of the European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, September 1996.
3. Vekeria H.D, Parmee I.C. "Reducing Computational Expense Associated With Evolutionary Detailed Design" Proceedings of the IEEE International Conference on Evolutionary Computation. Indianapolis, USA, April 1997.
4. Vekeria H.D, Parmee I.C. "Structural Design Using Multi-Level Evolutionary Strategies" Proceedings of The Mouchel Centenary Conference on Innovation in Civil & Structural Engineering" Cambridge, England, August 1997.
5. Parmee I.C, Vekeria H.D. "Co-operative Evolutionary Strategies for Single Component Design". Proceedings of the Seventh International Conference on Genetic Algorithms, Michigan State University, Michigan, July 1997
6. Vekeria H.D, Parmee I.C. "Evolutionary Search for Highly Dimensional Engineering Design Problems" Proceedings of the International Conference on Engineering Design. Tampere, Finland, August 1997.
7. Parmee I.C, Vekeria H.D. "Evolutionary / Adaptive Search Strategies and Model Representation in Engineering Design". Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Berlin, August 1997
8. Parmee I.C, Vekeria H.D. "Evolutionary / Adaptive Search Strategies for the Differing Requirements of Preliminary and Detailed Engineering Design". World Congress on Structural Engineering, San Francisco, July 1998.
9. Parmee I.C, Vekeria H.D, Bilchev G. "The Role of Evolutionary and Adaptive Search During Whole System, Constrained and Detailed Design". Journal of Engineering Optimisation, Vol. 29, Gordon & Breach Science Publishers, 1997, pp. 151-176.
10. Parmee I.C, Vekeria H.D. "Reducing Lead Time for Single Component Design via Appropriate Adaptive Search Strategies". Journal of Engineering Valuation and Cost Analysis. Sept. 1997., Vol. 2, Num. 3. (Special issue on Engineering Valuation and Computational Intelligence)
11. Vekeria H.D, Parmee I.C. "Improving the Performance of Evolutionary Algorithms Using Injection Island Strategies". Poster Proceedings of the International Conference on Adaptive Computing in Design and Manufacture", PEDC, University of Plymouth, Plymouth, April 1998.

Signed H. Vekeria

Date 10TH APRIL 2000

1. INTRODUCTION

1.1 Design

Pressure for more economic designs in industry has increased due to growing competition in the market place and advances in technology. Designers are constantly challenged to produce designs that meet all the performance specifications and yet can be produced at low cost. French [1994] describes design as a :

"... purposeful activity directed towards the goal of fulfilling human needs. It is also a typical intellectual task that human beings perform. Although things are built by many creatures, the nest of a bird, the dam of a beaver, the web of a spider are some examples, these creations are however instinctively produced. It is not the spider that decides the fundamental structure of it's web, but the programmed instinctive instructions that evolution has provided for the spider. Only humans have the ability to go beyond instinct and consciously create designs ".

Figure 1-1 illustrates three major stages of the design process : analysis, synthesis and evaluation [Balachandran 1993]. At the analysis stage the designer participates in the collection and classification of all relevant information to the problem and defines the objectives. In the synthesis phase the designer then seeks to formulate a potential solution. The potential solution is then considered at the evaluation phase, where it is judged against some criteria in order to select the most suitable solution. Failure to meet the required

criteria at the evaluation stage may necessitate a return to the analysis stage where the decisions must be appropriately corrected and then the whole process repeated.

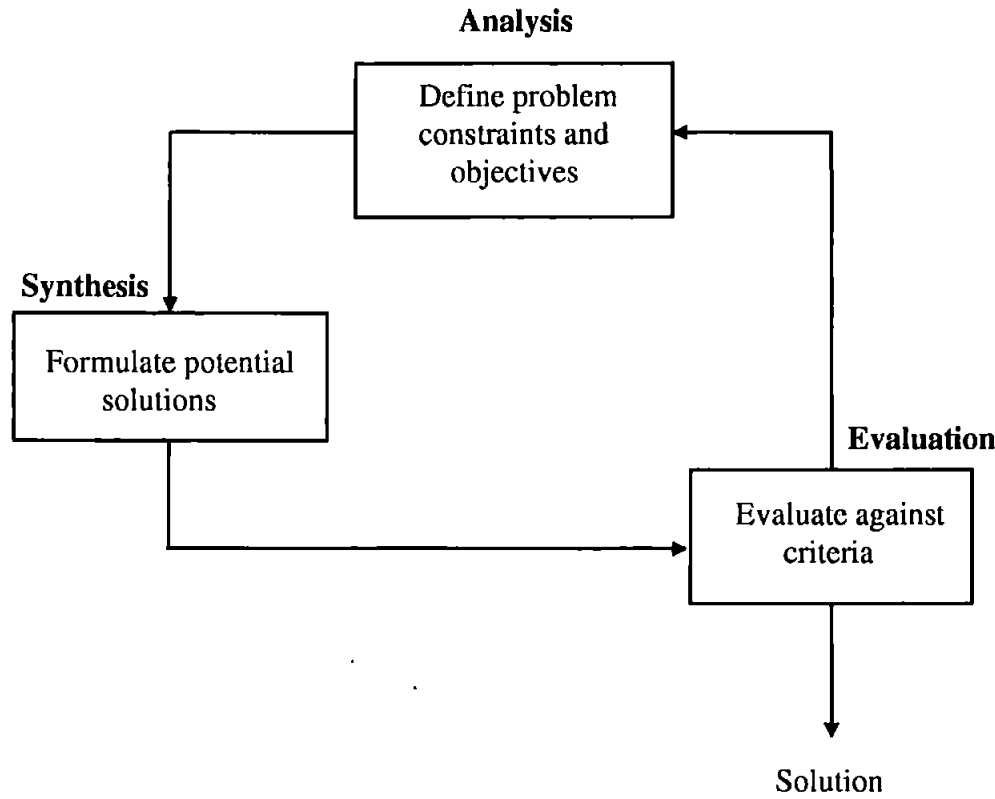


Figure 1-1 Asimow's three-phase design model [Balachandran 1993].

1.1.1 Importance of Engineering Design

There is no universally accepted definition of Engineering Design. Dym and Levitt, [1991] summarises the process of engineering design as follows

“ Engineering design is the systematic, intelligent generation and evaluation of specifications for artefacts whose form and function achieve stated objectives and satisfy specified constraints ”.

The design process has several distinct stages [Pahl and Beitz, 1984, Dym and Levitt, 1991] :

1. Conceptual or Preliminary Design takes the statement of the problem and generates broad solutions to it in the form of schemes. For example French [1985] describes it as :

“.. the phase that makes the greatest demands on the designer, and where there is considerable scope for improvements. Engineering science, practical knowledge, production methods and commercial aspects are all combined to make the most important decisions “

Maher highlights the explorative aspect of design, especially during conceptual design. Maher states that designers do not always have a complete problem description before commencing a design synthesis. During conceptual design, designers play around with ideas to get more understanding about the problem rather than focus on finding a design solution. Design therefore is an iterative process of searching the design problem space as well as the solution space. This phenomenon is referred to as exploration in design.

Gero [1993] defines exploration as follows :

“ Exploration in design can be characterised as a process which creates new design spaces or modifies existing design state spaces. “

Gero [1993] continues to suggest that exploration and search are related and that exploration precedes search. Maher and Poon [1995] also relates search as a part of exploration but highlights that search and exploration are not the same:

“ .. search becomes exploration where the focus of the search changes as the process continues .. “

2. Detailed design concerns the refinement of choices made in preliminary design. As we are further down in the design search tree, the decompositions and their interactions are better understood and therefore more easily manageable.
3. Analysis concerns performing calculations or deductions needed to assess whether the design satisfies other, less obvious specifications and constraints. This phase of the design process may be computationally expensive, especially if large complex programs such as finite element analysis are used.
4. Evaluation consists of predicting the behaviour of the current design by deriving the values of all relevant performance measures in order to determine whether the stated specifications and objectives of the current design are acceptable.
5. Iterative redesign concerns the redesign of products if the results are deemed unsatisfactory.

Many engineering design problems involve the quick development of the “best” or “near best” design in a complex engineering domain with the given material, technological and economical constraints. Here “best” can, for example refer to low cost or high quality. More often or not there is a trade-off. Solving this problem has tremendous commercial benefits.

Automated or semi-automated engineering design is likely to be extremely beneficial to commercial enterprises as it would allow them to reduce the cost of producing new designs, to produce better designs than their competitors, and to bring new concepts to the production line faster. All of these improvements would allow the companies that take advantage of them to increase market share and profitability.

1.1.2 Current Design Methods

The human design process is traditionally a prolonged, iterative one. Most complex engineering design is performed manually. Engineers often use computer aided design (CAD) software to create and edit their designs, and software such as finite element (FEA) to analyse their designs. The process often works as follows: the engineer initially creates a conceptual or preliminary design, which is then analysed, using appropriate software to determine which parts must be further redesigned or optimised. Further changes are then made using the CAD software. Iteration will continue until a design is developed that meets the original specification or is deemed acceptable. In some cases the design process may be aborted altogether due to time constraints, consequently resulting in a considerable waste of time and money. This design-evaluate-redesign process is extremely slow. It requires large amounts of calendar time, moreover it sometimes fails to produce an optimal or near optimal design solution. The longer the design process the more costly it becomes. Often, the intuitive redesign methods fail since the available design options are few and difficult to determine. Thus, computer software which helps to automate and speed up this process is highly beneficial. Moreover, with the cost of computers decreasing and the available computation power increasing, the computer is becoming an essential tool for the designer.

Whilst there is an abundance of computer aided design software and numerous analytical tools, software which automates the design process (e.g., identifying new designs or improving existing designs) is currently less common. Commercial optimisation software is often only capable of handling a relatively small number of design variables, which limits their use in industry to small problems. For example the ANSYS (release 5.3) finite element optimisation software is only capable of handling a maximum of 10 variables.

1.1.3 Optimum design

The purpose of design optimisation is to algorithmically search for the “best” or “near best” design solution relative to an overall criterion. Beightler [1979] et al describes what we are trying to accomplish when we optimise :

“ Man’s longing for perfection finds expression in the theory of optimisation. It studies how to describe and attain what is Best, once one knows how to measure and alter what is good or bad... Optimisation theory encompasses the quantitative study of optima and methods for finding them “.

“Optimum design” is defined as the design that is feasible and also superior to a number of other feasible alternatives [Balachandran 1993]. Papalambros and Wilde [1988] identify four steps in the design optimisation approach:

1. The selection of a set of variables to describe the design alternatives.
2. The selection of an objective (criterion), expressed in terms of the design variables, which we seek to minimise or maximise.

3. The determination of a set of constraints, expressed in terms of design variables, which must be satisfied by any acceptable design.
4. The determination of a set of values for the design variables, which minimise (or maximise) the objective, while satisfying all constraints.

An optimum design can be obtained in two ways [Balachandran 1993]:

1. By an iterative process, or
2. By solving an optimisation problem

In the first approach, the design is improved through repeated modification and the values of the design variables are changed or fixed sequentially. In the latter approach, all the design variables are determined simultaneously so as to satisfy a set of constraints and optimise a set of objectives. These objectives may coexist, conflict or be independent.

1.2 Evolutionary Optimisation

Evolution is a process of change over time. The driving force behind this change, as described by Darwin [Darwin, 1859], is natural selection. Evolutionary algorithms are inspired by and based upon evolution in nature. These algorithms typically use an analogy with natural evolution to perform search by evolving solutions to problems. Instead of working with one solution at a time in the search space, these algorithms consider a large collection or population of solutions at once. By maintaining a population of well adapted sample points, the probability of arriving at a sub-optimal solution is reduced. In any population, there are always individuals who are fitter than others. Such individuals live longer and thus get the chance to produce more offspring than individuals of average

fitness. Conversely, unfit individuals, or individuals poorly adapted to their environment, tend to produce less offspring than individuals of average fitness. In this way, the genes, and hence the characteristics, of fitter individuals propagate through a population, until, assuming those characteristics are better than others currently in the population, all of the population contains those characteristics.

The Genetic Algorithm (GA) is probably the best known and the most widely used of all evolutionary based algorithms. GA's were developed by Holland over twenty five years ago in an attempt to explain the adaptive processes of natural systems and to design artificial systems based upon these natural systems [Holland 1975].

Evolutionary algorithms are well suited to tackling highly complex optimisation problems [Goldberg 1989, Davis 1991]. Baeck et al [1997] argues that

“ The most significant advantage of using evolutionary search lies in the gain of flexibility and adaptability to the task at hand, in combination with robust performance and global search characteristics. They should be understood as a general adaptable concept for problem solving, especially well suited for solving difficult optimisation problems, rather than a collection of related and ready-to-use algorithms “.

Miles and Moore [1997] comment that the GA due to it's greater power and flexibility is better suited to design tasks than other adaptive learning techniques such as Neural Networks. Evolutionary design techniques have been around since the 1950's [Box, 1957], however, the potential of these technologies within the engineering design domain is only

now being realised. This is largely due to the computational expense associated with such population-based search strategies. Spurred by the recent advances in powerful desktop computing, there is growing interest in their realistic application to real-world problems, although computational requirement still represents a significant problem in some application areas [Parmee 1994] [Parmee, Vekeria & Bilchev 1997]. The Plymouth Engineering Design Centre is very active in the application of evolutionary search techniques to complex design problems and their integration with current design practice [Parmee 1994, Parmee 1996b].

1.3 Objectives of the Work

The aim of this work is to develop a system which is capable of creating design solutions automatically. By combining the automatic optimisation of a design alongside evaluation software which would automatically analyse the quality of the designs, there would be little or no need for human intervention in the design process. Such an automated system would be highly desirable to a human designer. It would speed up the whole design process by providing optimised design solutions to a problem. As the system is not restricted to pre-conceived ideas on certain ways of doing things, like that of a human designer, it would also be capable of delivering radically different design solutions.

The work will focus on the shape optimisation of flat plates. As mentioned earlier, computational expense is one of the major drawbacks of population based optimisation methods, however the advent of parallel processing and more efficient computing capability has helped to speed up this process. Finite element or other complex analysis techniques are commonly used to determine if a design will perform as expected. But, these

analyses are computationally very expensive to carry out and their use in an iterative manner for determining the optimal design is prohibitively expensive.

The rapid increase in computer power has also brought about an increase in the complexity of problems being tackled in the field of engineering design. There is a desire by companies to deal with increasingly more complicated problems. Unfortunately by increasing the accuracy in models employed, along with the use of appropriate algorithms, the resultant computations can often be of very high dimension, leading to practical difficulties in solving (“the curse of dimensionality”). A combination of computational expense (calls to analysis model), high-dimensionality and multi-modality presents a considerable challenge for any optimisation algorithm. This research therefore proposes methods by which these problems may be overcome. The main objectives of the research are therefore:

- Integration of evolutionary methods with a structural analysis model.
- Development of a system which is capable of structural optimisation
- Analysis of the performance of different evolutionary algorithms for shape optimisation of the flat plate
- Development of techniques for reducing the overall computational expense during population based search.
- Development of a technique for handling high dimensionality.

1.4 Overview of Thesis

The thesis examines the application of evolutionary techniques for the optimisation of roof tiles (referred to as the plate optimisation problem throughout the thesis). Several techniques are examined (CHC, PBIL, BGA and SGA) in order to determine their relative performance in minimising calls to the model and the overall quality of design solutions.

Due to limitations of the individual algorithms in handling high dimensionality (large numbers of design variables), several multi-level techniques were developed which included Dynamic Shape Refinement (DSR), Modified Injection Island Genetic Algorithm (MiiGA), Dynamic Injection Island Genetic Algorithm (DiiGA). The techniques exploit the differing levels of a problem representation. Problem dimensionality is increased as search progresses. These techniques were developed at the Plymouth Engineering Design Centre by Vekeria and Parmee [1997]. The MiiGA and DiiGA are extensions of the Injection Island GA (iiGA) developed at Michigan State University [Goodman et. al., 1997].

Based on the performance characteristics of the individual CHC and PBIL algorithms and the ability of the DiiGA technique in providing the capability of handling higher numbers of variables. Multi level co-evolution of the CHC and PBIL techniques is proposed to take advantage and further improve performance characteristics.

Chapter 1 has highlighted the design process and draws attention to some of the problems encountered in the design process and the benefits of automating parts of the design process.

Chapter 2 provides an introduction to the area of evolutionary computation. The chapter highlights common attributes of the various evolutionary techniques of particular relevance to engineering design processes. The chapter also provides a detailed discussion of some high performance evolutionary algorithms. Some of the algorithms detailed in this chapter are used in successive chapters on a structural optimisation problem.

Chapter 3 provides a literature review concerning the application of GA's to structural optimisation and shows that the area is receiving considerable interest. The chapter also discusses the development of software utilising a CHC genetic algorithm for the optimisation of a real world structural plate optimisation problem. The work was undertaken during a two year Teaching Company Programme. Two types of model are discussed, the first is based on bending moment and complex stress theory and the second on finite element analysis.

Chapter 4 provides a comparison in performance of different evolutionary algorithms on the plate optimisation problem. Results for the different types of evolutionary algorithms discussed in chapter 2 in relation to the plate problem are presented. Some of these techniques play a significant role in the thesis by guiding the research down certain avenues and laying the foundations for the development of various techniques.

Chapter 5 details some techniques for tackling some of the problems highlighted in chapter 4, concerning computational expense and problems with dimensionality. Methods that make use of different levels of representation for a problem are discussed. A comparison of the different methods that were developed which utilise multi-level representation is provided.

Chapter 6 discusses how hybrid approaches may provide further improvements in design performance. Distributed search techniques are proposed which take advantage of both different search techniques and multi-level representation.

Chapter 7 provides conclusions of the approaches that have been taken by drawing together the previous chapters, areas of further research are also discussed.

2. EVOLUTIONARY ALGORITHMS

2.1 Introduction

This chapter provides an introduction to the area of evolutionary optimisation. There are many GA variants which have been developed to improve the efficiency of evolutionary search for different problem classes. Several methods are discussed in detail. These techniques are used in successive chapters for the shape optimisation of roof tiles.

Optimisation has been studied for many years. Many methods have evolved and are detailed in a sizeable literature, with each method having advantages and disadvantages. Consider the 3 dimensional landscape of figure 2.1. Assuming we are maximising the solution, a traditional optimisation method such as a hill climber would climb the nearest hill from it's initial starting point. However, if the evaluation function defines a multi-modal landscape over the search space, then, depending upon the initial position in space the method may halt on some local optima of the space. Methods such as a random or an exhaustive search may overcome these problems. These methods are however computationally expensive and therefore better suited to small problems.

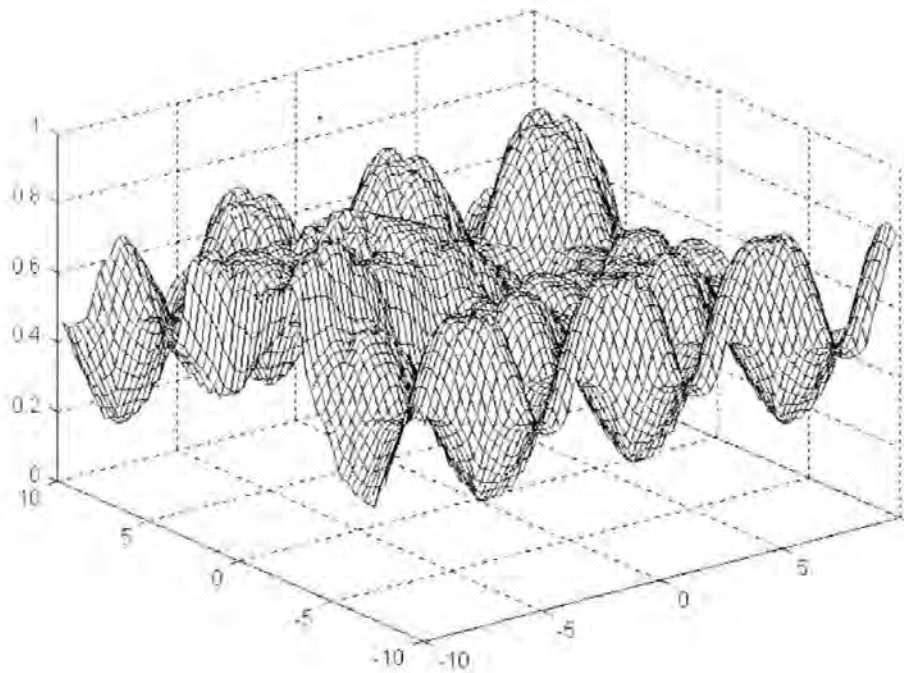


Figure 2-1: An example of a 3 dimensional landscape

Many engineering design problems cannot be tackled with classical optimisation methods. As such methods are severely restricted in their application due to the possibility of premature convergence on a local extrema, they are also limited to problems that have small search spaces if practical search times are required. Where classical techniques fail GA's may prove to be more successful especially when negotiating complex engineering design domains [Parmee 1994, Parmee 1996b]. Genetic Algorithms differ from traditional optimisation and search procedures in four ways [Goldberg 1989] :

1. GA's operate upon a coding of the parameter set, not the parameters themselves.
2. GA's search from a population of points, not a single point.
3. GA's use payoff (objective function) information, not derivatives or other auxiliary knowledge.
4. GA's use probabilistic transition rules as opposed to using deterministic rules.

2.2 Evolutionary Algorithms

Evolutionary algorithms (EA's) (Figure 2-2) are techniques for search and optimisation. They are based on the philosophy of natural selection, the driving process for the emergence of complex and well adapted organic structures. Like natural selection, EA's maintain a population of individuals. The *population* of structures evolve according to rules of selection and actions of "search operators", (or *genetic operators*), such as *recombination* and *mutation*. By manipulation of the genetic structure of these individuals (*genotypes*), EA's evolve progressively better *phenotypes*, the physical expression of a genotype i.e. the system. EA's treat their populations as though they were made up of living creatures. A single individual of a population is affected by other individuals of the population (e.g., by food competition, and mating). Each individual in the population receives a measure of it's *Fitness* in the environment. Reproduction focuses attention on high fitness individuals, thus exploiting the available fitness information. Recombination and mutation perturb those individuals, providing general heuristics for *exploration*. The better the individual performs under these conditions (exploration versus exploitation) the greater is the chance for the individual to live longer and generate more offspring who inherit the parental genetic information. Over the course of the evolution, this leads to a penetration of the population with the genetic information of individuals of above average fitness. The stochastic nature of reproduction leads to a permanent production of novel genetic information and therefore, to the creation of differing offspring.

Common attributes of the various evolutionary techniques of particular relevance to engineering design processes include [Parmee, Vekeria & Bilchev 1997]:

- Requirement for little, if any, apriori knowledge relating to the search environment.
- Excellent exploratory capabilities especially where population-based search is considered.
- Ability to avoid local optima. The stochastic nature of the various algorithms combined with continuing random sampling of the search space can prevent convergence upon a local sub-optima.
- Ability to handle high dimensionality.
- Robustness across a wide range of problem class.
- The provision of multiple good solutions.
- Ability to locate the region of the global optimum solution

There are many evolutionary based algorithms (Figure 2-3). The variations have differing philosophies on how to algorithmically model evolution. Evolutionary strategies (ES) and Evolutionary programming (EP) refer to two computational paradigms that utilise a population based search. There are many variants of evolutionary algorithms, their main differences lie in the [Baeck et. al. 1997] :

- Representation of individuals;
- Design of the variation operators (mutation and/or recombination);

Selection/reproduction mechanism.

```

// start with an initial time
t := 0;

// initialize a usually random population of individuals
initpopulation P (t);

// evaluate fitness of all initial individuals in population
evaluate P (t);

// test for termination criterion (time, fitness, etc.)
while not done do

    // increase the time counter
    t := t + 1;

    // select sub-population for offspring production
    P' := selectparents P (t);

    // recombine the "genes" of selected parents
    recombine P' (t);

    // perturb the mated population stochastically
    mutate P' (t);

    // evaluate it's new fitness
    evaluate P' (t);

    // select the survivors from actual fitness
    P := survive P,P' (t);
od
end EA.

```

Figure 2-2: Pseudo code for an EA

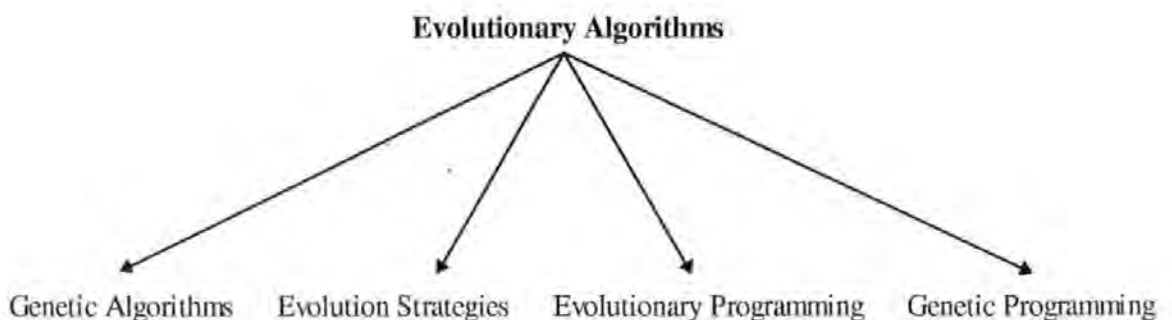


Figure 2-3: Types of evolutionary algorithms

2.2.1 The Simple Genetic Algorithm (SGA)

The SGA [Goldberg, 1989] (Figure 2-5) is the canonical genetic algorithm, it is composed of three operators:

1. Reproduction
2. Crossover
3. Mutation

Generally the SGA comprises of a population of initially randomly generated variable parameter sets (chromosomes). Variable values are generally represented in binary form although real-number representation can also be maintained.

The performance of each chromosome is determined by a mathematical model (fitness function) of the system under design.

2.2.1.1 Crossover

Crossover is applied to the reproduced chromosomes in order to imitate sexual reproduction. Crossover is usually applied with a high probability, with information being exchanged randomly between selected parent chromosomes. Simple crossover is implemented by choosing a random point in the selected pair of strings and exchanging the sub-strings defined by that point (Figure 2-4). The crossover operator thus mixes information from two parent strings producing offspring made up of parts from both parents. Crossover provides an exploratory capability. The canonical GA operates on a fixed-length binary string.

Crossover is an extremely important component of the genetic algorithm. Many genetic algorithm practitioners believe that if the crossover operator is not used, the result is no longer a true genetic algorithm. The same claim has not been made for mutation [Davis 1991].

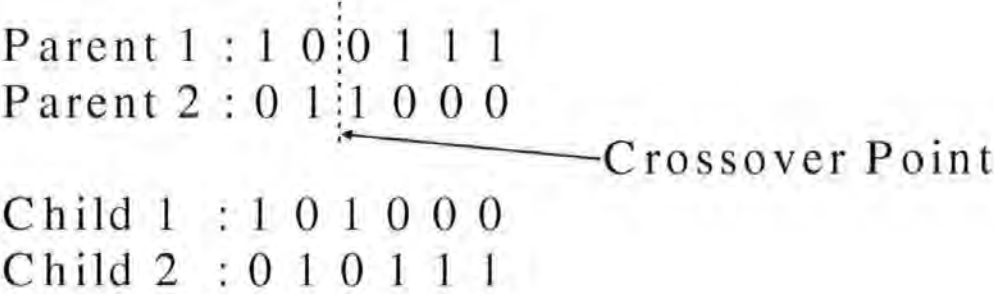


Figure 2-4: One point crossover

2.2.1.2 Reproduction

Reproduction in which individual strings are copied into the next generation is dependant upon the relative fitness of each chromosome. Those of high fitness have a greater probability of multiple reproduction whilst those of low fitness have a greater probability of rejection. The roulette wheel is a widely used method of selection. Roulette wheel selection may be viewed as allocating pie-shaped slices on a roulette wheel to population members, with each slice proportional to the member's fitness. The effect of the roulette wheel is to return a randomly selected parent. Although the selection procedure is random, the probability of each parent being selected is directly proportional to its fitness. On balance, over a number of generations the algorithm will drive out the least fit members and contribute to the spread of the genetic material of the fittest population members.

(fitness-proportionate selection). However it is still possible that the worst performing member could still be selected.

Another commonly used selection method is tournament selection. A basic form of tournament selection randomly selects two strings from the current population and their fitness values are compared. The string with the best fitness is placed in the intermediate population. This process is then repeated N times, where N is the population size.

2.2.1.3 Mutation

Like crossover the mutation operator is applied to the reproduced chromosomes in order to imitate biological evolution. Mutation in contrast is applied at a very low probability, it injects information into the genetic pool by mutating randomly selected bits. Mutation provides a small amount of random search, and helps to ensure that no point in the search space has a zero probability of being examined. It prevents premature convergence by ensuring that the genetic pool does not stagnate.

In "An Analysis of the behaviour of a class of genetic adaptive system" [De Jong, 1975] a study was performed of genetic algorithms in function optimisation. A series of parametric studies across a five-function suite of problems suggested that good GA performance requires the choice of a high crossover probability, a low mutation probability (inversely proportional to the population size), and a moderate population size.

Evolutionary algorithms are directed search techniques, but are inherently random. For this reason not every run is guaranteed to produce a satisfactory individual. The GA may need to be run several times (10 or more) utilising differing initial populations. It is therefore

important that robust GA's are developed especially for problems which are computationally expensive in order to keep run times to a minimum.

```
// start with an initial time
t := 0;

// initialize a usually random population of individuals
initpopulation P (t);

// evaluate fitness of all initial individuals of population
evaluate P (t);
// test for termination criterion (time, fitness, etc.)
while not done do

    // increase the time counter
    t := t + 1;

    // select a sub-population for offspring production
    P' := selectparents P (t);

    // recombine the "genes" of selected parents
    recombine P' (t);

    // perturb the mated population stochastically
    mutate P' (t);

    // evaluate it's new fitness
    evaluate P' (t);

    // select the survivors from actual fitness
    P := survive P,P' (t);
od
end GA.
```

Figure 2-5: Pseudo code for the canonical GA

2.2.2 Evolutionary Programming

Evolutionary programming (EP) (Figure 2-6) is described in an early book by Fogel, Owens and Walsh [Fogel et al 1966]. It is one of the earliest EAs. The basic EP method involves 3 steps which are repeated until a threshold for iteration is exceeded or an adequate solution is obtained.

1. Choose an initial POPULATION of trial solutions at random. The number of solutions in a population is highly relevant to the speed of optimisation.
2. It is in the creation of the new generations that EP differs from most other EA's, for it does not employ any crossover. Each solution is replicated into a new population. Each of these offspring solutions are mutated according to a distribution of MUTATION types, ranging from minor to extreme with a continuum of mutation types between. The severity of MUTATION is judged on the basis of the functional change imposed on the parents.
3. Each offspring solution is assessed by computing it's fitness. Typically, a stochastic tournament is held to determine N solutions to be retained for the population of solutions, although this is occasionally performed deterministically. There is no requirement that the population size be held constant, however, nor that only a single offspring be generated from each parent.

Unlike GA's, EP does not rely on fixed length structures, but permits individuals in the initial population to be of different lengths. These individuals are then tested, and parents for the subsequent generation are selected in a non-deterministic manner.

```

// start with an initial time
t := 0;

// initialise a usually random population of individuals
init. population P (t);

// evaluate fitness of all initial individuals of population
evaluate P (t);

// test for termination criterion (time, fitness, etc.)
while not done do

    // perturb the whole population stochastically
    P'(t) := mutate P (t);

    // evaluate it's new fitness
    evaluate P' (t);

    // stochastically select the survivors from actual fitness
    P(t+1) := survive P(t),P'(t);

    // increase the time counter
    t := t + 1;

od
end EP.

```

Figure 2-6: Pseudo code for EP

2.2.3 Evolution Strategies

Evolution Strategies are based on the work of Rachenberg [1973] and Schwefel [1975]. Like GA's, ES use fixed length structures, but instead of the binary representation used in GAs, ES have real valued genes.

The emphasis in ES is more on the acquisition of behaviour rather than structure [Angeline, 1993]. Each position in an ES (i.e. a real number) marks a behavioural trait, and an individual's behaviour is the composition of these traits.

Crossover in ES is intended to produce children that are behaviourally similar to their parents, and there are different approaches [Baeck, 1992].

The first, *discrete* recombination, is similar to a method often used in GA's, *uniform* crossover [Syswerda, 1989]. Discrete recombination consists of selecting the parameter value from either of the two parents. In other words, the parameter value in the child equals the value of one of the parents. Uniform crossover involves creating a *crossover mask*, a binary string the same length as the parents. A 0 in the mask results in the relevant gene being selected from the first parent, while a 1 results in the second parent donating the gene. The crossover mask is a random string, and generally ensures that each parent contributes equally to the child. An example is shown in Figure 2-7.

The other two methods exploit the fact that the genes are real valued. The first of these, the *intermediate recombination* operator, determines the value of the child's genes by averaging the two values in the parents genes. The second method, the *random intermediate* recombination, probabilistically determines the evenness of the contribution of each parent for each parameter.

Parent 1	:	0.8	0.3	0.2	0.5	0.6
Parent 2	:	0.3	0.1	0.8	0.4	0.1
Mask	:	1	0	1	0	1
Child 1	:	0.3	0.3	0.8	0.5	0.1
Child 2	:	0.8	0.1	0.2	0.4	0.6

Figure 2-7: Uniform Crossover applied to a real coded string

2.2.4 Genetic Programming

Genetic Programming (GP) was developed by Koza, [1992]. It is a relatively new technique for the evolution of computer programs. GP differs from the GA in the representation of individuals, using trees instead of fixed length strings. Thus, for example, the simple program "a + b * c" would be represented as:

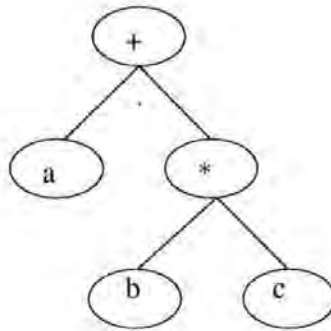


Figure 2-8: Example of a tree structure

The program trees are made up of two fundamental building blocks: nodes and leaves (Figure 2-8). Nodes can be simple functions such as + * which take one or more arguments, while the leaves are terminals, i.e. numbers or zero-argument function. The first major step in any implementation of GP is to select the necessary functions and terminals, and to ensure that any combination of them will result in a syntactically correct program.

GP uses a similar generational approach as the simple genetic algorithm, but, because of its tree structures uses a different crossover scheme. In GP the crossover operation is implemented by taking randomly selected sub-trees in the individuals (selected according to fitness) and exchanging them (Figure 2-9). Like the simple genetic algorithm this results in the creation of two new individuals. Like GA's, the GP reproduction operator simply

copies an individual unchanged into the next generation, however GP does not usually employ mutation as a genetic operator.

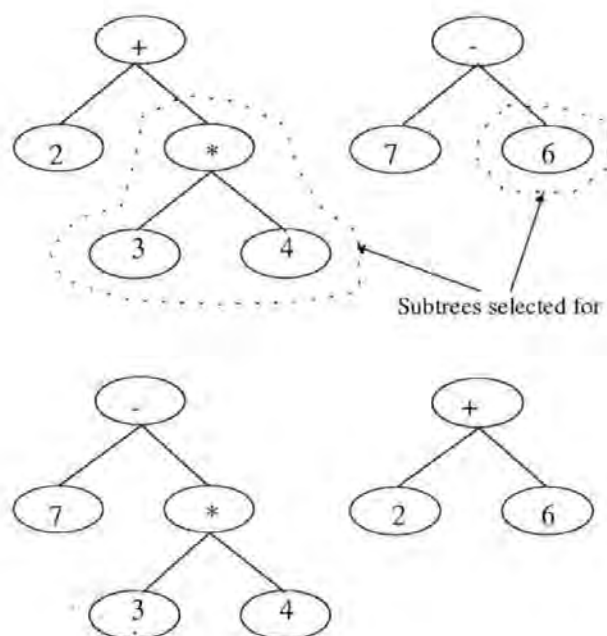


Figure 2-9: Crossing over two parent trees by swapping sub-trees.

2.3 Exploration and Exploitation

Two important but competing themes exist in an evolutionary search. Exploiting the best solution versus exploring the search space. Michalewicz [1994] provides a comparison between a number of search strategies, namely hill-climbing and random and genetic search. Hill-climbing is an example of a strategy which supports the exploitation of the search space whilst disregarding exploration. Random search is an example of a strategy which supports exploration of the search space, whilst ignoring exploitation of the search space. Genetic algorithms combine elements of both stochastic and directed search and hence provide a balance between exploration and exploitation. Selection according to fitness is the source for exploitation, so that the GA is able to focus the search on promising areas of the search space. The mutation and crossover operators are sources for

exploration in order to maintain population diversity so that important information is not lost. Whitley [1989] notes:

“ Many of the various parameters that are used to ‘tune’ genetic search are really indirect means of allocating selective pressure and population diversity. As selective pressure is increased, the search focuses on top individuals in the population, but because of this ‘exploitation’ genetic diversity is lost. Reducing the selective pressure (or using a larger population) increases ‘exploration’ because more genotypes and thus more schemata are involved in the search”

2.4 Variations of the Evolutionary Algorithms

There are a number of types of advanced EA's, all developed to improve the abilities of evolutionary search for different types of problems. This section describes three evolutionary algorithms that were initially selected because they were found (in the literature) [Eshelman, 1991, Baluja, 1994, Muhlenbein & Schlierkamp-Voosen 1993] to provide good results on various problems in comparison to those produced by other EA's.

When applying GA's to complex real world problems, a designer may face a number of difficulties. These problems include :

- Multimodality - the search space is characterised by a number of peaks and troughs [Goldberg, 1989].
- Constrained Space - Difficult to access and remain in a feasible region [Bilchev & Parmee, 1996].

- Premature convergence - population converges quickly onto non-optimal local minima [Davis, 1991].
- Deceptive - Contain isolated optima : the best points tend to be surrounded by the worst [Whitley, 1991].
- Highly sensitive - slight perturbation of the design variables causes large changes in relation to fitness [Parmee & Vekeria 1997].
- Multiple objectives - several objectives are present simultaneously [Fonseca & Fleming 1993, Coello Coello 1998].
- Uncertainty - vagueness or impreciseness due to poorly defined data, unsatisfactory formulation of design objectives or inability to evaluate the relative importance between objectives [Rao, 1984].
- Highly dimensional - large number of variables are present [Parmee and Vekeria 1997].
- Noise - noisy environment [Goldberg, 1989].

In an attempt to overcome these and other problems, new, and more advanced types of GAs have been developed. Much of the available GA literature concerns the development of new and more advanced GA's for tackling many of these problems, e.g. :

- Parallel and Distributed GAs help with exploration of search space and to reduce computational expense [Tenese, 1989] [Goodman et al, 1996].
- Structured GAs (sGAs) allow parts of chromosomes to be switched on or off using evolveable 'control genes' [Dasgupta & McGregor, 1992] [Parmee & Denham, 1994]
- Messy GA's use variable-length strings that may be over or under specified with respect to the problem being solved. [Goldberg et. al. 1991]
- CHC GA utilise population-elitist selection, a highly disruptive crossover operator, an "incest" prevention mechanism and a divergence process. [Eshelman, 1991].

- PBIL is combination of evolutionary optimisation and hill climbing. [Baluja, 1994].
- Breeder GA is a combination of evolution strategies (ES) and genetic algorithms (GA). [Muhlenbein & Schlierkamp-Voosen 1993].
- GAs with niching and speciation where the population within the GA is segregated into separate species [Deb & Goldberg, 1989].
- Hybrid GAs (hGA's) combine evolutionary search heuristics with traditional local search algorithms [Davis, 1991].
- GAANT involves a combination of a GA and ant colony based search [Parmee, 1996a]
- Multiobjective GAs (MOGAs) which allow multiple objectives to optimised [Fonseca and Fleming 1993].
- Combination of Fuzzy Logic with EA's [Zhao et. al. 1996].

A number of these variant techniques play a significant role in the thesis and these are now described in some detail.

2.4.1 The CHC Adaptive Search Algorithm

The CHC Adaptive Search Algorithm was developed by Larry Eshelman [Eshelman, 1991]. CHC stands for Cross generational elitist selection, Heterogeneous recombination (by incest prevention) and Cataclysmic mutation, which is used to restart the search when the population stagnates. The main differences between the CHC GA and the simple, canonical GA are:

- Population-elitist selection
- Highly disruptive crossover operator
- An “incest” prevention mechanism
- Divergence process

The rationale behind the CHC is to have a very aggressive search (by using monotonic selection through survival of the best strings) and to offset the aggressiveness of the search by using highly disruptive operators such as uniform crossover. With such small population sizes, however, the population converges to the point that it begins more or less to reproduce many of the same strings. At this point the CHC algorithm uses cataclysmic mutation. All strings undergo heavy mutation, except the best string which remains intact. After mutation, genetic search is restarted using only crossover.

procedure CHC

```

begin
    t = 0;
    d = L/4;
    initialize P(t);
    evaluate structures in P(t);
    while termination condition not satisfied do
    begin
        t = t + 1;
        selectr C(t) from P(t-1);
        recombine structures in C(t) forming C'(t);
        evaluate structures in C'(t);
        selects P(t) from C'(t) and P(t-1);
        if P(t) equals P(t-1)
            d--;
        if d < 0
            begin
                diverge P(t);
                d = r * (1.0 - r) * L;
            end
        end
    end
end.

```

Procedure select_r

```

begin
    copy all members of P(t-1) to C(t) in random order;
end.

```

Procedure select_s

```

begin
    form P(t) from P(t-1)
        by replacing the worst members of P(t-1)
        with the best members of C'(t)
        until no remaining member of C'(t)
        is any better than any remaining member of P(t-1);
end.

```

Procedure *recombine*

```
begin
  for each of the  $m/2$  pairs of structures in  $C(t)$ 
  begin
    determine the Hamming_distance
    if (Hamming_distance/2) > d
      swap half the differing bits at random;
    else
      delete the pair of structures from  $C(t)$ ;
  end
end.
```

Procedure *diverge*

```
begin
  replace  $P(t)$  with  $M$  copies of the best member of  $P(t-1)$ ;
  for all but one member of  $P(t)$ 
  begin
    flip  $r * L$  bits at random;
    evaluate structure;
  end
end.
```

Variables

M	population size
L	string length
t	generation
d	difference threshold
r	divergence rate

Figure 2-10: Pseudocode for CHC

2.4.1.1 Elitist Selection

The CHC employs a more direct emulation of Darwin's 'survival of the fittest'. In nature there are limited resources in the environment. As competition of these resources increases, the weakest individuals die. The fitter individuals survive longer and thus the greater their chance of having more offspring. The CHC replaces the traditional GA's "reproduction selection" with bias in favour of the "survival of the fittest". In traditional GA's the selection is performed according to a fitness criteria. Instead of biasing selection of

individuals towards the better performing members of the population, the CHC pairs each member randomly with another, regardless of the fitness. During the survival-selection process, instead of replacing the old parent population with the new child population, competition for survival is cross generational i.e. the child population must compete with the parent population for survival.

Several other GA's use fitness-biased survival selection -- Whitley's GENITOR [Whitley 1989], Syswerda's Steady State GA (SSSGA) [Syswerda 1989]. The SSSGA inversely ranks the parent population and replaces a certain number of the worst performing members of the parent population with children. The GENITOR algorithm is specifically designed to allocate reproductive trials according to rank. GENITOR only produces one genotype at a time, which is inserted in the population automatically ranking the individual relative to the existing pool. The CHC however differs from both of these algorithms in that the competition for survival is cross-generational - a child only replaces a member of the parent population if it is better.

2.4.1.2 Highly Disruptive Crossover (Uniform Crossover)

Eshelman [1991] argues the use of uniform crossover over the use of standard one point and two point crossovers, in order to combat parasitic bits (bits that tag along good performing schemata). The intuitive idea behind recombination is that the combination of features from two good parents may yield even better children. However the more bits copied from one parent into a child the more schemata of that parent are preserved at the expense of the other parent, and vice versa.

One general area of concern using EAs is premature convergence of solutions. Premature convergence refers to a situation where most of the population members have similar bit strings without reaching the optimal point in the space. One way of preventing the loss of diversity is not to allow new strings which have a hamming distance (number of differing bits in a string) below a specified threshold into the parent population, however, this limits the searching of the EA. A common strategy called incest prevention involves a random mating of parents but only if hamming distance is above a certain threshold. The CHC crosses exactly half the differing bits of the parents and these are exchanged randomly without replacement. This guarantees that the children are always at the maximum hamming distance from both parents therefore promoting diversity within the population (Figure 2-11). This is similar to uniform crossover [Syswerda, 1989], which recombines at bit level.

Parent 1 : 1 0 0 1 1 1

Parent 2 : 0 1 0 0 1 0

Child 1 : 0 0 0 0 1 1

Child 2 : 1 1 0 1 1 0

Figure 2-11: Disruptive Crossover

Eshelman [1991] argues that uniform crossover is much less likely than traditional one or two point crossover to produce the same offspring twice from the same parents. The reason for this is that the CHC preserves fewer schemata than one or two point crossover.

2.4.1.3 Divergence of Population

The CHC does not use mutation in the reproduction-recombination cycle. The use of HUX and incest prevention in conjunction with a population size large enough to preserve a number of diverse structures (e.g., 50) enables CHC to delay premature convergence. All these mechanisms cannot guarantee that no allele will prematurely converge. Some sort of mutation is required.

Since the CHC is extremely good at maintaining diversity, mutation is however less effective in the CHC than in the traditional GA. Mutation in the CHC is only introduced when the population has stagnated. Stagnation is said to have occurred once the difference threshold (this is set as length of string/4 at the beginning of the run) has dropped to zero and there have been several generations without any new offspring accepted into the parent population. The reinitialisation is only partial however as the best individual found so far is used as a template for creating a new population. Each new individual is created by flipping a fixed proportion (e.g., 35%) of the template's bits chosen at random. One instance of the best is added unchanged to the new population. This creates a population that preserves the progress made so far and is biased toward a good solution but with new diversity to continue the search. Moreover the search cannot converge to a worse solution than the previous search.

Eshelman [1991] argues that partial reinitialisations over chronic mutation are much more effective, performing considerably better on a large range of problems utilising the same parameter sets. Restarts provide many of the benefits of a large population without the cost of a slower search. Optimal solutions can be identified on easy problems in the first initialisation cycle whereas with more complex problems optimal solutions are identified only after repeated restarts.

2.4.1.4 Incest Prevention

Strategies for maintaining population diversity can naturally be grouped according to where they occur in the GA's reproduction-recombination-replacement cycle i.e. (1) how mates are selected, (2) how children are created by recombination, (3) how parents are replaced [Eshelman & Shaffer 1991]. The points concerning population selection and the creation of new individuals have already been addressed. The remaining point of the mating strategy has yet to be discussed. Mating strategies are usually considered in terms of speciation, where the goal is to prevent radically dissimilar individuals from mating. Goldberg & Richardson [1987] introduced penalties which reduce the fitness of individuals as a function of how similar they are to other individuals in the population. The effect of this is to reduce "incestuous" mating by increasing the likelihood of reproduction between diverse individuals. Eshelman's incest prevention mechanism is a more direct approach for preventing similar individuals from mating [Eshelman & Shaffer, 1991]. Individuals are randomly paired for mating and bias is introduced against mating individuals who are similar. Individuals are only crossed if their hamming distance, i.e. the number of differing bits between the two individuals, exceeds the difference threshold. The threshold is initially set to the expected average Hamming distance of the initial population ($\text{string length} / 4$), and then is allowed to drop as the population converges. The number of children produced each generation can vary from zero up to the population size. The disadvantage of this mating strategy is that more schemata are disrupted by crossover, since fewer schemata are shared.

2.4.1.5 No Duplicates

The CHC algorithm also utilises a no duplicates policy (Figure 2-10). This is to ensure that the number of evaluations are kept to a minimum. Once a child is produced by crossover, it is matched against all the members of the parent population. If a duplicate is found, the child is discarded, otherwise the child is evaluated and included in the child population of potential candidates for replacing members of the parent population. Another reason for implementing this strategy is to ensure that *super chromosomes* do not dominate the population, which would reduce the diversity within the parent population and ultimately cause premature convergence. The CHC GA always preserves the best individuals so far whilst maintaining a highly explorative search through disruptive crossover.

Several researchers have investigated the idea that diversity of a population may be maintained by restricting children from entering the parent population if they are similar to the parent members. De Jong [1975] suggested the crowding scheme in which an offspring replaces an existing individual according to its similarity in bit terms (hamming distance) with other individuals in a randomly drawn sub-population of size CF (crowding factor). Mauldin [1984] used a uniqueness operator to maintain diversity. An offspring would only be inserted into the population if it is genotypically different from all individuals in the population (specified by a given hamming distance).

2.4.1.6 CHC Performance

Eshelman [1991] compared the performance of the CHC with the canonical GA for a number of functions. For five of the six functions in which both algorithms found the optimum in all 50 searches, CHC, on average found the optimum in fewer evaluations, and on four of the functions, the CHC found the optimum more often than the GA. The only

function on which the canonical GA does significantly better than the CHC is a smooth, unimodal function. The CHC performed significantly better on all the multi-modal functions. Eshelman found that the CHC algorithm was relatively insensitive to parameter settings. Eshelman also reported the CHC to be a worthy competitor for Goldberg's messy GA [1991].

2.4.2 Population-Based Incremental Learning (PBIL)

Population-based incremental learning (PBIL) was introduced by Baluja in 1994. PBIL is an abstraction of a canonical GA without recombination. The statistics normally implicit in the population are explicitly maintained in a 'probability vector' which determines the frequency with which 0's and 1's are generated in each bit of the trial solutions. It is claimed that a standard form of PBIL performed as well as, or better than the canonical GA on a range of standard optimisation tasks. PBIL is a combination of evolutionary optimisation and hill climbing. The algorithm initially creates a real valued probability vector with values set to 0.5 which is utilised to create a trial set of binary encoded solution vectors where the probability of generating a 1 or 0 is equal. The performance of the real-numbered variable sets represented by these binary solution vector's are assessed via the fitness function. As search progresses, the values in the probability vector gradually shift relative to the fitness of the 'best' trial solution vectors. The distance the probability is pushed (towards either 0.0 or 1.0) depends upon a learning rate parameter. After the probability vector is updated, a new set of trial solution vectors is produced from the updated probability vector and the cycle is continued. As the search progresses, entries in the probability vector move away from their initial settings of 0.5 towards either 0.0 or 1.0 i.e. the binary representation of the trial solutions are pushed towards that of the current best solutions. Thus, PBIL does not store domain knowledge in a population but in a probability distribution.

PBIL is characterised by 3 parameters (Figure 2-12). The first is the number of samples to generate based upon each probability vector before an update (analogous to the population size of GA's). The second is the learning rate, which specifies the dimension of the steps towards a good solution. The third is the number of best solutions to update from.

Baluja [1994] suggests variants of the basic PBIL, such as updating the probability vector not only from the best trial but from several of the better performers. Although this method proved to be too problem dependent, some significant results were produced.

Greene [1996] suggests another variant on the basic PBIL. At each step of the search a record of the "best" and "worst" trial solutions are maintained. The probability vector is then maintained by moving it towards the "best" trial vector and moving it away from the "worst" trial solution [Greene 1996]. Greene concluded that this change worked well during the early stages of the search process, but began to fail as search progresses. In order to overcome this an element of the probability is moved away from that of the "worst" trial solution only in those bit positions where the "worst" and "best" probability vector differ. Greene also keeps a track of the highest value of the objective function attained and aborts the current step (and update's the probability vector) whenever this is exceeded. This results in an automatic adaptation in the number of trials per step. Greene argues that this allows a large number of trials per step to be used without spending time performing what amounts to an essentially random search in the early stages.

PBIL is susceptible to premature convergence. To overcome this, Baluja proposes an occasional random mutation of probability vector. In the canonical GA, mutation performs a clear role in maintaining diversity of the 'gene pool' by making it possible to regenerate a missing 0 or 1 at a particular bit position. This however is not possible with PBIL. Greene

suggests replacing mutation with a deterministic ‘forgetting’ operator. After each update of the probability vector, each element of the probability vector is pushed towards 0.5.

```

***** Initialize Probability Vector *****
for i :=1 to LENGTH do P[i] = 0.5;

while (NOT termination condition)
    ***** Generate Samples *****
    for i :=1 to SAMPLES do
        sample_vectors[i] := generate_sample_vector_according_to_probabilities (P);
        evaluations[i] :=Evaluate_solution (sample[i]);

    best_vector :=find_vector_with_best_evaluation (sample_vectors, evaluations);
    worst_vector := find_vector_with_worst_evaluation (sample_vectors, evaluations);

    ***** Update Probability towards best solution *****
    for i :=1 to LENGTH do
        P[i] :=P[i] * (1.0 - LR) + best_vector[i] * (LR);

    ***** Update Probability Away from Worst solution *****
    for i :=1 to LENGTH do
        if (best_vector[i] ≠ worst_vector[i] then
            P[i] :=P[i] *(1.0 - NEGATIVE_LR) + best_vector[i] *(NEGATIVE_LR);

    ***** Mutate Probability Vector *****
    for i := 1 to LENGTH do
        if (random (0,1)< MUT_PROBABILITY) then
            if (random (0,1) > 0.5) then mutate_direction :=1
            else mutate_direction :=0;
            P[i] :=P[i] * (1.0 - MUT_SHIFT) + mutate_direction * (MUT_SHIFT);

```

USER DEFINED CONSTANTS :

SAMPLES: the number of vectors generated before update of the probability vector.

LR: the learning rate, how fast to exploit the search performed.

NEGATIVE_LR: the negative learning rate, how much to learn from negative examples.

LENGTH: the number of bits in a generated vector.

MUT_PROBABILITY: the probability for a mutation occurring in each position.

MUT_SHIFT: the amount a mutation alters the value in the bit position.

Figure 2-12: the PBIL algorithm for a binary alphabet.

PBIL was shown to outperform GA's on several problems [Baluja, 1996]. One reason for PBIL's success may be attributed to it's capability of capturing first order dependencies between individual solution parameters and solution quality in a probability distribution [Baluja & Davies, 1997]. GA's on the other hand, maintain a population and rely on

crossover to sensibly combine parameters that are collectively responsible for favourable evaluations. Since the choice of crossover points is random, it may not be favourable [De Bonet et al, 1997]. Due to its disruptiveness it may tear apart previously discovered useful parameter groups.

2.4.3 The Breeder Genetic Algorithm (BGA)

The Breeder Genetic Algorithm (BGA) [Muhlenbein & Schlierkamp-Voosen 1993] is based on artificial selection similar to that used by human breeders. The BGA is a combination of evolution strategies (ES) and genetic algorithms (GA). The BGA (Figure 2-13) uses a selection scheme called truncation selection. The $T\%$ of the best individuals are selected and mated randomly until the number of offspring is equal to the size of the population. The search process of the BGA is mainly driven by recombination. The BGA depending upon the type of problem, may use one of a number of different recombination operators (discrete recombination, extended intermediate recombination, extended line recombination). The operator used in this work is a discrete crossover similar to the uniform crossover. It operates on the alleles of the selected parents chromosomes. Two parents, (u_1, \dots, u_n) and (v_1, \dots, v_n) , produce an offspring (w_1, \dots, w_n) so that w_i is either u_i or v_i with equal probability.

Mutation is an important background operator for the BGA. The BGA's objective is to give a small perturbation $\Delta x_i \times \delta$ on a variable x_i . Where Δx_i is a mutation range for the variable x_i and δ is the mutation probability. An allele x_i is chosen with probability pm to be mutated (Muhlenbein et al recommend it be set to 0.1). The mutation rate is inversely proportional to the number of parameters to be optimised and the mutation range is fixed.

as for mutation probability pm Muhlenbein suggests $1/n$, where n is the number of alleles in a chromosome.

Muhlenbein has applied the BGA using real coded chromosomes. The BGA has been used in this work utilising binary chromosomes. Similar principles are applied. In the case of recombination, uniform crossover is utilised as it is very similar in manner to the discrete crossover. The mutation operator is the same as that used in a traditional GA, however the rate is kept low, so as not to cause a large disruption.

STEP0: Define a genetic representation of the problem
STEP1: Create an initial population $P(0)$
STEP2: Each individual may perform local hill-climbing
STEP3: The breeder selects $T\%$ of the population for mating. This gives set $S(t)$
STEP4: Pair all the vectors in $S(t)$ at random forming N pairs. Apply the genetic operators crossover and mutation, forming a new population $P(t+1)$.
STEP5: Set $t = t + 1$, return to STEP2 if it is better than some criterion (acceptance)
STEP6: If not finished, return to STEP3.

Figure 2-13: Breeder Genetic Algorithm

2.5 Summary

This chapter has provided an introduction to the area of evolutionary computation. The chapter has discussed different types of evolutionary algorithms in existence and highlighted common attributes of the various evolutionary techniques of particular relevance to the engineering design processes of the following chapters.

3. INTEGRATION OF EVOLUTIONARY ALGORITHMS WITH MATHEMATICAL MODELS

This chapter firstly provides a literature review concerning the application of evolutionary algorithms to structural optimisation problems. The second half of the chapter then discusses the development of software utilising a CHC genetic algorithm for the optimisation of a real world structural plate optimisation problem concerning the shape optimisation of roof tiles. The problem concerns the optimal material distribution on the underside of this flat concrete plate, with varying load conditions. . The aim here is to enable the company to meet specifications of international markets, reduce lead times and costs through improved efficiency and a reduction in materials usage. The work was undertaken during a two year Teaching Company Programme. Two types of models are discussed, the first is based on bending moment and complex stress theory and the second on finite element analysis

3.1 Structural Optimisation

There is considerable literature on structural optimisation and structural shape optimisation [Leite, 1996]. This interest in shape design reflects the effectiveness of shape changes for improving structural performance [Haftka, 1986]. It also reflects a growing sophistication in structural analysis and optimisation tools, which allow more complex shape optimisation problems to be addressed. Shape optimisation is an integral part of the structural design process and tools available to assist the designer significantly affect the type of problems that can be attempted and to what extent optimisation can be performed.

There are three distinct classes of shape optimisation problems. In order of computational complexity these are : size, shape, and topological optimisation [Jensen, 1992]

- Size optimisation (also called cross-sectional optimisation) refers to the determination of specific geometric dimensions for a pre-selected design class, such as the thickness of a shell, the size of a truss member or the radius of a circular stress element.
- Shape optimisation (also called geometric optimisation) introduces additional design variables which allow for boundary movement. This process is more complex than size optimisation and geometrical changes have historically been limited. However, it is of significant importance for instance, in the aircraft and automotive industries, as well as others, providing improvements to turbine design and airfoil shapes. Size optimisation is a subset of shape optimisation.
- Topological optimisation involves topological as well as shape and size modifications. Topological modifications deal with assemblies of components. The components in the assembly may be modified and components may be added, deleted or moved in the assembly in an attempt to generate improved designs.

Literature concerning the application of evolutionary optimisation techniques to structural optimisation is becoming more prolific. There is a growing interest in the application of such techniques due to significant increases in computing and especially parallel processing capabilities. Engineering designers are now recognising the increasing potential of evolutionary search for real-world optimisation problems.

Many researchers have used the canonical genetic algorithm for the optimisation of trusses. Goldberg and Samtani [1986] used the GA to optimise a 10 bar truss. Jenkins [1991] used the GA for the minimum weight design of a trussed rafter roof structure. In order to avoid

stagnation and improve the progress of the GA during the latter stages of the search, a space condensation heuristic has been introduced. The method reduces the combinatorial space during the latter stages of search by removing discrete values of variables shown to be associated with low fitness individuals. The method also provides the additional advantage of reducing the overall processing time required [Jenkins 1994]. A study of a cable stayed bridge using the GA [Jenkins 1992] requires 500,000 evaluations of the structure thereby highlighting the problem related to a requirement for considerable processing time due to the large number of calls to the evaluation function.

Rajeev and Krishnamoorthy [1992] applied the GA to slightly more complex problems, concerning the minimisation of weight and satisfaction of stress and displacement constraints for a 25 bar truss and a 160-bar transmission tower. The 160-bar transmission tower utilises 22 variables. They conclude that the GA is a highly efficient technique for structural optimisation due to the ability to manipulate a large number of discrete variables. However computational expense proves to be a major drawback again due to the number of necessary function evaluations.

Hajela et al [1992] presented a two stage optimisation method for the sizing of skeletal structures. The first stage uses a GA to search for number of suitable low weight topologies whilst disregarding the stress and displacements. The second stage then uses these truss topologies as initial designs, for which the cross-sectional areas are then optimally sized using a GA for minimum weight and the satisfaction of stress and displacement criteria.

Jensen [1992] developed a GA based approach for topology optimisation. The design domain is discretised into small elements, where each element either contains material or is a void. No intermediate densities are allowed. The GA is used to determine the optimal

configuration of material and void within the domain such that the structure's weight is minimised subject to displacement and stress constraints.

Chapman et. al. [1993] extended the research of Jensen in the use of the GA for structural topology optimisation. Lighter designs were generated in comparison to homogenisation based solutions. The homogenisation method was developed by Bendsoe and Kikuchi [1988]. A design domain is discretised into small rectangular elements where each element contains composite material of continuously variable density and orientation. An optimality criteria method is used to determine how the material density and orientation in each element should change so that the compliance of the structure is minimised subject to a maximum volume constraint. The deterministic homogenisation based techniques require considerably fewer structural evaluations. However the GA is also able to offer a family of topologies (each unique in topology, weight and stiffness) which a designer can evaluate using a secondary criteria such as manufacturability.

Dhingra and Lee [1994] used the GA to optimise a 25 bar truss and found the GA to compare favourably to optimum solutions using gradient-based search techniques. They propose a co-operative game, theoretic approach for addressing multiple objective functions. In a non-co-operative game approach, each player is looking out for his own interests and is unconcerned about how his choice will affect payoffs of other players. The co-operative approach on the other hand assumes that each player is part of a team and is willing to compromise his own payoff in order to improve the situation as a whole.

Keane and Brown [1996] successfully applied the GA to the design of a satellite boom with regard to the efficient control of structural vibrations. The GA changes the geometry of the design by altering the three dimensional co-ordinates of its joints. The aim is to minimise

the band averaged noise along the boom. Keane found that in order to accurately assess the designs considered by the GA very significant computations were required, even when using a highly tuned and customised code to carry out the calculations. Furthermore he states that where global optima cannot be found utilising current levels of computing capability, rapid convergence to improved designs must be the alternative goal of the designer.

Kane and Schoenauer [1996] apply the GA to structural topology optimisation of cantilever plates. They suggest using specific genetic operators which are tailored for topology optimisation. The GA produced good results in comparison to the homogenisation based method. Computational expense was highlighted as major drawback of utilising a GA. Using coarse mesh representations of the plate, a single run may require up to 150,000 calls of an FEA fitness function, taking approximately 24 hours on a powerful HP workstation.

Cai and Thierauf [1996] have developed a two level parallel evolution strategy for the optimisation of a steel transmission tower. The objective is to minimise the weight of the structure under given stress, displacement and stability constraints. The discrete and continuous design variables are treated in parallel using two sub-populations. Periodically, the design variables in the two sub-problems are exchanged.

Genetic Algorithms have proved effective in the design of composite laminate structures. They are used to optimise ply thickness and orientation, and many studies concerning the improvement of the GA's reliability and efficiency are evident in this area. Mingra [1986] performed some of the earlier studies concerning optimisation of laminations on honeycomb structures. Le Riche and Haftka [1994] studied the problem of composite panel weight minimisation subject to buckling and strength constraints. Feasible designs were

generated by utilising a combination of penalty parameters and the tuning of various genetic operators. This method also increased the overall efficiency of the genetic search and provided a 56% reduction in the computational cost of search. Haftka et. al. [1996] also explored the possibility of specially tuning the GA in order to take advantage of repeated runs. The concept was to maximise the efficiency of the GA during the early stages of search by increasing selection pressure. This however may result in premature convergence to a solution which is significantly inferior to one which may be found by using a combination of explorative and exploitive search strategies.

Kogiso et al [1994] uses a binary tree to store appropriate information regarding laminate designs that had already been analysed. After the generation of a new population of designs, the tree is searched for laminate designs with either an identical stacking sequence or similar performance (e.g. laminates with identical in-plane strains). Depending on the retrieved information a given laminate may not be required. This process does however require a large amount of computer memory and the search through the tree also has a computational cost. Kogiso also proposed a local improvement approach to reduce the number of analyses required by a GA.

Yamazaki [1996] reduces computational expense by using a two-level optimisation technique in maximising the critical buckling load of composite plates. The first level of optimisation involves the computationally expensive structural and sensitivity analysis. Once the optimum lamination parameters have been determined, the second level of optimisation implements the GA to find the stacking sequence that best matches the optimal lamination parameters. The second level does not require expensive structural analysis. This combination of optimisation methods allows Yamazaki to reduce the complexity of the analysis required during the GA run.

Goodman et. al. [1996] uses injection island genetic algorithms (iiGA) for the design of composite cantilever plates for the weight minimisation and selection of appropriate structural responses for given loading conditions. Goodman et. al. [1997] also applies the iiGA to optimise the Specific Energy Density (SED) of elastic flywheels. Injection island GA's search at various levels of resolution in parallel within a given space. Islands (sub-populations) which have a low level of resolution inject high performance individuals into an island of higher resolution to "fine-tune" the designs. Convergence of the low resolution processes occurs quickly and is then discontinued, saving valuable CPU time. The technique provides a reduction in the computational time plus an increase in the robustness of a typical GA.

Soremekun et al [1996] utilises the GA for the minimum weight design of a cantilever laminated composite plate. Somemekun outlines three multiple elitist and one variable elitist selection strategies. The strategies involve passing a prescribed number of the best individuals from the parent population to the new parent population. Depending upon the strategy employed the rest of the individuals for the parent population are either selected from the top performing members of the child population or a combination of the top performing and randomly selected individuals of the child population. The number of top performers passed to each successive generation remains constant throughout the genetic search in multiple elitist scheme and is varied in variable elitist selection. Small reductions in computational cost have been realised using these strategies.

Mill et. al. [1996] have utilised different types of shape representations these include methods based on parametrics, lines, primitives, spline curves etc. They found approximate splines for curves and surfaces to be powerful methods of describing shapes and also

amenable to GA manipulation. Splines consist of curves whose basic shapes are influenced by the positioning of a set of control points. The final shape will be influenced by the type of spline used and the position of the control points. The curve does not necessarily have to pass through all the control points.

3.2 Development of Evolutionary Software for Single Component Design

The initial two years of the research described within the thesis was carried out as part of a Teaching Company Scheme between the University of Plymouth and Redland Technologies Ltd (now Lafarge Brass). During this period research was performed in a real world problem domain concerning the shape optimisation of roof tiles. As the work can be used to optimise any flat single component plate the roof tile is referred to as the flat plate problem throughout out the thesis. The problem concerns the optimal material distribution on the underside of this flat concrete plate, with varying load conditions.

The Teaching Company Scheme is a partnership between industry and academia. The role of a Teaching Company Associate is to provide a link between the University and the company in the transfer of new knowledge. The overall aim of the scheme was to improve the competitive position of the company through the implementation of new technology.

The main aim of the programme was therefore the development of software utilising evolutionary algorithms for the optimisation of concrete flat plates, thus enabling the company to meet specifications of international markets, reduce lead times and costs through improved efficiency and a reduction in materials usage.

There are six major modules in the developed software :

User Interface - Allows the user to modify the operator settings for the GA

Representation - The optimal shape will depend on the plate representation and the selected variables to represent the modifiable elements.

Analysis - Analyses the design using FEA or complex stress theory

Optimisation - Modifies the values of the design variables.

Evaluation - Determines the fitness of the design

Termination - Checks to see if any of the termination criteria are met. Stops at the maximum number of evaluations or restarts depending upon the requirement of the engineer.

Two forms of structural analysis have been utilised to evaluate the phenotypes. The first is based on complex stress and bending moment theory, and is computationally inexpensive. The designs produced by this method must be considered high risk due to the simplicity of the analysis. The second is the finite element method which is computationally expensive, but produces significantly lower risk design solutions due to the in-depth analysis performed upon the phenotype.

Most real-world optimisation problems, particularly those related to design, require the simultaneous optimisation of more than one objective function. Some examples include:

- In bridge construction, a good design is characterised by low total mass and high stiffness.
- Aircraft design requires simultaneous optimisation of fuel efficiency, payload, and weight.

- In chemical plant design the objectives to be considered include total investment and net operating costs.

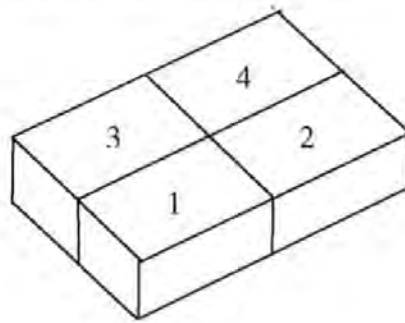
Some method relating to trade-off between the criteria is needed to ensure a satisfactory design.

The fitness of the design in relation to the plate problem takes into account the degree of maximum allowable stress violation on the plate and the plate's overall weight. There are two main objectives to the research. The first relates to the achievement of high-performance designs, i.e. to minimise the weight of the plate whilst satisfying maximum stress requirements. This conflict of criteria plus the high dimensionality results in a highly sensitive optimisation problem with many local optima. The dimensionality in this case refers to the number of variables on the plate. The second concerns the minimisation of required function evaluations. It is essential that the second objective is achieved in order that computationally expensive analysis techniques such as FEA can be realistically utilised.

3.2.1 Genetic Representation of the Plate

Figure 3-1 shows an example of how the genetic representation of the plate is decoded. The figure shows how a chromosomal representation (12 bits long) is used to represent a plate consisting of four elements. The chromosome is converted into four real numbers to represent the depths of the elements on the plate. Further information on problem representation may be found in Davis [1991]. The programs for the various algorithms and the mathematical model were written in Fortran 77.

The following plate utilises 4 elements.



The plate is encoded using a chromosome :

100111101110

The chromosome is partitioned in order to represent the four elements on the plate:

100 111 101 110

These bit strings are converted from base 2 to base 10 to yield :

4 7 5 and 6

If we assume the following :

Maximum plate depth of 20mm

Minimum plate depth of 9mm

Step size of 7mm

Therefore :

The variable depth on the plate is $20 - 9 = 11$ mm

The variation step size is $11 / 7 = 1.57$ mm

This gives 8 possible depths (i.e. 9, 10.57, 12.1, 13.7, 15.29, 16.86, 18.43 and 20.0)

The base 10 values are multiplied by 1.57 and then added to the minimum plate depth to provide the overall depth of the elements :

Element 1 $(4 * 1.57) + 9 = 6.29$ mm

Element 2 $(7 * 1.57) + 9 = 11.0$ mm

Element 3 $(5 * 1.57) + 9 = 7.86$ mm

Element 4 $(6 * 1.57) + 9 = 9.43$ mm

Figure 3-1: Decoding process used by the plate problem

3.2.2 Current Design Practice at the Company

Engineers at the company use a very similar design practice to the one outlined in chapter 1. Computer aided design (CAD) software is used to create and edit designs, whilst finite element analysis (FEA) is used to analyse the designs. A conceptual design is initially developed (based on previous designs and engineer's insight and knowledge related to the problem) which is then analysed using FEA software to determine which areas require redesign. Further changes are then made using the CAD software. This loop continues until a design is developed that meets the original specifications or is deemed acceptable.

In order to save money and become market leaders in plate design, the company must design lightweight components which meet predefined stress criteria. This design-evaluate-redesign process as stated in chapter 1 is extremely slow and often requires large amounts of human and calendar time, furthermore it sometimes fails to produce an optimal or near optimal design solution. The longer the design process the more costly it becomes. Automating the whole or even part of the design process would therefore be highly desirable.

The plate problem poses a considerable challenge in comparison to standard test functions such as De Jongs test suite [Goldberg 1989]. These test functions were developed in order to visualise and measure the relative performance of various algorithms. The flat plate problem is a real world problem where there is no apriori knowledge relating to the nature of the search space, due to the high dimensionality. The lack of prior knowledge makes it extremely difficult to determine whether the algorithm has converged to an optimum or near optimum solution unless an exhaustive search is executed. The goal therefore is to arrive at a "good" design solution, with minimum computational expense.

3.2.3 The Evaluation Model (Complex Stress)

In order to allow extensive experimental work, a simple mathematical model utilising bending moment and complex stress analysis has been utilised to ensure computational cost is kept to a minimum.

The plate is represented in a grid type manner being divided into rectangular or square elements each with variable depth (Figure 3-2). However, if required, a set number of elements may be considered as one variable to promote uniformity in depth. The plate may be subjected to one or more load conditions. The plate is supported on each of the four corners. For most problems (unless otherwise stated) the overall plate dimensions are 200mm x 200mm. The variables are relatively continuous in nature. The depth of each element is allowed to vary between the lower and upper bounds. The minimum plate depth is fixed at 8mm for most problems. However the upper limit on the plate depth is either 18mm or 24mm and the variation in depth between upper and lower bounds is discretised by introducing nine intermediate element depths (Figure 3-3). In order to achieve a certain degree of symmetry for ease of manufacture, an option is also available whereby neighbouring elements whose angles exceed a preset aspect ratio (the ratio describing relative depth at the element interfaces) may be penalised.

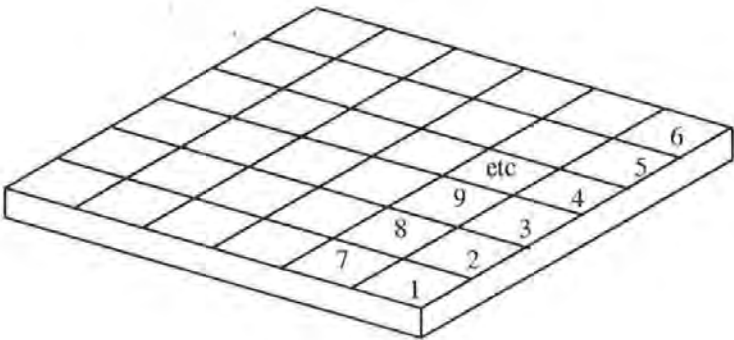
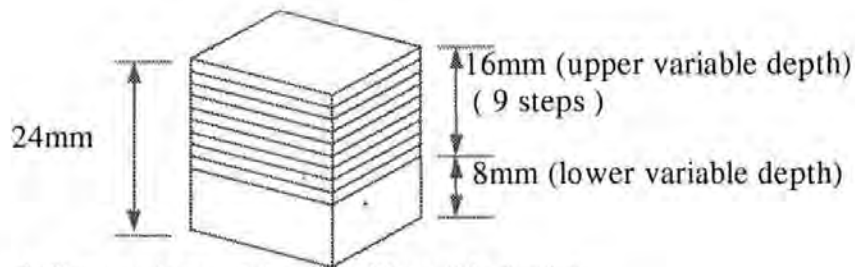


Figure 3-2: Representation of plate elements



$16 / 9 = 1.78\text{mm}$ variations (therefore 10 possible depths)
 (8.0, 9.7, 11.6, 13.3, 15.1, 16.9, 18.7, 20.5, 22.3, 24.0)

Figure 3-3: Discretisation of element depth variation

The shear and normal stresses are calculated for all transverse (X) and longitudinal (Y) sections using the following formulas:

$$\text{Bending Moments} \quad (M = W.x) \quad (\text{Equation 3-1})$$

$$\text{Second Moment of Area} \quad (I = bd^3 / 12) \quad (\text{Equation 3-2})$$

$$\text{Section Modulus} \quad (Z = I / y) \quad (\text{Equation 3-3})$$

$$\text{Normal Bending Stress} \quad (\sigma = M / Z) \quad (\text{Equation 3-4})$$

$$\text{Shear Stress} \quad (\tau = F / A) \quad (\text{Equation 3-5})$$

Where :

W = Force

x = Distance from support

b = Breadth

d = Depth

y = Distance from Neutral Axis

F = Force

A= Cross Sectional Area

Principal stress (σ_r) is as the stress criterion and is calculated for individual elements using the following formula:

$$\sigma_{1 \text{ or } 2} = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2} \quad (\text{Equation 3-6})$$

σ_1 & σ_2 = Principal Stresses

σ_x & σ_y = Normal Stresses

τ_{xy} = Shear Stress

The fixed parameters of the material are: flexural stress limit = 10N/mm^2 , density = $2.2 \times 10^{-7} \text{ Kg/mm}^3$. Designs exhibiting a high degree of stress violation are penalised to ensure that the generated designs satisfy relevant criteria. Although preliminary design solutions for the flat plate problem can be achieved with a relatively small number of variable elements (15 to 50) in excess of 300 elements are required during detailed design to ensure accurate stress evaluation.

3.2.4 Multi-Criteria Optimisation

The objective with the plate problem is the minimisation of weight whilst satisfying maximum stress requirements. The fitness of the stress criteria (Fs) is calculated by summing the number of elements that have stresses greater than the flexural limit which is then multiplied by a factor. The degree of violation is then taken into account by summing the difference for all elements which exceed the flexural limit. The number of

stress violations and degree of stress violations are then summated to form the overall stress violation (SV). This stress violation is divided by 100 then inversed in order to convert to a maximisation problem. The +1 in the formula is to prevent run time errors when the program is executed. The weighting for the stress criteria (W_s) is 1000.

The criteria weighting relating to the weight (W_w) of the plate increases as the degree of stress violation decreases. Designs which have high stress violations are therefore penalised to a greater extent as plate weight is reduced. The weight of the plate is also inversed to convert the problem to one of maximisation. The weighting (W_w) therefore depends upon the extent the stress criteria has been satisfied. In order to arrive at an overall fitness rating (F') for the plate the fitness values F_s and F_w are summated.

$$F_s = (1/((SV / 100) + 1)) * W_s \quad (\text{Equation 3-7})$$

$$F_w = (1/W_t) * W_w \quad (\text{Equation 3-8})$$

$$\text{if } F_s \geq 1000 \text{ then } W_w = 500$$

$$\text{if } F_s > 700 \text{ then } W_w = 400$$

$$\text{if } F_s > 500 \text{ then } W_w = 300$$

$$\text{if } F_s > 300 \text{ then } W_w = 150$$

$$\text{if } F_s > 200 \text{ then } W_w = 100$$

$$\text{if } F_s \leq 200 \text{ then } W_w = 50$$

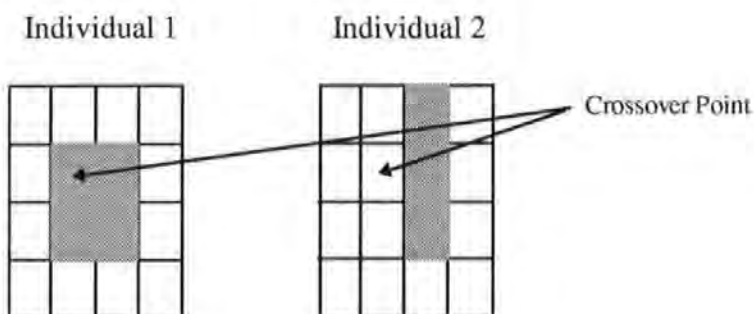
$$F' = F_s + F_w \quad (\text{Equation 3-9})$$

The above approach was taken in order to avoid equal emphasis on both objectives. If equal emphasis is placed on both objectives and there are large stress violations on the plate, the GA rapidly reduces the weight of the plate at the expense of meeting the stress requirements. This is due to the weight of the plate being an attractor. Placing a higher weighting on the stress moves the search towards the region of the design space containing solutions with low stress violations. As the degree of stress violation decreases the weighting for the weight increases, therefore once the stress criteria is satisfied the problem becomes a single objective relating to the minimisation of plate weight.

3.2.5 Two Dimensional Crossover

The use of a two dimensional string representation was considered, to provide a more realistic picture of the plate [Cartwright and Harris, 1993]. However due to the disruptive uniform crossover in the CHC algorithm, it is not possible to crossover individuals in the manner shown in Figure 3-4. The figure shows how the method would be applied to the plate problem if disruptive crossover was not utilised. Depending upon the operator settings one or more genes would be crossed. An individual would be defined as an $n \times n$ grid. Individual genes would be held on the 2-D grid. The grid is connected together to form the surface of a torus. It is therefore possible to combine promising section(s) of different plates through the action of 2-D crossover.

Before crossover



After crossover

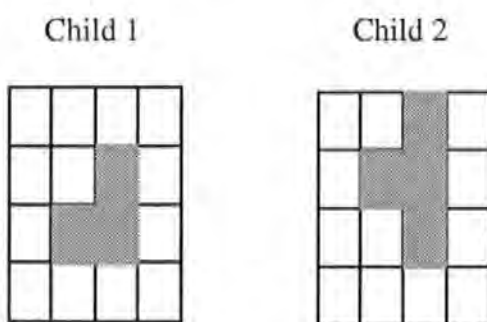


Figure 3-4 : 2 dimensional crossover representation

3.2.6 Development of Automated Design Tool

In order to develop a GA which provides good robust solutions the simplified model based on complex stress and bending moment theory was used to ensure extensive experimentation could take place. While this method provides stress results in the centre of the plate which are reflective of those obtained by FEA, the results on the periphery of the plate are not comparable. However the method provides a way of rapidly addressing issues such as multi-modality, multi-criteria, sensitivity and high dimensionality.

The initial implementation divided the plate into horizontal and vertical strips. The GA alters the periphery of the plate. An iterative optimisation loop is used to determine the depth of individual strips for the main body of the plate so as to satisfy the stress criteria. The highest depth of individual strips defined the resultant shape.. This method proved to be an extremely fast (due to the small number of variables) way of generating feasible preliminary design solutions. The design was then further refined utilising FEA. The main drawback with the technique is that it is not very flexible and excludes a large number of possible design solutions due to iterative loop and strip representation.

3.2.7 Selection of Design Variables

The advent of the finite element analysis (FEA) and the development of increasingly powerful computational processing capability has allowed the complex analysis of large problems and the identification of low-risk design solutions i.e. solutions with a low probability of error. However, every type of analysis requires input which is determined from a set of design variables. The time required to initially develop this input and perform the evaluation can be extensive, and there is no guarantee that the resulting design will be feasible. If it is not, new values for the design variables must be determined. The determination of an optimal set of design variables and their upper and lower bounds is not always directly intuitive, and consequently, is often found through trial and error.

Using evolutionary search to tackle design problems imposes certain restrictions and requirements on phenotypic representation. A popular choice of representation that would describe the plate relates to nodal co-ordinates. However a major problem with this choice is the resulting large number of design variables to define even the simplest of shapes. The advantage is the ability to obtain a general curved boundary, consistent with the finite

element model in which the structure is allowed to assume whatever shape is necessary to obtain the minimum weight. The problem with this generality is that an undesirable or impractical shape may be produced.

The appropriate selection of the shape representation techniques for a particular problem is necessary for effective optimisation. There are two main considerations in the selection of design variables. First, the number of design variables must be kept to a minimum since each design variable adds the burden of a number of analyses to the total computational effort required in the optimisation process. In terms of evolutionary optimisation, a large number of variable parameters are required to produce even the simplest of shapes. The more variables in the phenotype, the more genes there are in the genotype, making the search problem larger and thus more complex. Secondly a limit on the number of design variables restricts the changes in shape during optimisation and may exclude a good practical shape which might lead to a better design. There are no general sets of rules governing the task of optimum selection of a shape representation technique. Engineering insight and a compromise for the particular problem is therefore required to make this choice.

Taking the above issues into consideration the plate is split into individual elements (similar to brick elements when using FEA) in order to provide a higher degree of resolution. The plate is represented by regions which are described by a set of key nodes that control the geometry. The nodes are allowed to move in one dimension (i.e. depth) during the evolutionary design process. This method was developed in order to allow the designer flexibility whilst keeping the number of variables to a minimum. A major problem with this choice is the resulting large number of design variables which increase the computational expense. To produce even a simple shape requires large numbers of

elements. However the advantage is the ability for the plate to assume a large variety of shapes.

3.2.8 NISA FEA Software

During the course of teaching company programme a software package was developed for optimising flat plates. The package consisted of the CHC algorithm integrated with FEA software. Before deciding to integrate the CHC with FEA, various algorithms were investigated and experiments performed using the computationally inexpensive complex stress model. The results are discussed in chapter 4. Before performing the optimisation the designer must firstly define a Finite Element model and identify the variables. The plate is initially designed with minimum thickness throughout the body. Eight noded brick elements are used for modelling the plate. The files describing the model are used as a template which are amended automatically by the CHC software to include the new values for the variables for each evaluation during the optimisation process. The nodes may be linked in several ways to allow flexibility to the designer. This also helps to reduce the overall number of variables.

There are generally two types of relationships between the variables and nodes. The first is a one to many relationship, where one variable may have several nodes or elements attached to it. This is referred to as a variable area. The second is a one to one relationship where one node is equivalent to one variable (Figure 3-6 and Figure 3-7). The models may be created with small tapers to aid blending between the elements (Figure 3-5). It was found that if all nodes were allowed to vary the resultant shape was usually impractical, moreover there is also a large increase in the number of variables.

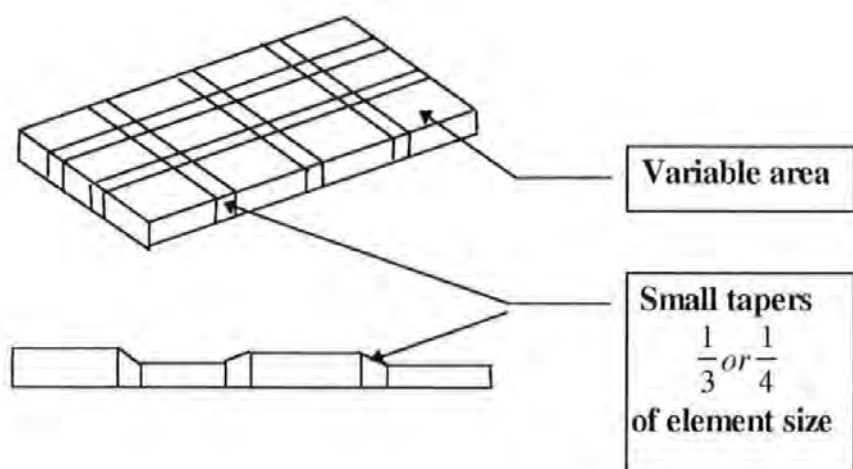


Figure 3-5: A diagram to show element tapers

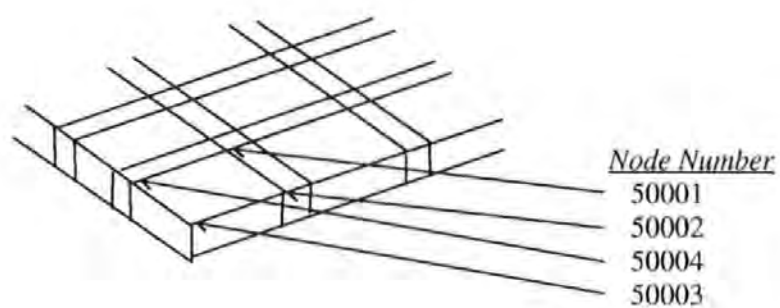


Figure 3-6: A diagram to show node id's











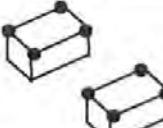
Variable	Number of Variables	Number of Elements	Number of Nodes attached
	1	1	4
	1	2	6
	1	3	8
	1	5	12
	1	9	16
	1	4	9
	4	1	4 (each node is a variable)
	6	2	6 (each node is a variable)
	8	3	8 (each node is a variable)
	9	4	9 (each node is a variable)
	1	2	8

Figure 3-7: Explanation of the different types of variables

A variable may have any number of nodes attached to it, e.g :

1st variable (4 nodes) starting id 50001 (last node number = 50004)

2nd variable (6 nodes) starting id 50005 (last node number = 50010)

3rd variable (100 nodes) starting id 50011 (last node number = 50110)

Once the designer has developed the model and created the appropriate files which allow communication between the FE software and the CHC GA, the evolutionary design software may be run. The software runs until such a time the user feels that the design is acceptable, or until the search process has stagnated. The designer may view a graphical representation of the best design solution at any stage of the optimisation process. Due to the considerably long run time information is recorded regarding the run which is automatically saved every n number of evaluations, in case of a system failure. This allows the user to recommence the program close to the point it was stopped. The design software does not permit human intervention during evolution.

3.3 Summary

This chapter has discussed the application of evolutionary algorithms to structural optimisation problems and shows that this area is receiving considerable interest. This chapter has also discussed the development of the CHC genetic algorithm for the optimisation of a real world structural plate problem, during a two year Teaching Company Programme. By combining the automatic optimisation of a design alongside evaluation software which automatically analyses the quality of the designs, considerable time on the part of the designer may be saved. The developed software is currently being utilised in industry for the optimisation of flat plates. The software not only provides practical solutions but, as it does not commence from a feasible point (a common practice in shape

optimisation) it can also provide novel design solutions. Plates designed using the developed software have now been mass manufactured. They have provided design solutions superior to those in existence and therefore have made significant savings for the industrial partner.

The next chapter provides a comparison in performance of different existing evolutionary algorithms in order to determine which is the most effective on the plate optimisation problem.

4. APPLICATION OF EVOLUTIONARY / ADAPTIVE ALGORITHMS

4.1 Application of Evolutionary Algorithms to the Plate Problem

As described in chapter 2, there are a number of advanced GA variants, all developed to improve the efficiency of evolutionary search for problem classes. The plate problem has several levels of complexity relating to multi-objectives, high dimension and high sensitivity to slight perturbation of design variables. It is therefore necessary to introduce high performance evolutionary algorithms that can best handle such characteristics.

The genetic algorithm is only one of many non-linear adaptive search algorithms known in computer science. It is currently not possible to define exactly which of these search algorithms is best for which problem or even class of problem [Fogel 1995]. However, it is possible to identify algorithms that continuously produce “good” results (in comparison to those produced by other techniques) for a wide range of different problems. The GA exhibits robust behaviour having been successfully applied to many problem classes.

The objective of the following sections is to provide a comparison in performance of different evolutionary algorithms on the plate optimisation problem. The algorithms have been selected on the basis of their performance on various problems in comparison to other search techniques. The objective here is not to optimise all possible operator parameter settings for any particular problem. The process of the selection of optimal settings is complex and has been investigated many times before on different classes of problems [Grefenstette 1986, Schaffer & Morishima 1987, Fogerty 1989, Davis 1989, Goldberg

1989]. In fact the “no free lunch” theorem for search states that no such optimal settings exist for all possible problem classes [Wolpert and Macready 1995].

There are three main measures of performance :

1. As the analysis module dominates the expenditure of resources on the plate optimisation problems it is therefore considered the base cost and is used as a measure of efficiency in this thesis. The criteria utilised relates to total number of calls required to arrive at good feasible design solutions and the CPU cost of the analysis.
2. The effectiveness of the algorithm at locating a good design solution, i.e. to minimise cost and degree of stress violation
3. The robustness of the method, i.e. standard deviation of the results.

This chapter presents the results for the different types of evolutionary algorithms discussed in Chapter 2 in relation to the plate problem. Some of these techniques play a significant role in the thesis by guiding the research towards certain avenues and laying the foundations for the development of various techniques. The chapter firstly looks at the application of the various evolutionary algorithms utilising the complex stress model. This model was developed in order to produce design solutions quickly by keeping computational expense to a minimum during experimentation. The simplified model does not perform an in-depth structural analysis as a result the designs produced by this method must be considered high risk. The simplified model helps to determine the best performing evolutionary algorithms which are then integrated with the more computationally expensive FEA model. The second half of the chapter discusses the results of this integration.

4.2 Operator Settings for the Evolutionary Algorithms

The operator settings for the different evolutionary algorithms are shown in Table 4-1. In some problem cases, an attempt was made to improve the performance of the algorithm further through the identification of better operator settings.

Algorithm	Operator	Setting
PBIL	Positive Learning Rate	1.0
	Negative Learning Rate	1.0
	Forgetting Factor	0.005
	Mutation Shift	0.05
	Mutation Probability	0.02
	Trials per Iteration	40
	Number of Vectors to Update from	1
	Max. No. of Evaluations	10000
CHC	Population Size	40
	Divergence Rate	30%
	Maximum Number of Restarts	3
Canonical GA	Population Size	50
	Mutation Rate	0.001
	Crossover Rate	0.7
	Selection Method	Roulette Wheel
BGA	Population Size	50
	Mutation Rate	0.001
	Crossover Rate	0.7
	Top T% of Individuals Selected for Mating	20%

Table 4-1 : Operator Settings for the Various Algorithms

The PBIL algorithm used throughout this thesis differs slightly from the one outlined in chapter 3. The probability vector $P(i)$ is moved away from that of the “worst” trial solution only in those bit positions where the “worst” and “best” probability vector differ [Greene 1996]. The best solutions are tracked during the search and the evaluation of individuals is aborted as soon as one is found which is better than the previous best. This individual is then used to update the probability vector. Greene [1996] found that this process provides a

rapid improvement in the early stages of the search process. In order to maintain diversity and prevent the elements of the probability vector drifting rapidly towards 0 or 1, a “forgetting factor” (Equation 4-10) is utilised. This has the effect of moving each element of the probability vector a small amount towards 0.5. In addition the mutation operator is also utilised to maintain diversity.

$$P(i) = P(i) - \gamma (P(i) - 0.5) \quad (\text{Equation 4-10})$$

γ = forgetting factor

4.2.1 Results for the Flat Plate Problems Utilising the Complex Stress Model

During the plate optimisation a simplified model has been utilised as described in chapter 3 to keep computational expense to a minimum during experimentation. The simplified model does not carry out an in-depth structural analysis and therefore generated results are not as reliable as those produced by the finite element method. As a result the designs produced by the simplified method must be considered high risk. Using such a simplified model which is still characterised by dimensionality, multi-modality and sensitivity, a technique may be developed to cope with such conditions during comparative experimentation without the burden of running large computationally expensive analysis software.

4.2.1.1 Single Load Case Problems

Figure 4-1 displays a plate simply supported on four corners with a central load of 1500 Newtons. An initial study has been performed on plates of varying resolution with a single load case. This type of problem should not pose a particular challenge to the algorithms. As

it is a one load case problem, material must be concentrated around the area of the load to minimise stress violation. However as the problem is further complicated by increasing the dimensionality or constraining the design by reducing the upper limit on material it becomes more challenging for the various adaptive search algorithms. The reduction in the upper limit has the effect of reducing the number of feasible design solutions in the search space, whilst the search space remains the same size. Reducing the upper limit on variable depth means there is less material available and finding a solution with low stress violation therefore becomes difficult.

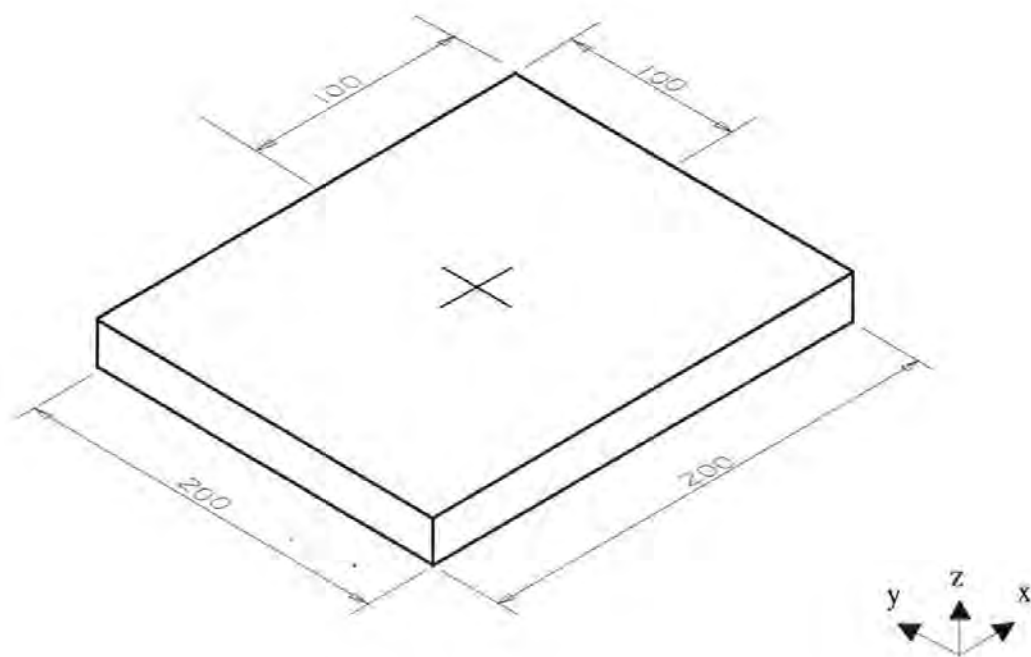


Figure 4-1: Simply supported plate with a central load

4.2.1.1.1 Comparison of Results for single Load Cases

Initial results using a simple, canonical GA with optimised parameter settings were disappointing [Vekeria and Parmee 1996]. Due to the perceived sensitivity of the problem the processing capabilities of the canonical GA does not seem appropriate. Severe

degradation of the convergence characteristics is evident as the number of variable elements increases (Figure 4-2). Ten runs have been executed for each test case.

Figure 4-2 shows that for relatively small numbers of variables (i.e. 15), the canonical GA is able to converge on high fitness design solutions. As the number of variables increases, the canonical GA becomes increasingly less efficient and is unable to converge on “good” design solutions. The figure also shows that the subsequent integration of a breeder GA, PBIL and the CHC GA results in significant improvements in fitness of solution, although performance degradation is still evident with increasing dimensionality. The CHC algorithm performs well on all plate representations however on the higher dimensional representations (400 +) the CHC is slightly exceeded in performance by PBIL, utilising a high learning rate of 1.0 (Table 4-2 and Table 4-3). The results however show the CHC to be an extremely effective form of search algorithm when utilising varying numbers of elements compared to the other methods when integrated with a single load case problem.

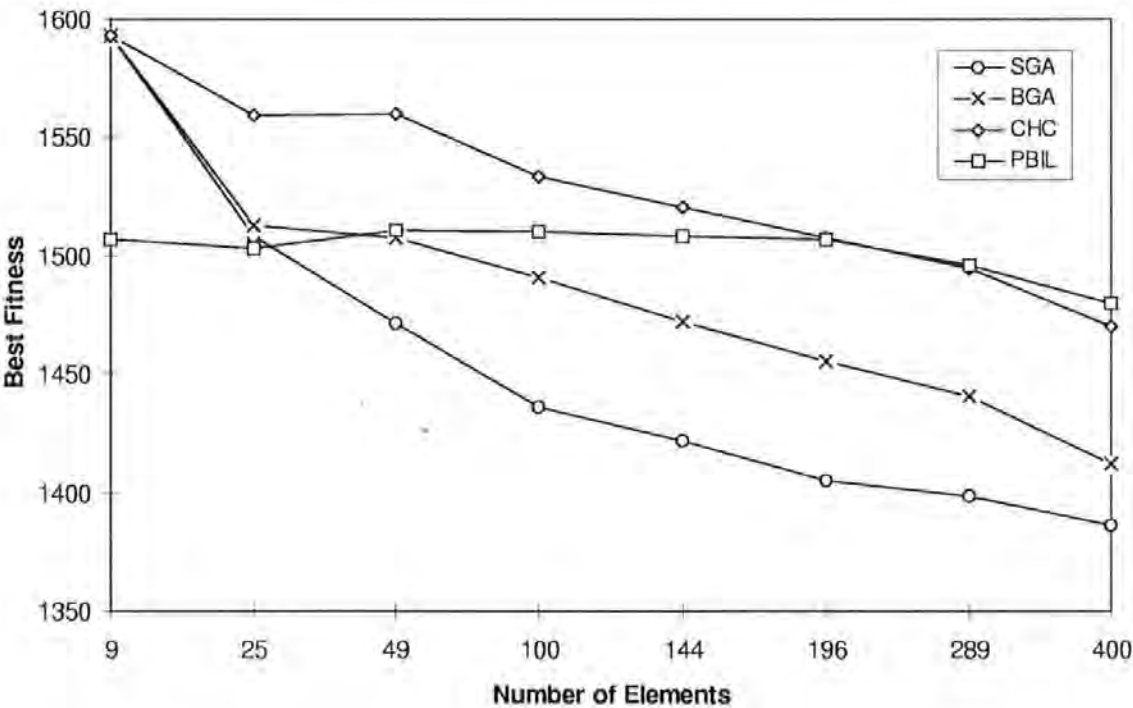


Figure 4-2: Performance comparison of the various search techniques (1 load case)

Max=24mm.

CHC (1 load case)							
Test Number	Plate size	Max upper limit	Best Fitness	Best Weight	Average Fitness	Average Weight	Fitness (SD)
CHC_1	20x20	24	1482.236	1.04	1471.522	1.06	5.748839
CHC_2	20x20	18	1474.914	1.05	1468.695	1.07	2.929435
CHC_3	24x24	24	1455.896	1.10	1450.828	1.11	3.390425
CHC_4	24x24	18	1464.129	1.08	1457.714	1.09	3.032191

Table 4-2 : Results for CHC for various problem cases utilising a single load case (no. of runs = 10)

PBIL (1 load case)							
Test Number	Plate size	Max upper limit	Best Fitness	Best Weight	Average Fitness	Average Weight	Fitness (SD)
PBIL_1	20x20	24	1487.756	1.03	1480.513	1.04	4.322704
PBIL_2	20x20	18	1480.944	1.04	1472.525	1.06	5.484269
PBIL_3	24x24	24	1455.107	1.10	1452.505	1.11	2.784014
PBIL_4	24x24	18	1462.017	1.08	1458.732	1.09	2.249974

Table 4-3 : Results for PBIL for various problem cases utilising a single load case (no. of runs = 10)

The evolution curves are shown for the fittest individual of the 10 runs in Figure 4-3 to Figure 4-6 which show a rapid increase in fitness for both CHC and PBIL due to the initial satisfaction of the stress criteria. Once stress satisfaction is minimised the fitness increases gradually in terms of weight reduction of the plate. Figure 4-2 shows that PBIL's performance improves with increasing grid resolution when dealing with a single load case, suggesting that a tendency for premature convergence is offset by the sheer number of possible design directions available at higher dimensions. Whereas the more diverse search of the CHC begins to lose its way, PBIL manages to sustain a better compromise between exploration and exploitation and finally outperforms the CHC as the 400 element representation is approached. Another feature of PBIL is its rate of convergence during

early generations with medium to high grid resolutions as shown in Figure 4-3 to Figure 4-6. This suggests that different techniques may be better suited to varying stages of the evolutionary process. However, rapid convergence during the early stages may not prove beneficial in the longer term as this may lead to convergence upon a local optima later in the search. A major advantage in using the CHC algorithm is that it requires little or no operator tuning. Although progression for the CHC is slower than PBIL due to it's explorative nature, it manages to converge on good design solution during the latter stages of the search. The CHC also proves to better handle exploration of the search space across a number of different resolution plates (see Figure 4-2). The Nisa FEA software discussed in following sections was therefore initially integrated with the CHC GA.

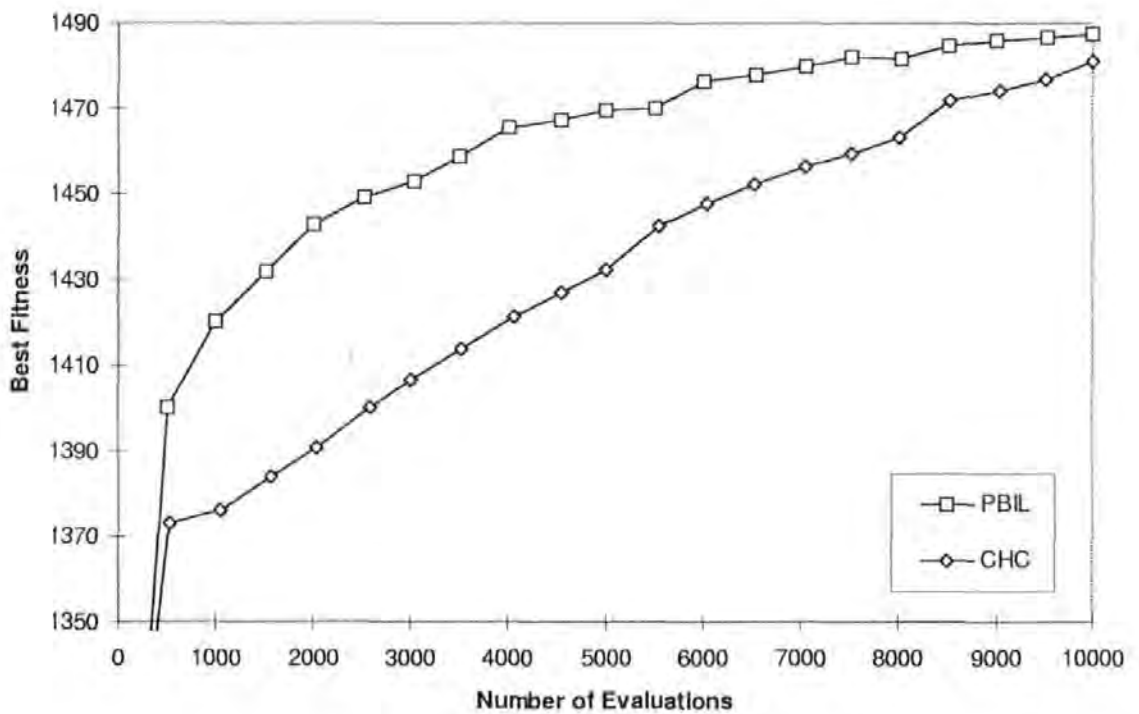


Figure 4-3: Comparison of PBIL and CHC GA (1 load case) (20 x 20 plate) max =24mm

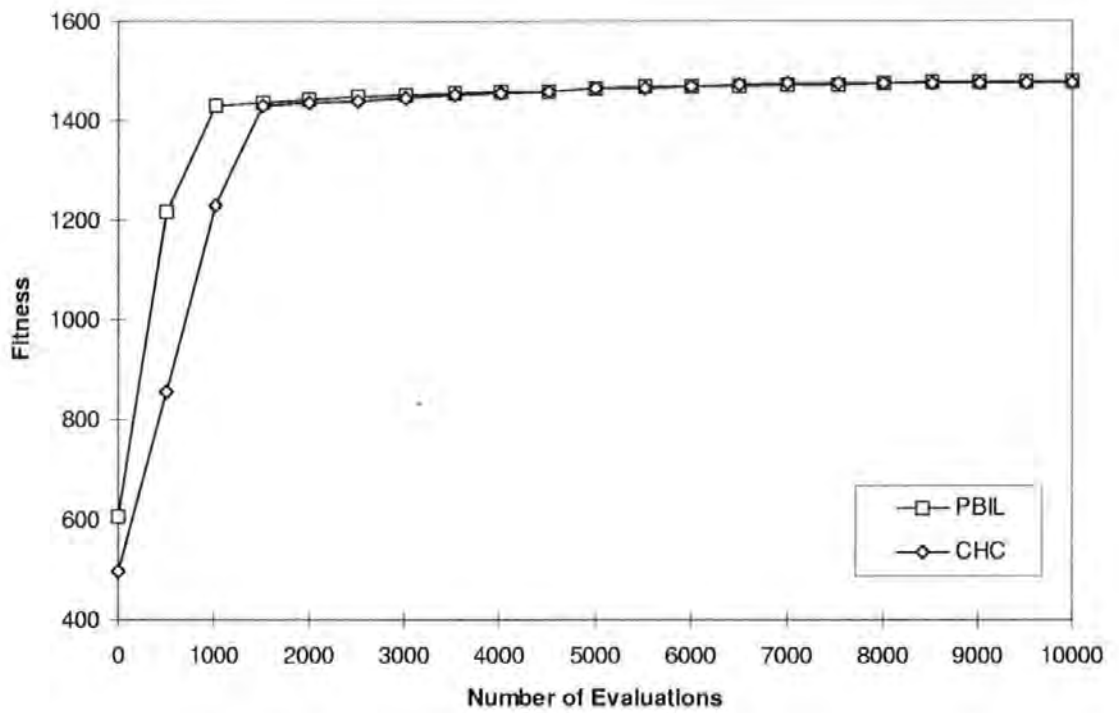


Figure 4-4: Comparison of PBIL and CHC GA (1 load case) (20 x 20 plate) max =18mm

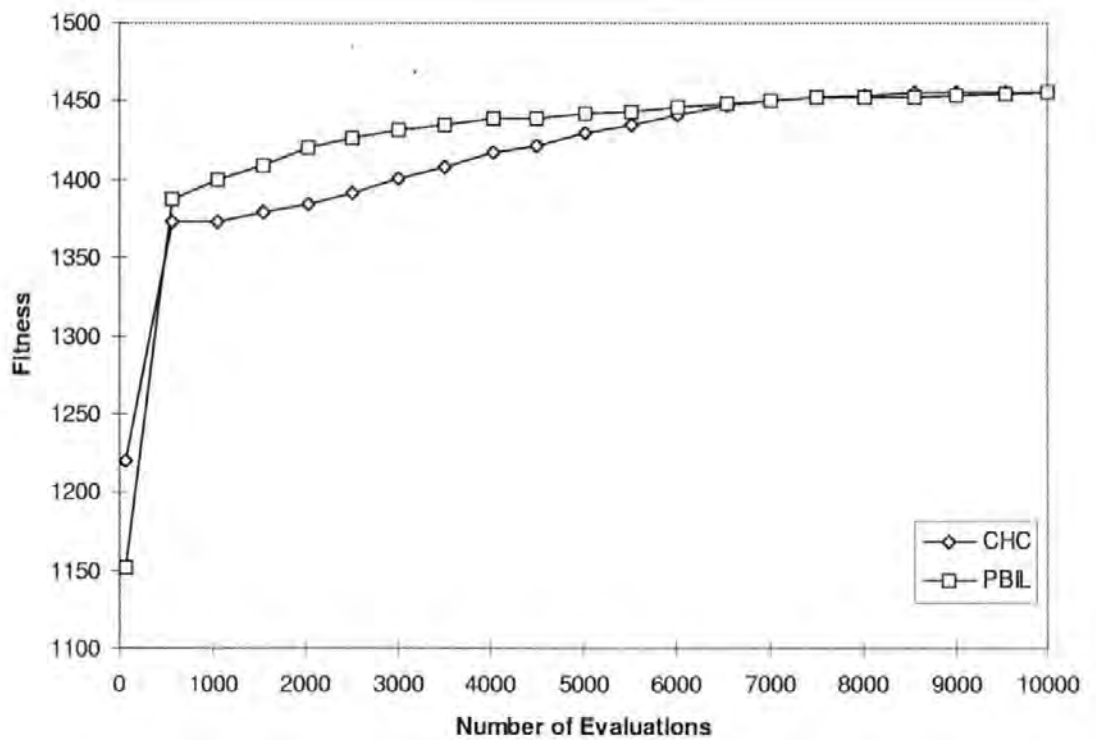


Figure 4-5: Comparison of PBIL and CHC GA (1 load cases) (24 x 24 plate) max =24mm

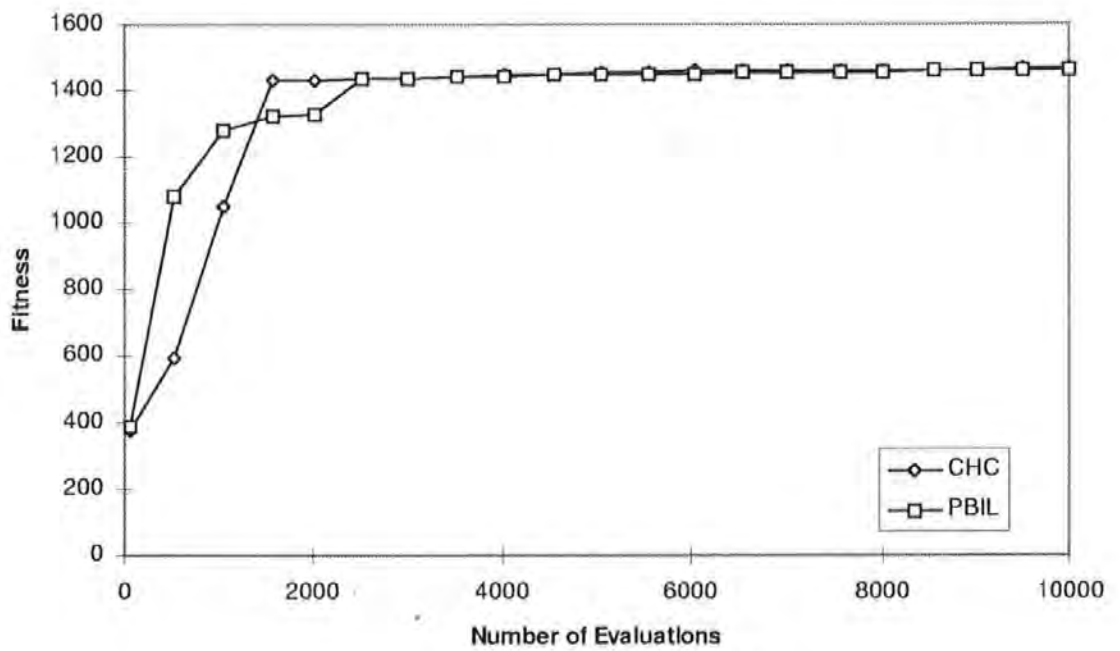


Figure 4-6: Comparison of PBIL and CHC GA (1 load case) (24 x 24 plate) max = 18mm

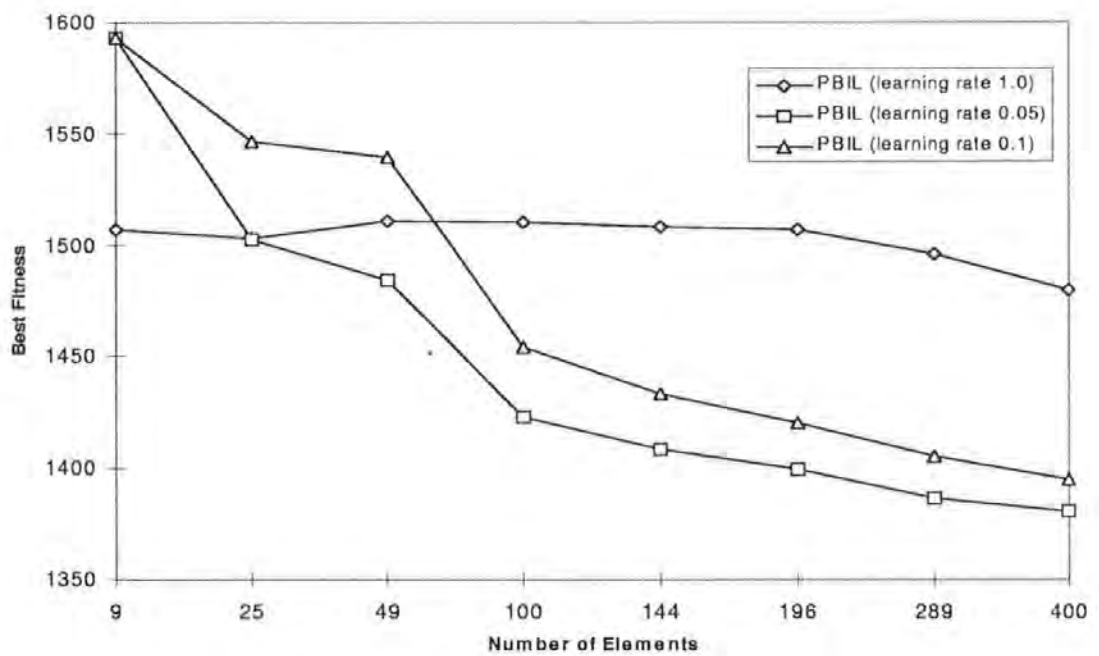


Figure 4-7: Comparison in performance of different learning rates (1 load case)

Figure 4-7 shows the comparison in performance for PBIL with different learning rates for a single load case. This illustrates the utility of lower learning rates for lower dimensional cases whilst high learning rates are better suited to higher dimensions. Reducing the learning rate has a direct impact on the trade-off between exploration and exploitation of the search space. For example if the learning rate is 0.1, there is little exploitation of solutions. As the learning rate is increased, the amount of exploitation increases, and the ability to sample large portions of the space diminishes. The learning rate provides the selection pressure for PBIL. In lower dimensions there are fewer design solutions which meet the design criteria therefore a high learning rate results in a poor solution due to the search algorithm focusing on the top individuals in the space, i.e. the overall diversity in the gene pool reduces resulting in premature convergence. In this case it is suggested that lower learning rates are utilised to promote better sampling of the search space. In the higher dimensions (>200) with a single load case there are many feasible solutions which meet design objectives. This results in many possible design directions for the algorithm. A high learning rate exploits good solutions and rapidly negotiates this very large search space to identify a high performance locally optimal solution.

When using the simple GA, a degree of operator tuning must be performed. This is also the case with PBIL, as the problem is scaled up i.e. utilises more variables. Further experiments utilising higher dimensions are only performed on the best performing of the four algorithms, i.e. the CHC and PBIL. Constraining the designs by reducing the upper limit of variable depth does not pose a problem for the CHC or the PBIL algorithm. As mentioned earlier in the chapter, the reduction in the upper limit has the effect of reducing the number of feasible design solutions in the search space, whilst still keeping the search space the same size. Less material is available during optimisation resulting in a reduced number of design solutions with lower stress violations.

4.2.1.2 Multiple Load Case Problems

Unlike the previous one load case problem where the material is more localised, this section examines multi-load cases. With such a problem material is spread across the plate resulting in significant increases in stress violations. The problem is further complicated by increasing the dimensionality or constraining the design by reducing the upper limit on material. Multiple load cases are common in structural design, a component may be subjected to various load conditions during it's lifetime. Optimising several load cases poses a considerable challenge to the algorithms. Figure 4-8 displays a plate simply supported on four corners with a central load of 1500 Newtons and two further loads of 800N each.

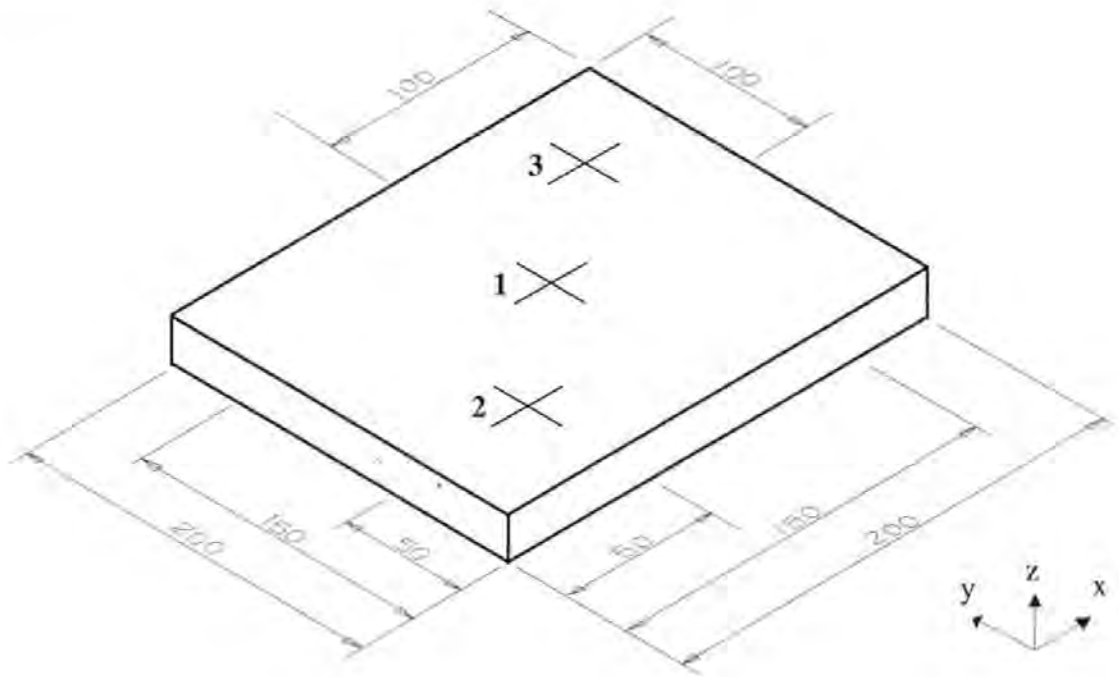


Figure 4-8: Simply supported plate with three load cases

4.2.1.3 Comparison of Results for Multi Load Cases

Results for the BGA and the canonical GA are disappointing (Figure 4-9). As the number of variables are increased, they become increasingly less efficient and are unable to converge on high fitness solutions. Ten runs are performed for each test case.

CHC (3 Load cases)							
Test Number	Plate size	Max upper limit	Best Fitness	Best Weight	Average F	Average Wt	Fitness (SD)
CHC_5	20x20	24	1422.33	1.18	1419.221	1.19	2.71
CHC_6	20x20	18	1427.38	1.17	1403.781	0.12	52.92
CHC_8	24x24	24	1410.92	1.22	1405.492	1.23	4.33
CHC_9	24x24	18	1419.57	1.19	1307.997	1.22	178.22
CHC_7	20x20	24	1486.46	1.03	1481.210	1.04	4.26
Evaluations = 50000							

Table 4-4 : Results for CHC for various problem cases utilising three load cases (no. of runs = 10)

PBIL (3 Load cases)							
Test Number (table No.)	Plate size	Max upper limit	Best Fitness	Best Weight	Average F	Average Wt	Fitness (SD)
PBIL_5 L.R = 1.0	20x20	24	1446.37	1.12	1437.81	1.14	4.59
PBIL_6 L.R = 1.0	20x20	18	1441.21	1.12	1293.16	1.14	97.22
PBIL_7 L.R = 0.1	20x20	24	1367.14	1.36	1363.54	1.38	1.96
PBIL_8 L.R = 0.1	20x20	18	1215.41	1.26	1137.90	1.27	36.73
PBIL_9 L.R = 1.0 Evaluations = 50000	20x20	24	1470.52	1.06	1462.08	1.08	5.02
PBIL_11 L.R = 0.1	24x24	24	1360.04	1.39	1355.42	1.41	2.80
PBIL_16 L.R = 0.1	24x24	18	774.46	1.24	703.09	1.26	80.49
PBIL_12 L.R = 0.4	24x24	24	1394.03	1.27	1388.25	1.29	3.07
PBIL_17 L.R = 0.4	24x24	18	1399.59	1.25	1271.78	1.26	111.11
PBIL_13 L.R = 0.6	24x24	24	1399.50	1.25	1395.60	1.27	3.55
PBIL_18 L.R = 0.6	24x24	18	1218.30	1.24	1041.01	1.25	88.45
PBIL_14 L.R = 0.8	24x24	24	1411.28	1.22	1405.15	1.23	3.58
PBIL_19 L.R = 0.8	24x24	18	1287.78	1.22	1138.97	1.23	144.73
PBIL_10 L.R = 1.0	24x24	24	1417.91	1.20	1413.54	1.21	3.27
PBIL_15 L.R = 1.0	24x24	18	1316.44	1.19	1136.69	1.21	118.34

Table 4-5 : Results for PBIL for various problem cases utilising three load cases (no. of runs = 10)

PBIL and the CHC GA perform significantly better than the BGA or the canonical GA although performance degradation is still evident with increasing dimensionality (Figure 4-9, Table 4-4 and Table 4-5). The results show that the CHC and PBIL are still extremely effective when compared to the other methods.

Figure 4-9 once again shows PBIL's performance significantly improving with increasing grid resolution, furthermore PBIL, as with the single load case still has a high rate of evolution during early stages of the search (Figure 4-10 to Figure 4-13).

Table 4-5 shows the performance of PBIL on a three load case 20x20 and 24x24 representations utilising different learning rates. This illustrates that lower learning rates which should help counteract premature convergence by promoting better exploration of the search space do not in this case provide any improvement in design performance.

Further experiments utilising higher dimensions are only executed using the best performing algorithms namely the CHC GA and PBIL. The problem is made more complex by increasing the number of load cases to 3 and reducing the maximum depth of the plate to 18mm. More material is now distributed across the plate and due to reduction in plate depth more stress violations occur across the plate resulting in fewer feasible design solutions. This poses a problem for the PBIL algorithm, it is rendered ineffective at negotiating the highly complex search space. The highly exploitive nature of PBIL is unable to locate the reduced number of feasible design solutions now present in the search space, resulting in premature convergence. CHC in comparison due to it's more explorative nature performs far better, but in some cases still fails to produce a feasible design solution. Reducing the learning rate in order to promote exploration in many of the cases does not provide better performance solutions .

Experiments relating to the CHC and PBIL algorithms utilising higher numbers of evaluations provides small improvements in overall design performance, but at a high cost in overall computational expense (Table 4-4 test CHC_7 and Table 4-5 test PBIL_9).

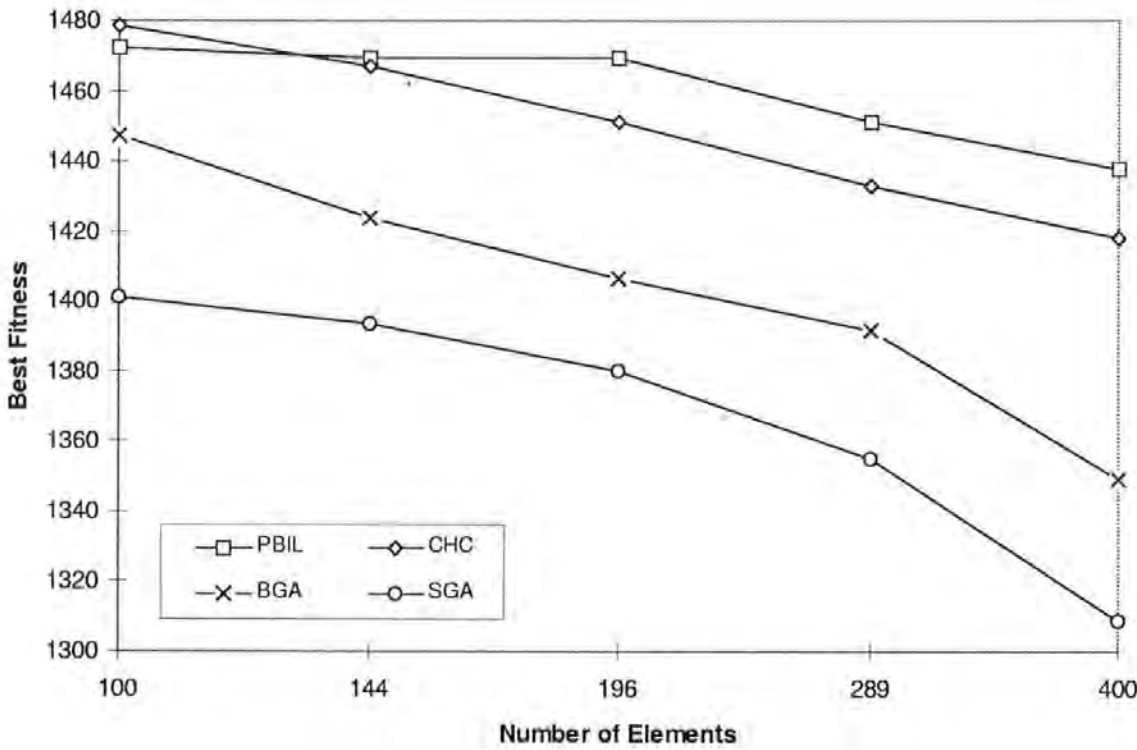


Figure 4-9: Performance comparison of the various search techniques (three load cases) (20x20), dmax = 24mm, dmin = 8mm.

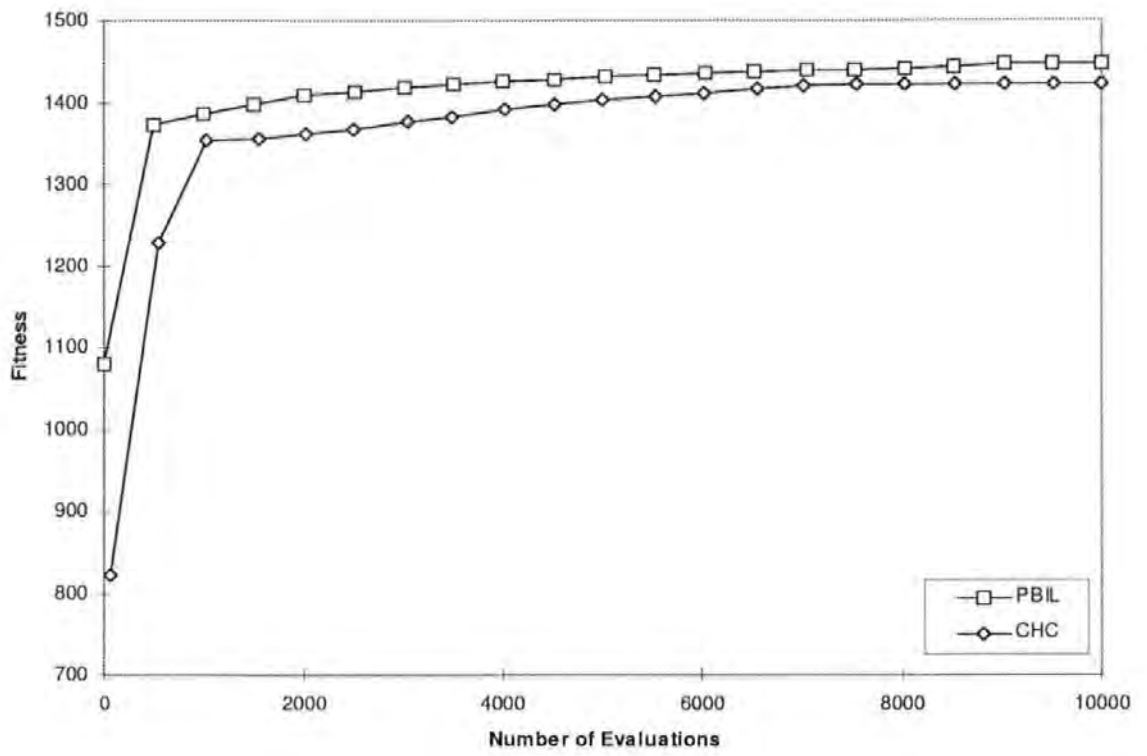


Figure 4-10: Evolution of PBIL and CHC (20x20), $d_{max} = 24$, $d_{min} = 8$, load cases = 3

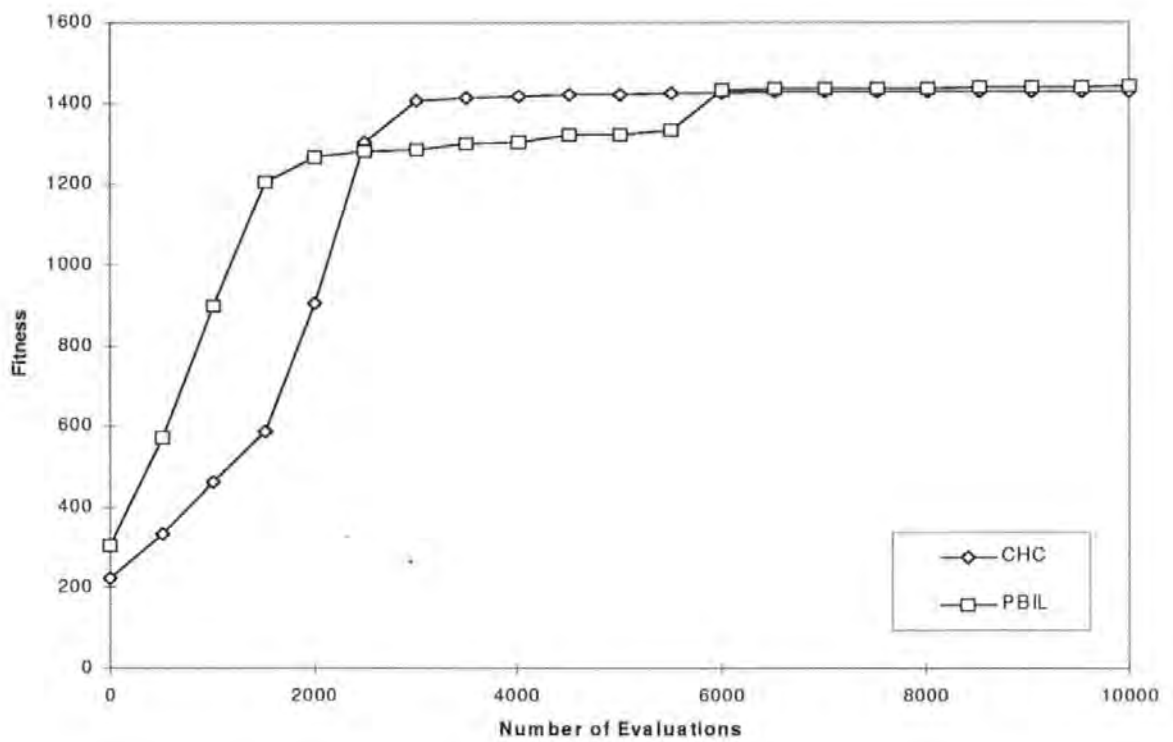


Figure 4-11: Evolution of PBIL and CHC (20x20), $d_{max} = 18$, $d_{min} = 8$, load cases = 3

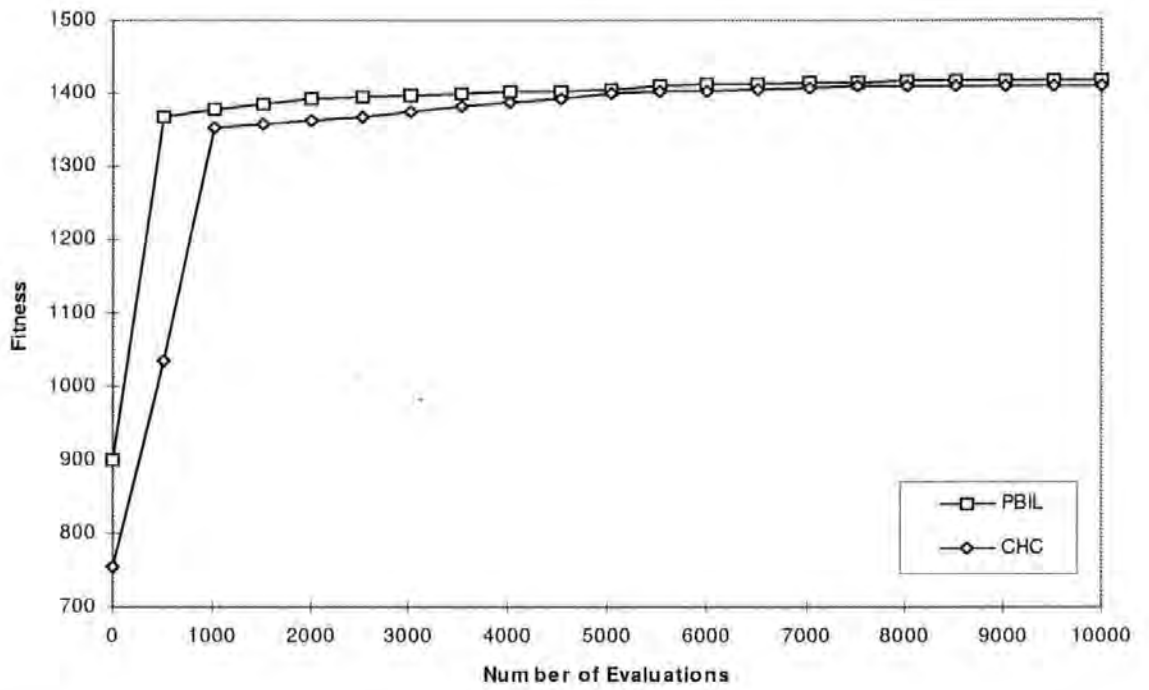


Figure 4-12: Evolution of PBIL and CHC (24x24), dmax = 24, dmin = 8, load cases = 3

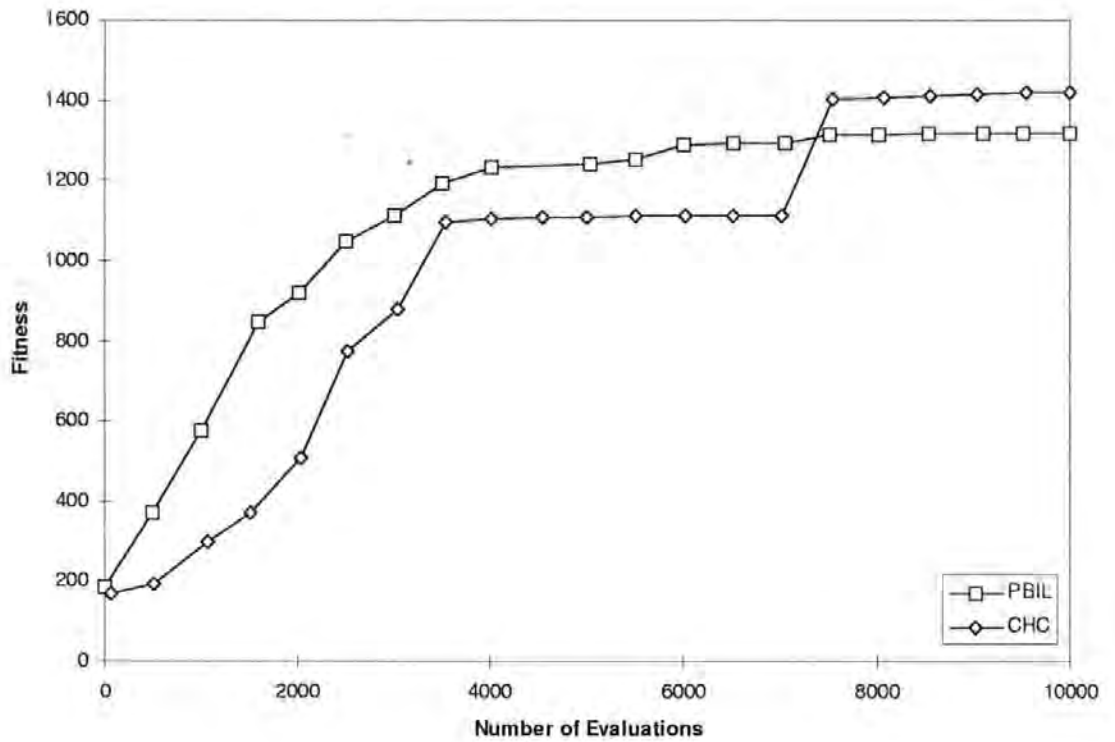


Figure 4-13: Evolution of PBIL and CHC (24x24), dmax = 18, dmin = 8, load cases = 3

An advantage of the CHC algorithm is that relatively small populations may be utilised (Figure 4-14). The CHC's many interrelated mechanisms support better exploration of the search space. These different mechanisms ensure that the CHC can explore without the disadvantage of using large population sizes and slower convergence. Increasing the population size however, tends to result in poorer performance.

Figure 4-14 shows the results for a 3 load case problem utilising different population sizes. It can be seen that the lower population sizes (< 100) provide solutions of better performance than those utilising larger population sizes (> 100). The use of disruptive crossover and incest prevention enables the CHC to delay premature convergence and arrive at good design solutions without the need for large population sizes.

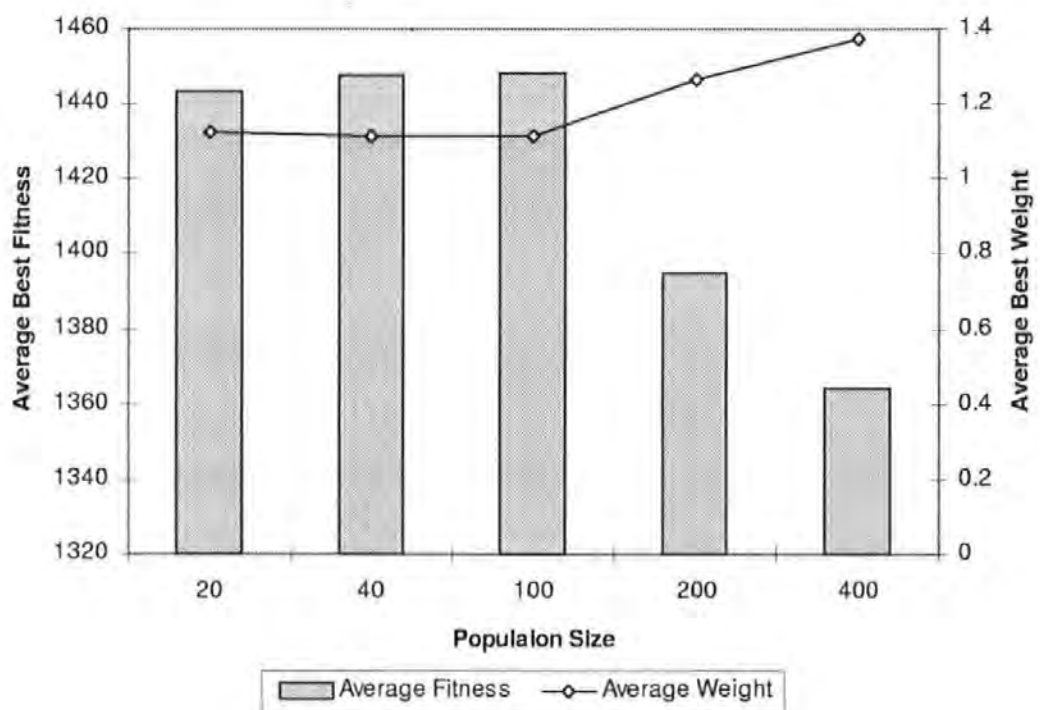


Figure 4-14: 3 load case problem utilising different population sizes for the CHC algorithm (20000 evaluations) max 24, min 8.

4.2.1.4 Drawbacks of Using the same GA for Various Levels of Problem Complexity

It is widely accepted that the major drawback of genetic algorithms to complex engineering design problems is the large number of analyses [Hafika et al 1996, Goodman et al 1996]. The number of analyses for the plate problem depends upon factors such as the level of representation, and the loading conditions. The designer would therefore need to tune the operator settings of a GA in order to suit the problem. If the problem is computationally expensive the tuning may take a considerable time, and would therefore have to take place with a simpler version of the actual problem of interest. The designer is assuming that the same algorithm with its operators and settings will do just as well when more complex problems are introduced. It has been shown that most of the algorithms utilised with the plate problem generally perform well on coarse representations when utilising a single load case and a small number of design variables. However when they are applied to more complex higher dimensional problems, some of the algorithms notably the canonical GA and BGA deteriorate considerably in performance. Therefore the use of algorithms and operator settings based on a simple version may not be sufficient to solve more complex problems.

4.2.2 Results for the Flat Plate Problems Utilising FEA

As discussed earlier in the chapter the simplified model does not carry out an in-depth structural analysis and is therefore not as reliable as the finite element method. As a result the designs produced by the simplified method must be considered high risk. If confidence in design performance is required then FEA should be utilised to provide a low risk detailed design solution. This confidence can only be achieved if there is also sufficient numbers of elements in order to allow accuracy of plate representation. Whilst providing increased confidence in the design solution it does so at the cost of greatly increased

computational expense. This next section of the chapter presents results on the utilisation of FE analysis model during evolutionary optimisation.

4.2.2.1 Single Load Case Problem

Figure 4-15 shows the FEA plate simply supported continuously along edges on the y plane. A total central line load of 396N is equally distributed as forces on the nodes. Nodes on the outer edges are half the force of the inner nodes. The plate has a minimum thickness of 10mm and a maximum thickness of 13mm. There are three possible incremental steps of 1mm each. The fixed parameters of the material are: flexural limit = 10N/mm^2 , density = $2.2 \times 10^{-7} \text{ Kg/mm}^3$, Poisson's Ratio = 0.2, Modulus of Elasticity = 14000. This is a real world problem provided by the industrial partner.

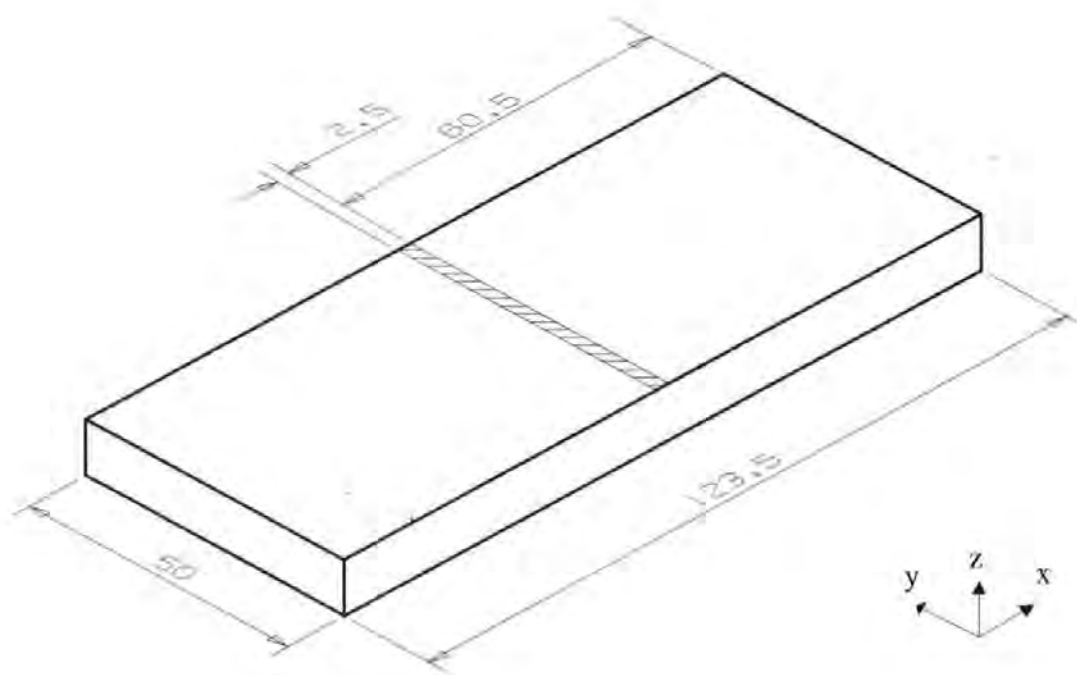


Figure 4-15 : Simply supported plate with a central load (utilising FEA)

Table 4-6 shows the information relating to two types of single load case test problems. The table highlights the considerable computational expense when performing a single evaluation even when utilising high performance workstations to perform the analysis (SUN Ultra Enterprise 4000). Due to the high computation of individual evaluations, the number of runs performed and the overall number of evaluations within each run had to be kept to a minimum in order to generate results in an acceptable period of time. As a result four runs and 3000 evaluations were performed on each problem case. The large CPU time expended on each evaluation demonstrates the need for keeping the total number of evaluations to a minimum.

Number of Variables (grid size(x,y))	Number of Load Cases	Number of Nodes	Number of Elements	CPU time (seconds / evaluation)
48 (8x6)	1	864	506	8.5
200 (10x20)	1	2880	1786	43.3

Table 4-6 : Computational expense for individual FE evaluations utilising a single load case

When utilising PBIL higher learning rates for problems employing less than 50 variables results in premature convergence (Table 4-7). Based on results from experiments with the complex stress model, in order to maintain diversity a learning rate of 0.1 is utilised. Figure 4-16 shows the average best fitness of the 4 runs for the 48 variable problem. Rapid evolution by the PBIL algorithm is apparent, however due to its highly exploitive nature it converges prematurely, resulting in the CHC algorithm eventually exceeding it in performance. A feasible solution is found relatively early, however further weight minimisation seems to pose a problem for the PBIL algorithm (Figure 4-17).

Test Number	Best Fitness	Best Weight	Average Fitness	Average Weight	Fitness (SD)
FEA_CHC1	1437.07	0.143	1436.96	0.144	0.0880
FEA_PBIL2	1436.65	0.145	1436.58	0.145	0.0645

Table 4-7 : Results for 48 variables 1 load case problem

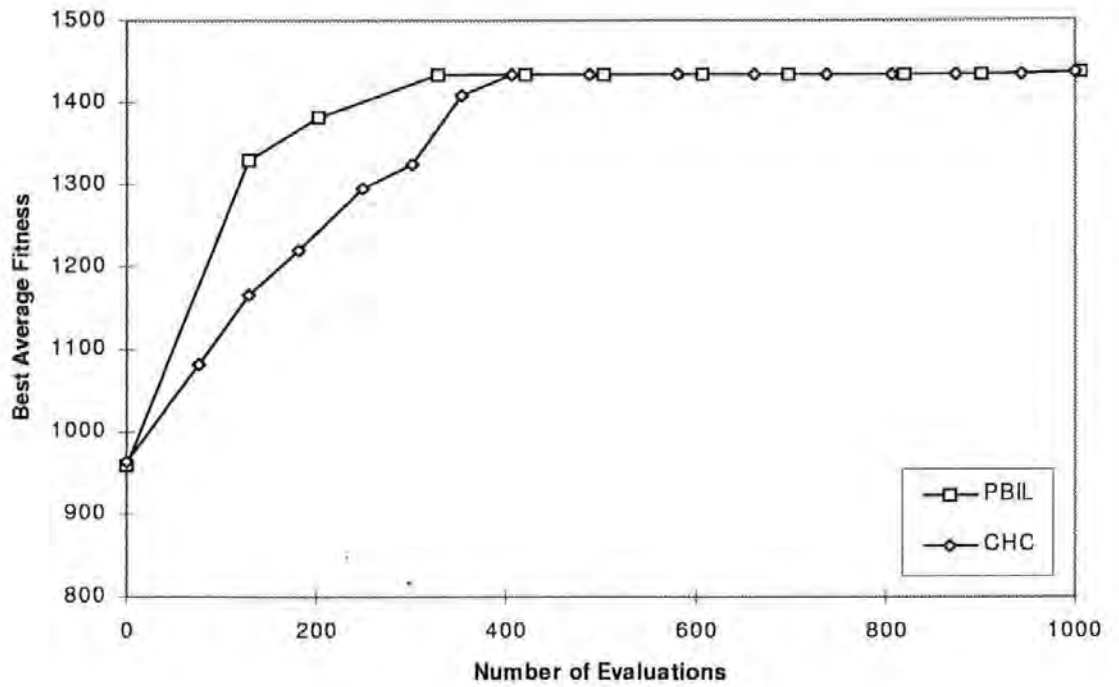


Figure 4-16 : Best Average Fitness utilising FEA (1 load case, 48 variables)

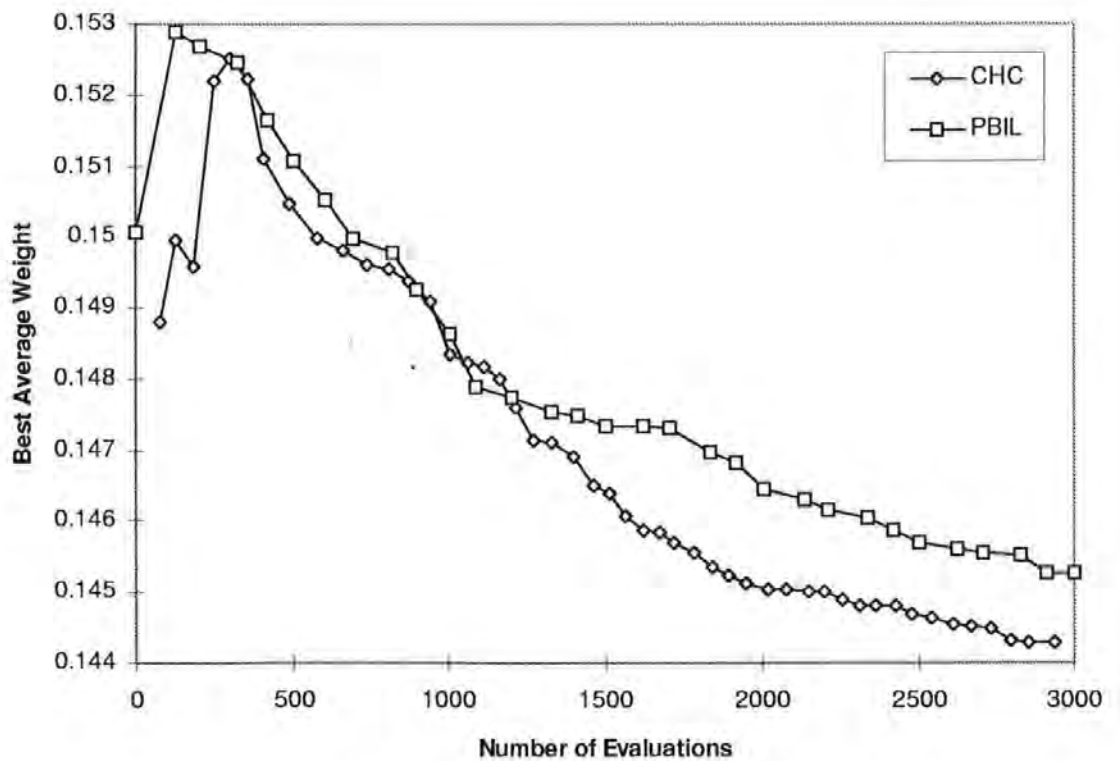


Figure 4-17 : Best Average Weight utilising FEA (1 load cases, 48 variables)

The learning rate is increased to 1.0 for the 200 variable, single load case problem (Table 4-8). Figure 4-18 shows the average best solution of the 4 runs. It is interesting to note PBIL's superior performance in comparison to the CHC. A rapid evolution of fitness by PBIL is again apparent. Figure 4-19 shows that once a feasible region is located, the PBIL algorithm performs better than the CHC at identifying lower weight. PBIL requires approximately 1300 evaluations on average to arrive at comparable design solution to the one generated by the CHC at 3000 evaluations. In a typical run to arrive at comparable design solutions CHC requires approximately 55% greater CPU time than PBIL.

Test Number	Best Fitness	Best Weight	Average Fitness	Average Weight	Fitness (SD)
FEA_CHC3	1434.72	0.150	1434.49	0.151	0.2043
FEA_PBIL4	1436.35	0.146	1436.04	0.147	1.8006

Table 4-8 : Results for 200 variables 1 load case problem

Research utilising the simpler model also showed the PBIL method to provide reduced performance on a single load case, low dimensional problems and increased performance on single load case, high dimensional problems in comparison to the CHC algorithm. These experiments therefore show to some extent that employing simpler analysis software can aid in the selection and optimisation of evolutionary and adaptive algorithms, before moving to more computationally expensive analysis tools.

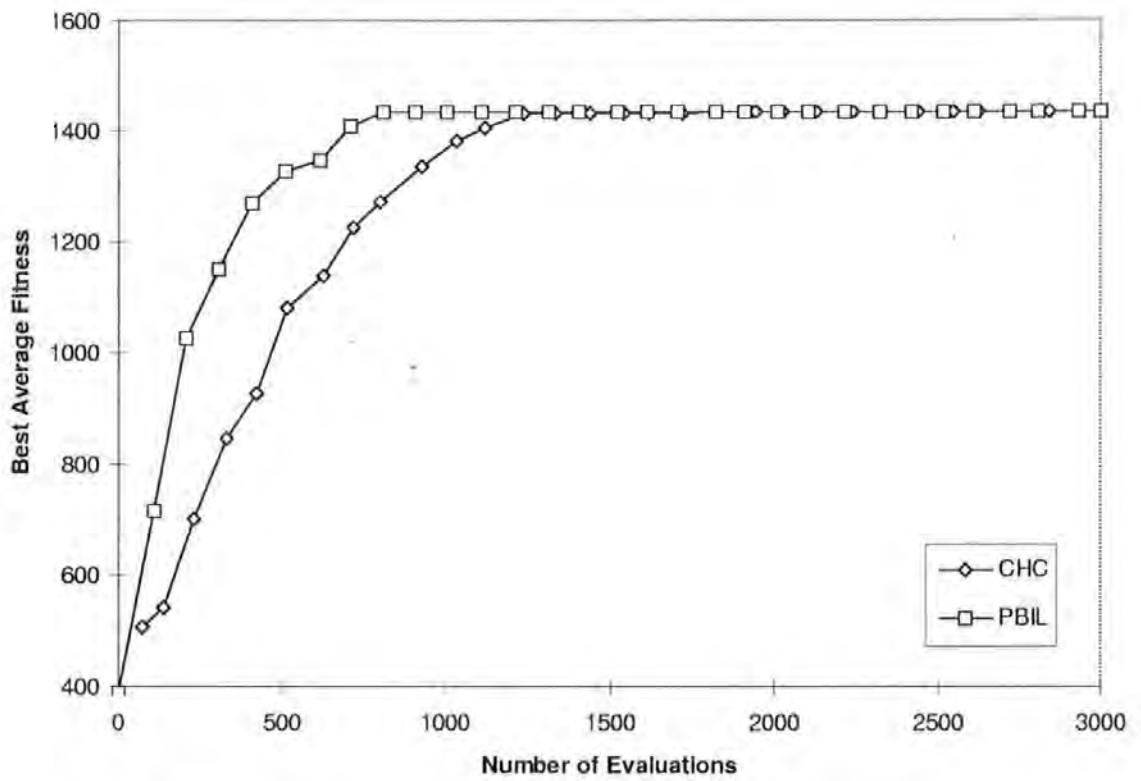


Figure 4-18 : Best Average Fitness utilising FEA (1 load case, 200 variables)

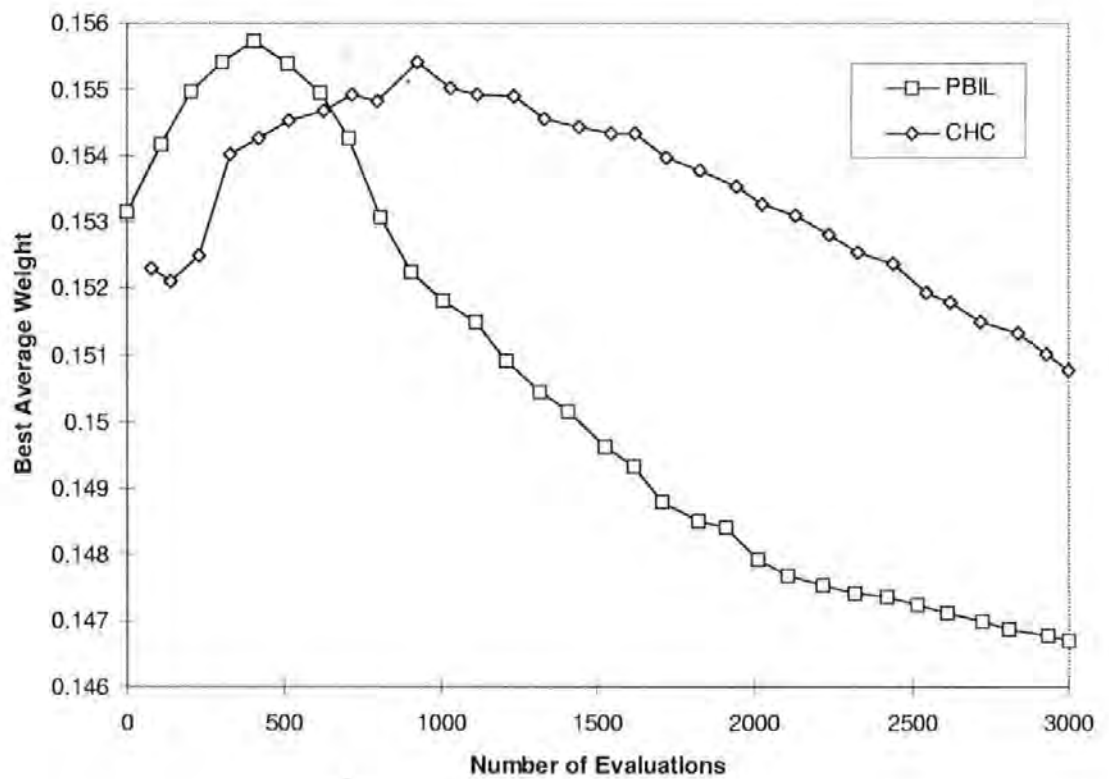


Figure 4-19 : Best Average Weight utilising FEA (1 load case, 200 variables)

4.2.3 Multiple Load Case Problem

Figure 4-20 shows the FEA plate simply supported continuously along edges on the y plane. Load case 1 has a load of 396N equally distributed as forces on nodes except outer nodes which are half of those of the inner nodes. Load cases 2 and 3, each have a pressure load of 396N. As with the single load case the plate has a minimum thickness of 10mm and a maximum variable thickness of 13mm. There are three increment steps of 1mm each. The fixed parameters of the material are: flexural limit = 10N/mm^2 , density = $2.2 \times 10^{-7}\text{ Kg/mm}^3$, Poisson's Ratio = 0.2, Modulus of Elasticity = 14000. This again represents a real world structural design problem supplied by the industrial collaborator.

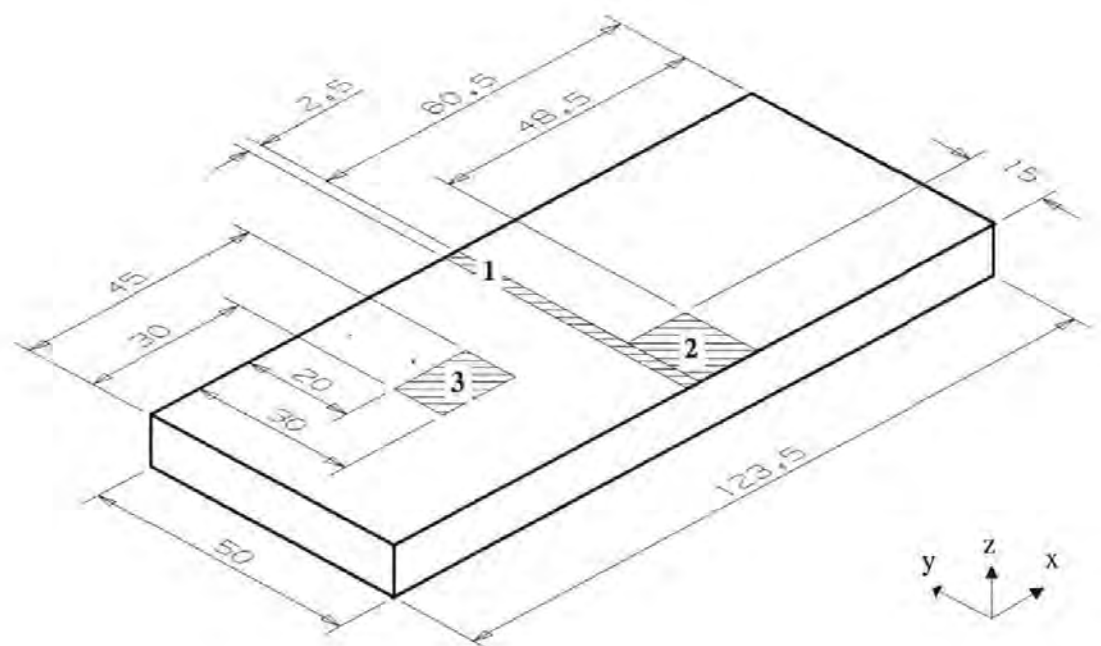


Figure 4-20 : Simply supported plate with three load cases (utilising FEA)

Table 4-9 shows the information relating to the test problem. As with the single load case FEA problems, all runs are performed on a SUN Ultra Enterprise 4000 server with 6, 167Mhz processors. Due to the high computational cost of individual evaluations, a maximum of 4 runs and 3000 evaluations were performed on each problem case.

Number of Load Cases	Number of Nodes	Number of Variables (grid size(x,y))	Number of Elements	CPU time (seconds / evaluation)
3	864	48 (8x6)	506	14.5

Table 4-9 : Computational expense for individual FE evaluations utilising three load cases

Test Number	Best Fitness	Best Weight	Average F	Average Wt	Fitness (SD)
FEA_CHC5	1434.72	0.15017	1434.46	0.150858	0.360372
FEA_PBIL6	1434.237	0.151445	1434.088	0.151841	0.13757

Table 4-10 : Results for 48 variables 3 load case problem

Table 4-10 shows the results of using the CHC and the PBIL algorithms for the test problem. A learning rate of 0.1 is utilised with the PBIL algorithm to maintain diversity. Figure 4-21 shows the average best fitness of the 4 runs for the 48 variable problem. The best average fitness refers to the fitness of the best individual in each of the runs being summated this figure is then divided by the number of runs (i.e.4). Rapid evolution of the PBIL algorithm is again apparent, however in this case due to its highly exploitive nature it converges prematurely, resulting in eventual out performance by the CHC algorithm. A feasible solution is found relatively early, however the further minimisation of the weight

again poses a problem for the PBIL algorithm (Figure 4-22). CHC requires approximately 2000 evaluations on average to arrive at comparable design solution to the one generated by the PBIL at 3000 evaluations. Therefore in a typical run to arrive at comparable design solutions PBIL requires approximately 33% greater CPU time than the CHC. The behaviour of the algorithms on the 3 load case problem utilising the FEA and the simple complex stress mathematical models are again comparable.

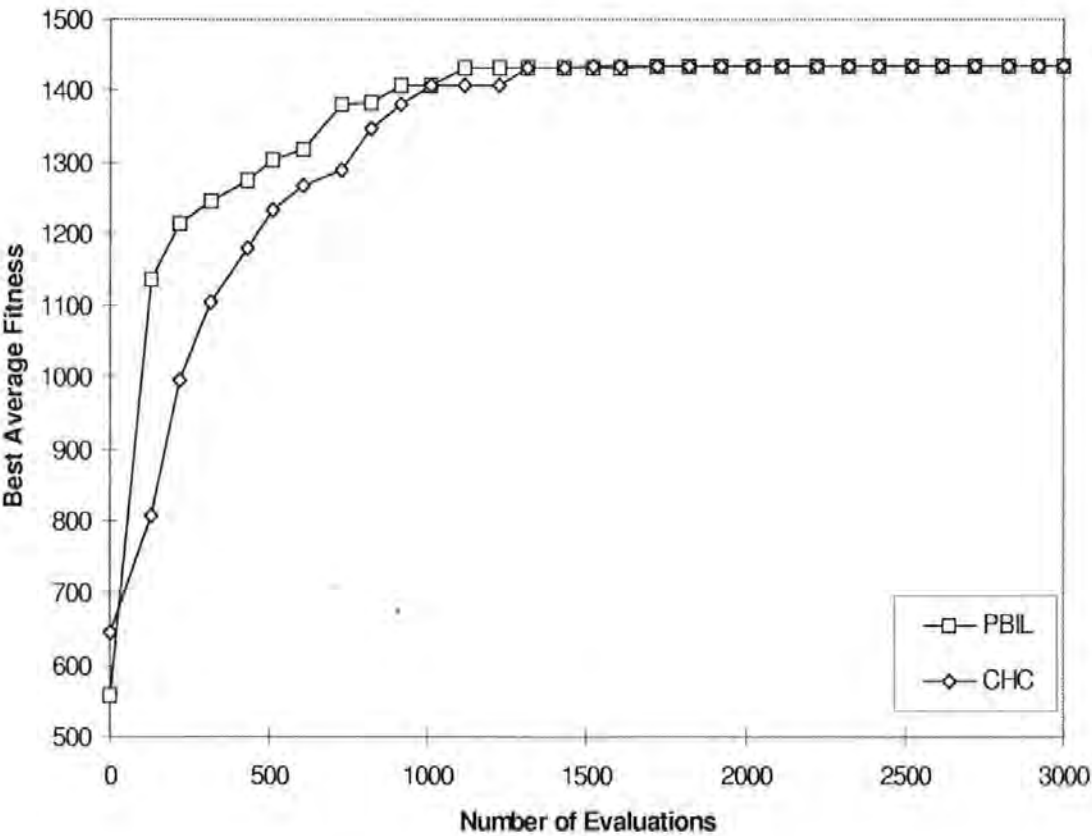


Figure 4-21 ; Best Average Fitness utilising FEA (3 load cases, 48 variables)

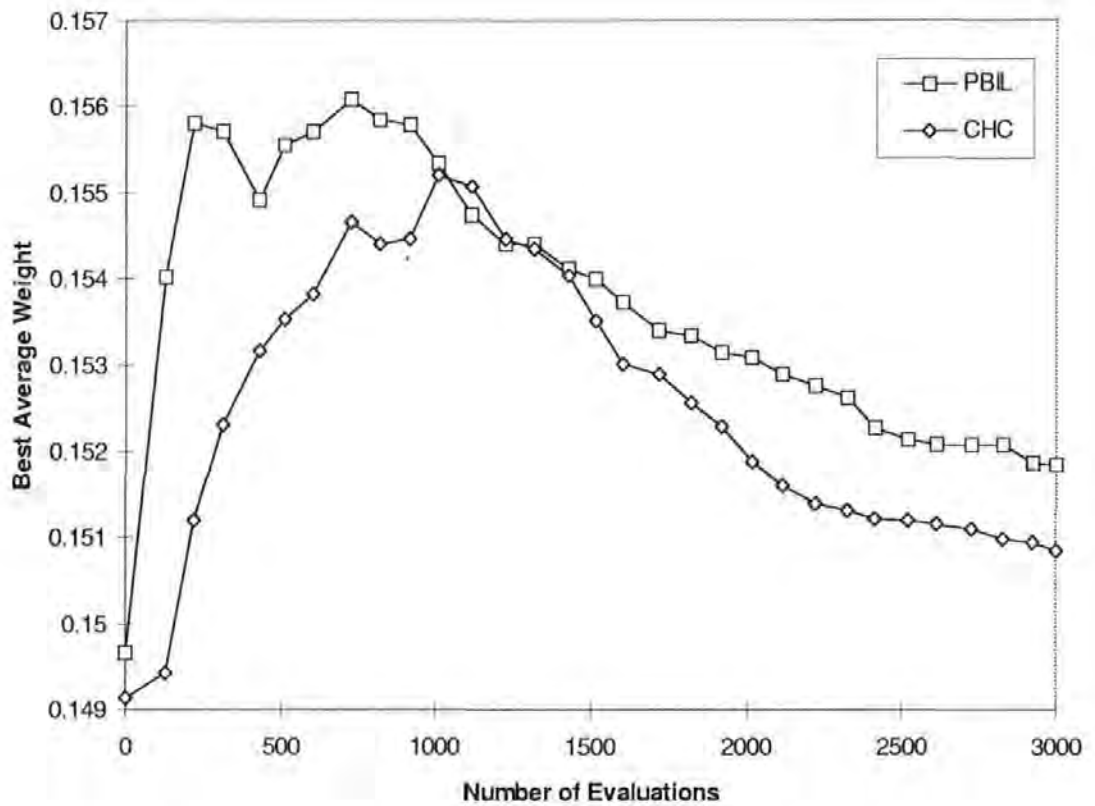


Figure 4-22 : Best Average Weight utilising FEA (3 load cases, 48 variables)

4.3 Amended PBIL with a given population size (PBIL_POP)

One of the main problems with PBIL is premature convergence. A source of diversity in the traditional GA is the number of individuals in the population. As the population size is increased more information is available in the gene pool resulting in greater diversity.

A new modified PBIL algorithm is presented in Figure 4-23 which introduces a population. The algorithm is similar to the compact GA [Harik et. al, 1997]. The compact GA proposes the generation of two individuals from the probability vector, which are then put into tournament. The better performing individual then updates the probability vector.

The algorithm proposed in this section utilises a population of individuals (Figure 4-23). The assumption is that by maintaining a population of individuals, the probability of premature convergence may be reduced, whilst still utilising PBIL's highly exploitive nature. Baluja and Caruana [1995] state that the number of samples to generate based upon each probability vector before an update is analogous to the population size of GA's. As with the traditional PBIL a real valued probability vector with values set to 0.5 is generated. The probability vector is utilised to create a population of binary encoded individuals where the probability of generating a 1 or 0 is equal. The population is then assessed via the fitness function.

The values in the probability vector gradually shift relative to the fitness of individuals in the population. The degree of variation of the probability (between 0.0 or 1.0) as in the original PBIL algorithm depends upon the learning rate parameter. Updating the probability vector results in the generation of a new population and the cycle is continued. As the search progresses, entries in the probability vector move away from their initial settings of 0.5 towards either 0.0 or 1.0 i.e. the binary representation of the individuals in the population are pushed towards that of the current best solutions. As with the original PBIL algorithm domain knowledge is not stored in the population but in the probability distribution.


```

***** Initialize Probability Vector *****
for i :=1 to LENGTH do P[i] = 0.5;

***** Generate Samples *****
for i :=1 to POPSIZE do
    sample_vectors[i] := generate_sample_vector_according__to_probabilities (P);
    evaluations[i] :=Evaluate_solution (sample[i]);

best_vector :=find_vector_with_best_evaluation (sample_vectors, evaluations);
worst_vector := find_vector_with_worst_evaluation (sample_vectors, evaluations);

***** Update Probability towards best solution *****
for i :=1 to LENGTH do
    P[i] :=P[i] * (1.0 - LR) + best_vector[i] * (LR);

***** Update Probability Away from Worst solution *****
for i :=1 to LENGTH do
    if (best_vector[i] ≠ worst_vector[i] then
        P[i] :=P[i] *(1.0 - NEGATIVE_LR) + best_vector[i] *(NEGATIVE_LR);

***** Push each element in the Probability Vector towards 0.5 by a small amount *****
***** ( Forgetting Factor ) *****
for i :=1 to LENGTH do
    P[i] :=P[i] - FF * (P[i]-0.5);

***** Mutate Probability Vector *****
for i := 1 to LENGTH do
    if (random (0,1)< MUT_PROBABILITY) then
        if (random (0,1) > 0.5) then mutate_direction :=1
        else mutate_direction :=0;
        P[i] :=P[i] * (1.0 - MUT_SHIFT) + mutate_direction * (MUT_SHIFT);

```

USER DEFINED CONSTANTS :

POPSIZE: the number of individuals in the population.

LR: the learning rate, how fast to exploit the search performed.

NEGATIVE_LR: the negative learning rate, how much to learn from negative examples.

LENGTH: the number of bits in a generated vector.

FF: the forgetting factor.

MUT_PROBABILITY: the probability for a mutation occurring in each position.

MUT_SHIFT: the amount a mutation alters the value in the bit position.

Figure 4-23: The amended PBIL algorithm (PBIL_POP)

4.3.1 Comparison of Results for PBIL_POP Utilising the Complex Stress Model

PBIL with a population performs better than the original PBIL algorithm on lower dimensions utilising a learning rate of 1.0 on a single load case (Figure 4-25). Intermediate populations i.e 20 40 and 60 seem to work best (Figure 4-24). As explained earlier in this chapter, fewer feasible design solutions exist due to lower dimensions and as a result the PBIL with a population is more effective. On the higher dimensions and still utilising a single load case it's performance is exceeded by that of the original PBIL algorithm, because there are more feasible designs the more highly exploitive nature of the original PBIL algorithm outperforms the modified PBIL with a population .

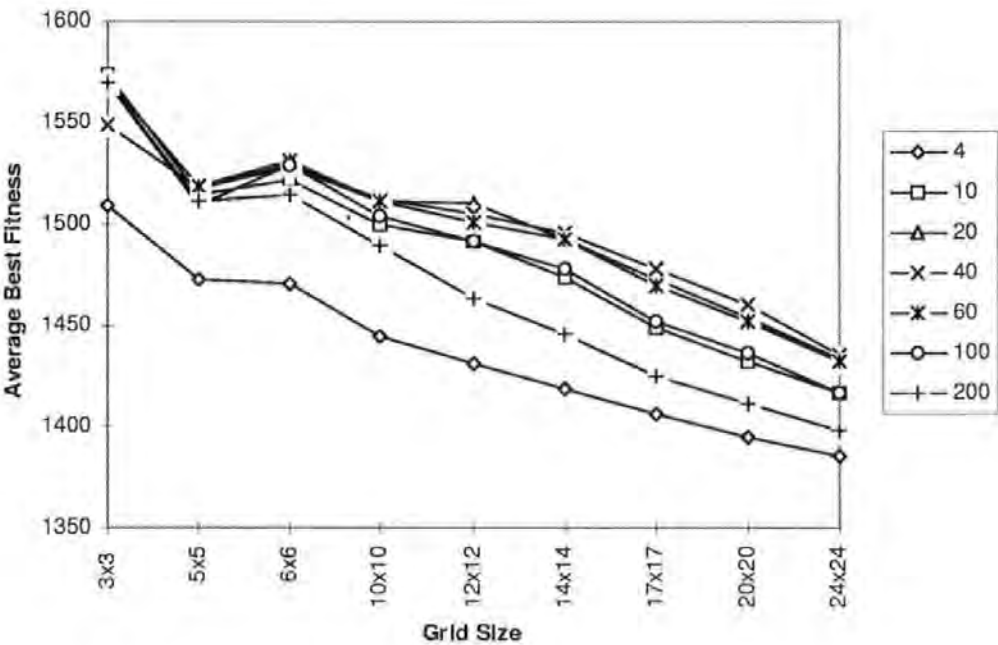


Figure 4-24: Effect of different population sizes on amended PBIL algorithm for a single load case.

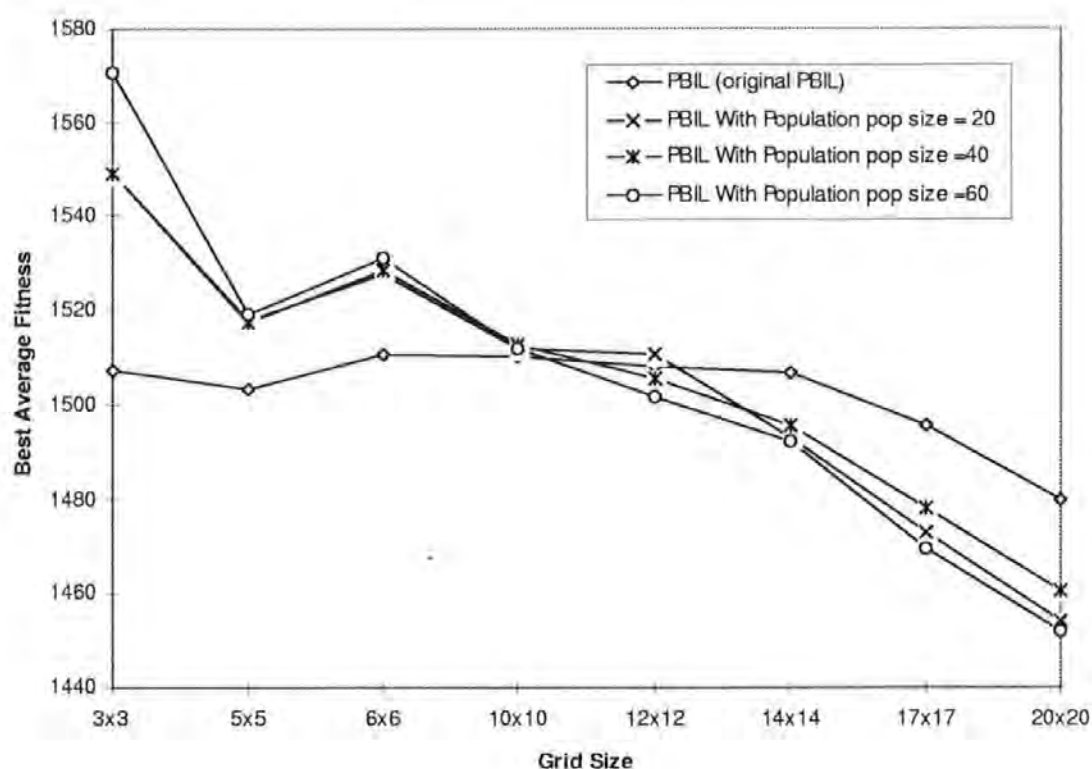


Figure 4-25 : Effect of different population sizes on amended PBIL algorithm for single load case (learning rate =1.0)

On the more complex 3 load case problems where exploration of the search space is of great importance, the modified PBIL fails to provide better solutions when compared to the original PBIL algorithm. In Table 4-11 five learning rates are explored. The more complex of the problems such as the 20x20 and 24x24 3load case plates, with an upper limit of 18 mm show improved results when utilising lower learning. This is due to better exploration of the search space. However altering the learning rates (Table 4-11) or the population sizes (Table 4-12) does not seem to have a major impact upon the algorithm's relative performance.

Learning Rate		(1 load case)		(3 load cases)		(3 load cases)	
		20x20 (24mm)	20x20 (18mm)	20x20 (24mm)	20x20 (18mm)	24x24 (24mm)	24x24 (18mm)
1.0	fitness	1454.046	1462.435	1415.509	1062.597	1393.531	978.8119
	weight	1.099	1.082	1.203	1.086	1.271	1.247
0.8	fitness	1446.96	1456.797	1407.447	1215.169	1388.842	1064.932
	weight	1.119	1.096	1.228	1.23	1.286	1.26
0.6	fitness	1438.05	1450.825	1402.382	1238.287	1384.261	1068.923
	weight	1.141	1.108	1.243	1.244	1.302	1.261
0.4	fitness	1429.837	1444.043	1392.482	1354.494	1377.309	1115.695
	weight	1.164	1.127	1.273	1.249	1.327	1.265
0.1	fitness	1389.811	1420.203	1358.223	905.2703	1351.105	536.3241
	weight	1.284	1.19	1.397	1.263	1.424	1.245

Table 4-11: Amended PBIL (with population) (PBIL1) Population Size = 20

Population Size		20x20 (24mm)	20x20 (18mm)	24x24 (24mm)	24x24 (18mm)
20	fitness	1415.509	1062.597	1393.531	978.8119
	weight	1.203	1.086	1.271	1.247
40	fitness	1420.533	1243.312	1401.1	1005.725
	weight	1.189	1.207	1.246	1.246
60	fitness	1412.136	1132.855	1398.176	857.7043
	weight	1.213	1.223	1.256	1.243
100	fitness	1400.057	981.5387	1383.774	711.7879
	weight	1.249	1.247	1.303	1.241

Table 4-12: Amended PBIL (with population) (PBIL1) (3 load cases) learning rate = 1.0

4.4 Other Techniques

The messy genetic algorithm [Goldberg et. al. 1991] was considered but was not included in the test suite because of the computational expense associated with its two evolutionary phases. Messy Genetic Algorithms (mGA's) use variable-length strings that may be over or under specified with respect to the problem being solved, (this is why they are called messy), and also have two distinct phases. The first Primordial Phase ensures that good

building blocks are enriched through a number of generations of self reproduction without genetic action. Selection alone is run to enrich population with high proportion of the best building blocks. The second Juxtapositional Phase is closer to the canonical GA process in that genetic operators such as mutation are used. However the mGA uses cut and splice operators as opposed to the classical crossover.

A preliminary study in the application of GP to the generation of optimal plate surfaces has been undertaken by Birkenhead [1997]. The method was found to be computationally expensive, requiring in some cases 17 times more evaluations than the CHC algorithm to produce a design comparable in performance. However, this was a short-term preliminary study and further research is required to better assess the GP approach.

4.5 Summary

Many researchers have focused on the comparison of an evolutionary algorithm in relation to a canonical GA (Baluja 1994, Eshelman 1991). This chapter realising the limitations of the canonical GA has compared the performance of different high performance evolutionary algorithms.

It has been shown that EA's are extremely effective at solving the flat plate problem. The use of GA's for solving the plate problem does however involve a large number of calls to the analysis model. This chapter has highlighted that the use of more advanced GA's may help reduce the overall number of calls, and thus make it feasible to integrate complex models such as FEA with an EA.

The single load case promotes the generation of material concentrations in one area of the plate and it is suggested that the highly exploitive characteristics of PBIL is better suited to a less complex distribution of material upon the plate than that required by the three load case problem. With three load cases the material is distributed across a wider area of the plate to best satisfy stress characteristics. Moreover a reduction in plate depth results in more stress violations across the plate resulting in fewer feasible design solutions. It is therefore assumed that the greater diversity of the CHC algorithm results in the better identification of this more complex material distribution. The rapid convergence characteristics of PBIL prevent it from fully exploring the search space, which eventually results in the algorithm premature converging. The performance of the CHC GA is extremely competitive in comparison to the other algorithms. It is extremely robust in the sense that little, if any parameter tuning is required to achieve good results.

The computational demand increases with structural complexity i.e. number of elements, number of load cases. The computational expense to arrive at a feasible solution depends upon a number of factors such as the level of representation of the plate, loading conditions, constraints and the type of optimisation algorithm utilised. Depending on the factors, the search techniques perform in different ways. We need to select a method that yields relatively good results across a broad, spectrum of problem configuration and not limited to one that only provides good results on a particular aspect of the problem through extensive operator tuning. To keep computational expense to a minimum during optimisation (when utilising computationally expensive models) designers often restrict themselves to either optimising coarse representations of the design or sections of a detailed design by focusing on the problem areas so as to reduce the overall number of variables.

The algorithms discussed in this chapter show a degradation in performance with increased numbers of elements on the plate (>100) and multiple load cases. The following chapter discusses ways in which these problems may be overcome.

5. MULTI - LEVEL SEARCH STRATEGIES

Chapter 4 has illustrated a degradation in performance as plate resolution (i.e. number of elements) is increased. In order to solve realistic problems a strategy is required which can handle large numbers of design variables (i.e. >100). This chapter proposes methods which can tackle such problems by utilising co-evolution of multi-representations.

In many optimisation problems there may exist a number of ways in which the problem can be represented. An optimisation algorithm can utilise coarse or fine representations to produce design solutions. A typical example may be a coarse FEA mesh as compared to a refined one for stress analysis. In relation to the plate problem, the coarse representation would provide a preliminary design solution which must be considered high risk due to the low level of accuracy. However such a representation will be relatively inexpensive in computational terms. Conversely a fine representation provides a low risk detailed design solution due to a higher accuracy of plate representation, but also incurs greater computational expense. However a combination of simple (coarse) and complex (fine) representations may lead to a design which is as good as those resulting from a single fine representation, but at a lower computational cost.

Evolutionary design optimisation may involve search utilising different numbers of variables. This presents an opportunity for the development of a strategy that would exploit the differing levels of a problem representation. A strategy that gradually increases in problem dimensionality as the search process progresses would take advantage of this concept. As the plate is a single component it is not reliant on the design of any other

associated assemblies of components, it may therefore be possible to design from preliminary through to detailed design using evolutionary techniques. As coarse representations are computationally less expensive savings can be made in terms of reduced evaluation time. As the solutions for coarse representations evolve more rapidly, these may be used to assist the more refined representations in order to reduce the number of calls to the evaluation function involving the computationally expensive fine representation analysis. To accomplish this the developed technique must successfully progress from a coarse representation to a fine one. The following sections describe three different processes which utilise such multi-level representations for the plate problem. The first termed Dynamic Shape Refinement (DSR) is a sequential technique developed by Vekeria and Parmee [1997]. The other two techniques namely the Modified Injection Island Genetic Algorithm (MiiGA) and the Dynamic Injection Island Genetic Algorithm (DiiGA) also developed by Vekeria and Parmee [1997] involve concurrent processing of models of different resolutions. All three techniques use the CHC GA for the optimisation phase.

5.1 Dynamic Shape Refinement (DSR)

The Dynamic Shape Refinement was developed at the Plymouth Engineering Design Centre by Vekeria and Parmee [Vekeria & Parmee, 1996] and is loosely based on finite element adaptive shape refinement [Kohli & Carey, 1993]. The DSR technique mimics natural evolution in that simple life forms are initially evolved which become increasingly more complex through several generations, the higher life forms displacing the lower. The DSR technique utilises problem representation of varying resolution, starting with a coarse representation which gradually increases in resolution until the desired level of representation is obtained (Figure 5-1). The technique also imitates the process by which a

designer may tackle a design problem. The designer would initially develop a simple (coarse) design, which he or she would gradually refine by adding more details (fine).

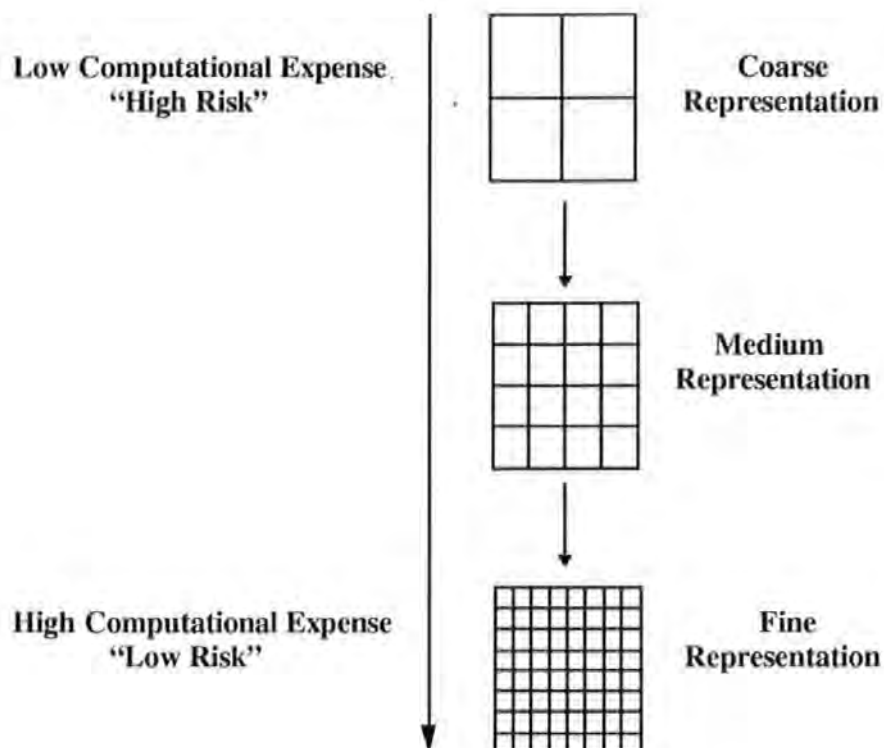


Figure 5-1: An Example of the Dynamic Shape Refinement Technique utilising Three Levels of Representation

The size of the optimisation problem can be varied by increasing (or decreasing) the number of variables as the shape evolves. Generally the initial shape is relatively coarse and high fitness solutions may be obtained within a relatively small number of evaluations. The coarse representation also results in less costly analytic computation. These solutions are however high risk due to lack of resolution associated with the small number of elements. The designer must decide on the level of accuracy required in order to determine the total number of representation levels to utilise. The evolution of a representation ceases once convergence or the maximum permissible number of function evaluations for a particular level is reached. The maximum number of function evaluations may be

representative of that normally required for that level of representation. There are three options as progression is made from the coarse to more fine level representations:

1. Reduce the number of function evaluations as finer resolution levels are computationally more expensive.
2. Increase the number of function evaluations and expend a larger amount of computation of resource in fine tuning the high resolution designs as coarse levels are less accurate.
3. Keep the number of evaluations constant on all levels of representations.

Once search at the coarse level ceases the population is mapped onto a finer more accurate representation and the evolutionary process allowed to continue until the next level of representation is introduced. The mapping of encoding attempts to focus search around “good” solutions that have already been discovered utilising the coarse representation. The final population of the coarse representation becomes the initial population of the next level, which is re-evaluated once it has been mapped into a finer representation. There are however other options other than mapping the whole of the coarse population on to a finer population. The method used is based on the re-initialisation phase of the CHC algorithm. Here the best or a randomly selected individual from the coarse representation is mapped to the next level. The individual is then copied M times (M = population size). Each new individual is created by flipping a fixed proportion (e.g., 35%) of the template’s bits chosen at random. One instance of the best is added unchanged to the new population. Evolution takes place in only one direction from coarse to fine representations.

A simple 3 level representation is presented below:

Representations:

- a) $5 \times 5 = 25$ elements
- b) $10 \times 10 = 100$ elements
- c) $20 \times 20 = 400$ elements

Process:

- Commence evolution of representation a).
- Stop evolution of a) if it has converged or reached maximum permissible number of evaluations.
- Map population a) to produce population b), and continue evolution.
- Stop evolution of b) if it has converged or reached maximum permissible number of evaluations.
- Map population b) to produce population c), and continue evolution.
- Stop evolution if c) converges or reaches maximum permissible number of evaluations.

5.1.1 Mapping of Encoding

When using the DSR technique the issue of mapping a low resolution encoding to a higher resolution must be addressed as there is an increase in the size of the chromosomal representation. The high resolution model must be as close as possible in terms of representation to its more coarse counterpart. This may be achieved relatively easily in the case of the plate problem, so long as there is one to one mapping in one direction (i.e. from coarse to fine representation) the number of elements may be multiplied by two or four. Figure 5-2 shows 2 grid representations (2×2 and 4×4) which illustrate how the mapping

of an individual is accomplished from a 4 to a 16 element representation. In this case the number of elements are multiplied by 4 every time a new representation is introduced. The corresponding depth of a in the 4 element representation is mapped onto $a1, a2, a3$ and $a4$ in the 16 element representation. These then form the new variables, as the next level of representation is optimised. The other components b, c and d are migrated in the same manner in order to form a complete individual.

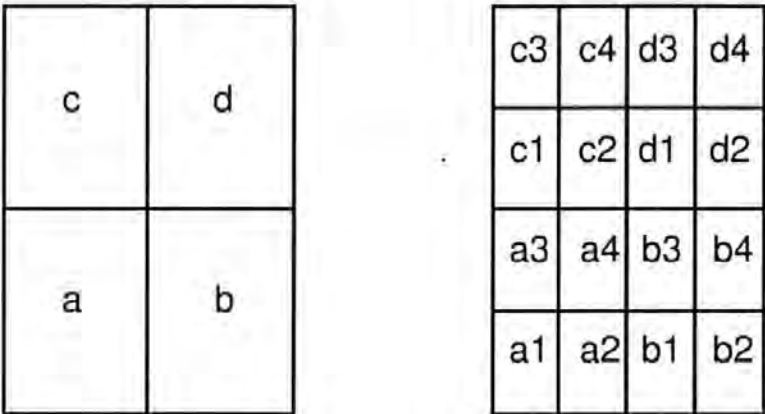


Figure 5-2: The migration of encoding

5.1.2 Discussion and Results for the DSR Technique

The DSR technique allows a major part of the optimisation to be performed during the early coarse levels, with “fine-tuning” being carried out at the finer levels. The solution should not only be of minimum weight within relevant stress criteria but also be considered low-risk in terms of the final resolution of plate representation i.e. there is a sufficiently high number of elements to provide confidence in the stress evaluation.

shows the evolution curve for the CHC, PBIL and DSR CHC processes. Details of the algorithm and plate representation follow:

Stand Alone CHC and DSR CHC:

Population size = 40; (DSR population kept constant on all levels of representations)

Divergence rate = 30%;

Maximum number of restarts = 3

Stand alone CHC plate resolution = 20x20 elements.

DSR CHC plate resolutions = 5x5, 10x10 and 20x20 elements (whole population is mapped during transition from one representation to another)

Stand Alone PBIL

Positive Learning Rate = 1.0

Negative Learning Rate = 1.0

Forgetting Factor = 0.005

Mutation Shift = 0.05

Mutation Probability = 0.02

Trials per Iteration = 40

Number of Vectors to Update from = 1

Stand alone PBIL plate resolution = 20x20 elements

Total Number of Calls to the Model = 10000 (unless otherwise stated)

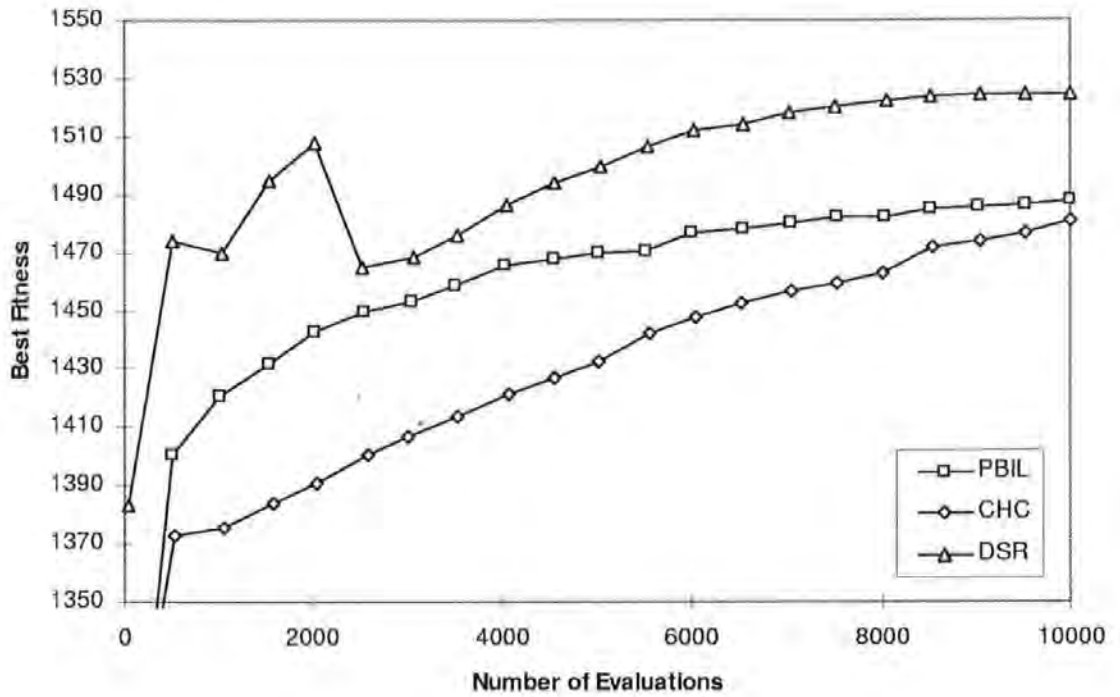


Figure 5-3: Performance of stand-alone CHC, CHC DSR and PBIL for 20x20 plate (1 load cases). Max = 24mm.

The evolution curves are shown for the fittest individual found from the 10 runs. The high fitness achieved during the early stages of the DSR CHC approach must be treated with caution because fitness is measured in terms of weight versus stress violation and the coarser representations although seemingly of high fitness are also high-risk due to the lack of resolution during stress evaluation. A higher resolution stress evaluation returns a greater degree of violation and a related degradation of fitness as shown by the dips in the DSR curve as finer resolutions are introduced. The first dip in the graph represents the transition from a 5x5 representation to a 10x10 one. The second represents the transition from 10x10 to a 20x20 representation. It can be seen that as a result of the highly fit genetic material from the 10x10 representation seeding the 20x20 representation fitness achieved from the DSR technique quickly exceeds that of the single population GA.

In comparison the CHC and PBIL single population representations (20x20 plate) show a constantly improving fitness throughout the evolutionary process. Comparison of the 20x20 representations of the DSR (Table 5-1) approach with those of PBIL and CHC (Chapter 4) show the DSR achieving a higher fitness than the single representation approach with far fewer calls to the analysis routine. Therefore to some extent the DSR satisfies the primary objectives of the research i.e. feasible design solution with minimum weight and number of calls to the fitness function.

DSR								
Test Number	Plate size	Max upper limit	Load cases	Best Fitness	Best Weight	Fitness (SD)	Average Fitness	Average Weight
DSR_1	20X20	24	1	1524.37	0.95	14.11	1496.92	1.01
DSR_2	20X20	18	1	1477.41	1.05	6.70	1466.10	1.07
DSR_3	24X24	24	1	1506.58	0.99	12.22	1479.45	1.04
DSR_4	24X24	18	1	1497.95	1.00	6.49	1487.37	1.03
DSR_5	20X20	24	3	1452.05	1.11	7.60	1444.03	1.13
DSR_6	20X20	18	3	1446.51	1.12	74.82	1416.48	1.14
DSR_7	24X24	24	3	1449.99	1.11	8.60	1439.10	1.14
DSR_8	24X24	18	3	1453.41	1.10	5.02	1446.47	1.12

Table 5-1: Results for the DSR technique utilising various problem cases (no. of runs = 10)

Figure 5-4 shows the evolution curve of the DSR process utilising the CHC algorithm.

Details of the algorithm and plate representation follow:

Stand Alone CHC and DSR CHC:

Population size = 40; (DSR population kept constant on all levels of representations)

Divergence rate = 30%; Maximum number of restarts = 3

DSR plate resolutions = 6x6, 12x12 and 24x24 elements

Stand alone CHC plate resolution = 24x24 elements.

A high fitness is once again achieved during the early stages of the DSR approach. The first dip in the graph represents the transition from a 6x6 representation to a 12x12 one. The second represents the transition from 12x12 to a 24x24 representation. It can be seen that as a result of the highly fit individuals from the 12x12 representation the fitness of the DSR technique quickly exceeds that of the single population evolutionary methods.

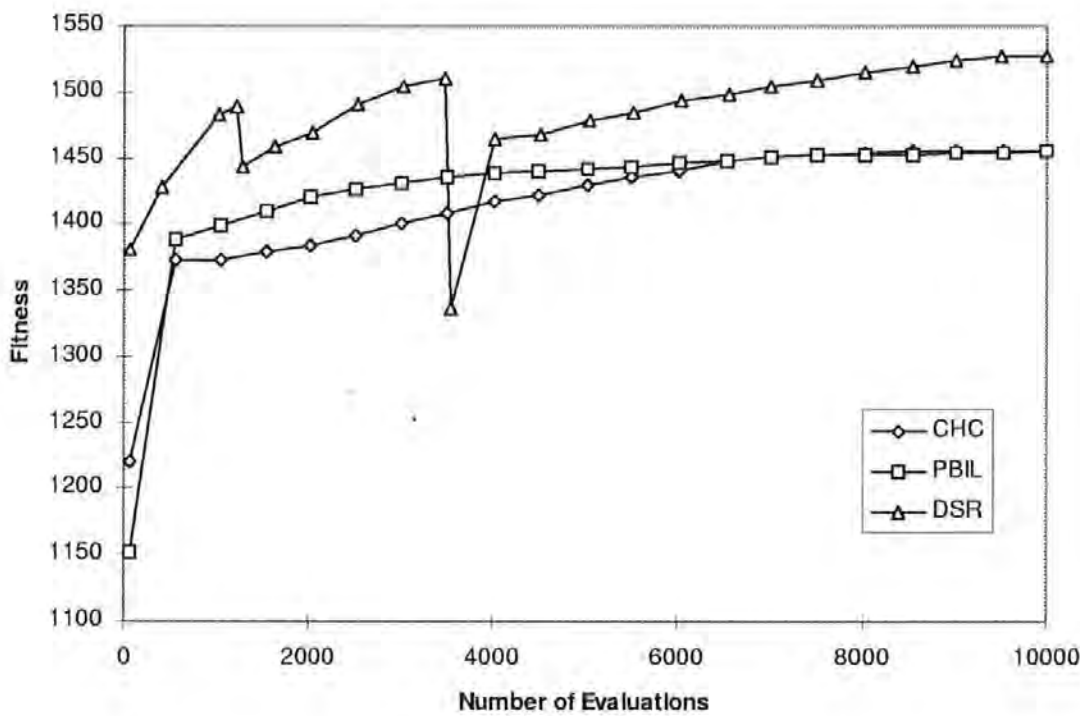


Figure 5-4: Performance of stand-alone CHC, DSR and PBIL for 24x24 plate (1 load case).
max 24mm

Table 5-1 shows the results for the DSR technique. It shows the final best fitnesses utilising 400 elements and a single load case problem. These results are comparable to those achieved with the single population CHC and PBIL methods shown in chapter 4, but they are however achieved with less computation due to the use of coarser representations

(Figure 5-13 and Figure 5-7). On the finer 576 element representation (24x24 plate) the final best fitnesses are far superior to those achieved using the CHC and PBIL methods. This indicates that the DSR technique achieves a better compromise between exploration and exploitation of the search space on single load case problems of high dimension.

On the three load case problems the results again show comparable final best fitnesses to those achieved by CHC and PBIL. As in the single load case problems they are achieved at less computational cost through the use of less costly coarse representations. Unlike the CHC and PBIL methods, the standard deviations shown in Table 5-1 indicates that the DSR technique also provides more robust solutions to the problem involving three load cases, 24x24 plate and upper material limit of 18mm.

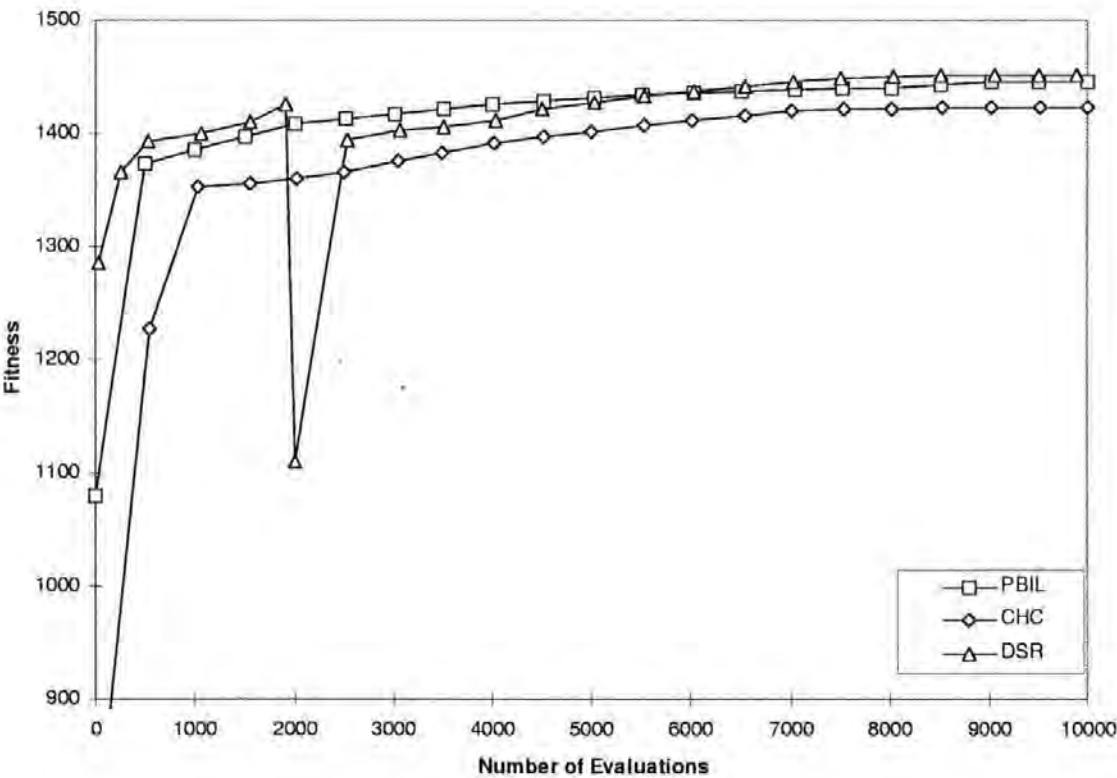


Figure 5-5: Performance of stand-alone CHC, DSR and PBIL for 20x20 plate (3 load cases) max 24mm

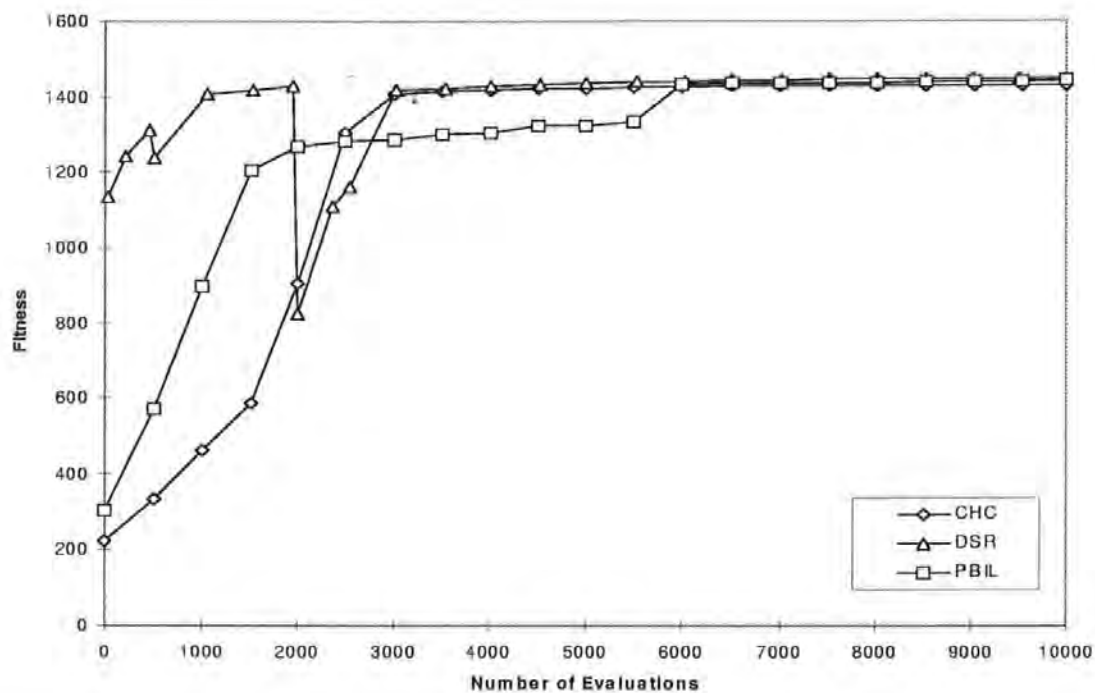


Figure 5-6: Performance of stand-alone CHC and DSR for 20x20 plate (3 load cases) max 18mm.

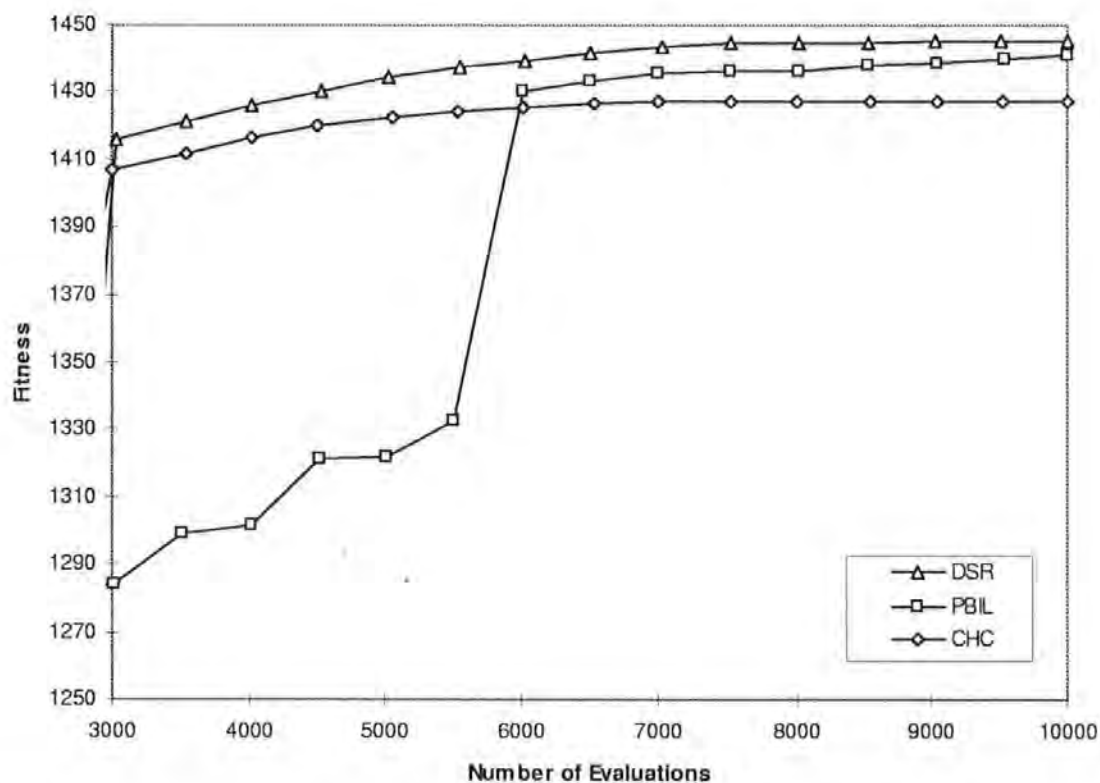


Figure 5-7: Performance of stand-alone CHC and DSR for 20x20 plate (3 load cases) max 18mm during Latter stages of search

In summarising the results it can be said that the DSR technique presents a novel way of dealing with a large numbers of variables and reducing the number of calls to the model during a GA run. By utilising a combination of simple and complex representations the DSR technique on most of the problems described leads to a design which is as good as those resulting from those of a single fine representation, but at lower computational cost through the utilisation of coarse representations.

5.2 Parallel Genetic Algorithms

The task of finding an optimal solution for a complex structural analysis problem poses a considerable challenge to the engineer, not only because of high dimensionality but also because of the high computational expensive. There are two ways in which to lessen the computational expense. Firstly to accelerate convergence of individual algorithms whilst minimising the overall number of evaluation calls and thus CPU time; secondly the distribution of the problem through the utilisation of parallel architectures. The research described here represents a combination of these two approaches.

This section presents a method by which multi-level representations are used in a parallel manner in an attempt to make further improvements in performance. The Injection Island GA (iiGA) [Goodman et. al., 1997] is a technique that maintains multiple interacting sub-populations of different resolution. However, unlike the DSR technique which is sequential in nature, several levels of representations evolve at the same time, with occasional migration from one sub-population to another. The technique presented not only reduces the calls to a model but is also very accessible to the parallel method.

Parallel Genetic Algorithms (PGA) address the convergence problem of single population GA's by subdividing the populations and evolving the sub-populations independently so they are more likely to explore different portions of the search space. The main motivations to use PGA's are :

- to increase speed and efficiency
- to allow the application of the GA to a larger problem
- to try to follow biological metaphor more closely

Various researchers have utilised PGAs for complex structural design problems. For instance Leite (1996) applied many parallel models and environments to the design of a cable-stayed bridge. The studies show that, especially for large engineering problems, the parallel GA performs better than serial algorithms both in execution speed and quality of solution. Doorly et al [1996] utilised parallel genetic algorithms to reduce computational expense for optimisation in computational fluid dynamics (CFD) for the design of optimal airfoils. Poloni et al [1996] utilise parallel GA's for aerodynamic design optimisation problems. A massively parallel Cray computer is utilised to reduce the computational effort required for the accurate evaluation of a design configuration. Goodman et. al. uses injection island genetic algorithms (iiGA) for the design of composite cantilever plates [1996] and to optimise the Specific Energy Density (SED) of elastic flywheels [1997]. The iiGA searches at various levels of resolution in parallel within a given space. Adeli and Cheng [1995] use the parallel GA for the optimisation of high rise building structures and space stations with several hundred members.

There are several types of Parallel GA's which differ in the nature of the population structure and / or the method of selection.

5.2.1 Micro-grain GA (mgGA)

Micro-grained GA's [Punch et. al. 1993] maintain a single population with multiple processors being used to run the evaluation function. No migration is employed. Every processor that is used (up to the number of members in the population), results in an increase in performance. If fewer processors than the number of members of the population are available, then each processor is responsible for processing a subset of the population, making the populations evaluation time equivalent to the evaluation time of the most costly subset. Genetic operations such as crossover and mutation are typically conducted sequentially by a single "master" node which controls the system. The Micro-grained GA is especially useful when the evaluation function is computationally expensive as in the case of FEA and CFD packages which may take in the order of several minutes for a single evaluation. The mgGA's do not address the problem of premature convergence, their primary goal is speed in comparison to sequential GA's.

5.2.2 Fine-Grain GA's (fgGA's)

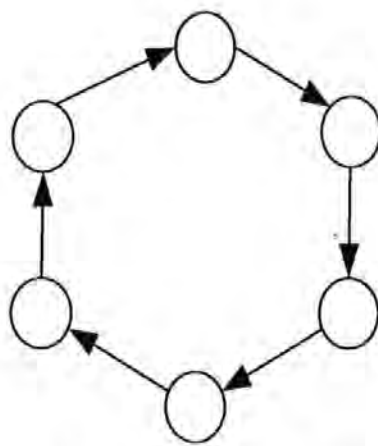
Fine Grained GA's (fgGA's) [Manderick and Spiessens, 1989] are sometimes also termed massively parallel GA's. A large population is divided into a series of smaller sub populations by placing one individual at each location on a toroidal 2-dimensional grid. With each individual assigned this way, the grid locations are not necessarily related to the individual's solutions, rather they are arbitrary designations used to perform selection.

Sub populations are defined in terms of neighbourhood on the grid. One method is to utilise a fixed size neighbourhood where for any given location (individual) a sub population would be that location plus its eight immediate neighbours. With this method

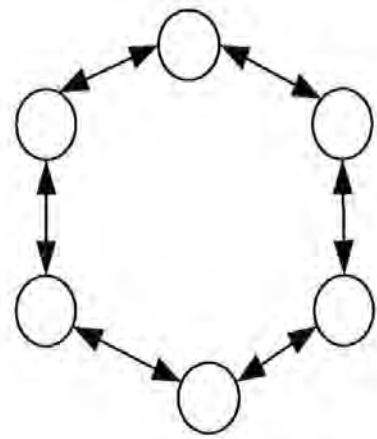
there is a natural local mating scheme within each sub population. The entire population may be viewed as numerous small sub-populations which overlap. This results in a mixing of individuals between the sub-populations. Sub populations that are within a close locality (relative to grid size) will exert more of an influence on each other than those a greater distance apart, whilst the more distant sub populations should evolve comparatively unique chromosomes. High connectivity between neighbours increases the spread of high fitness individuals, making sub-populations susceptible to domination and perhaps premature convergence.

5.2.3 The Distributed Genetic Algorithm (DGA)

Tenese [1989] proposed the distributed genetic algorithm (DGA) also termed coarse grained GA as a way of efficiently parallelising the canonical genetic algorithm (CGA). In the DGA, the global population is divided into several sub-populations, one per processor (Figure 5-8). Inter-processor communication occurs during the migration phase at regular intervals (i.e. migration interval). During migration, a fixed proportion of each sub-population is selected and sent to another sub-population. In return, the same number of migrants are received from some other sub-population and replace individuals according to some criteria. This migration can occur either asynchronously or synchronously. Because the time-consuming measurement of fitness is performed independently at each separate processing node, this approach to parallelisation delivers an overall increase in performance that is nearly linear with the number of independent processing nodes.



Ring Migration Topology



Neighbourhood Migration Topology

Figure 5-8 : Two examples of PGA topologies

5.2.4 Cooperative Co-evolutionary Optimisation

The use of multiple interacting sub-populations has been widely explored as an alternate mechanism for co-evolving niches using the distributed GA. Co-operative co-evolutionary algorithms (CCA) [Potter and De Jong, 1994] combine and extend these ideas in several ways. A CCA consists of a collection of independent sub-populations, each attempting to evolve sub-components (species) which are useful as modules for achieving more complex structures. The CCA evolves each species (function variable) in a round robin fashion using the current best values from the other species. This is quite similar in style to numerical optimisation techniques which proceed by optimising one function variable at a time while holding the other variables constant. Unlike the island model, the individuals from the separate sub-populations do not interbreed. Complete solutions are obtained by assembling representatives from each of the species present. Credit assignment at the species level is defined in terms of the fitness of the complete solution in which the members participate. This provides evolutionary pressure for species to co-operate rather than compete. However competition still exists among individuals within the same sub-

population. Such a procedure works well on problems whose variables are reasonably independent, but difficulties arise with problems such as that of the plate which has high interacting variables i.e. slight perturbation of a single variable may have an effect on the overall fitness of the design.

Barbosa [1997] proposes a co-evolutionary GA for solving structural optimisation problems. Two GA's are run independently. A GA evolves for a certain number of generations on population *A* while population *B* is kept frozen. The GA is then allowed to operate on population *B* while population *A* is kept frozen. The cycle is repeated *n* number of times. The fitness is based on function $f(x,y)$ where *x* is taken from population *A* and *y* from population *B*. As a result the fitness of each individual in one population depends on all individuals of the other population.

5.3 The Injection Island GA (iiGA)

As highlighted in section 5.2 parallel processing is often used to increase the speed of convergence. However, before introducing parallel architectures it is extremely important to develop and optimise the underlying adaptive algorithms with respect to their efficiency, effectiveness and overall robustness.

The injection island architecture (iiGA) [Goodman et. al., 1996] offers a concurrent rather than a sequential shape refinement process. The iiGA is an extension of the coarse grained PGA, whereby lower resolution representations are explored on some islands, which inject approximate solutions into higher resolution populations for further refinement. The iiGA is therefore characterised by: (1) sub-populations using different data representations and (2) exchange of genetic material one way. To illustrate the technique, an example of an

iiGA topology is shown in Figure 5-9. In this figure each circle represents a separate sub-population or “island”. Sub-population at islands 1 and 2 use a low resolution representation for problem solutions. Best results from islands 1 and 2 are migrated to islands 3 and 4 respectively at a set number of evaluations. Sub-populations at islands 3 and 4 employ a medium resolution representation, and individuals injected into these sub-populations from islands 1 and 2 are expanded as appropriate to the new higher resolution representation. Similarly, individuals evolved on islands 3 and 4 are injected into the sub-populations at islands 5 and 6 respectively, and expanded into this higher resolution representation. Island 5,6,7,8,9,10 form the basis of the more traditional island GA, exchanging individuals at the same, highest resolution at a set number of evaluations.

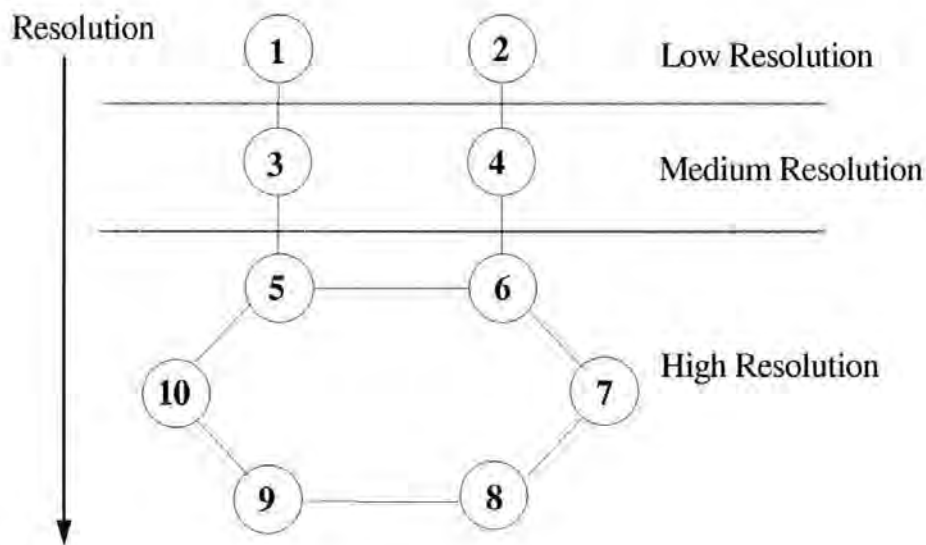


Figure 5-9 : Michigan Injection Island Topology

Goodman et. al., [1997] utilised the iiGA in a distributed environment where numerous sub-populations of different representations exist each on individual processors. The objective is a reduction in overall run time whilst maintaining diversity. However, instead of reducing overall time of the run through the use of multi-processors an alternative architecture is introduced here that borrows from both DSR and Michigan’s iiGA.

The iiGA as with the DSR technique offers a method of reducing calls to the model by utilising various levels of representations. However iiGA's concurrent evolution of the different levels of representations, ensures that feasible design solutions are available relatively early in the search process thereby aiding a further reduction in run times.

The architectures described differ to those used by Goodman et. al., [1997]. Firstly the sub-populations are not distributed, they are contained as subsets of each population. The flat plate is represented by a number of different resolution grids each resolution being allocated to a population subset. Members of each sub-population are evaluated one at a time. Secondly relatively few sub-populations are utilised in comparison to typical implementations of Michigan's iiGA. The overall number of sub-populations have been reduced in order to decrease the number of calls to the model. Each level of plate representation is usually represented by one sub-population at the start unlike Michigan's iiGA, which may have several sub-populations for a given level.

Due to these differences and to avoid confusion the technique will be referred to as the Modified Injection Island GA (MiiGA). The objective is to establish co-evolutionary, multi-level representation processes with appropriate migration regimes that support the design of single components from preliminary through to detailed design.

5.3.1 Application of MiiGA's on the Plate Problem

This and following sections focus on the use of multiple representations as a method for maintaining genetic diversity and reducing the number of calls made to an evaluation function.

When implementing an MiiGA the designer must address certain issues such as :

- The number of individuals to migrate
- Which individuals to migrate (the best or arbitrary ones)
- Which individuals to replace (the worst or arbitrary ones)
- Synchronous or Asynchronous migration of individuals
- Exchange between neighbours or between arbitrary subpopulations.

A 3 level MiiGA representation is presented below:

Representations: a) $5 \times 5 = 25$ elements b) $10 \times 10 = 100$ elements c) $20 \times 20 = 400$ elements

Process:

- Commence co-evolution of representations a) and b) and c).
- Migrate fit individuals from a) to b), a) to c) and b) to c) every n evaluations
- Continue the process until maximum number of restarts or maximum number of evaluations has been reached

The solutions of the coarse design representations are injected into the more detailed designs for fine grained refinement. The coarse migrated individual must therefore be converted to the required level of representation before migration. This is accomplished in the same manner as outlined in section 5.1.1. Migration of information is from low to high resolution at a set number of evaluations. Figure 5-10 shows the co-evolution of 3 representations 3×3 , 6×6 and a 12×12 . Individuals are migrated from 3×3 to 6×6 , 3×3 to 12×12 and 6×6 to 12×12 at a pre-set number of evaluations (this is normally 100 on most problems unless stated otherwise). Migration allows the passing of highly fit schemata by injecting the best individuals that have evolved from a proportionally smaller search space into higher resolution representations replacing the worst individuals present at that time. There are a large number of possibilities when deciding the migration method to utilise. The research does not discuss the best migration methods to use.

Subpopulation 1 Subpopulation 2 Subpopulation 3

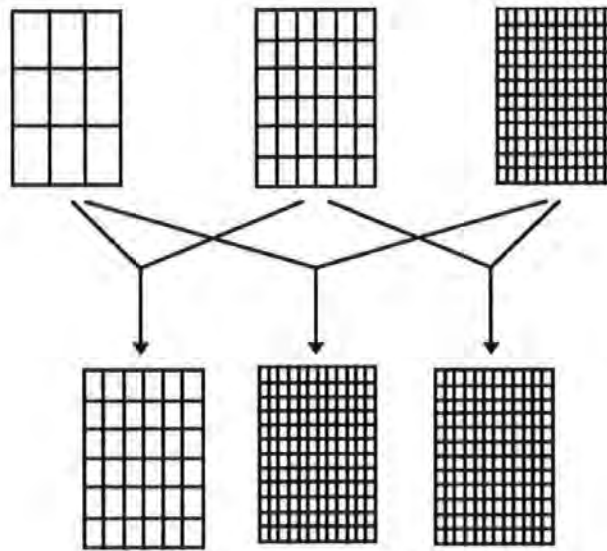


Figure 5-10: Migration between subpopulations

The fittest individual from the coarse representation is selected for migration into the finer representations, replacing the least fit in those subpopulations. Each subpopulation passes the fittest individual to all higher resolution subpopulations. However other strategies may be used such as :

- Every subpopulation passing individuals to only the finest subpopulation
- Every subpopulation passing individuals only to its neighbour
- Every subpopulation passing individuals to both its neighbour and the subpopulation containing the highest resolution.

MiiGA								
Test Number	Plate size	Max upper limit	Load cases	Best Fitness	Best Weight	Fitness (SD)	Average Fitness	Average Weight
MiiGA_1	20X20	24	1	1454.30	1.1	11.02	1438.36	1.14
MiiGA_2	20X20	18	1	1471.09	1.06	8.06	1459.78	1.09
MiiGA_3	24X24	24	1	1451.31	1.11	11.43	1421.04	1.19
MiiGA_4	24X24	18	1	1459.28	1.09	9.87	1448.44	1.12
MiiGA_5	20X20	24	3	1443.89	1.13	18.01	1417.03	1.20
MiiGA_6	20X20	18	3	1451.24	1.11	190.11	1374.58	1.16
MiiGA_7	24X24	24	3	1387.80	1.29	4.92	1380.08	1.32
MiiGA_8	24X24	18	3	1295.28	1.15	265.37	848.35	1.21

Table 5-2 : Results for MiiGA utilising various problem cases (no. of runs = 10)

Figure 5-11 and Figure 5-12 illustrates the effect of CHC integration with the MiiGA architecture in terms of the best fitness of a stand-alone CHC GA of 40 chromosomes (400 elements) and an MiiGA using 3 sub-population islands (consisting of 25, 100 and 400 elements) of 20 chromosomes each. The curve displayed for the MiiGA represents the fittest individual found from the 10 runs in the finest sub-population (400 elements) utilising a single load case. The number of evaluations is the summation of all evaluations of the sub-populations. Rapid progress is apparent when compared with the single population CHC GA. This is due to the injection of high performance individuals from the coarse representations. Rapid progress is also apparent in comparison to the single population CHC GA when using 3 sub-population islands on a single load case 24x24 variable plate problem (Figure 4-13).

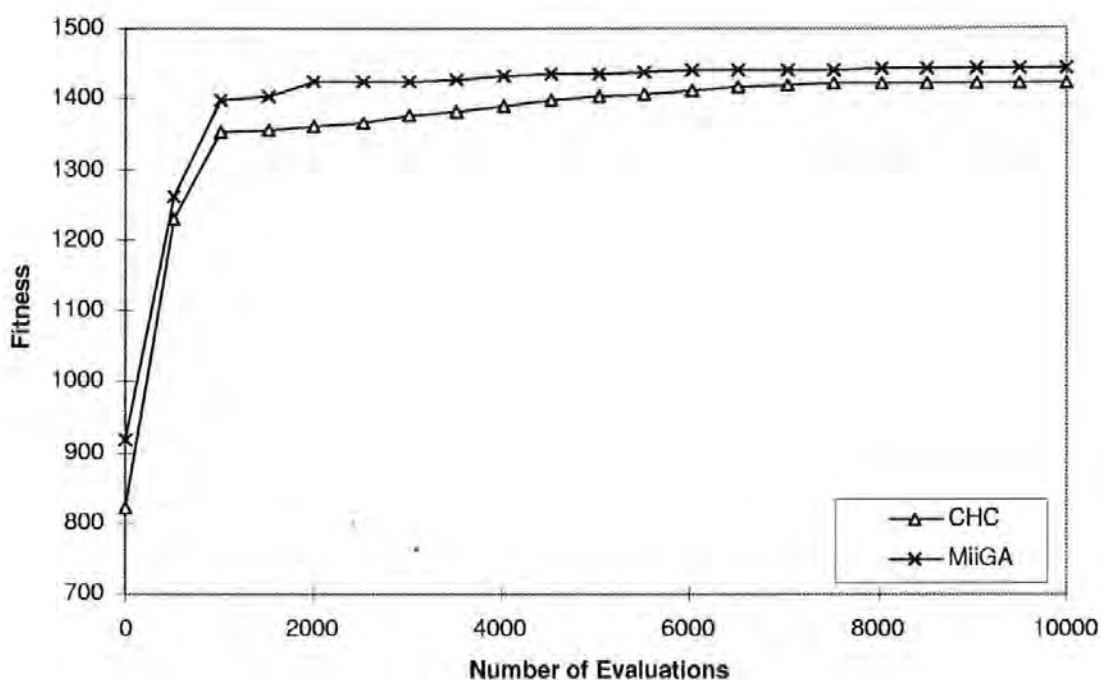


Figure 5-11: Performance of stand-alone CHC GA and MiiGA CHC (20 x20 1 load case)
max 24mm

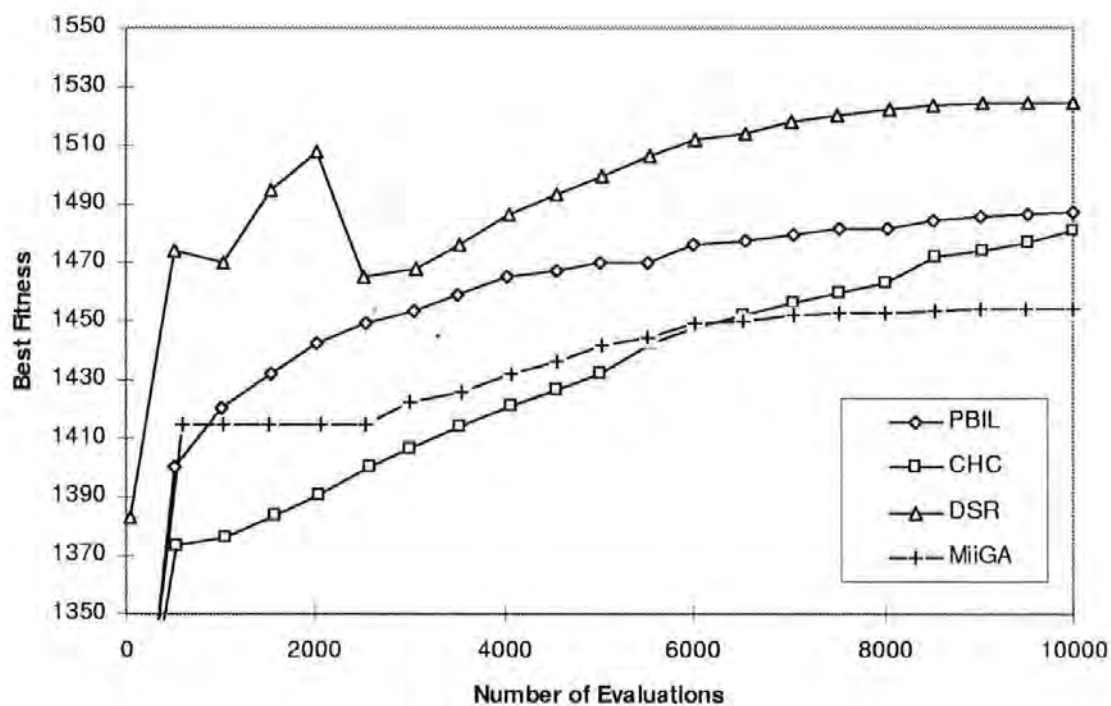


Figure 5-12: Performance of stand-alone CHC GA and MiiGA CHC (20 x20 1 load case)
max 24mm

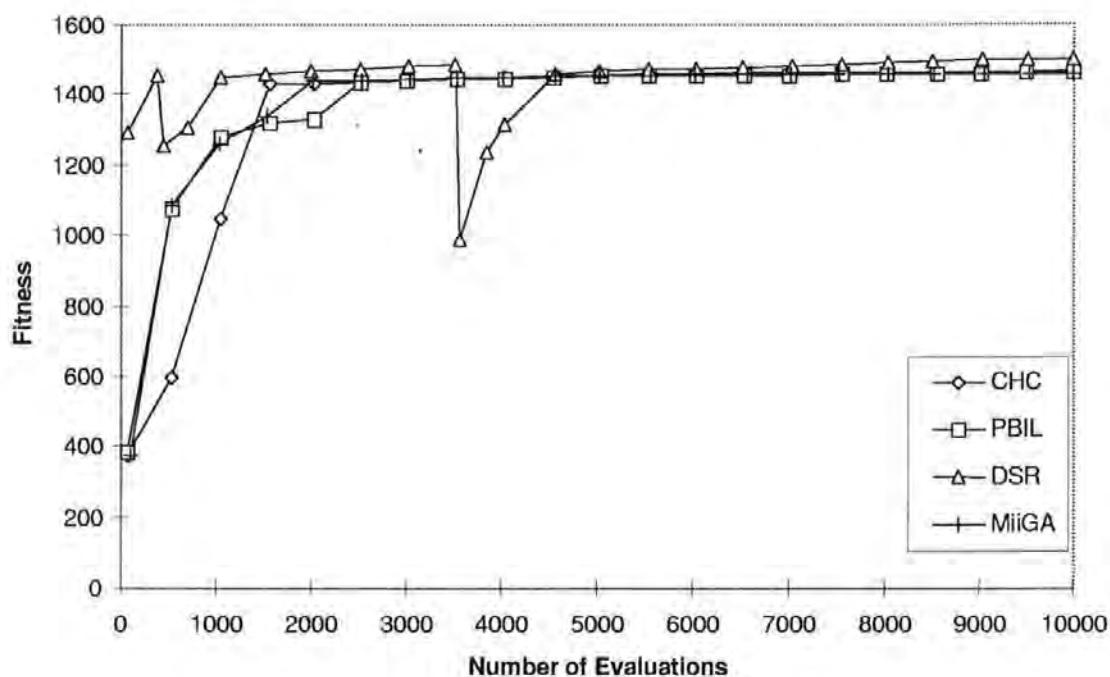


Figure 5-13: Performance of stand-alone CHC GA and MiiGA CHC (24 x24 1 load case)
max 18mm

On the more complex problem utilising three load cases (20x20 plate with an upper limit of 24mm on material), the MiiGA fails to produce final design solutions as good as those resulting from the single population methods or the DSR technique (Figure 5-14). However a more rapid progress is apparent in comparison to the single load case problem. The curve displayed for the MiiGA represents the fittest individual found from the 10 runs in the finest sub-population (400 elements).

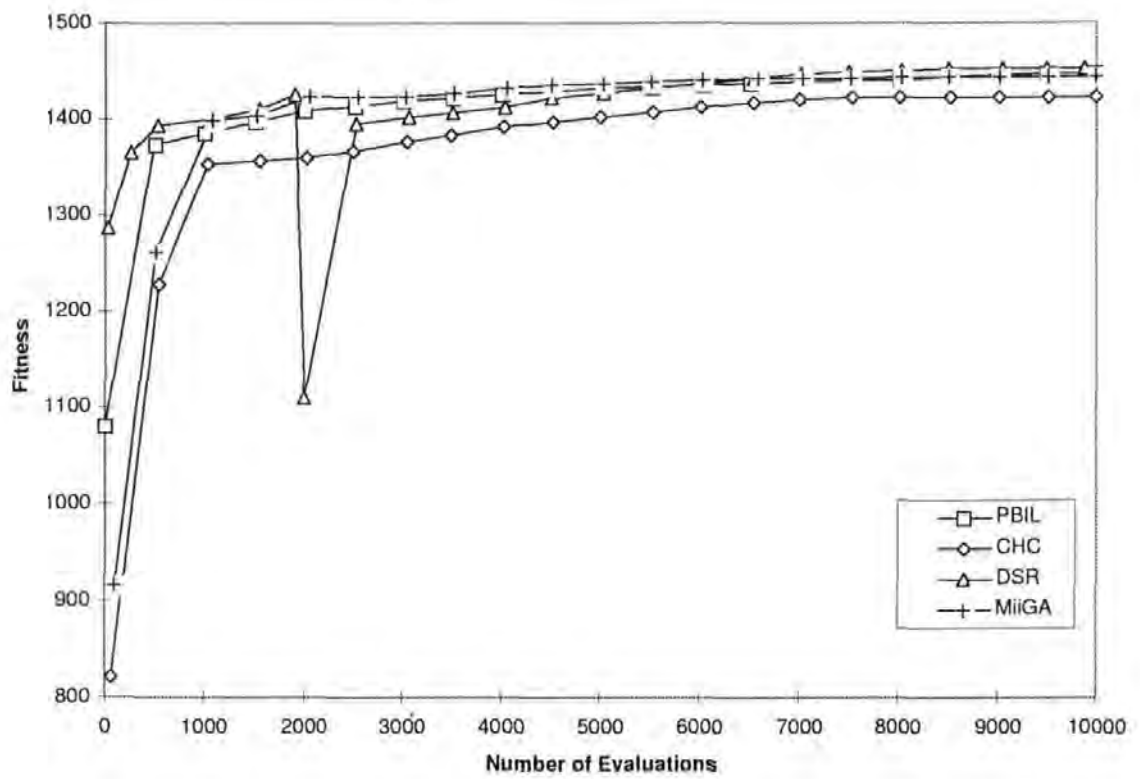


Figure 5-14: Performance of stand-alone CHC GA and MiiGA CHC (20 x20 3 load cases, max. upper limit = 24mm)

On the most complex problem utilising three load cases with an upper limit of 18mm on material , the MiiGA fails to produce feasible design solutions. Figure 5-15 illustrates the MiiGA consisting of 6x6, 12x12 and 24x24 grids of 20 chromosomes each. The curve displayed for the MiiGA represents the finest sub-population (576 elements). Rapid progress is again apparent when compared with the single population CHC GA. The single representation CHC approach does however eventually outperform the MiiGA in terms of maximum fitness. This is due to the fast convergence of the lower resolution MiiGA representations limiting the injection of useful material into the higher resolution populations which eventually results in a stagnation of the co-evolutionary process. Due to this reason the average best solution after 10,000 evaluations is not as good as those from the single population CHC, PBIL or DSR techniques. The initial motivation for

implementing the described architecture was that coarse representation may still have something to contribute during the latter stages of search process.

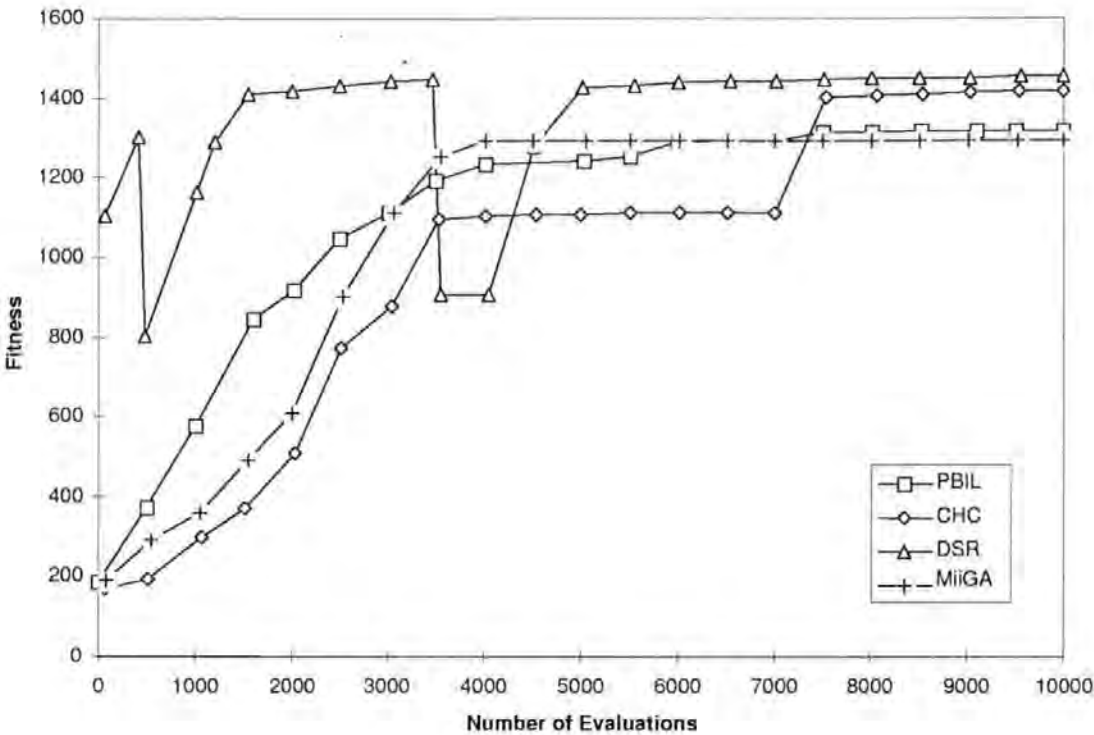


Figure 5-15: Performance of stand-alone CHC GA and MiiGA CHC (24 x24 3 load cases, max upper limit =18 mm)

In summary the MiiGA manages to locate feasible design solutions for a single load case problem, producing a final fitness which is not as good as those produced by the CHC, PBIL or DSR methods. On the more complex problems utilising multi-load cases the final design solutions are again not as good as the CHC, PBIL or DSR techniques due to stagnation of the co-evolutionary process. This is due to the coarse representations continued evolution, even though they have little to contribute during the latter stages of the design process. The advantage however in using the MiiGA is that feasible design solutions especially on simpler problems evolve faster due to concurrent evolution and are

therefore available earlier in the design process, thus saving computational effort and calendar time.

5.4 Dynamic Injection Island GA (DiiGA)

In order to address the stagnation problem a dynamic aspect has been introduced to the MiiGA paradigm by Vekeria and Parmee. The Dynamic Injection Island Genetic Algorithm (DiiGA) method of representation addresses the problem of stagnation as discussed in the previous section. The technique is a combination of the DSR and iiGA techniques. Two or more levels of representations evolve. Previous results have shown that the DSR technique is highly effective at reducing the overall computational effort through the utilisation of several levels of representations and by the phasing out of lower representations. The MiiGA due to its concurrent evolution of the different levels of representations ensures that, unlike the DSR technique, a feasible design solution can be generated relatively early in the search process. The DiiGA is a strategy which takes advantage of the better mechanisms of the two approaches (Figure 5-16). As a lower resolution process ceases to inject useful information into the higher resolution processes so it is removed and replaced by a resolution that is higher than any currently in existence. The new higher representation is seeded from the new lower sub-population. This is accomplished by mapping the fittest individual from the lower sub-population to the higher sub-population. The remainder of individuals in the new higher sub-population are formed by copying the mapped individual and mutating a fixed percentage (30%) of it's bits at random. This creates a sub-population which is biased towards a good solution from the lower level but with new diversity. The desired behaviour is one of constant improvement in fitness, avoiding the levelling of the MiiGA curve as displayed in Figure 5-15. A simple 3 level representation involving 4 processes is presented:

Representations:

a) $5 \times 5 = 25$ elements

b) $10 \times 10 = 100$ elements

c1) $20 \times 20 = 400$ elements

c2) $20 \times 20 = 400$ elements

Process:

- Commence co-evolution of representations a) and b).
- Migrate from a) to b) every n evaluations until a) converges and ceases to pass useful information to b) or until maximum permissible evaluations
- Remove a) and introduce c_1) using the best individual from b) to seed new population
- Migrate individuals from b) to c_1) every n evaluations until b) converges and ceases to pass useful information to c_1) or until maximum permissible evaluations
- Remove b) continue to evolve c_1). Introduce another co-evolving subpopulation (c_2), seeded from (c_1).

Individuals are prevented from migrating if a duplicate exists in the host subpopulation in order to maintain search diversity. Further migration only takes place if the individual is fitter than the least fittest individual in the host sub-population. The run continues until its termination condition is met (when the sub-populations have converged, the maximum number of evaluations have been reached or the maximum number of re-initialisations has been achieved). It should be noted that there is no danger that the best individual will rapidly take over the new sub-population. The CHC GA's incest preventing mechanism (the dropping difference threshold), in combination with elitist selection and disruptive recombination will prevent this. Eshelman [1991] found that partial re-initialisations

perform better using smaller population sizes when compared with chronic mutation and provide many of the benefits of a large population without the cost of a slower search. The number of evaluations is the summation of all evaluations of the sub-population 1 and sub-population 2. Migration takes place every 100 evaluations.

The following sections show the performance of the DiiGA, firstly using the complex stress model (section 5.4.1) and then the finite element model (section 5.4.2).

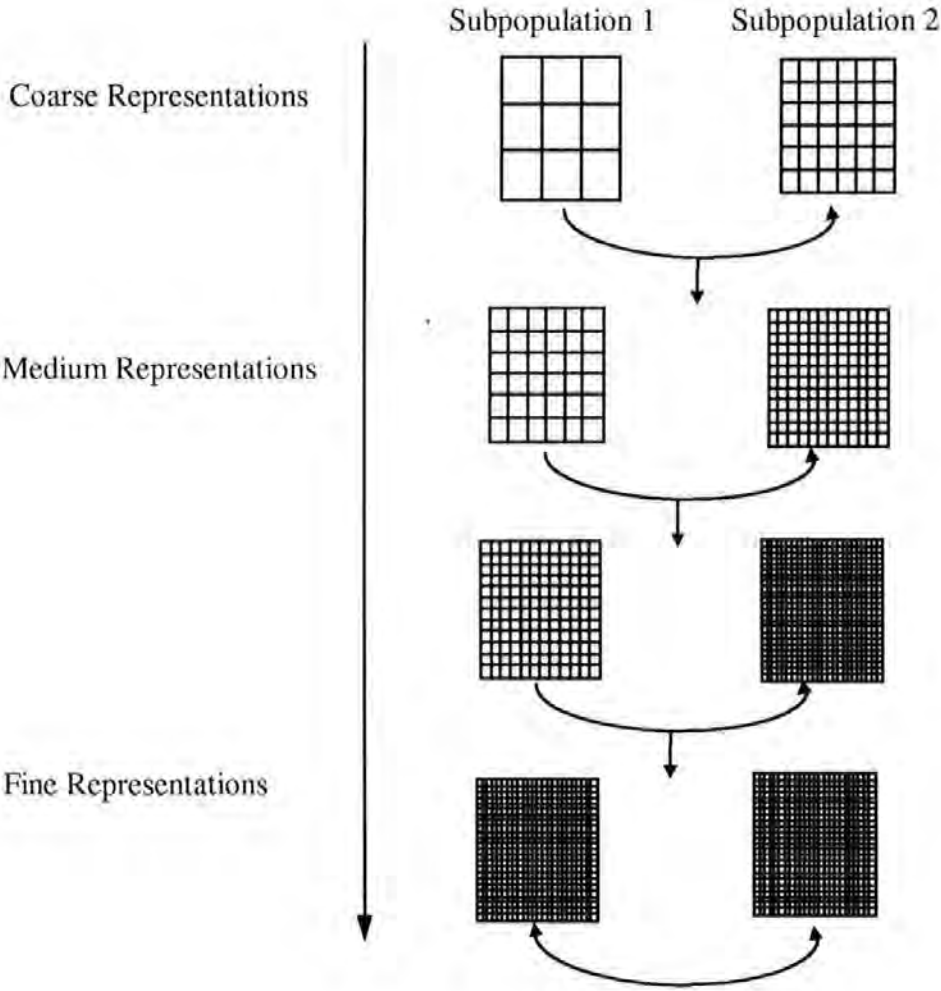


Figure 5-16 : Grid Representation for the DiiGA

5.4.1 Application of Dynamic Injection Island GA to the Plate Problem utilising the Complex Stress Model

On the single load case problem with a maximum limit of 24mm on the plate, there are relatively small differences in the performance of the CHC and PBIL presented in chapter 4 in comparison to the DiiGA (Table 5-3). However on the more difficult single load case problems where the upper limit on material is 18mm, the DiiGA performs better, through the maintenance of better diversity.

DiiGA								
Test Number	Plate size	Max upper limit	Load cases	Best Fitness	Best Weight	Fitness (SD)	Average Fitness	Average Weight
DiiGA_1	20X20	24	1	1468.80	1.07	11.95	1450.97	1.11
DiiGA_2	20X20	18	1	1480.49	1.04	8.59	1468.73	1.07
DiiGA_3	24X24	24	1	1461.84	1.08	9.03	1448.99	1.11
DiiGA_4	24X24	18	1	1474.68	1.05	5.50	1467.94	1.07
DiiGA_5	20X20	24	3	1430.88	1.16	12.14	1410.50	1.22
DiiGA_6	20X20	18	3	1428.64	1.17	67.08	1396.80	1.20
DiiGA_7	24X24	24	3	1409.82	1.22	14.33	1388.94	1.29
DiiGA_8	24X24	18	3	1424.22	1.18	5.37	1416.71	1.20

Table 5-3: Results for DiiGA utilising various problem cases (no. of runs = 10)

Figure 5-17 and Figure 5-18 represents the fitness of sub-population 1 of the 10 runs for the finest resolution grid (400 elements). Three load cases are utilised. The first dip shows the transition from a 5x5 to a 10x10 representation. The second dip shows the transition from a 10x10 to a 20x20 representation. Rapid evolution is apparent in the graphs resulting from highly fit genetic material from the more coarse representations boot strapping the overall fitness. Continuous improvement is maintained, resulting in superior performance in terms of degree of stress violation and weight in comparison with the 3 sub-population MiiGA.

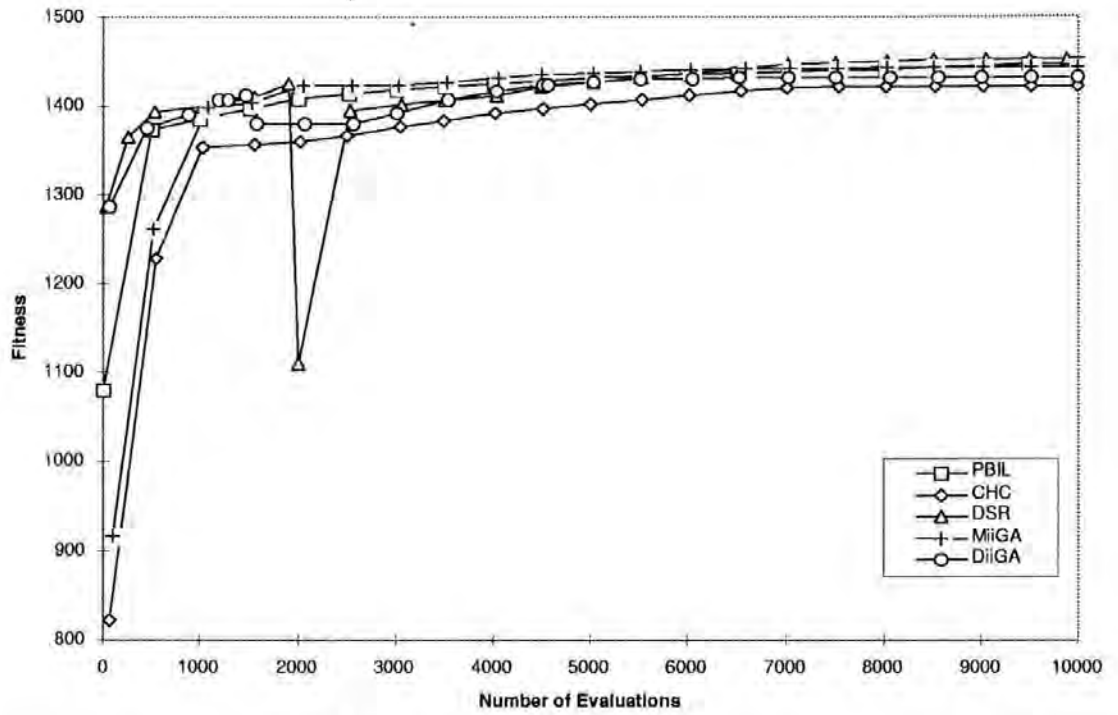


Figure 5-17: Performance of DiiGA against other techniques for 20x20 plate (3 load cases) max 24mm min 8mm.

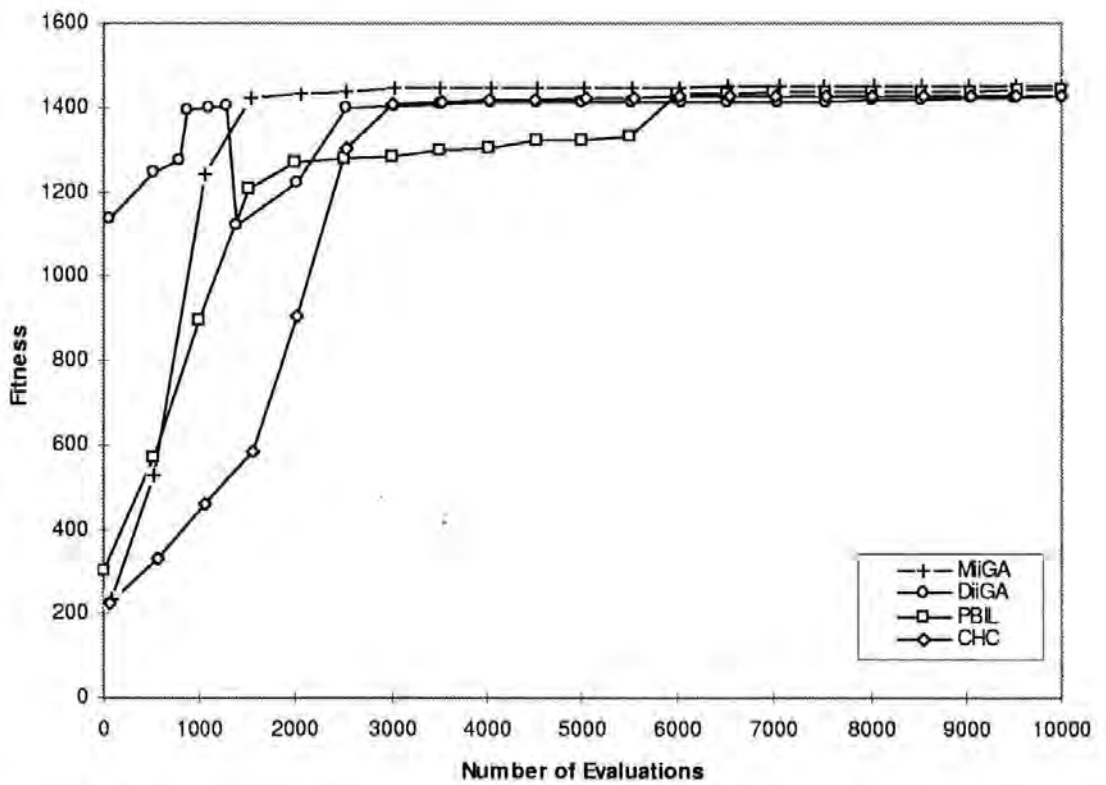


Figure 5-18: Performance of DiiGA against other techniques for 20x20 plate (3 load cases) max 18mm min 8mm.

Figure 5-19 and Figure 5-20 represents the fitness of sub-population 1 for a 24x24 plate. Rapid evolution is again apparent. The first dip shows the transition from a 6x6 to a 12x12 representation. The second dip shows the transition from a 12x12 to a 24x24 representation. The DSR technique does manage to produce fitter design solutions than the DiiGA on most problems, however these solutions are at the latter stages of the search and therefore requires the GA to run for a certain number of evaluations. In the case of DiiGA feasible design solutions are available relatively early in the design stage. A designer may look at these results and halt the evolutionary process if the design is deemed acceptable, thus saving computational effort and calendar time. The DiiGA is also considerably more robust (refer to standard deviation in Table 5-3) than the other techniques on the most complex of the problems which is a 24x24 plate utilising an upper limit on material of 18mm, highlighting it's explorative capabilities.

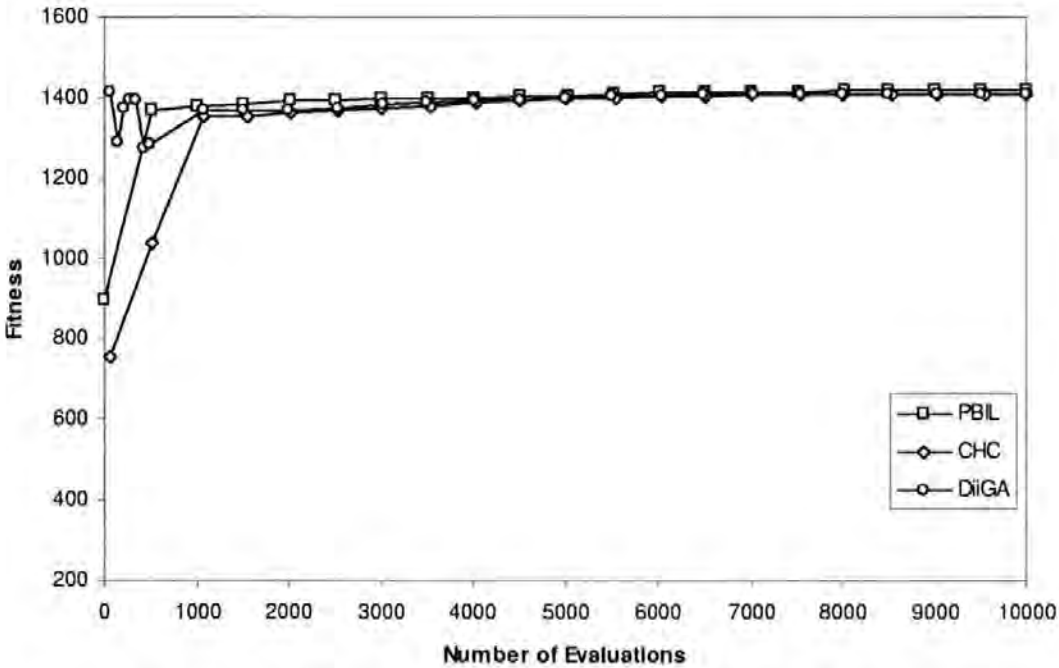


Figure 5-19: Performance of DiiGA against PBIL and CHC for 24x24 plate (3 load cases) max 24mm min 8mm.

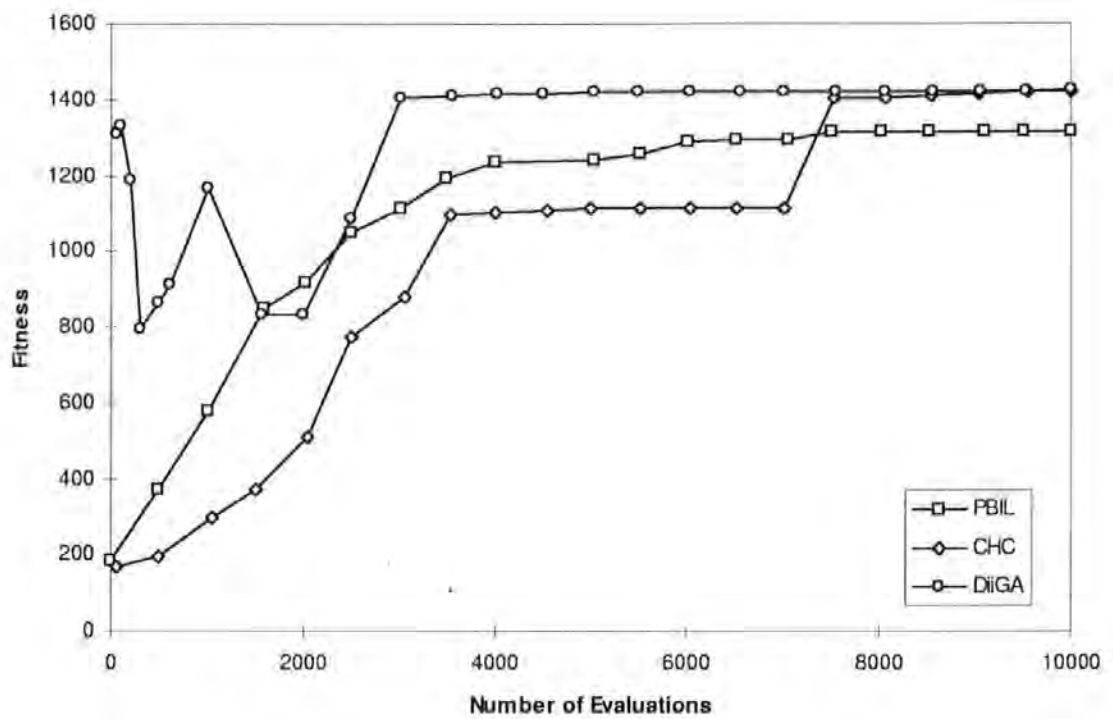


Figure 5-20: Performance of DiiGA against PBIL and CHC for 24x24 plate (3 load cases) max 18mm min 8mm.

The convergence of coarse representations in the MiiGA prevents the migration of further useful genetic material into the more detailed representations, thus promoting premature convergence. The DiiGA removes sub-populations as they cease to migrate fit individuals, these are replaced by new sub-populations which introduce new genetic material and thus more diversity. Because useful genetic information is gained during the coarse levels of the search, the information is passed to the finer representations resulting in a rapid increase in fitness as finer representations are introduced.

5.4.2 Application of Dynamic Injection Island GA on the Plate Problem utilising FEA

The previous section has shown the DiiGA to perform well in comparison to the other evolutionary techniques when utilising a complex stress analysis model. This section therefore investigates the performance improvement of the DiiGA in comparison to single representation techniques when utilising an FE analysis model. As with the previous chapter the FE problems were provided by industry.

Table 5-4 shows the computational expense for individual FE evaluations, it shows that the CPU time increases considerably with structural complexity i.e. number of elements, number of load cases. Table 5-5 shows the results for 48 and 200 variable problems.

Figure 5-22 shows the best average fitness of the 4 runs for the 48 variable single load case problem, whilst Figure 5-23 shows the average best weight of the 4 runs for the same problem. Two levels of representations are used. Results from the 12 variable sub-population are injected into the 48 variable sub-population. The initial fitness of the 12 variable sub-population is relatively high due to a low number of stress violations on the plate. Once the 12 variable sub-population ceases to inject useful material into the 48 variable sub-population it is discarded and replaced by another 48 variable representation, seeded from the old 48 variable sub-population. The DiiGA manages to reach lower weight design solutions earlier in comparison to the CHC and PBIL saving calls to the analysis model. As the lower 12 variable sub-population is cheaper to analyse it further saves computational expense. Approximately 350 evaluations in every run are performed on the 12 variable sub-population.

Number of Variables Number of Variables (grid size(x,y))	Number of Load Cases	Number of Nodes	Number of Elements	CPU time (seconds / evaluation)
12 (4x3)	1	216	83	1.7
12 (4x3)	3	216	83	2.9
48 (8x6)	1	864	506	8.5
48 (8x6)	3	864	506	14.5
50 (5x10)	1	720	414	6.57
200 (10x20)	1	2880	1786	43.3

Table 5-4 : Computational expense for individual FE evaluations

Test Number	Plate Size	Load Cases	Best Fitness	Best Weight	Average Fitness	Average Weight	Fitness (SD)
DiiGA_FEA1	48 (8x6)	1	1437.03	0.1440	1436.98	0.1442	0.020
DiiGA_FEA2	200 (10x20)	1	1435.59	0.1479	1435.19	0.1489	0.337
DiiGA_FEA3	48 (8x6)	3	1434.64	0.1504	1434.50	0.151	0.125

Table 5-5 : Results for 48 and 200 variable problems

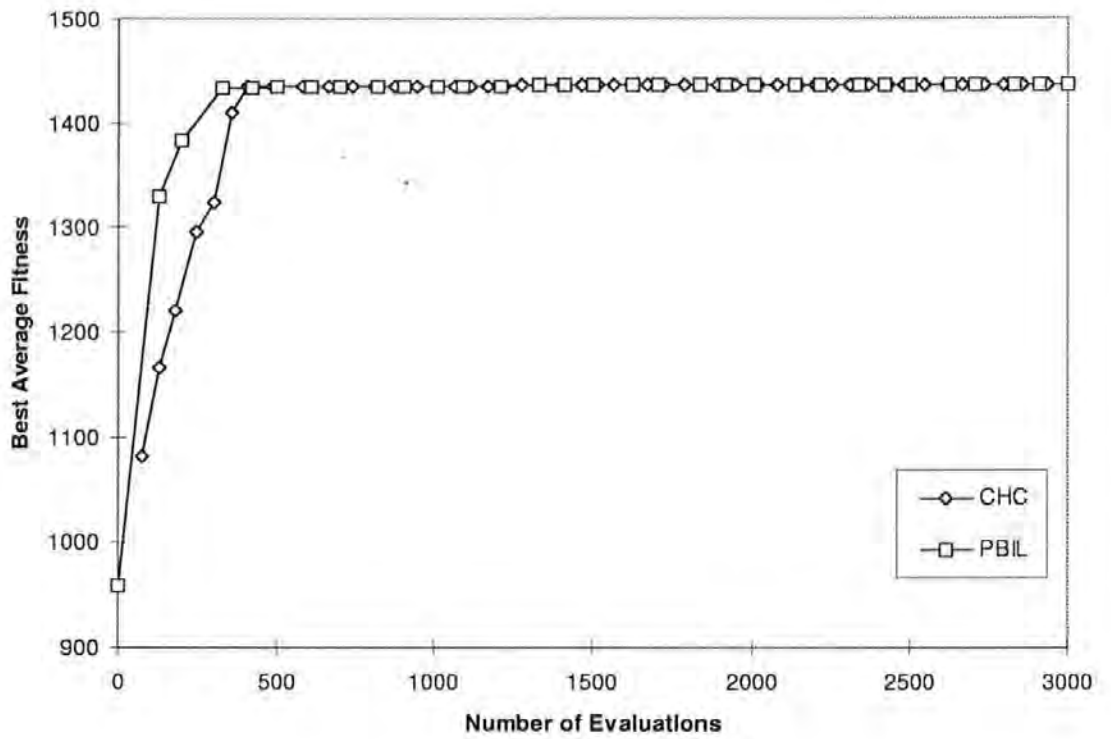


Figure 5-21 : Best Average Fitness utilising FEA (1 load cases, 48 variables)

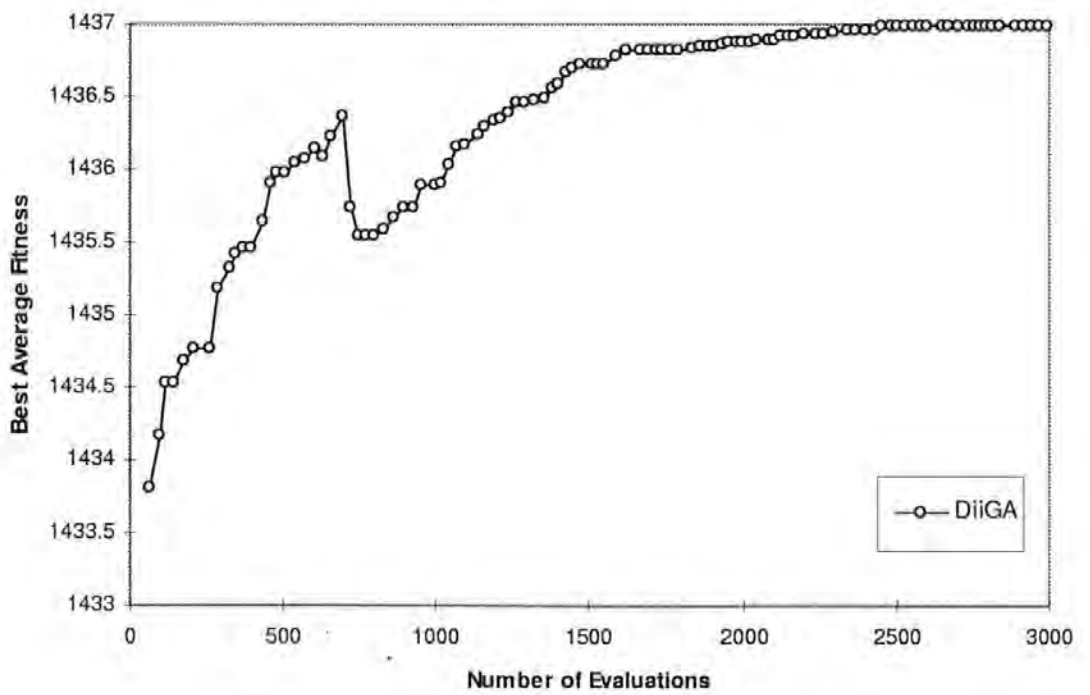


Figure 5-22 : Best Average Fitness utilising FEA (1 load cases, 48 variables)

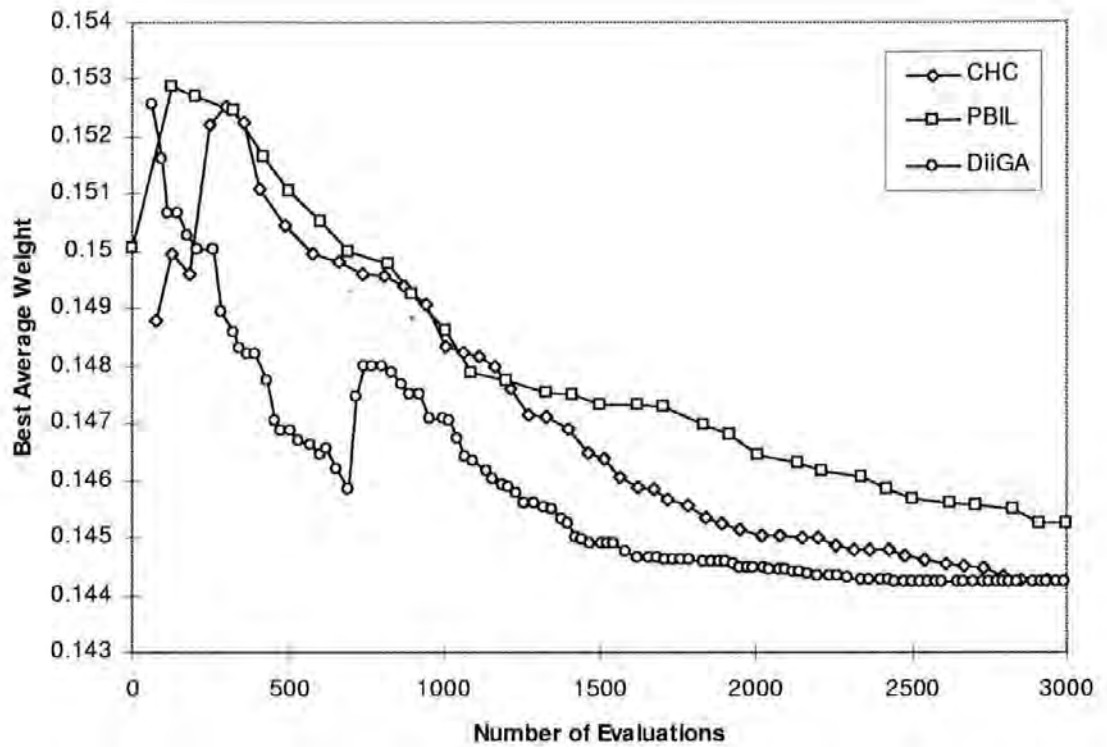


Figure 5-23 : Best Average Weight utilising FEA (1 load cases, 48 variables)

Figure 5-24 and Figure 5-25 shows the average best fitness of the 4 runs for the 48 variable three load case problem. 12 and 48 variable sub-populations are again used. Results from 12 variable sub-population are injected into the 48 variable sub-population. The initial fitness of the 12 variable sub-population is relatively high due to a low number of stress violations on the plate. However once the 48 variable sub-population is introduced it drops sharply due to larger stress violations resulting from the three load cases. Due to the new sub-population being seeded from a high fitness representation, and the injection of high fitness solutions from the other 48 variable sub-population the fitness rapidly picks up. This eventually leads to lower weight solutions being available to the designer earlier than the other two techniques, thus saving calls to the analysis model. The DiiGA on average requires 700 fewer evaluations to arrive at design solutions which are comparable

to those generated by the CHC at 3000 evaluations. Moreover approximately 350 of these evaluations are performed on the 12 variable sub-population. Therefore in a typical run the DiiGA requires approximately 30% less CPU time in comparison to the single population CHC GA. Also as with the 1 load case problem due to approximately 350 evaluations in every run being performed on the 12 variable sub-population further computational expense is saved.

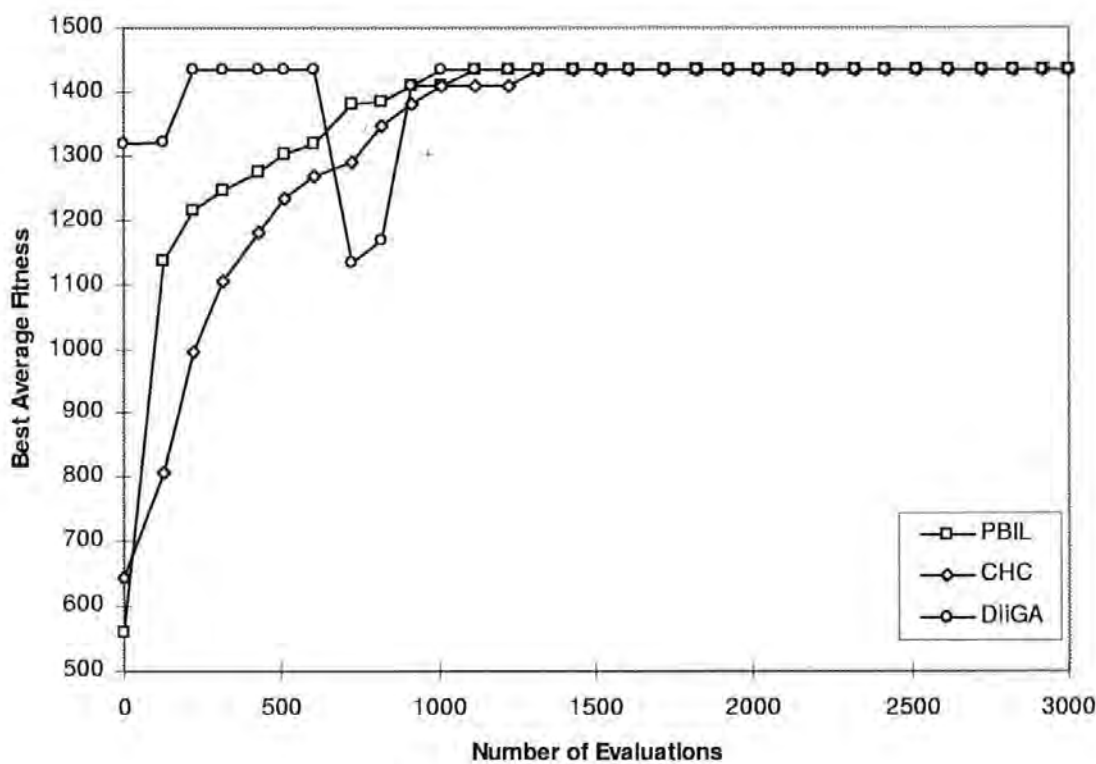


Figure 5-24 : Best Average Fitness utilising FEA (3 load cases, 48 variables)

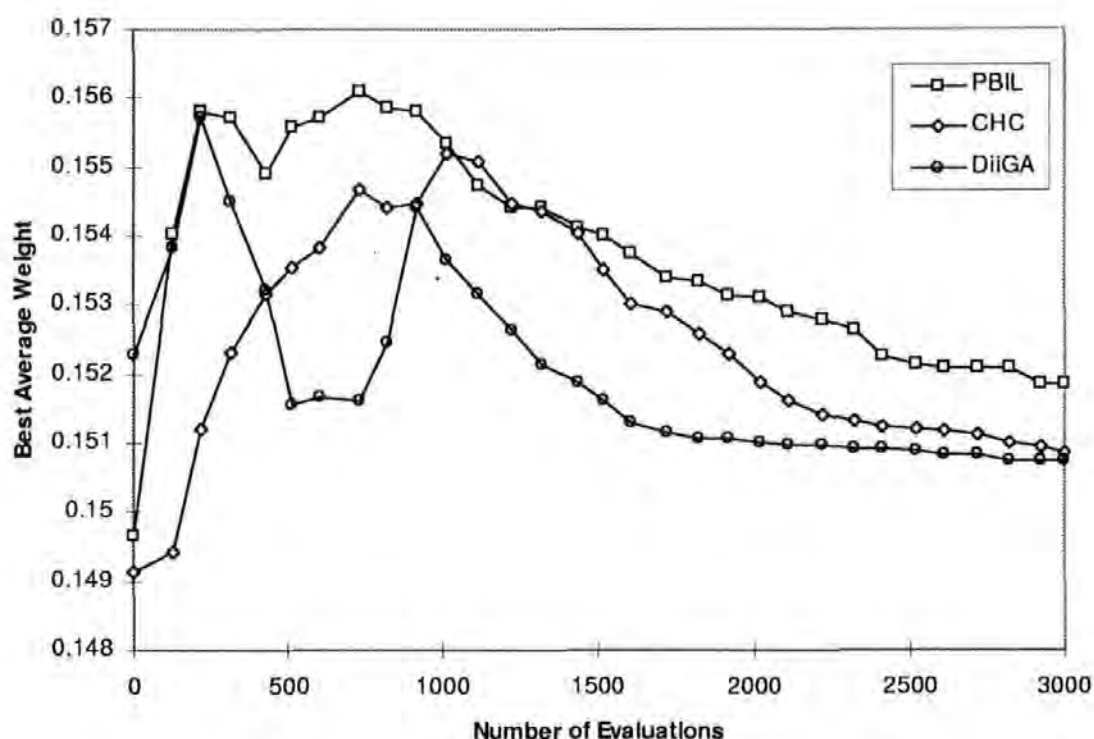


Figure 5-25 : Best Average Weight utilising FEA (3 load cases, 48 variables)

Figure 5-26 and Figure 5-27 show the average best fitness of the 4 runs for the 200 variable one load case problem. 50 and 200 variable sub-populations are used. Results from 50 variable sub-population are injected into the 200 variable sub-population. The initial fitness of the 50 variable sub-population is relatively high due to a low number of stress violations on the plate. Due to the new sub-population being seeded from a high fitness representation, and the injection of high fitness solutions from the other 50 high performance solutions evolve rapidly. The DiiGA on average requires 1000 fewer evaluations to arrive at a comparable design solution to the one generated by the CHC at 3000 evaluations. More over approximately 400 of these evaluations are performed on the 50 variable sub-population. So in a typical run to arrive at comparable design solutions CHC requires approximately 55% greater CPU time than the DiiGA. It is interesting to note PBIL's superior performance in comparison to the CHC and the DiiGA.

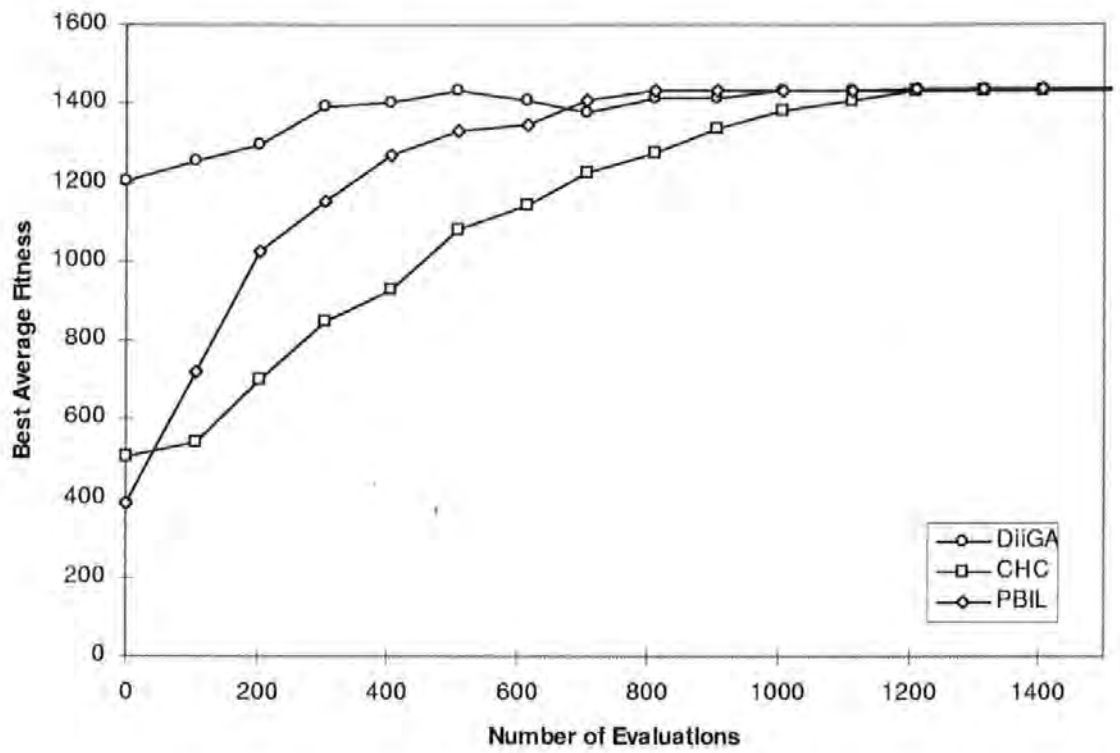


Figure 5-26 : Best Average Fitness utilising FEA (1 load case, 200 variables)

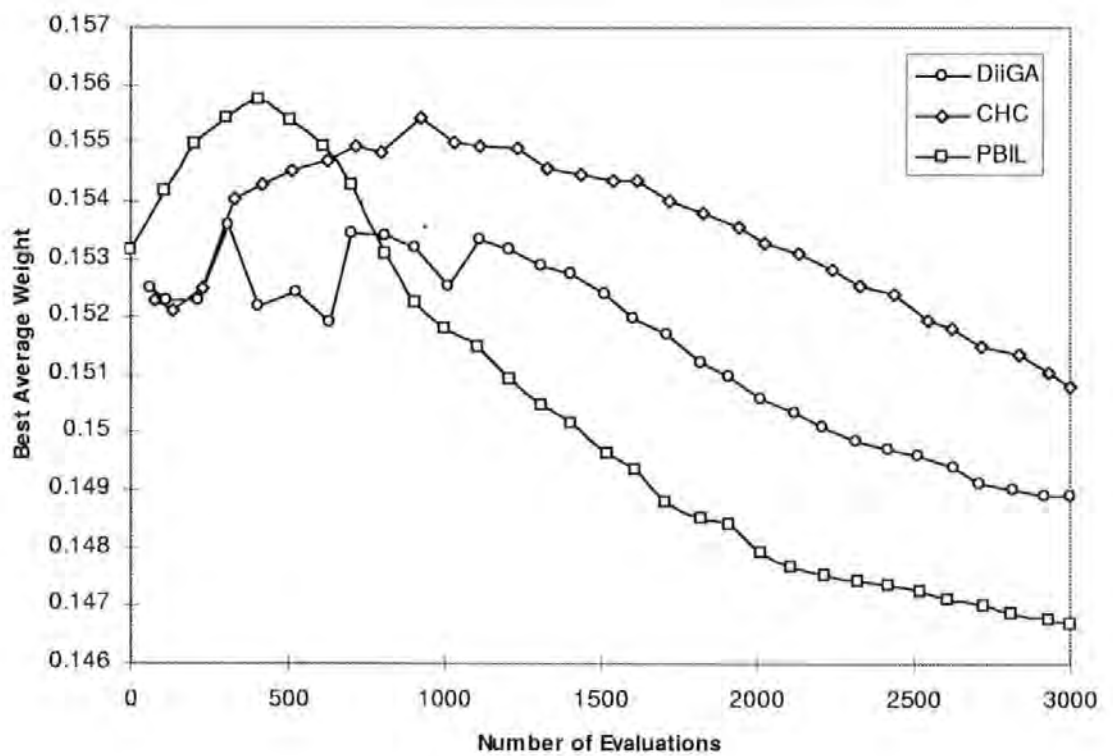


Figure 5-27 : Best Average Weight utilising FEA (1 load case, 200 variables)

The DiiGA has helped to considerably cut overall computational expense. By reducing the number of calls required and CPU time (through the use of coarser problem representations), complex models such as FEA may realistically be used. In the problem cases outlined only two levels of representations have been utilised. More levels may further reduce computational expense.

5.5 Summary

Practical computability of structural problems are often limited by high dimensionality. This chapter has outlined methods which aid tackling such problems in order to reduce computational expense. The DSR, MiiGA and DiiGA are all effective at reducing the overall number of evaluations to arrive at feasible design solutions. The advantage the MiiGA and DiiGA methods have over the DSR technique is that feasible design solutions evolve faster and are therefore available earlier in the design process, unlike the DSR technique where low risk design solutions are available during the latter stages of the search when the finest level of representation has been introduced. The designer may then stop the evolutionary process if a particular design is deemed acceptable thereby saving computational effort and calendar time.

The DiiGA is better suited to the task of exploration of the search space on problems utilising multi load cases and low limits on material. The experiments utilising FEA show that the computational savings appear to become more pronounced as the number of variables and load cases are increased. The results show that the DiiGA method can reduce computational expense by up to 55% in comparison to single population techniques. The operator settings for the DiiGA have not been optimised, so further savings may still be

possible. The introduction of higher levels of plate resolution would also further reduce the overall computational expense.

In order to expand the limits of practical computability, researchers have used parallel processing. However, before introducing parallel architectures it is extremely important to select, develop and optimise the underlying algorithms with respect to their cost and accuracy. The multi-level algorithms highlighted in this chapter behave differently on the problems presented. This however is not very surprising, based on the performance of the different search techniques and their dependency upon problem specifics outlined in chapter 4. There are many possible variations of the techniques and their relative operator setting. Amending these strategies may yield greater savings in calls to the model. However what this chapter has shown is that multi-level techniques are highly effective at reducing the overall computational effort in comparison to single level representation algorithms presented in chapter 4.

The single population algorithms discussed in chapter 4 display interesting performance characteristics and provide a better way of searching the design space based on problem specifics (exploration versus exploitation). The DiiGA has been shown to be very effective at reducing computational expense through the use of multi-level representations. The next chapter discusses how the properties of these various techniques may be combined to further improve design performance.

6. MULTI AGENT SEARCH TECHNIQUES

As highlighted in Chapter 4, different techniques may be better suited to varying stages of the evolutionary process. The CHC is capable of performing good exploration of the search space. PBIL, whilst capable of maintaining a reasonable level of exploration, is more efficient at exploiting the search space. Based upon the performance of these search techniques as outlined in chapter 4, and the ability of the DSR technique at handling higher numbers of variables two multi-agent search techniques are proposed by Vekeria and Parmee [1997] as a way of further reducing computational expense. (where agents refer to the individual algorithms).

The second half of this chapter discusses distributed techniques, the process resembles that of the DiiGA except the subpopulations manipulating the grid representations utilise different evolutionary strategies. This again is intuitively based upon the performance of the various adaptive search techniques detailed in chapter 4, the benefit of multi-level representations detailed in chapter 5 and improved performance of multi-agent strategies outlined in section 6.2 of this chapter.

6.1 Hybrid Search Techniques

Evolutionary algorithms can be very effective at solving certain classes of optimisation problem. There are, however, many problem areas where EA does not perform particularly well. As a result several hybrid EA's have been proposed [Seront and Bersini 1996]. The most common being local optimisation techniques incorporated within the GA. The GA is

a good global optimiser and explores the search space very efficiently. Conversely local search techniques are good local optimisers and perform good exploitation of solutions.

Hybrid approaches have been found to work well on some problems as a result of these complementary properties of the search algorithms.

For instance:

- Koumoussis and Georgiou [1994] introduced a mixed strategy that utilises GA's to search for optimal geometries of steel truss roofs, and a logic program, developed by the authors, to solve the sizing problem.
- Parmee [1996a] utilises a GAANT algorithm which involves aspects of an ant colony model in combination with a GA. This results in a dual-agent approach to achieve a multi-level search across a design hierarchy described by mixed discrete/continuous variable parameters.

Most evolutionary algorithms depend on a set of control parameters. Often the optimal setting of these parameters is dependent on the particular problem. Furthermore the optimal parameter settings may vary for different stages of the search. Similarly a search technique may work well for different types of problems or be better suited to different stages of the search process, as in chapter 4.

Adamidis and Petridis [1996] propose a method called Co-operating Populations with Different Evolution Behaviours (CoPDEB) where subpopulations are allowed to exhibit different evolution behaviours to overcome the problem of operator parameter setting. A coarse-grained parallel GA where a number of sub-populations co-evolve is utilised. Each

sub-population runs a GA with a different evolutionary behaviour by amending rules regarding selection, recombination and mutation.

6.2 Multi-Search Techniques

The first approach utilises the CHC and then the PBIL algorithm (CHC_PBIL). The reasoning being that the more diverse search of the CHC will provide an optimal starting individual for the PBIL-based search. The CHC will perform an initial explorative search, the PBIL method will then quickly exploit the surrounding local search space. The establishment of a multi-agent co-operative strategy may therefore provide a partial solution to the problem of balancing the two competing themes of exploration and exploitation. The proposed structure initially runs the CHC for 3000 function evaluations followed by PBIL for a further 7000 function evaluations. The fittest design solution from the CHC GA is used as the sample solution from which the initial probability vector is updated once PBIL is introduced. The probability vector is only updated when a better individual is located.

The second approach, initially utilises the PBIL algorithm which then switches to the CHC algorithm (PBIL_CHC). The reasoning here is that the PBIL method will quickly exploit solutions and identify promising regions which can then be explored by the CHC method. This presents an alternative strategy to the CHC_PBIL method. Similar settings are used to those of the CHC_PBIL method. PBIL initially runs for 3000 function evaluations followed by the CHC for a further 7000 function evaluations. The fittest design solution from PBIL is used as a template for creating the new CHC population, whereby each new individual is created by flipping a fixed proportion (e.g., 30%) of the template's bits chosen at random.

One instance of the best is added unchanged to the new population. This ensures that the CHC search cannot converge to a worse solution than the previous PBIL search.

6.2.1 Application of Multi-Search Techniques to the Plate Problem

CHC_PBIL								
Test Number	Plate size	Max upper limit	Load cases	Best Fitness	Best Weight	Fitness (SD)	Average Fitness	Average Weight
CHC_PBIL_1	20X20	24	1	1483.33	1.03	4.93	1477.37	1.05
CHC_PBIL_2	20X20	18	1	1484.93	1.03	2.46	1481.33	1.04
CHC_PBIL_3	24X24	24	1	1461.50	1.08	4.21	1455.57	1.10
CHC_PBIL_4	24X24	18	1	1473.69	1.06	4.16	1466.99	1.07
CHC_PBIL_5	20X20	24	3	1447.31	1.12	4.56	1438.65	1.14
CHC_PBIL_6	20X20	18	3	1448.96	1.11	33.75	1433.44	1.13
CHC_PBIL_7	24X24	24	3	1421.94	1.19	4.38	1417.47	1.20
CHC_PBIL_8	24X24	18	3	1427.61	1.17	81.42	1362.52	1.18

Table 6-6: Results for CHC_PBIL utilising various problem cases (no. of runs = 10)

PBIL_CHC								
Test Number	Plate size	Max upper limit	Load cases	Best Fitness	Best Weight	Fitness (SD)	Average Fitness	Average Weight
PBIL_CHC_1	20X20	24	1	1486.09	1.03	5.69	1474.67	1.05
PBIL_CHC_2	20X20	18	1	1489.70	1.02	6.39	1479.39	1.04
PBIL_CHC_3	24X24	24	1	1460.69	1.09	7.15	1452.18	1.11
PBIL_CHC_4	24X24	18	1	1475.75	1.05	4.19	1468.04	1.07
PBIL_CHC_5	20X20	24	3	1437.51	1.14	4.47	1430.29	1.16
PBIL_CHC_6	20X20	18	3	1445.44	1.12	3.95	1437.85	1.14
PBIL_CHC_7	24X24	24	3	1418.87	1.19	3.91	1413.75	1.21
PBIL_CHC_8	24X24	18	3	1426.49	1.17	4.99	1421.53	1.19

Table 6-7: Results for PBIL_CHC utilising various problem cases (no. of runs = 10)

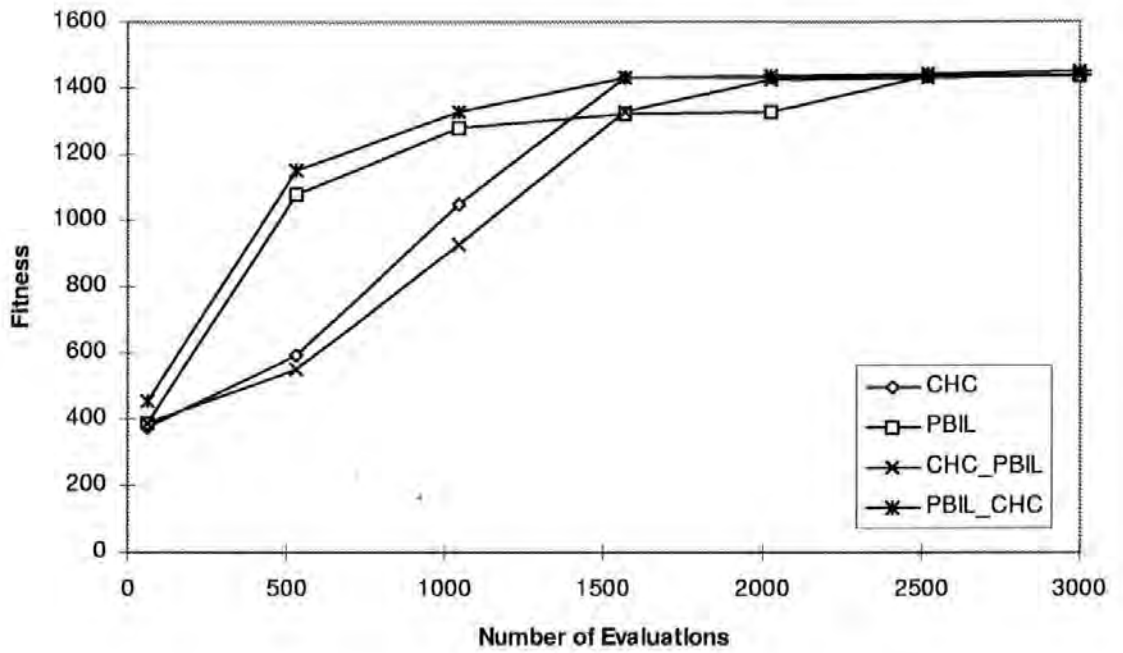


Figure 6-1 : Graph to show a comparison of the early stages of the search process for different search methods -(1 load case 24x24 plate max. 18mm, min. 8mm)

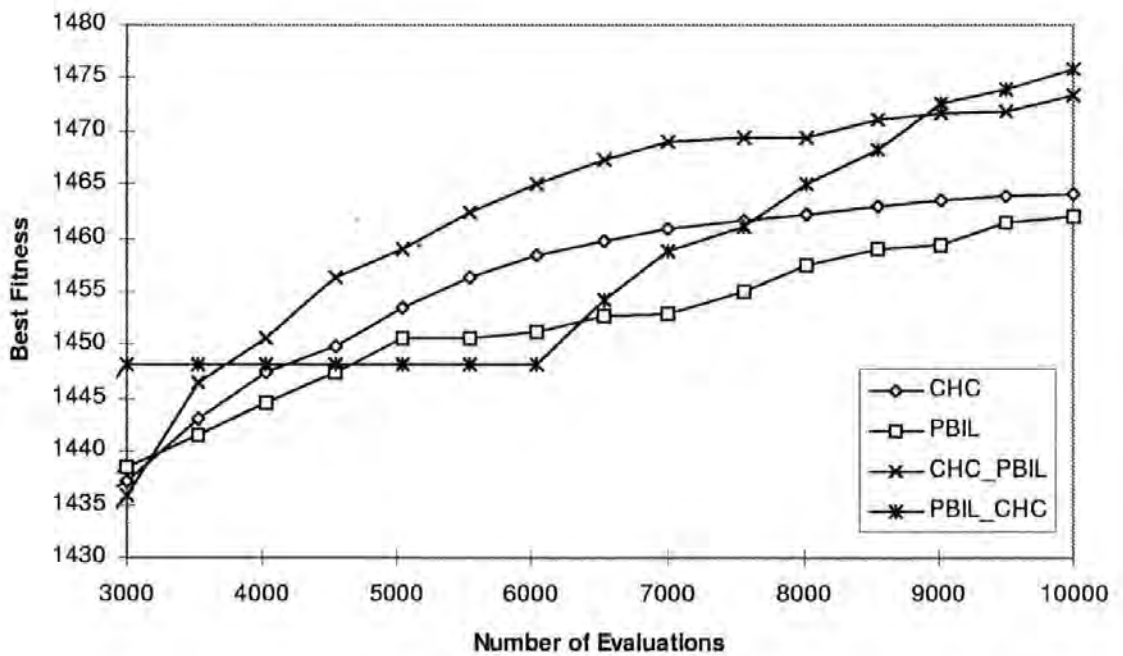


Figure 6-2 : Graph to show a comparison of latter stages of the search process for different search methods -(1 load case 24x24 plate max. 18mm, min. 8mm)

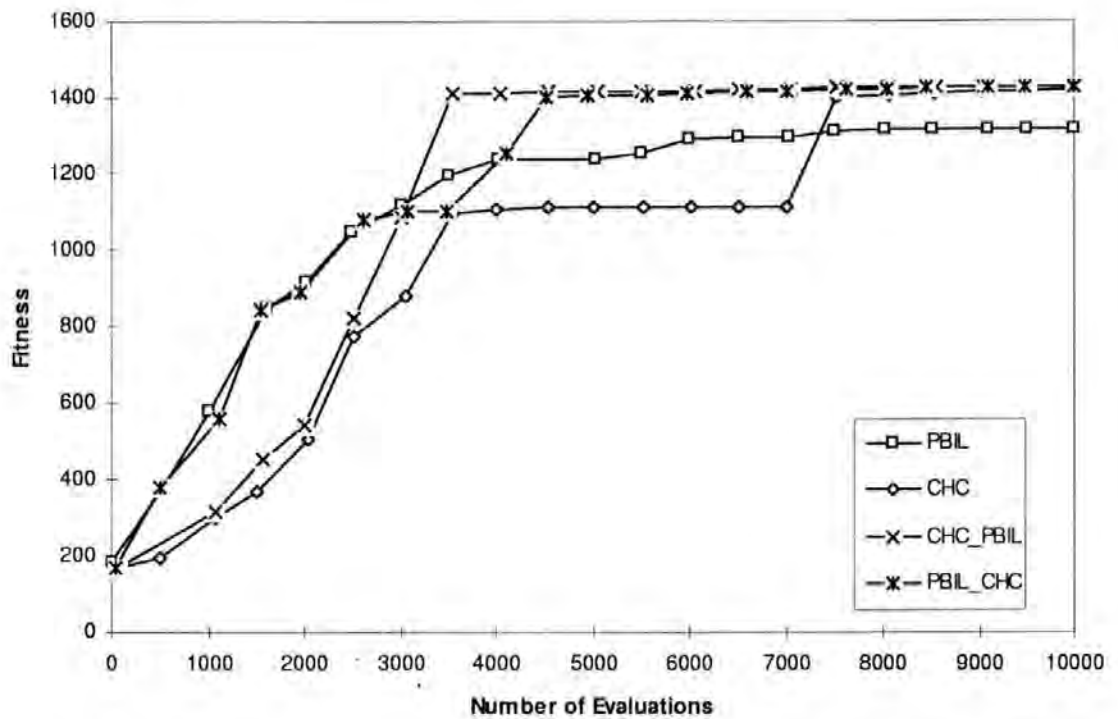


Figure 6-3 : Graph to show a comparison of different search methods -(3 load cases 24x24 plate max. 18mm, min. 8mm)

Figure 6-1 and Figure 6-2 shows the evolution curve for the PBIL, CHC sub-populations running independently against the proposed hybrid technique utilising both algorithms. The graphs show the fittest individual from 10 runs for each algorithm. The CHC_PBIL method shows an initial slow increase in the performance as a result of its explorative behaviour. The PBIL method then rapidly negotiates the surrounding search space and converges to a design solution. The CHC method on its own takes a further 6000 evaluations to arrive at a comparable solution. In comparison the PBIL-CHC curve shows a rapid increase in fitness during the early stages of search but then levels off during the introduction of the CHC method after 3000 evaluations. The PBIL-CHC method does eventually find a solution fitter than the CHC_PBIL combination due to the CHC's explorative behaviour.

Table 6-6 and Table 6-7 show that in all the test cases utilising 3 load cases the CHC_PBIL method produced the best final fitness. Due to its initial exploration and then the rapid exploitation, this technique is well suited especially to problems of high dimensionality utilising single load cases. However on the more complex problem of the 24x24 3 load case plate with a minimum 18mm maximum variable thickness the PBIL_CHC algorithm seems better suited (Table 6-7 and fig 6.3). This problem has a more complex distribution of material on the plate. Also as we have reduced the upper limit on material the number of possible design directions is reduced thus requiring more exploration of the search space. This is illustrated in Figure 6-3 which shows rapid evolution by the PBIL and PBIL_CHC methods. Due to the initial PBIL phase, rapid exploitation of solutions takes place which are then explored by the CHC. Although in this case the CHC_PBIL method shows a high performance solution with rapid evolution, the method is not robust and on average produces much lower performance design solutions in comparison to the PBIL_CHC method (refer to standard deviation and average fitness values for test cases 6 and 7 in Table 6-6 and Table 6-7). The PBIL_CHC combination seems to provide better complementary properties which result in fitter and more robust design solutions than those produced by CHC, PBIL and the hybrid CHC_PBIL algorithms for this class of problem.

Other techniques may also be incorporated in order to take advantage of multi-level representations and multi-agent search strategies. Two methods are proposed where the Dynamic Search Refinement (DSR) technique may be used in conjunction with a multi-agent search strategy. The first approach may utilise a coarse representation manipulated by PBIL which then switches to finer representation manipulated by the CHC algorithm, after convergence or a certain number of evaluations. The reasoning here is that the PBIL process may rapidly converge to a high-performance region utilising a coarse representation, which is then explored by the CHC technique utilising a finer

representation. The second approach may utilise a coarse representation manipulated by CHC, which then switches to finer representation manipulated by the PBIL algorithm. The reasoning here is that the CHC process utilising a coarse representation may initially better explore the search space and identify a number of diverse high-performance solutions, then a finer PBIL representation rapidly converge to a local, optimum solution. The process would be performed in a sequential manner, gradually increasing the complexity of the representation whilst taking advantage of the differing characteristics of the search algorithms.

6.3 Distributed Search Techniques

The distributed search technique utilises different resolution grids, each evolving upon a separate island. The process is similar to the DiiGA except the subpopulations manipulating the grid representations utilise different evolutionary strategies. This is intuitively based upon the performance of the various adaptive search techniques detailed in chapter 4, the benefit of multi-level representations detailed in chapter 5 and improved performance of multi-agent strategies outlined in section 6.2 of this chapter.

The establishment of a distributed architecture supporting several search algorithms and their subsequent removal / re-introduction depending upon relative performance during the evolution process may provide a partial solution to the problem of selecting the most appropriate search technique for a particular problem.

6.3.1 Application of Distributed Search Techniques to the Plate Problem

Two simple configurations are assessed. The first method is termed chc-pbil-pbil (c-p-p) (Figure 6-4).

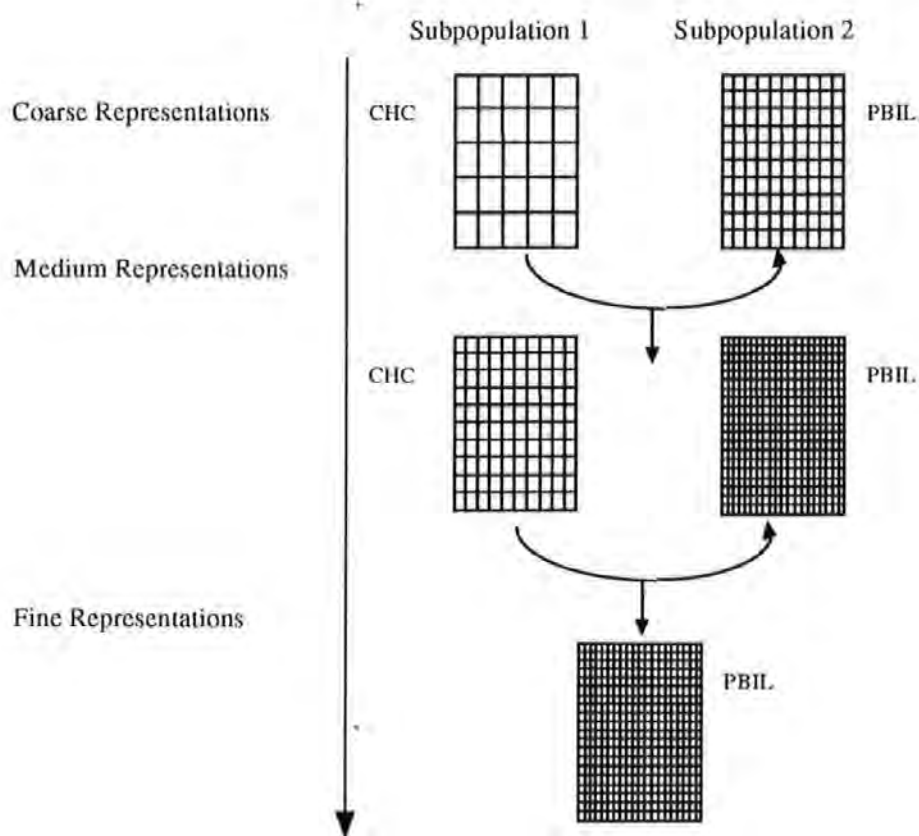


Figure 6-4 : Grid Representation for the chc-pbil-pbil (c-p-p) configuration

The CHC algorithm manipulates a 5x5 grid representation which co-evolves with a PBIL manipulation of a 10x10 representation. Migration is allowed every 200 evaluations. The 5x5 CHC process is killed as it ceases to provide sufficiently high-performance solutions for injection to the 10x10 PBIL process. The CHC now manipulates the 10x10 representation. The fittest individual is used as a template to create the new sub-population. Each new individual is created by flipping a fixed proportion (30%) of the template bits chosen at random. One instance of the best is left unchanged. A further 20x20 PBIL representation is introduced and co-evolves with the lower 10x10 CHC representation. The PBIL process receives injected solutions every 200 generations. Once the 10x10 CHC representation ceases to pass useful information to the 20x20 PBIL process it is killed. The 20x20 PBIL process continues to evolve until it has converged. The reasoning here is that

the more diverse search of the CHC which leads to higher performance on the coarser resolutions interacts with the more rapid convergence characteristics of PBIL to provide an optimal starting population for the final PBIL-based search.

The objective is a higher-performance solution within a lesser number of function evaluations than would be attainable using the CHC alone within a DiiGA architecture. Results from a single load-case representation are shown in Figure 6-5 and Figure 6-6 and compared to the results from a three load-case representation.

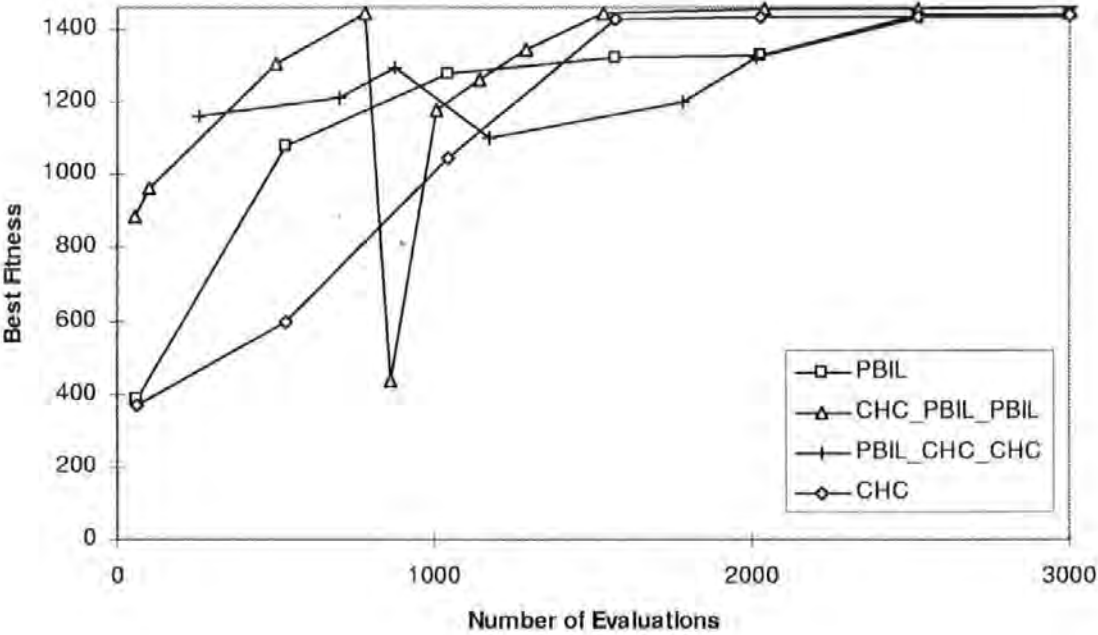


Figure 6-5 : Graph to show a comparison of the early stages of the search process for different search methods -(1 load cases 24x24 plate max. 18mm, min. 8mm)

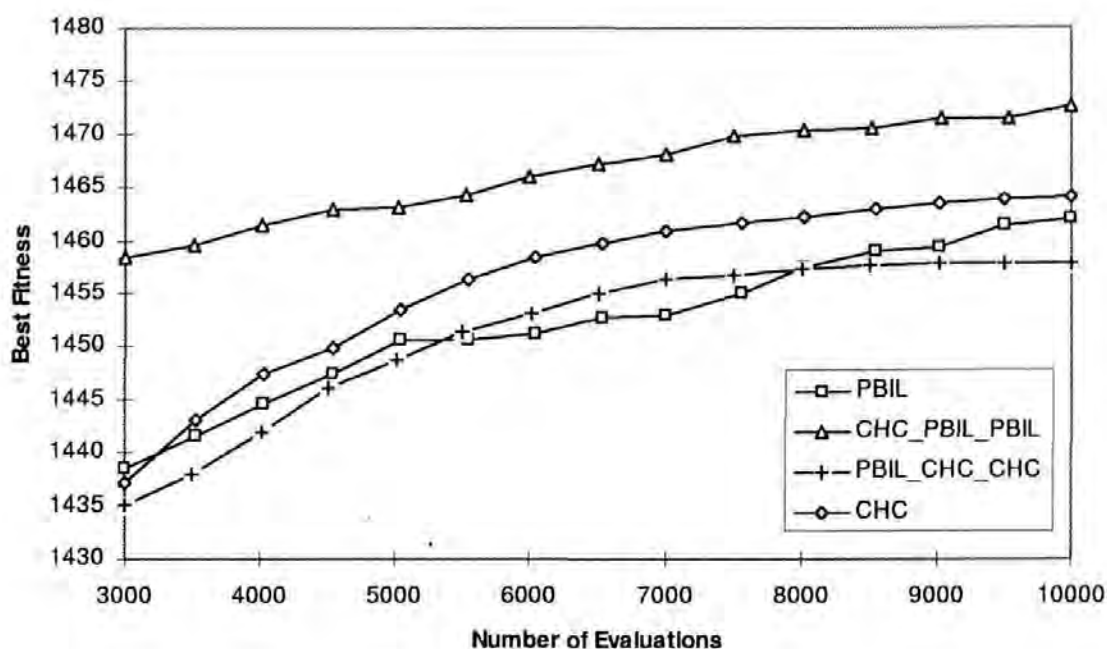


Figure 6-6 : Graph to show a comparison of latter stages of the search process for different search methods -(1 load cases 24x24 plate max. 18mm, min. 8mm)

CHC_PBIL_PBIL								
Test Number	Plate size	Max upper limit	Load cases	Best Fitness	Best Weight	Fitness (SD)	Average Fitness	Average Weight
C-P-P_1	20X20	24	1	1482.78	1.04	4.87	1474.90	1.05
C-P-P_2	20X20	18	1	1486.90	1.03	4.35	1481.44	1.04
C-P-P_3	24X24	24	1	1454.32	1.10	2.11	1451.51	1.11
C-P-P_4	24X24	18	1	1472.70	1.06	3.61	1466.18	1.07
C-P-P_5	20X20	24	3	1449.04	1.11	7.51	1437.58	1.14
C-P-P_6	20X20	18	3	1450.05	1.11	90.00	1330.29	1.13
C-P-P_7	24X24	24	3	1423.78	1.18	5.01	1415.48	1.20
C-P-P_8	24X24	18	3	1427.24	1.17	76.56	1365.16	1.19

Table 6-8: Results for CHC-PBIL-PBIL utilising various problem cases (no. of runs = 10)

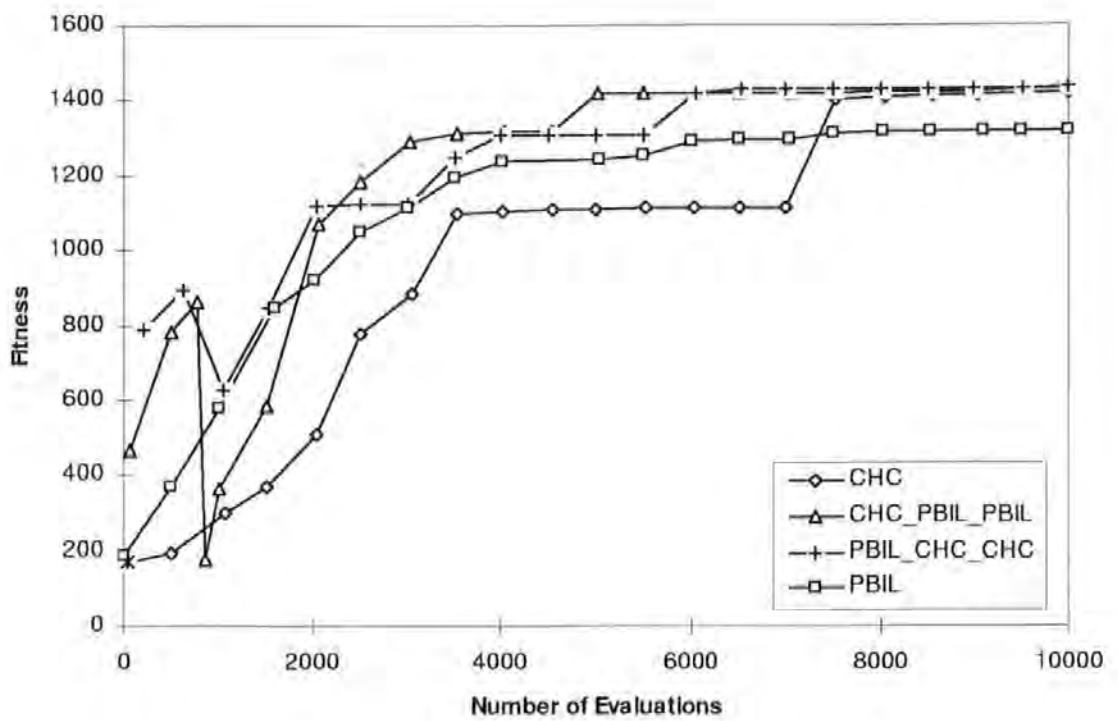


Figure 6-7 : Graph to show a comparison of different search methods -(3 load cases 24x24 plate max. 18mm, min. 8mm)

The second method is termed pbil-chc-chc (p-c-c) (Figure 6-8). This configuration involves a PBIL manipulation of the 5x5 grid co-evolving with a CHC manipulation of the 10x10 grid. The 5x5 PBIL process is killed as it ceases to pass useful information to the CHC process. PBIL now manipulates the 10x10 resolution. A further 20x20 CHC process is introduced and co-evolves with the 10x10 PBIL representation. Once the 10x10 PBIL representation ceases to pass useful information to the 20x20 CHC process it is killed. The 20x20 CHC representation continues to evolve until it has converged. This strategy therefore investigates an alternative dynamic where PBIL injects locally high-performing solutions into the more diverse search processes of CHC. Results from a single load case problem are shown in Figure 6-5 and Figure 6-6 and compared to the results from a three load-case representation

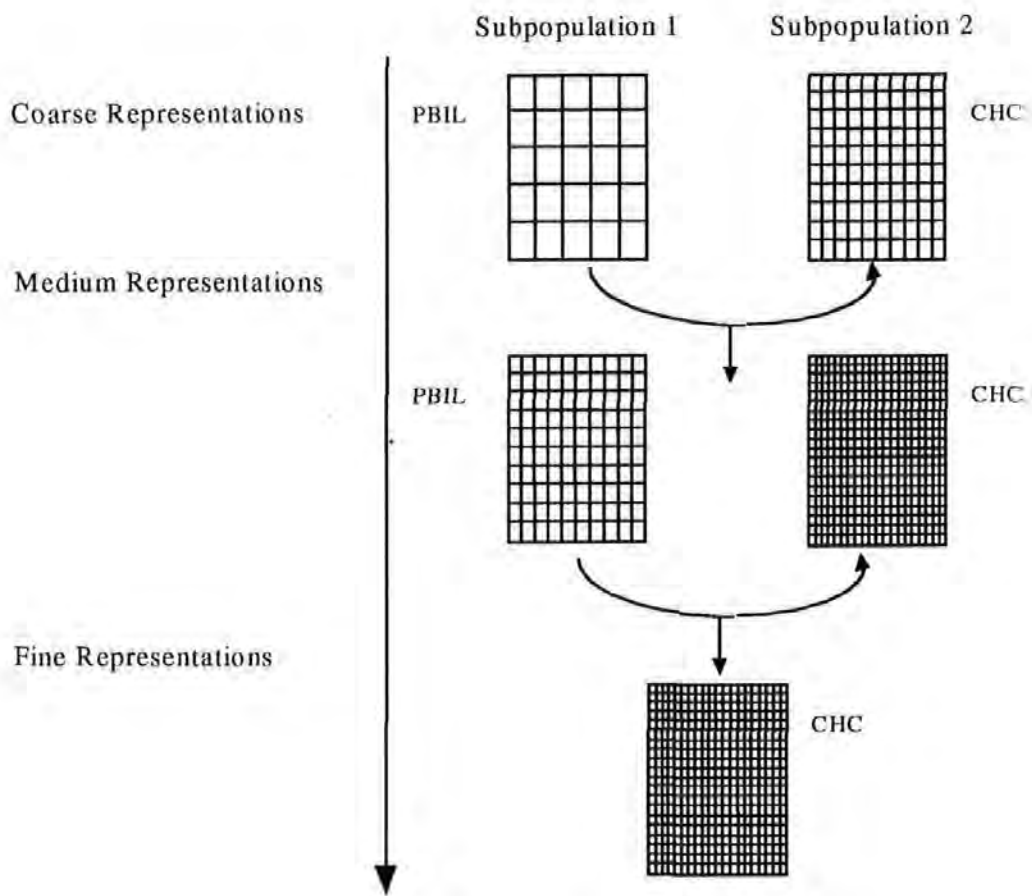


Figure 6-8: Grid Representation for the pbil-chc-chc (p-c-c) configuration

PBIL_CHC_CHC								
Test Number	Plate size	Max upper limit	Load cases	Best Fitness	Best Weight	Fitness (SD)	Average Fitness	Average Weight
P-C-C_1	20X20	24	1	1460.69	1.09	5.10	1455.54	1.10
P-C-C_2	20X20	18	1	1468.28	1.07	6.60	1459.56	1.09
P-C-C_3	24X24	24	1	1444.45	1.12	4.88	1437.66	1.14
P-C-C_4	24X24	18	1	1457.92	1.09	9.95	1450.33	1.11
P-C-C_5	20X20	24	3	1422.75	1.18	6.28	1416.32	1.20
P-C-C_6	20X20	18	3	1431.90	1.16	6.19	1423.34	1.18
P-C-C_7	24X24	24	3	1412.39	1.21	7.29	1404.56	1.24
P-C-C_8	24X24	18	3	1431.81	1.16	4.54	1424.58	1.18

Table 6-9: Results for PBIL-CHC-CHC utilising various problem cases (no. of runs = 10)

The c-p-p co-evolution results in increased performance both in terms of reduced calls to the evaluation function and improved overall fitness in the single load case situation. However, as a more realistic three load case problem is introduced the c-p-p is very significantly out performed in robustness by the p-c-c co-evolution (Table 6-8 and Table 6-9). Although further experimentation is required to better determine the dynamics for these comparative performances upon differing problem representations, the following reasoning appears sound. The single load case promotes the generation of material concentrations in one area of the plate and it is suggested that the convergence characteristics of the c-p-p are better suited to a less complex distribution of material upon the plate than that required by the three load case problem. With three load cases material is distributed across a wider area of the plate to best satisfy stress characteristics. It is assumed that the greater diversity of the later stages of p-c-c search results in the better identification of this more complex material distribution. The rapid convergence characteristics of PBIL however, greatly accelerates this identification resulting in far less evaluation calls than is required by a DiiGA process utilising CHC alone. It is interesting to note, however, that performance of the CHC alone finally equals that of the p-c-c co-evolution, whereas the characteristics of the c-p-p process results in rapid convergence upon a significantly lower robust solution.

6.4 Variable Complexity Modelling

The integration of FEA representation and concurrent processing of simple evaluation models alongside complex analyses may also yield greater savings in computational expense. Giunta et al [1995] have developed an interesting technique termed “variable-complexity modelling”, it is a process by which simple, computationally inexpensive analysis techniques are used together with more detailed, expensive techniques in the design optimisation process.

This technique is applied in the area of multidisciplinary optimisation (MDO) in the aerodynamic-structural optimisation of the High Speed Civil Transport (HSCT), which is computationally expensive due to the analysis of the vehicle and its many systems. A typical optimisation problem is to minimise the takeoff gross weight of an HSCT configuration. Starting with a large number of candidate HSCT configurations, the designs are screened using algebraic weight equations (which is relatively inexpensive in comparison to detailed analysis methods) to eliminate impossible design points. Detailed finite element analysis is then applied to selected configurations in the remaining design space to provide more accurate results.

Ellman et al [1996] use several strategies for the design optimisation of a sailing yacht. One of the strategies uses a simple model to get near an optimum, before relying upon a complex model during the last stage of the design similar to Giunta et al. [1995]. Ellman writes:

“ An optimisation algorithm can often utilise relatively simple models to make search control decisions, and rely on complex models only when needed to verify optimality of a solution and satisfaction of constraints. For this reason, a combination of simple and complex models can lead to designs as good as those resulting from a single complex model, but at far lower computational expense. ”

6.5 Summary

The experiments performed on the multi-load cases shows that the manner in which the algorithms are used is extremely important. The sequence of the algorithms seems highly dependent upon problem specifics. The CHC_PBIL approach is especially well suited to tackling problems of high dimensionality utilising single load cases. This method initially explores but spends majority of it's time exploiting information regarding the search space and as a result is better suited to this class of problem. The PBIL_CHC performs rapid exploitation and then expends the majority of it's time exploring. As a result, it is better suited to problems which are more complicated, where fewer high performance solutions exist in the search space and a higher degree of exploration is required in order to identify them.

The results from the distributed search strategies indicate a potential for a further reduction in calls to the evaluation model. The p-c-c method manages to achieve better results than the DiiGA technique and is also more robust. The selected search configurations are however very sensitive to problem specifics e.g. the performance differences between the one and three load case scenarios. This second point may be addressed by improving the dynamics of the introduction / removal of individual search algorithms.

7. CONCLUSIONS

7.1 Conclusions

The aim of this work has been to develop a semi automated system which is capable of providing high performance design solutions. By combining evolutionary optimisation of the design utilising evaluation software to provide a measure of the quality of the designs, there is little need for human intervention in the design process. Such a system would be highly desirable in terms of cost as design lead time and associated man-hours is reduced.

The research has focused on evolutionary / adaptive strategies that allow the machine based design of a single engineering component from preliminary problem definition through to detailed definition. In order to achieve this it was necessary to overcome two main problem areas i.e. the successful optimisation of large numbers of interactive design variables and the minimisation of calls to the fitness evaluation model. These objectives have been achieved to a significant extent by the introduction of high performance advanced computational strategies.

Many researchers have focused on the use of a canonical GA for design optimisation tasks. This work, realising the limitations of the canonical GA in terms of calls to the model and design performance has compared the performance of different high performance evolutionary algorithms. It has been demonstrated that the use of high performance evolutionary search algorithms such as CHC and PBIL help to reduce the overall number

of calls to the evaluation function, and thus make it feasible to integrate computationally intensive models such as FEA with an evolutionary design process.

The performance of the CHC and PBIL algorithms on the plate problem shows them to be extremely competitive in comparison to the Breeder GA and the canonical GA. The overall results have shown firstly that the CHC is an extremely robust and highly explorative algorithm and secondly that the rapid convergence characteristics of PBIL help it to rapidly exploit solutions in large search spaces.

Many factors effect the total number of analyses that need to be performed. In the case of the plate problem factors such as the level of representation, and the loading conditions have a bearing on this figure. It has been shown that most of the algorithms utilised with the plate problem generally perform well on coarse representations when utilising a single load case and a small number of design variables. However when they are applied to more complex higher dimensional problems, some of the algorithms notably the canonical GA and BGA deteriorate considerably in performance. Therefore the use of algorithms and operator settings based on a simpler representation may not be sufficient to solve more complex problems. All of the algorithms displayed a marked degradation in performance as the complexity of the problem increases with the introduction of more variables (>80).

The need to handle higher numbers of variables to allow lower risk design solutions has led to the development of techniques which exploit differing levels of a problem representation. These strategies involve a gradual increase in dimensionality as the search process advances. The advantage in using this approach is that coarse representations

provide a good starting point for the finer representations in addition to being less computationally expensive.

Three multi-level techniques were developed by Vekeria and Parmee namely the DSR, MiiGA and DiiGA. The DSR technique presents a novel way of dealing with a large numbers of variables and reducing the number of calls to the model during a GA run. By utilising a combination of simple and complex representations the DSR technique on most of the problems outlined leads to a design which is as good as those resulting from those of a single fine representation but at lower computational cost through the utilisation of coarse representations. This reduction is however during the latter more detailed stages of the design process.

The Modified injection island architecture (MiiGA) was introduced to allow feasible design solutions to evolve faster and be made available earlier in the design process. This allows significant savings in computational effort and calendar time. The MiiGA offers a concurrent rather than a sequential shape refinement process. Alternative architectures are suggested as a way of minimising the number of calls to the model and CPU cost through the use of different representations. The MiiGA manages to locate feasible designs earlier than those utilising the CHC and PBIL methods. Although this approach provides rapid evolution of design solutions it also has the draw back of premature convergence resulting in stagnation of the optimisation process and final designs which are worse than the DSR technique.

The Dynamic Injection Island Genetic Algorithm (DiiGA) addresses the problem of stagnation by phasing out lower representations as their performance declines. This method combines the better mechanisms of both the DSR and MiiGA approaches. The DiiGA

achieves a significantly higher fitness overall whilst still maintaining the initial rapid improvements exhibited by the MiiGA. The DiiGA is well suited to the task of exploration of the search space especially on problems utilising multi load cases and low limits on material. The experiments utilising FEA show that the computational savings appear to become more pronounced as the number of variables and load cases are increased. Results in chapter 5 show the potential of the DiiGA technique on a 200 variable one load case problem. Typically, the CHC requires approximately 55% (on average) greater CPU time than the DiiGA to identify a comparable design solution. While these results are very encouraging, it is expected that further computational savings could be achieved by firstly optimising the operator settings and secondly using more levels of representations.

The main advantages therefore in using multi-level techniques such as the DSR, MiiGA and DiiGA lies in the reduction of computational expense, handling high dimensionality and the ability to identify higher fitness design solutions in comparison to single representation GA's. There are many possible variations of the techniques examples include number of islands, number of levels of representations, migration strategies etc. Amending these strategies should yield further savings in calls to the model.

Based on the performance characteristics of the individual CHC and PBIL algorithms and the ability of the DiiGA technique in providing the capability of handling higher numbers of variables and solutions earlier in the search process a multi agent approach has been proposed. The reason for this is to take advantage and further improve performance characteristics. However more extensive experimentation is required to properly assess the utility of co-evolving processes involving several differing search algorithms. However the preliminary findings of chapter 6 indicate that:

- It is possible to improve performance both in terms of overall fitness and reduced evaluation calls.
- The selected search configurations are very sensitive to problem specifics e.g. the performance variation between one and three load case scenarios.

This second point can be addressed by improving the dynamics of the introduction / removal of individual search algorithms. A performance based scenario is envisaged whereby algorithms are removed / re-introduced dependent upon on-line measurement of their relative performance. This could result in the automatic selection of appropriate search configurations.

The initial results have indicated a considerable potential for a significant reduction in the number of evaluation calls during evolutionary search. Refinement of the basic strategies introduced here are likely to further reduce computational expense related to evaluation calls. In generic terms this will allow a more efficient integration with complex analysis techniques during detailed design and contribute significantly to those preliminary stages of the design process where a degree of complex analysis is required to validate results from more simplistic preliminary design models.

Initial introduction of the stand-alone CHC GA incorporating FEA analysis with the design process within an industrial environment has shown that it is possible to achieve improved solutions whilst significantly reducing design lead time. This involves a more machine-based process where designer interaction is required to fine-tune GA-generated designs. This interaction is largely required due to the high-risk aspects related to insufficient resolution of the grid representation. The use of multi-level representations has increased the number of elements that can be successfully manipulated from eighty to four hundred.

This higher resolution plus the achieved reductions in computational expense indicate that a total machine-based approach is possible. This can only result in further reductions in lead time and related cost reduction. It is expected that these new techniques will be integrated with the industrial design process as a prototype system in the near future.

Further work can look at incorporating an interesting technique developed by Giunta et al [1995] termed “variable-complexity modelling”, where simple computationally inexpensive analysis techniques are used together with more detailed, expensive techniques in the design optimisation process. The integration of FEA representation and concurrent processing of simple evaluation models alongside complex analyses would also yield greater savings in computational expense if applied to the plate optimisation problem. Parallelisation of the problem would further reduce the computational expense.

When dealing with low dimensional problems a number of adaptive search algorithms which can perform the task of optimisation may perform relatively well. However if the problem is complex (i.e. high dimensional, multi-modal, constrained and multi-objective) and computationally expensive analysis is required, the more sophisticated evolutionary adaptive strategies such as those outlined in this thesis will likely be required. Although parallel processing as outlined in chapter 5 initially appears to be a good solution to the problem of computational expense it is extremely important to develop and optimise the underlying algorithms with respect to their efficiency and accuracy as shown in this thesis.

The strategies outlined in this thesis have not only proved to be efficient and robust when tackling high dimensionality, multi-modality and sensitivity, but have also reduced overall computational expense. The results show the strategies and techniques to be highly capable of locating high performance solutions. The incorporation of these strategies into industrial

design practice and the resulting mass production of the designs being produced by the collaborating company shows the potential of the algorithmic structures presented here in a real world design and manufacture environment.

References

- Adamidis and Petridis (1996). "Co-operating populations with different evolution behaviours", *Proceedings of the IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, pp. 188-191.
- Adeli H., and Cheng N.T. (1995). "Augmented Lagrangian Genetic Algorithm for Structural Optimisation", *ASCE Journal of Aerospace Engineering*, Vol. 8, No. 3, pp 156-163.
- Angeline P. J. (1993). "Evolutionary Algorithms and Emergent Intelligence" PhD Thesis, Ohio State University, Columbus.
- Baeck T. (1992). "Self-adaptation in genetic algorithms". In *First European Conference on Artificial Life*.
- Baeck T., Hammel U. and Schwefel H. (1997). *Evolutionary Computation: Comments on the History and Current State*, *IEEE Transactions on Evolutionary Computation*, Vol 1, No. 1, pp 3-17.
- Balachandran M. (1993). "Knowledge-Based Optimum Design" *Topics in Engineering*, Vol. 10, Hobbs the Printers Ltd.
- Baluja S. (1994). *Population Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimisation and Competitive Learning*. Tech. Rep, School of Computer Science, Carnegie Mellon University, Pittsburgh, CMU-CS-94-194.
- Baluja. S. (1996). "An empirical Comparison of Seven Iterative and Evolutionary Function Optimisation Heuristics", Technical Report. CMU-CS-95-193. Pittsburgh, PA, Carnegie Mellon University.
- Baluja S. And Caruana R. (1995). "Removing the Genetics from the Standard Genetic Algorithm" Technical Report No. CMU_CS_95_141, Pittsburgh, Pennsylvania: Carnegie Mellon University.
- Baluja S. and Davies S. (1997).. "Using Optimal Dependency-trees for Combinatorial Optimisation : Learning the Structure of the Search Space", Technical Report : CMU- CS-97-107, Pittsburgh, PA, Carnegie Mellon University.
- Barbosa H.J.C. (1997). "A Coevolutionary genetic algorithm for a game approach to structural optimisation", *Proceedings of the Seventh International Conference on Genetic Algorithms*. Michigan State University, East Lansing, Thomas Back (Ed), pp 545- 552.
- Beightler C. S., Phillips D.T. and Wilde D.J. (1979), *Foundations of Optimisation* (2nd edition), Englewood Cliffs, NJ, Prentice-Hall.
- Bendsoe M.and Kikuchi (1988). "Generating Optimal Topologies in Structural Design Using a Homogenization Method", *Journal of Computer Methods in Applied Mechanics and Engineering*, 71,pp 197-224.

Bilchev G. and Parmee I.C. (1996) "Constraint Handling for the Fault Coverage Code Generation Problem: An Inductive Evolutionary Approach", Lecture Notes in Computer Science, The 4th International Conference on Parallel Problem Solving from Nature, Berlin, Springer, pp 880-889.

Birkenhead D. (1997). "Genetic Programming for Shape Definition and Optimisation", MSc Dissertation, University of Plymouth, Plymouth.

Box G. E. P. (1957). Evolutionary Operation: A Method for Increasing Productivity. Journal of the Royal Statistical Society, vol. 6 No. 2 pp 81-101

Cartwright. H. M. and Harris S. P. (1993). "Analysis of the Distribution of Airborne Pollution using Genetic Algorithms", Journal of Atmospheric Environment, Vol. 27A, No. 12, pp. 1783-1791, Pergamon Press Ltd.

Cai J. and Thierauf G. (1996). "Structural Optimisation of a Steel Transmission Tower by using Parallel Evolutionary Strategy", Proceedings of the second International Conference on Adaptive Computing in Engineering Design and Control - '96', PEDC, Plymouth, pp 18-25.

Chapman C.D., Saitou K., Jakiela M.J.(1993). "Genetic Algorithms as an approach to Configuration and Topology Design", Proceedings of the 1993 Design Automation Conference, DE-Vol. 65-1, Published by the American Society of Mechanical Engineers, Albuquerque, New Mexico, pps. 485-498.

Coello Coello C. (1998). "Two new approaches to Multiobjective Optimisation using Genetic Algorithms", Proceedings of the Adaptive Computing in Design and Manufacture, Ed. Parmee I. C., Springer, pp. 151-160

Darwin C. (1859). on the Origin of Species by Means of Natural Selection. John Murray.

Dasgupta D. and MacGregor (1991). "A Structured Genetic Algorithm" Research Report IKBS - 2-91, University of Strathclyde, UK.

Davis L. (1989). "Adapting Operator Probabilities in Genetic Algorithms", Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufman, New York.

Davis L. (1991). (Ed), Handbook of Genetic Algorithms, New York, Van Nostrand Reinhold.

De Bonet. J.S., Isbell C. L., & Viola P. (1997). "Mimic: Finding Optima by Estimating Probability Densities", In M. Jordan, M. Mozer & M. Perrone (Eds.) Advances in Neural Information Processing Systems, Vol. 9, Cambridge, MA, MIT Press, pp. 424-430.

De Jong K. (1975). An analysis of the behaviour of a class of Genetic Adaptive Systems, University of Michigan.

Deb K. and Goldberg D. E. (1989). "An Investigation of Niche and Species Formation in Genetic Function Optimisation" Proceedings of Fifth ICGA and their Applications, Morgan Kaufmann, pp 42-50

- Deb K. and Goyal M. (1997). "Optimizing Engineering Designs Using a combined Genetic Search", Proceedings of the Seventh International Conference on Genetic Algorithms, Michigan State University, East Lansing, MI, pp. 521-528.
- Dhringa A. And Lee B.H. (1994). A Genetic Algorithm Approach to Single and Multiobjective Structural Optimisation with Discrete-Continuous Variables, International Journal for Numerical Methods in Engineering, Vol. 37, 1994, pp 4059-4080.
- Doorly D. J., Peiro J., Kuan T. and Oesterle J. P. (1996). Optimisation of Airfoils using Parallel Genetic Algorithms, Proceedings of the 15th AIAA International Conference on Numerical Methods in Fluid Mechanics, June 96, Monterey, CA, USA.
- Dym C. L. and Lewitt R. E.(1991). Knowledge Based Systems in Engineering. McGrawHill Inc. (Civil Engineering Series)
- Ellman T., Keane J., Schwabacher M. and Ke-Thia Y.(1996). "Multi-Level Modelling for Engineering Design Optimisation", Department of Computer Science, Hill Centre for Mathematical Sciences, Rutgers University, Internal Report Number HPCD-TR-44.
- Eshelman L. J. (1991). "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Non-traditional Genetic Recombination". In G.J.E Rawlins (editor), Foundations of Genetic Algorithms and Classifier Systems. Morgan Kaufmann, San Mateo, CA.
- Eshelman L. J. and Shaffer J. D. (1991). "Preventing Premature Convergence in Genetic Algorithms by Preventing Incest". Proc. of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA.
- Fogel L. J., Owens A. J., and Walsh M.J. (1966). Artificial Intelligence Through Simulated Evolution. John Wiley.
- Fogel D.B. (1995). Evolutionary Computation : Towards a New Philosophy of Machine Intelligence. IEEE Press.
- Fogerty T.C. (1989). "Varying the Probability of Mutation in the Genetic Algorithm", Proceedings of the Third International Conference on Genetic Algorithms", San Mateo, CA, Morgan Kaufmann, pp. 104-109.
- Fonseca C.M. and Fleming P.J. (1993). An overview of Evolutionary Algorithms in Multiobjective Optimisation, Evolutionary Computation 3, 1-16.
- French M. J. (1985). Conceptual Design for Engineers, 2nd edition, The Design Council Books -london Springer Verlag.
- French M.J. (1994). Invention and Evolution: Design in Nature and Engineering, 2nd Edition, Cambridge University Press.
- Gero J.S. (1993). Towards a Model of Exploration in Computer-Aided Design, Proceedings of the workshop on Formal Design Methods for Computer-Aided Design, Tallinn, Estonia, eds John S. Gero and Fay Sudweeks, pp. 271-291.

Giunta A.A., Balabanov V., Burgee S., Grossman B., Haftka R.T., Mason W.H. and Watson L.T. (1995). "Variable- Complexity Multidisciplinary Design Optimisation Using Parallel Computers". International Conference on Computational Engineering Science, Hawaii, USA.

Goldberg D. (1989). "Genetic Algorithms in Search, Optimisation and Machine Learning", Addison - Wesley Publishing Company, Inc.

Goldberg D. E. and Samtani M. P. (1986). Engineering Optimisation via Genetic Algorithms, Proceedings of the 9th Conference on Electronic Computation, ASCE, New York, pp 471-482.

Goldberg D. E. and Richardson J. (1987). "Genetic Algorithms with Sharing for Multimodal Function Optimisation". Genetic Algorithms and their Applications : Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, pp 41-49.

Goldberg D. E, Deb K. and Korb B. (1991). "Don't Worry; Be Messy" Proceedings of the Fourth International Conference on genetic Algorithms, Ed's, Belew R & Booker I. B, Morgan Kaufmann Publishers, pp 24-30.

Goodman E.D. Averill R. C., Punch W.F., Ding Y., Mallot B. (1996). "Design of Special-Purpose Composite Material Plates Via Genetic Algorithms". Proc. of the Second International Conference on Adaptive Computing in Engineering Design and Control, ed. I.C Parmee, University of Plymouth.

Goodman E., Eby D., Averill R.C., Gelfand F., Punch W.F., Mathews O. (1997). "An Injection Island GA for Flywheel Optimisation", Proceedings of the European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, Sept '97, pp. 687-691.

Greene J.R. (1996). Population-Based Incremental Learning as a Simple Versatile Tool for Engineering Optimisation, First International Conference on Evolutionary Computation and its Applications, Moscow, Russia, pp. 258-269.

Grefenstette J. (1986). "Optimisation of Control Parameters for Genetic Algorithms", IEEE Transactions on System, Man, and Cybernetics, SMC-16(1).

Haftka R.T.(1986). Structural Shape Optimization - A Survey. Computer Methods in Applied Mechanics and Engineering 57, Elsevier Science Publishers B.V.

Haftka R. and Kamat M.(1985). Elements of Structural Optimisation. Eds. Haftka, R. and Kamat, M. Martinus Nijhoff Publishers, Dordrecht.

Haftka R., Le Riche R. and Harrison P. (1996). "Genetic Algorithms for the Design of Composite Panels", Emergent Computing Methods in Engineering Design - Applications of Genetic Algorithms and Neural Networks, Eds. D. Grierson and P. Hajela. Nato ASI Series Springer, pp 10-29.

Hajela P., Lee E., Lin C-Y.(1992). "Genetic Algorithms in Structural Topology Design", Proceedings of the NATO Advanced Research Workshop on Topology Design of Structures, eds., MP. Bendsoe and C.A. Mora Soares, Kluwer Academic.

- Harik G.R., Lobo G.F., Goldberg D.E.(1997). "The Compact Genetic Algorithm", IlliGAL Report No. 97006, Dept. of General Eng. University of Illinois at Urbana - Champaign, Urbana, Aug '97.
- Holland J.H. (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor.
- Jenkins W.M.(1991). Structural Optimisation with the Genetic Algorithm, *The Structural Engineer*, Vol. 69, No. 24, Dec '91, pp 418-422.
- Jenkins W.M. (1992). A Plane Frame Optimum Design Environment based on the Genetic Algorithm, *Journal of Structural Engineering*, Proceedings of the American Society of Civil Engineers, Vol. 118, No. 11, pp. 3103-3112.
- Jenkins W.M.(1994). A Space Condensation Heuristic for Combinatorial Optimisation. *Advances in Structural Optimisation*, Civil Comp Press Ltd, Edinburgh.
- Jensen E. (1992). *Topological Structural Design Using Genetic Algorithms*. PhD Thesis, Perdue University, November '92.
- Kane C., Schoenauer M. (1996) "Topological Optimum Design using Genetic Algorithms", *Journal of Control and Cybernetics*, Vol. 25, No. 5.
- Keane A.and Brown (1996). "The design of a Satellite Boom with Enhanced Vibration Performance using Genetic Algorithm Techniques", Proceedings of the second International Conference on Adaptive Computing in Engineering Design and Control , PEDC, Plymouth, March '96, pp. 107-113.
- Kogiso N., Watson L.T., Gurdal Z. and Haftka R.T. (1994). Genetic Algorithms with Local Improvement for Composite Laminate Design, *Structural Optimisation*, Vol. 7, No. 4, pp. 207-218.
- Kohli H. S.and Carey G.F. (1993). "Shape Optimisation Using Adaptive Shape Refinement". *Int. Journal for Numerical Methods in Engineering*, Vol. 36 pp- 2435-2451.
- Koumoussis V.K. and Georgiou, P.G. (1994). "Genetic Algorithms in Discrete Optimisation of Steel Truss Roofs", *ASCE - Journal of Computing in Civil Engineering*, Vol. 8, No. 3, July'94.
- Koza J. R. (1992). "Genetic Programming - on the Programming of Computers by Means of Natural Selection". The MIT Press.
- Le Riche R. and Haftka R.T. (1994). Improved Genetic Algorithm for Minimum Thickness Composite Laminate Design. *Proceedings of International Conference on Composite Engineering*, New Orleans, LA
- Leite J. P. (1996). PhD thesis in *Parallel Adaptive Search Techniques for Structural Optimisation*" Heriot-Watt University, Edinburgh.
- Maher M. L. and Poon J. (1995). Modelling design exploration as co-evolution, *Microcomputers in Civil Engineering*, Volume11, Part3, pp. 195-209.

- Manderick B. and Spiessens P. (1989). "Fine-Grained Parallel Genetic Algorithms", Proc. Third ICGA, Jun '89, pp. 428-433.
- Mauldin M. (1984). "Maintaining Diversity in Genetic Search" International Conference on Artificial Intelligence, pp 247-520.
- Michalewicz Z. (1994). Genetic Algorithms + Data Structure = Evolution Programs, 2nd ed., Springer Verlag, New York.
- Miles J. and Moore C. J.(1997). "Innovative Computer Systems For Designers", Innovation in Civil and Construction Engineering, Ed. M. B. Leeming and B.H.V Topping, Civil-Comp Press, pp. 271- 279.
- Mill F., Warrington S.W., Smith R., Sherlock A. (1996) "Shape and Topology Optimisation in Engineering Design with Genetic Algorithms", Proc. of the Second International Conference on Adaptive Computing in Engineering Design and Control, ed. I.C Parmee, University of Plymouth, pp. 270-276.
- Mingra A.K (1986). "Genetic Algorithms in Aerospace Design", AIAA Southeastern Regional Student Conference, Huntsville, AL,USA.
- Muhlenbein H. and Schlierkamp-Voosen. D. (1993). Predictive Models for the Breeder Genetic Algorithms. Journal of Evolutionary Computation. Vol. 1, Part 1. pp. 25-49.
- Nagendra S., Jestin D., Gurdal Z., Haftka R.T., and Watson L.T. (1994). Improved Genetic Algorithm for the Design of Stiffened Composite Panels, 15th International Symposium on Mathematical Programming, Ann Arbor.
- Pahl G. and Beitz W. (1984). Engineering Design, 1984, Design Council Books, London, Springer Verlag.
- Papalambros P. & Wilde D. (1988). Principles of Optimal Design, Cambridge University Press, New York.
- Parmee I.C. (1993). The Concrete Arch Dam - An Evolutionary Model of the Design Process, Proceedings of the International Conference on Neural Nets and Genetic Algorithms, Innsbruck, Austria, Springer-Verlag Wein.
- Parmee, I.C. (Ed.) (1994). Proceedings of the First International Conference on Adaptive Computing in Engineering Design and Control, PEDC, Plymouth, Sept. '94.
- Parmee I. C. and Denham M. J. (1994) "The Integration of Adaptive Search Techniques with Current Engineering Design Practice", Proceedings of Adaptive Computing in Engineering Design and Control, Ed. I.C Parmee, Sept '94, pp. 1-13
- Parmee, I. C. (1996a). "The Maintenance of Search Diversity for Effective Design Space Decomposition using Cluster-Oriented Genetic Algorithms (COGA's) and Multi-Agent Strategies (GAANT), Proceedings of the International Conference on Adaptive Computing in Engineering Design and Control", University of Plymouth, Plymouth, ed. Parmee I. C., pp. 128-138.

- Parmee I. C. (Ed.) (1996b). Proceedings of the second International Conference on Adaptive Computing in Engineering Design and Control, PEDC, Plymouth, March '96.
- Parmee I. C. and Vekeria. H. D. (1997). "Co-operative Evolutionary Strategies for Single Component Design". Proc. of the Seventh International Conference on Genetic Algorithms, Michigan State University, Michigan.
- Parmee I.C., Vekeria H. D., Bilchev G. (1997). "The Role of Evolutionary and Adaptive Search During Whole System, Constrained and Detailed Design". Journal of Engineering Optimisation, Vol. 29, Gordon & Breach Science Publishers, pp. 151-176.
- Poloni C., Fearon M., Ng D. (1996). Parallelisation of Genetic Algorithm for Aerodynamic Design Optimisation, Proceedings of ACEDC 96, University of Plymouth, pp 59-64.
- Potter M. A. and De Jong K.A. (1994). A cooperative coevolutionary approach to function optimisation. Proceedings of the Third Parallel Problem Solving From Nature, Jerusalem, Israel, pp-249-2257, Springer-Verlag, pp 249-257.
- Punch W., Goodman E. Pei M., Chai-Shun L., Hovland P., Enbody R. (1993). "Further Research on Feature Selection and Classification Using Genetic Algorithms", Proceedings of the Fifth ICGA, June '93, pp. 557-564.
- Rachenberg I. (1973) Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart.
- Rajeev S. and Krishnamoorthy (1992). Discrete Optimisation of Structures Using Genetic Algorithms, Journal of Structural Engineering, Vol. 118, No. 5, pp 1233-1250.
- Rao S. (1984). "Multiobjective. Optimisation in Structural Design with Uncertain Parameters and Stochastic Processes", American Institute of Aeronautics and Astronautics Journal, Vol. 22, No. 11, Nov '84, PP. 1670-1678.
- Rao S., Sundararaju K., Prakash B.G., and Balakrishna C. (1992). "Multiobjective Fuzzy Optimisation Techniques for Engineering Design", Computers and Structures, Vol. 42, No. 1, Pergamon Press plc, pp. 37-44.
- Schwefel, H.P. (1975). Evolutionstrategie and numerische Optimierung. Dissertation, Technische Universität, Berlin.
- Seront G., and Bersini H. (1996). "Simplex GA and Hybrid Methods", Proceedings of the IEEE International Conference on Evolutionary Computation", May '96, Nagoya, pp. 845-848.
- Shaffer D. and Morishima A. (1987). An Adaptive Crossover Distribution Mechanism for Genetic Algorithms", Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Assoc. New York.
- Soremekun G., Gurdal Z., Haftka T., Watson L.T. (1996). Improving Genetic Algorithm Efficiency and Reliability in the Design and Optimisation of Composite Structures. Proceedings of the Sixth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, Bellevue, W.A. Sept. 4-6, 1996, pp. 372-383.

- Syswerda G. (1989). Uniform crossover in genetic algorithms. In ICGA3.
- Tenese R. (1989). "Distributed Genetic Algorithms" Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, San Mateo, California, June '89 pp 434-439.
- Vekeria H. D and Parmee I. C. (1996). "The Use of a Co-operative Multi-Level CHC GA for Structural Shape Optimisation", Proceedings of the European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, Sept'96.
- Vekeria H. D., Parmee I. C. (1997). "Reducing Computational Expense Associated With Evolutionary Detailed Design". Proc. of International Conference on Evolutionary Computation, University of Purdue, Indianapolis, 1997.
- Whitley D. L. (1989). The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best. In Schaffer.J, editor, Proceeding of the third International Conference on Genetic Algorithms, Pages 116-121, San Mateo, Morgan Kaufmann.
- Whitley D. L. (1991). "Fundamental Principles of Deception in Genetic Search" Foundations of Genetic Algorithms, Morgan Kaufmann Publishers Inc., pp 221-241
- Wolpert D.H. and Macready W.G. (1995). "No Free Lunch Theorems For Search", Technical Report, Santa Fe Institute, Santa Fe, New Mexico.
- Yamazaki K. (1996). Two Level Optimisation Technique of Composite Laminate Panels by Genetic Algorithms. AIAA Paper 96-1539-CP.
- Zhao L., Tsujimura Y., Gen M. (1996). "Genetic Algorithm for Fuzzy Clustering". Proceedings of the IEEE International Conference on Evolutionary Computation, Nagoya, Japan, May '96, pp. 716-719.

PUBLICATIONS

**INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 97 TAMPERE, AUGUST 19-21, 1997**

**EVOLUTIONARY SEARCH FOR HIGHLY DIMENSIONAL
ENGINEERING DESIGN PROBLEMS**

H.D Vekeria and I.C Parmee

Keywords : Evolutionary Search, Injection Island GA, Detailed Design

1 Introduction

Stochastic search and optimisation methods which model natural evolutionary processes are receiving considerable interest in Engineering Design due to their wide range of applications to problems [Parmee, Denham and Roberts 1993] This paper presents a comparison of Evolutionary / Adaptive Search methods for the reduction in computational expense associated with the optimisation of highly dimensional structural design problems. Complex analysis packages such as Finite Element Analysis (FEA) can lead to excessive computational expense when utilised as an evaluation function. There is a need therefore to minimise the number of calls to the fitness function. Distributed, co-operative injection island strategies are presented which allow dynamic refinement of component representation. This paper shows that by utilising multi-level Genetic Algorithm (GA) architectures, dramatic improvements in design performance may be gained whilst significantly reducing the overall number of evaluations in comparison with single level representations.

2 The Evaluation Model

The design domain involves a real-world problem concerning the optimal material distribution on the underside of a flat concrete plate with varying load and support conditions. The plate is represented in a grid type manner being divided into rectangular or square elements each with variable depth. However, if required, a set number of elements may be considered as one variable to promote uniformity in depth. The overall plate dimensions are 200mm x 200mm. In order to achieve a certain degree of symmetry for ease of manufacture, neighbouring elements whose angles exceed a preset aspect ratio (the ratio describing relative depth at the element interfaces) are penalised. In order to allow extensive experimental work, the GA has been integrated with a simple mathematical model utilising bending moment and complex stress analysis to ensure computational cost is kept to a minimum. Principal stress (σ_p) is calculated using the following formula:

$$\sigma_{1or2} = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2} \quad \sigma_{x.or.y} = \text{Direct Stress} \quad \tau = \text{Shear Stress}$$

The fitness of the design relates to the level of stress violation and the overall weight of the plate i.e. weight must be minimised within given stress criteria. The fixed parameters of concrete are: flexural limit = 9MPa, Density = 2.2 g/cc. Theoretical direct stresses in both the X and Y planes are increased by a factor of 1.18 to account for errors incurred in applying simple beam theory. Designs exhibiting a high degree of stress violation are penalised to ensure that the generated designs satisfy relevant criteria. Although preliminary design solutions for the flat plate problem can be achieved with a relatively small number of variable elements (15 to 50) in excess of 300 elements are required during detailed design to ensure accurate stress evaluation for a number of support and load conditions.

3 Initial Results

Initial results using a simple, canonical GA [Goldberg 1989] with various parameter settings were disappointing with severe degradation of the convergence characteristics with an increase in dimensionality i.e. variable element number. Due to the perceived sensitivity and the very high number of local optima, the processing capabilities of the simple, canonical GA are not appropriate for this class of problem. Subsequent integration of a breeder GA (BGA) [Muhlenbein, Schlierkamp-Voosen 1993], Population-based Incremental Learning (PBIL) [Baluja 1994] and the CHC GA [Eshelman 1991] resulted in significant improvements as shown in figure 1 although performance degradation is still evident with increasing dimensionality. A brief overview of the different forms of evolutionary algorithms is given below.

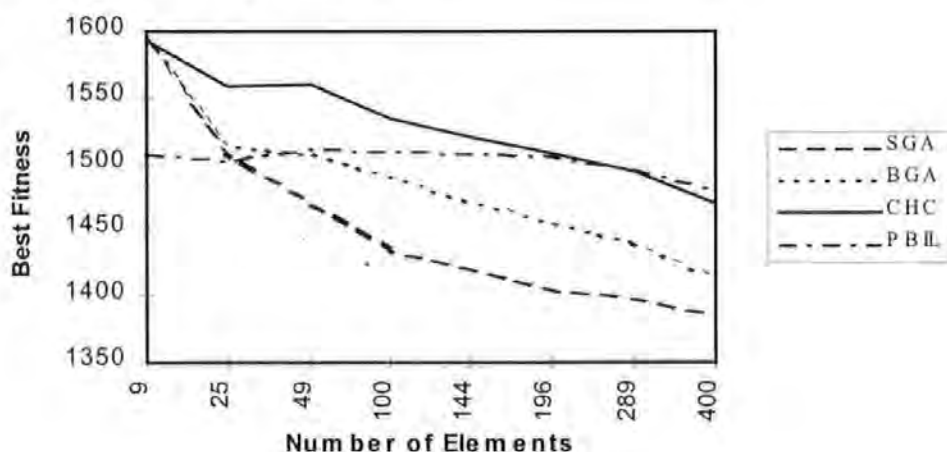


Figure 1 - Graph to show a comparison in performance between different GA's (single load case)

3.1 The CHC Genetic Algorithm

The CHC GA differs from the simple, canonical GA in a number of respects :

- It is highly elitist biased. Instead of replacing the old parent population with the child population, competition for survival is cross generational, the child population must compete with the parent population for survival.
- It maintains diversity through incest prevention. Individuals are randomly paired for mating, bias is introduced against mating individuals who are similar. Individuals are only crossed if their Hamming distance exceeds the difference threshold.

- It utilises a highly disruptive crossover. Half of the differing bits are swapped at random. This promotes diversity by producing children that are different from both parents.
- Mutation is only introduced when the population has converged or the search has stagnated.

3.2 Population-Based Incremental Learning (PBIL)

Population-Based Incremental Learning (PBIL) is a combination of evolutionary optimisation and hill climbing. The object of the algorithm is to create a real valued probability vector which, when sampled, reveals high quality solution vectors with high probability. Initially the values of the probability vector are set to 0.5. Sampling from this vector yields random solution vectors as the probability of generating a 1 or 0 is equal. As search progresses, the values in the probability vector gradually shift to represent highly fit solution vectors. The distance the probability is pushed (towards either 0.0 or 1.0) depends upon the learning rate parameter. After the probability vector is updated, a new set of solution vectors is produced by sampling from the updated probability vector and the cycle is continued.

3.3 The Breeder Genetic Algorithm (BGA)

The Breeder Genetic Algorithm (BGA) is based on artificial selection similar to that used by human breeders. The BGA is a combination of evolution strategies (ES) and genetic algorithms (GA). The BGA uses a selection scheme called truncation selection. The T% of the best individuals are selected and mated randomly until the number of offspring is equal to the size of the population. The BGA uses one of a number of recombination and mutation operators.

4 Dynamic Shape Refinement (DSR)

The DSR technique based on Adaptive Shape Refinement (ASR) [Kohli and Carey 1993] utilises problem representations of varying resolution. A sequential evolutionary process utilising the CHC algorithm (with a population size of 40, a divergence rate of 30% and a maximum number of restarts of 3) commences upon a relatively coarse (in terms of number of elements) plate representation. As convergence is achieved so the best solution from this process is mapped onto a finer resolution elemental grid and a population based upon mild perturbation of this solution is established. The CHC then manipulates the finer representation until convergence is again achieved and the mapping procedure is repeated. This sequential evolution process continues, utilising finer representations until a satisfactory solution is identified. Such a satisfactory solution should not only be of minimum weight within relevant stress criteria but also be considered low-risk in terms of the final resolution of plate representation i.e. there is a sufficiently high number of elements to provide confidence in the stress evaluation. Figure 2 compares a DSR approach utilising the CHC manipulation of 5x5, 10x10 and 20x20 element representations to a CHC manipulation of a single 20x20 representation. The results have been averaged over twenty runs of the algorithms.

The high fitness achieved during the early stages of the DSR approach must be treated with caution. Fitness is measured in terms of weight versus stress violation and the coarser representations although seemingly of high fitness are also high-risk due to the lack of

resolution during stress evaluation. A higher resolution stress evaluation returns a greater degree of violation and a related degradation of fitness as shown by the dips in the DSR curve as finer resolutions are introduced. The DSR achieves a significantly higher fitness than the single representation approach with far fewer calls to the analysis routine.

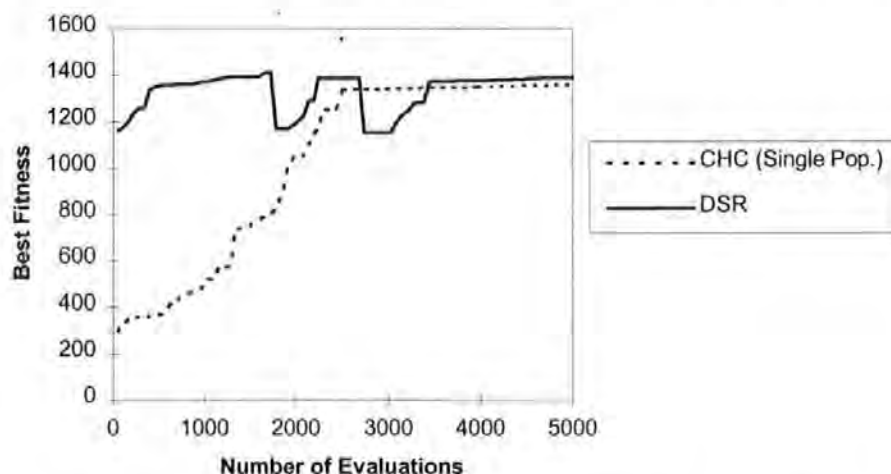


Figure 2 - Graph to show the performance of the single population GA against a GA utilising DSR (2 load cases).

5 The Dynamic Injection Island GA (DiiGA)

The Dynamic Injection Island GA (DiiGA) [Vekeria and Parmee 1997] is based on the injection island architecture (iiGA) [Punch, Averill, Goodman, Lin and Ding 1995]. The DiiGA utilises a concurrent shape refinement process. The flat plate is represented by a number of different resolution grids, each evolving upon a separate island. The solutions of the coarse design representations are injected into the more detailed designs for fine grained refinement. As a lower resolution process ceases to inject useful information into the higher resolution processes so it is removed and replaced by a resolution that is higher than any currently in existence. Migration of information is from low to high resolution at a set number of evaluations which requires translation of the differing grid size to maintain true representations. Migration allows the passing of highly fit schemata by injecting the best individuals that have evolved from a proportionally smaller search space into higher resolution representations replacing the worst individuals present at that time. A more detailed discussion of this technique may be found in [Vekeria and Parmee 1997].

A simple 3 level representation is presented below:

Representations:

- | | |
|-----------------------|------------------------|
| a) 5x5=25 elements | c1) 20x20=400 element |
| b) 10x10=100 elements | c2) 20x20=400 elements |

Process:

- Commence co-evolution of representations a) and b).
- Migrate from a) to b) every n generations until a) converges and ceases to pass useful information to b)
- Remove a) and introduce c₁) using the best individual from b) to seed new population
- Migrate individuals from b) to c₁) every n generations until b) converges and ceases to pass useful information to c₁)

- Remove b) continue to evolve c_1). Introduce another co-evolving subpopulation (c_2), seeded from (c_1).

Individuals are prevented from migrating if a duplicate exists in the host subpopulation in order to maintain search diversity. Further migration only takes place if the individual is fitter than the least fittest individual in the host subpopulation. The run continues until its termination condition is met (when the subpopulations have converged, the maximum number of evaluations have been reached or the maximum number of reinitialisations has been achieved).

Figure 3 shows the performance of the DiiGA. The graph represents the fitness of the finest resolution grid (400 elements). There is a rapid increase in fitness due to the migration of highly fit individuals from the coarse representation and constant improvement is maintained, resulting in superior performance in terms of degree of stress violation, weight and material distribution in comparison with the single population CHC GA. After the rapid increase in fitness due to meeting the stress criteria the fitness increases gradually by reducing the overall weight of the plate. The plateau after 6000 evaluations shows that the subpopulation has reinitialised, in order to introduce more diversity. The final designs produced using the DiiGA technique were also on average 3 percent lighter than those produced using the iiGA [Vekeria and Parmee 1996].

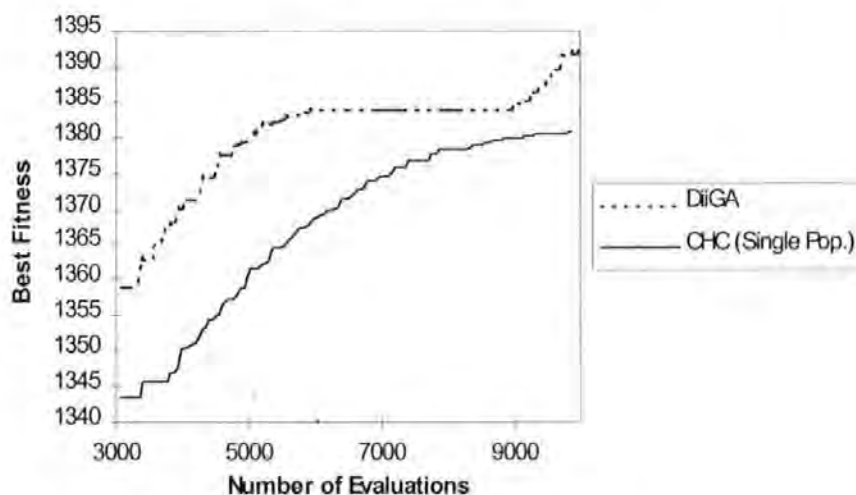


Figure 3 - Graph to show a comparison in performance between a CHC GA and a DiiGA (latter stages of search)

6 Conclusions

The DSR technique has shown that that a significant reduction in the number of calls to the model may be made during the latter more detailed stages of the design process, producing light weight, low risk design solutions. The DiiGA satisfies the initial objective of the research i.e. to converge upon a high performance design solution with a minimum number of function evaluations. The main advantage therefore in using injection island techniques is the reduction in computational expense and the ability to identify better design solutions when compared to single population GA's.

These initial results indicate a considerable potential for a significant reduction in the number of evaluation calls during evolutionary search. Refinement of the basic strategies

introduced here are likely to further reduce computational expense related to evaluation calls. In generic terms this will allow a more efficient integration with complex analysis techniques during detailed design and contribute significantly to those preliminary stages of the design process where a degree of complex analysis is required to validate results from more simplistic preliminary design models.

Acknowledgements

The research has been carried out at the Plymouth Engineering Design Centre, funded in the main by the UK Engineering and Physical Science Research Council (EPSRC). We wish to thank them for their continuing commitment to this research.

References:

- Baluja, S., "Population Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning", CMU-CS-94-194, Tech. Rep, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1994.
- Eshelman, L.J., "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination", In G.J.E Rawlins (editor), *Foundations of Genetic Algorithms and Classifier Systems*. Morgan Kaufmann, San Mateo, CA, 1991, pp. 265-283.
- Goldberg, D., "Genetic Algorithms in Search, Optimisation and Machine Learning", Addison - Wesley Publishing Company, Inc., Reading, MA., 1989.
- Goldberg, D., Deb, K., Korb, B., "Don't Worry, Be Messy", Proc. of the Forth International Conference on Genetic Algorithms. Ed. R.K Belew and L.B Booker, University of California, San Diego, July 1991, pp. 24-30.
- Kohli, H.S., Carey, K.S., "Shape Optimisation Using Adaptive Shape Refinement", Int. Journal for Numerical Methods in Engineering, Vol. 36, 1993, pp. 2435-2451.
- Muhlenbein H, Schlierkamp-Voosen D., "Predictive Models for the Breeder Genetic Algorithms", Journal of Evolutionary Computation. Vol. 1, Part 1, 1993, pp. 25-49.
- Parmee I.C, Denham M.J, Roberts A., "Evolutionary Engineering Design Using the Genetic Algorithm", Procs. International Conference on Engineering Design, The Hague, August 1993, pp. 1295-1302.
- Punch, W.F., Averill, R.C., Goodman, E.D, Lin, S., Ding, Y., "Using Genetic Algorithms to Design Laminated Composite Structures", Journal of IEEE Expert, Intelligent Systems and Their Applications, Feb 1995.
- Vekeria, H.D., Parmee, I.C., Vekeria H. D., Parmee I. C. "Reducing Computational Expense Associated With Evolutionary Detailed Design", Procs. International Conference on Evolutionary Computation, University of Purdue, Indiannapolis, 1997, pp. 391-396.

Harish Vekeria

Plymouth Engineering Design Centre,
University of Plymouth, Drake Circus,
Plymouth, Devon, PL4 8AA, UK.
Phone: +44(0)1752 233508
Fax: +44(0)1752 233529
Email: hvekera@plymouth.ac.uk

Ian Parmee

Plymouth Engineering Design Centre,
University of Plymouth, Drake Circus,
Plymouth, Devon, PL4 8AA, UK.
Phone: +44(0)1752 233509
Fax: +44(0)1752 233529
Email: iparmee@plymouth.ac.uk

Co-operative Evolutionary Strategies for Single Component Design

Ian C. Parmee

Plymouth Engineering Design Centre University
of Plymouth, Drake Circus Plymouth, Devon
PL4 8AA, UK
iparmee@plymouth.ac.uk

Harish D. Vekeria

Plymouth Engineering Design Centre University
of Plymouth, Drake Circus Plymouth, Devon
PL4 8AA, UK
hvekeria@plymouth.ac.uk

Abstract

The paper introduces the preliminary development of co-operative strategies that will enable the machine-based design of a single engineering component from initial configuration definition through to product realisation. The initial utilisation and comparison of basic evolutionary approaches leads to the introduction of high-performance evolutionary and adaptive search algorithms in order to improve performance within the high dimensional space that describes the component topology. A requirement for computationally expensive finite element analysis provokes the development of a sequential method for Dynamic Shape Refinement (DSR)[1] in an attempt to minimise calls to the fitness function and further improve solution performance. This leads to the utilisation of distributed, co-operative injection island strategies [2,3] and the development of strategies both for the dynamic refinement of component representation and the introduction / removal of differing search algorithms within the injection island architecture.

therefore to minimise the number of calls to the fitness function. This initially led to the development of a sequential method of shape refinement (DSR) where improvement is achieved sequentially by utilising increasingly refined representations of the plate and 'injecting' results from lower order representations to higher order.

Improvements gained in this manner have led on to the introduction of Michigan State University's Injection Island Architecture [2] and the achievement of significantly better designs with reduced calls to the models. Further improvement is achieved by introducing a dynamic refinement to the injection architecture where lower order plate representations are removed as they cease to contribute and are replaced by representations of a higher resolution than currently exists within the co-evolving processes [3]. Finally, initial investigation involving the utilisation of differing adaptive search algorithms integrated with the dynamic shape refinement is described and preliminary results are presented. This approach involves the use of two co-evolving adaptive search algorithms within an injection island architecture and their subsequent introduction / removal depending upon their relative performance

1 Introduction

The evolutionary design of a building component primarily consisting of a concrete flat plate is introduced. It is necessary for the plate to be represented by circa 400 elements of variable depth in order to provide accurate stress evaluation. The objective is to minimise the weight of the plate whilst satisfying maximum stress requirements. This conflict of objectives plus the high dimensionality results in a highly sensitive optimisation problem with many local optima.

The utilisation of a number of evolutionary and adaptive algorithms manipulating simple models of the plate illustrates a degradation in performance as plate resolution (i.e. number of elements) is increased. Finite element analysis is required to achieve accurate stress analysis but this leads to excessive computational expense. There is a need

In all cases simple analysis techniques are utilised in the evaluation function to allow extensive experimentation at low computational expense. Finite element analysis is now being introduced into the co-evolutionary processes to allow concurrent evolution with appropriate communication between both simple and complex models of differing resolution. The overall objective of the research is to establish co-evolutionary processes with appropriate migration regimes that support the design of single components from preliminary through to detailed design and product realisation. There are two main objectives to the research: the first relates to the achievement of high-performance designs whereas the second concerns the minimisation of required function evaluations. It is essential that the second objective is achieved in order that computationally expensive analysis techniques can be realistically utilised.

2 The Evaluation Model

The design domain involves a real-world problem concerning the optimal material distribution on the underside of a flat concrete plate with varying load and support conditions. The plate is represented in a grid type manner being divided into rectangular or square elements each with variable depth. However, if required, a set number of elements may be considered as one variable to promote uniformity in depth. The overall plate dimensions are 200mm x 200mm. In order to achieve a certain degree of symmetry for ease of manufacture, neighboring elements whose angles exceed a preset aspect ratio (the ratio describing relative depth at the element interfaces) are penalised. In order to allow extensive experimental work, the GA has been integrated with a simple mathematical model utilising bending moment and complex stress analysis to ensure computational cost is kept to a minimum. Principal stress (σ_p) is calculated using the following formula:

$$\sigma_{1or2} = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2}$$

$\sigma_{x,or,y}$ = Direct Stress τ = Shear Stress

The fitness of the design relates to the level of stress violation and the overall weight of the plate i.e. weight must be minimised within given stress criteria. The fixed parameters of concrete are: flexural limit = 9MPa, Density = 2.2 g/cc. Theoretical direct stresses in both the X and Y planes are increased by a factor of 1.18 to account for errors incurred in applying simple beam theory. Designs exhibiting a high degree of stress violation are penalised to ensure that the generated designs satisfy relevant criteria. Although preliminary design solutions for the flat plate problem can be achieved with a relatively small number of variable elements (15 to 50) in excess of 300 elements are required during detailed design to ensure accurate stress evaluation for a number of support and load conditions.

3 Initial Results

Initial results using a simple, canonical GA with various parameter settings were disappointing with severe degradation of the convergence characteristics with an increase in dimensionality i.e. variable element number. Due to the perceived sensitivity and the very high number of local optima, the processing capabilities of the simple, canonical GA [9] are not appropriate for this class of problem. Subsequent integration of a breeder GA (BGA) [4], Population-based Incremental Learning (PBIL) [5] and the CHC GA [6] resulted in significant

improvements as shown in figure 1 although performance degradation is still evident with increasing dimensionality.

The messy genetic algorithm [7] was also considered and would seem best suited to this class of problem due to its ability to maintain good linkage between individual genes. However it was not included in the test suite because of the computational expense associated with its two evolutionary phases and dual loop structure. The computational expense of finite element evaluation currently necessitates an alternative approach. Although the CHC algorithm does not necessarily support the genetic correlation provided by the messy GA it offers the best performance of those algorithms tested in addition to its proved robustness across a wide range of standard test functions [6,8].

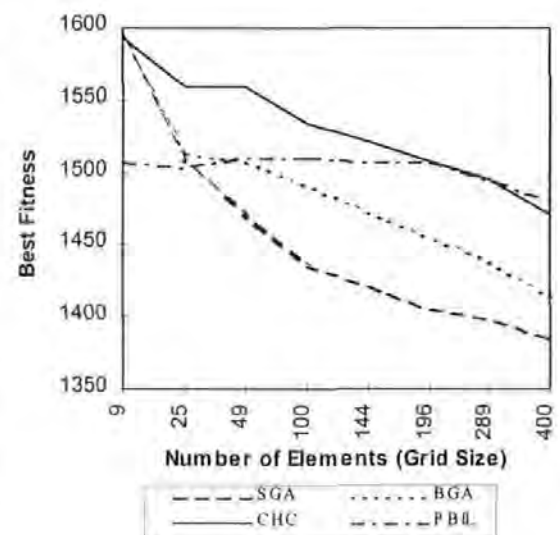


Figure 1 - Graph to show a comparison in performance between different GA's (single load case)

4 Dynamic Shape Refinement (DSR)

An initial alternative approach utilises problem representations of varying resolution. A sequential evolutionary process utilising the CHC algorithm (with a population size of 40, a divergence rate of 30% and a maximum number of restarts of 3) commences upon a relatively coarse (in terms of number of elements) plate representation. As convergence is achieved so the best solution from this process is mapped onto a finer resolution elemental grid and a population based upon mild perturbation of this solution is established. The CHC then manipulates the finer representation until convergence is again achieved and the mapping procedure is repeated [3]. This sequential evolution process continues, utilising finer representations until a satisfactory solution is identified. Such a satisfactory solution should not only be of minimum weight within relevant stress criteria but also be considered low-risk in terms of the final resolution

of plate representation i.e. there is a sufficiently high number of elements to provide confidence in the stress evaluation. Figure 2 compares a DSR approach utilising the CHC manipulation of 5x5, 10x10 and 20x20 element representations to a CHC manipulation of a single 20x20 representation. The results have been averaged over twenty runs of the algorithms.

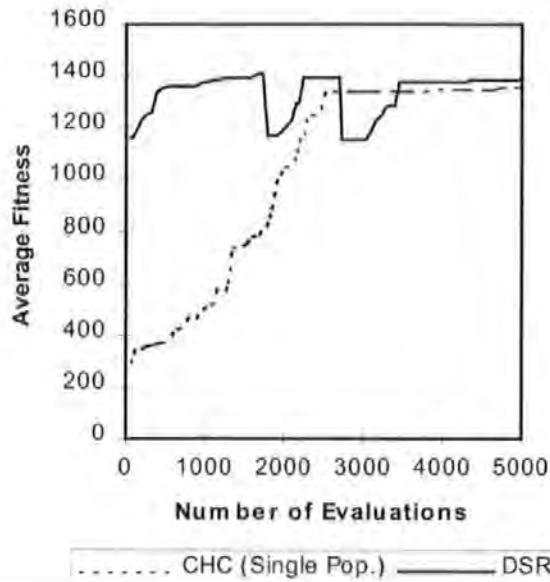


Figure 2 - Graph to show the performance of the single population GA against a GA utilising DSR (2 load cases).

The high fitness achieved during the early stages of the DSR approach must be treated with caution. Fitness is measured in terms of weight versus stress violation and the coarser representations although seemingly of high fitness are also high-risk due to the lack of resolution during stress evaluation. A higher resolution stress evaluation returns a greater degree of violation and a related degradation of fitness as shown by the dips in the DSR curve as finer resolutions are introduced. The CHC representation, being of a fine (20x20) resolution throughout the evolutionary process shows constantly improving fitness.

Comparison of the 20x20 representations of the two approaches reveals that DSR achieves a significantly higher fitness than the single representation approach with far fewer calls to the analysis routine. Therefore to some extent the DSR satisfies the primary objectives of the research i.e. minimum weight with minimum calls to the fitness function.

5 The Injection Island GA (iiGA)

Can we improve upon DSR by introducing a concurrent rather than sequential shape refinement process? The injection island architecture (iiGA) [2] offers this facility. The flat plate is represented by a number of different resolution grids, each evolving upon a separate island. Unlike the DSR method all

sub-populations are initiated at the same time. The solutions of the coarse design representations are injected into the more detailed designs for fine grained refinement. Migration of information is from low to high resolution at a set number of evaluations which requires translation of the differing grids to maintain true representations (figure 3).

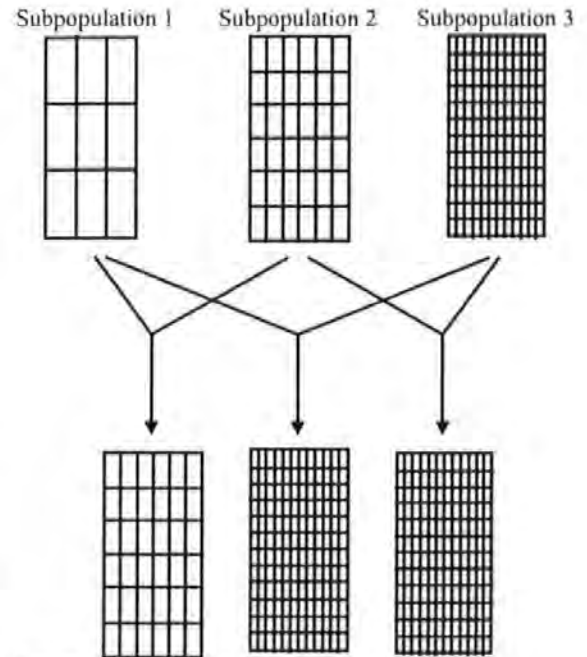


Figure 3 - Migration between Subpopulations

Migration allows the passing of highly fit schemata by injecting the best individuals that have evolved from a proportionally smaller search space into higher resolution representations replacing the worst individuals present at that time.

Figure 4 illustrates the effect of CHC integration with the iiGA architecture in terms of the average population fitness of a single population CHC GA of 60 chromosomes (400 elements) and an iiGA using 3 subpopulation islands (consisting of 25, 100 and 400 elements) of 20 chromosomes each. The curve displayed for the iiGA represents the finest subpopulation (400 elements). The number of evaluations is the summation of all evaluations of the subpopulations. Migration takes place every 100 evaluations. Rapid progress is apparent when compared with the single population CHC GA. A significant reduction in the number of evaluations to achieve similar fitness is apparent from the iiGA throughout the co-evolutionary process.

The single representation CHC approach will however eventually outperform the iiGA in terms of maximum fitness (figure 5). This is due to convergence of the lower resolution iiGA representations limiting the injection of useful material into the higher resolution populations which eventually results in a stagnation of the co-evolutionary process.

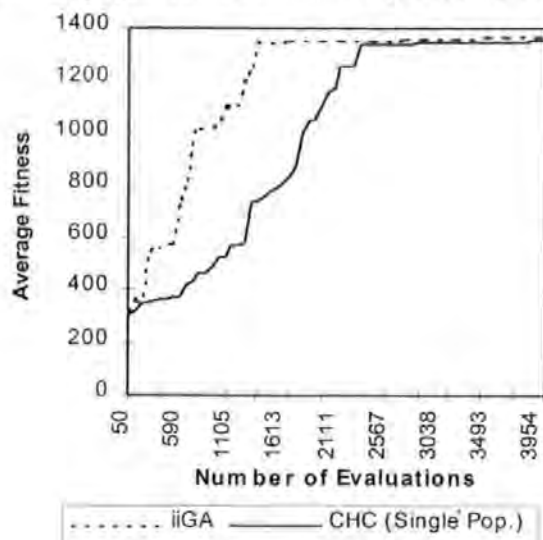


Figure 4 - Graph to show a comparison in performance between a single population CHC GA and an iiGA (early stages of the search for 2 load cases)

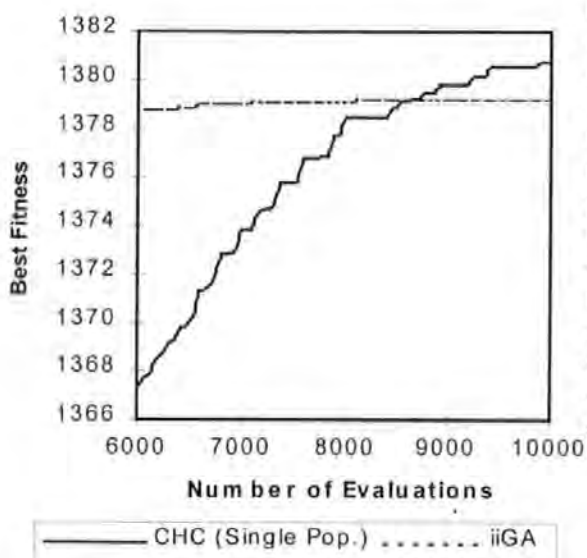


Figure 5 - Graph to show a comparison in performance between a CHC GA and a iiGA (latter stages of search for 2 load cases)

6 Dynamic Injection

The problem of process stagnation is addressed by introducing a dynamic injection (DiiGA) method of element representation. As a lower resolution process ceases to inject useful information into the higher resolution processes so it is removed and replaced by a resolution that is higher than any currently in existence. A simple 3 level representation involving 4 processes is presented below:

Representations:

- a) $5 \times 5 = 25$ elements
- b) $10 \times 10 = 100$ elements
- c1) $20 \times 20 = 400$ elements
- c2) $20 \times 20 = 400$ elements

Process:

- Commence co-evolution of representations a) and b).
- Migrate from a) to b) every n generations until a) converges and ceases to pass useful information to b)
- Remove a) and introduce c_1) using the best individual from b) to seed new population
- Migrate individuals from b) to c_1) every n generations until b) converges and ceases to pass useful information to c_1)
- Remove b) continue to evolve c_1). Introduce another co-evolving subpopulation (c_2), seeded from (c_1).

Individuals are prevented from migrating if a duplicate exists in the host subpopulation in order to maintain search diversity. Further migration only takes place if the individual is fitter than the least fittest individual in the host subpopulation. The run continues until its termination condition is met (when the subpopulations have converged, the maximum number of evaluations have been reached or the maximum number of reinitialisations has been achieved). It should be noted that there is no danger that the best individual will rapidly take over the new subpopulation. The CHC GA's incest preventing mechanism (the dropping difference threshold), in combination with elitist selection and disruptive recombination will prevent this. Eshelman found that partial reinitialisations perform better using smaller population sizes when compared with chronic mutation and provide many of the benefits of a large population without the cost of a slower search [1,2]. Initial results are shown in figure 6.

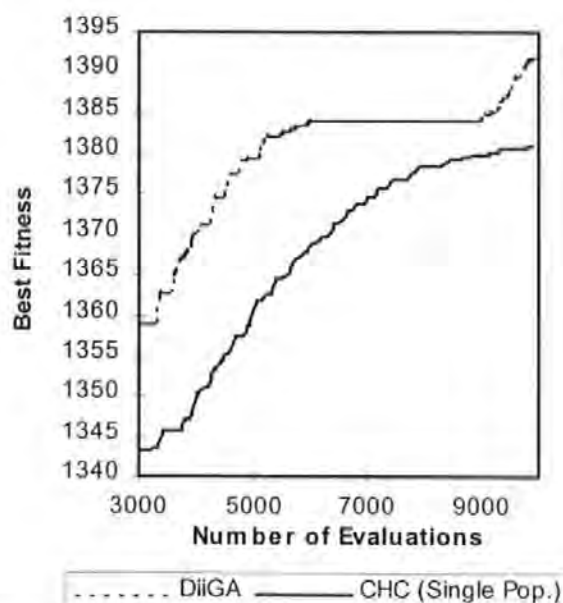


Figure 6 - Graph to show a comparison in performance between a CHC GA and a DiiGA (latter stages of search for 2 load cases)

7 Distributed Search Techniques

Trials per Iteration	40
Max. No. of Evaluations	10000

A final hypothesis is that further improvement in overall fitness and in the number of function evaluations may be possible by introducing different search techniques to the individual co-evolving processes. This is intuitively based upon the performance of the various adaptive search techniques of figure 1 and the behaviour of PBIL in particular. It is assumed that the initial poor relative performance of PBIL on the coarser resolution grids is due to premature convergence upon some local optima resulting from the implementation of a high learning rate.

It is interesting to note however that PBIL's performance significantly improves with increasing grid resolution suggesting that this tendency for premature convergence is offset by the sheer number of possible design directions available at higher dimensions. Whereas the more diverse search of the CHC begins to lose its way, PBIL manages to sustain a better compromise between exploration and exploitation and finally outperforms the CHC as the 400 element representation is approached. Another feature of PBIL is its rate of convergence during early generations with medium to high grid resolutions as shown in figure 7. This suggests that different techniques may be better suited to varying stages of the evolutionary process (although rapid convergence during the early stages may not prove beneficial in the longer term).

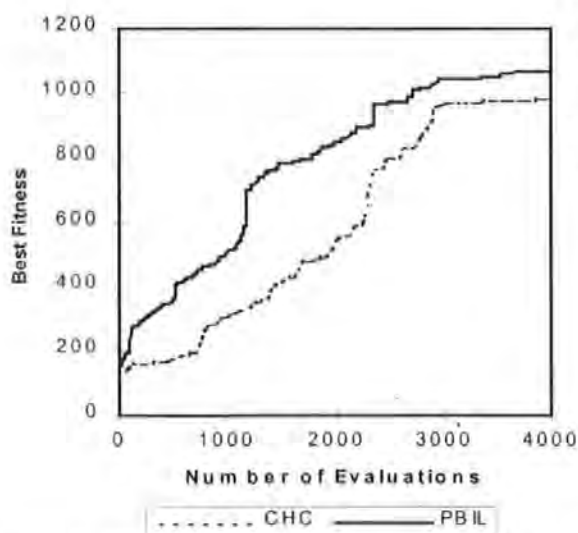


Figure 7 - Graph to show the initial rapid progress of PBIL in comparison to the CHC (3 load cases are utilised)

The following settings are used in relation to all PBIL runs.

Positive Learning Rate	0.1
Negative Learning Rate	0.1
Mutation Probability	0.02
Mutation Shift	0.05

The establishment of a distributed architecture supporting several search algorithms and their subsequent removal / re-introduction depending upon relative performance during the evolution process may provide a partial solution to the problem of selecting the most appropriate search technique for a particular problem. Preliminary experimentation here, however, concentrates upon the utility of this approach for the achievement of the two primary objectives of the flat plate problem i.e. minimal weight with minimum function evaluations. Two simple configurations are assessed. In the first the CHC algorithm manipulating a 5x5 grid representation co-evolves with a PBIL manipulation of a 10x10 representation. Migration is allowed every 200 evaluations with the better solutions from the CHC process updating the probability vector of the PBIL process. When the CHC ceases to provide sufficiently high-performance solutions for injection the process is killed and replaced by a second PBIL process manipulating a 20x20 grid representation. This continues to co-evolve with the lower resolution PBIL process receiving injected solutions every n generations. The reasoning here is that the more diverse search of the CHC which leads to higher performance on the coarser resolutions interacts with the more rapid convergence characteristics of PBIL to provide an optimal starting population for the final PBIL-based search. The objective is a higher-performance solution within a lesser number of function evaluations than would be attainable using the CHC alone within a DiiGA architecture. Results from a single load-case representation are shown in figure 8 and compared to the results from a three load-case representation (figure 9).

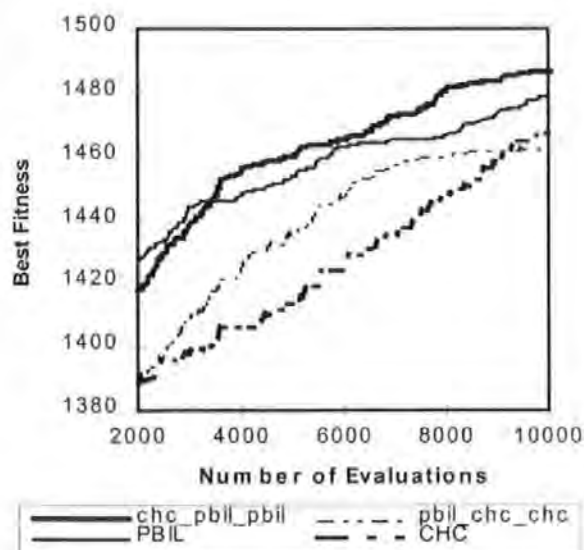


Figure 8 - Graph to show the performance of the different configurations for a 1 load case representation.

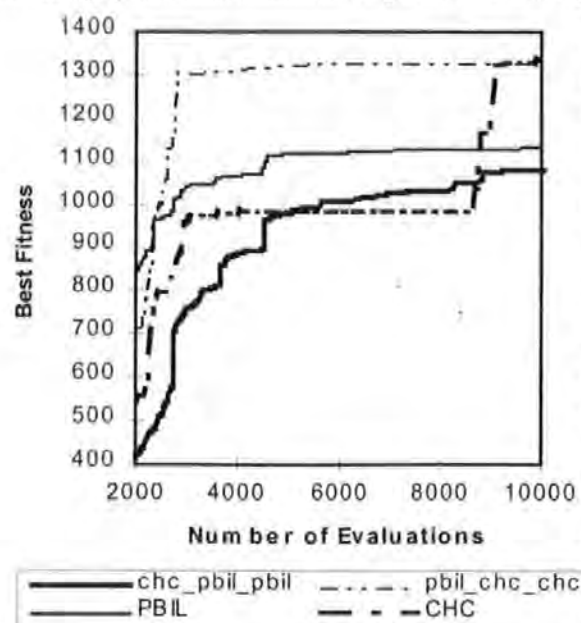


Figure 9 - Graph to show the performance of the different configurations for a 3 load case representation.

The second configuration involves a PBIL manipulation of the 5x5 grid co-evolving with a CHC manipulation of the 10x10 grid. The 5x5 PBIL process is killed as it ceases to pass useful information to the CHC process and at this point the CHC algorithm is replaced by PBIL which now manipulates the 10x10 resolution. A further 20x20 CHC process is introduced and co-evolves with the 10x10 PBIL representation. This strategy therefore investigates an alternative dynamic where PBIL injects locally high-performing solutions into the more diverse search processes of CHC.

The chc-pbil-pbil (c-p-p) co-evolution results in increased performance both in terms of reduced calls to the evaluation function and improved overall fitness in the single load case situation. However, as a more realistic three load case problem is introduced the cpp is very significantly out performed by the pbil-pbil-chc (p-p-c) co-evolution. Further experimentation is required to determine the reasons for these comparative performances upon differing problem representations. The single load case promotes the generation of material concentrations in one area of the plate and it is suggested that the convergence characteristics of the c-p-p are better suited to a less complex distribution of material upon the plate than that required by the three load case problem. With three load cases material is distributed across a wider area of the plate to best satisfy stress characteristics. It is assumed that the greater diversity of the later stages of p-c-c search results in the better identification of this more complex material distribution. The rapid convergence characteristics of pbil however, greatly accelerates this identification resulting in far less evaluation calls than is required by a DiiGA process utilising CHC alone. It is interesting to note,

however, that performance of the CHC alone finally equals that of the p-c-c co-evolution, whereas the characteristics of the c-p-p process results in rapid convergence upon a significantly lower performance solution.

8 Conclusions

The DSR technique has shown that that a significant reduction in the number of calls to the model may be made during the latter more detailed stages of the design process, producing light weight, low risk design solutions.

Results from the CHC GA utilising the iIGA architecture show dramatic improvements in the strength to weight ratio characteristics exhibited by the plate at significantly less computational cost than that required by a single population CHC GA. The CHC DiiGA achieves a significantly higher fitness overall whilst still maintaining the initial rapid improvements exhibited by the CHC iIGA. The DiiGA satisfies the two objectives of the research i.e. to converge upon a high performance design solution with a minimum number of function evaluations. The main advantage therefore in using injection island techniques is the reduction in computational expense in addition to the ability to identify better design solutions when compared to single population GA's.

More extensive experimentation is required to properly assess the utility of co-evolving processes involving several differing search algorithms. However the preliminary findings of section 7 indicate that :

- it is possible to improve performance both in terms of overall fitness and reduced evaluation calls.
- the selected search configurations are very sensitive to problem specifics e.g. the performance differences between one and three load case scenarios.

This second point may be addressed by improving the dynamics of the introduction / removal of individual search algorithms. A performance based scenario is envisaged whereby algorithms are removed / re-introduced dependent upon on-line measurement of their relative performance. This could result in the automatic selection of appropriate search configurations.

The integration of FEA representation and concurrent processing of simple evaluation models alongside complex analyses is now under investigation. This will allow the introduction of more high resolution levels to the DiiGA architecture. High resolution simple stress analyses will initially inject information into coarse resolution FEA representations before dying off and allow the process to move into a secondary detailed design

phase. The objective here is to achieve a continuous process from preliminary design of the plate through to final product realisation.

These initial results indicate a considerable potential for a significant reduction in the number of evaluation calls during evolutionary search. Refinement of the basic strategies introduced here are likely to further reduce computational expense related to evaluation calls. In generic terms this will allow a more efficient integration with complex analysis techniques during detailed design and contribute significantly to those preliminary stages of the design process where a degree of complex analysis is required to validate results from more simplistic preliminary design models.

Acknowledgments

The research has been carried out at the Plymouth Engineering Design Centre (PEDC), funded in the main by the UK Engineering and Physical Science Research Council (EPSRC). We wish to thank them for their continuing commitment to this research.

PEDC publications are available at :
<http://www.tech.plym.ac.uk/soc/research/edc/>

References

- [1] H.S Kohli, G.F Carey. "Shape Optimisation Using Adaptive Shape Refinement". *Int. Journal for Numerical Methods in Engineering*, Vol. 36 pp- 2435-2451, 1993.
- [2] E.D Goodman, R.C Averill, W.F Punch, Y. Ding, B Mallot. "Design of Special-Purpose Composite Material Plates Via Genetic Algorithms". *Proc. of the Second Int. Conf. on Adaptive Computing in Engineering Design and Control*, ed. I.C Parmee, University of Plymouth, 1996
- [3] H.D Vekeria, I.C Parmee. "The Use of a Co-operative Multi-Level CHC GA for Structural Shape Optimisation", *Procs. Forth European Congress on Intelligent Techniques and Soft Computing*, Aachen, 1996.
- [4] H. Muhlenbein, D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithms. *Journal of Evolutionary Computation*. Vol. 1, Part 1. pp. 25-49, 1993.
- [5] S. Baluja. Population Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Tech. Rep, School of Computer Science, Carnegie Mellon University, Pittsburgh, CMU-CS-94-194.
- [6] L.J Eshelman. "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination". In G.J.E Rawlins (editor), *Foundations of Genetic Algorithms and Classifier Systems*. Morgan Kaufmann, San Mateo, CA, 1991.
- [7] D.E Goldberg, K. Deb, B. Korb. "Don't Worry, Be Messy". *Proc. of the Forth International Conference on Genetic Algorithms*. Ed. R.K Belew and L.B Booker, University of California, San Diaego, July 1991.
- [8] L.J Eshelman, J.D Shaffer. "Preventing Premature Convergence in Genetic Algorithms by Preventing Incest". *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1991.
- [9] D. Goldberg. "Genetic Algorithms in Search, Optimisation and Machine Learning", Addison - Wesley Publishing Company, Inc., 1989.