Shape Optimisation using Traditional and Morphogenetic Evolutionary Algorithms:

Integrated Representation of Geometry and Physical Behaviour

Andrew Sherlock

Ph.D.

University of Edinburgh

2003

Declaration

I declare that this thesis has been composed by myself and is all my own work except where otherwise stated.

Acknowledgements

I would like to acknowledge and thank a number of people for their help in making this thesis possible.

Thanks to members of the Manufacturing Planning Group, who have over the years provided a stimulating environment in which to work, along with necessary encouragement, help and sometimes criticism, also to members, formerly, of the School of Mechanical Engineering, and latterly the School of Engineering and Electronics. Support from academics, support staff, postgraduates and researchers over the years has greatly helped my studies.

Thanks to Frank Mill for his (extended) supervision throughout my Ph.D. studies.

Thanks to my parents, Jim and Sheila Sherlock for their support, over the years of my studies, as well as their encouragement to finally write-up the thesis.

Finally, thanks to my wife, Jess Rogaly, for her tolerance, patience and support whilst writing this thesis, for her proof reading (although any errors are all my own) and for the constant cups of tea.

Abstract

The primary aim of the work described in this thesis was to develop novel techniques for shape optimisation that can effectively search through a large generality of shapes. This would allow a computer to be used in a more creative way to synthesise shapes for engineering components, given a specification of their desired function.

To achieve this, investigations into the use of novel shape representations and algorithms for shape optimisation were undertaken. Three examples of work done on shape optimisation for engineering components, using evolutionary algorithms and various shape representations, and the problems encountered in linking them together effectively with the analysis module, are described. These examples are aerofoil profile optimisation with a genetic algorithm, optimisation of a constructive solid geometry solid model with genetic programming and structural optimisation of a voxel shape representation with a genetic algorithm.

One conclusion of these investigations was that, when the set of shapes to be searched is large and general, it is often difficult to analyse correctly all the possible shapes. This can cause optimisation algorithms to optimise to shapes that are in practice sub-optimal. Additionally, there is only limited opportunity for the search through the set of shapes to be influenced by the analysed performance of the shape. This thesis argues that using a common representation for both the geometry and physical behaviour would allow a number of novel and effective algorithms for shape optimisation to be developed. The representation proposed is based on Chain models using cell complexes and chains from algebraic topology.

As an example of a new approach to shape optimisation enabled by the new representation, a novel algorithm that adds a morphogenetic stage to a genetic algorithm for structural optimisation, is also described. In initial studies this method, in which a shape is grown in response to both its genetic growth rules and structural performance, was found to be an effective approach to structural optimisation.

Contents

List of Figures

1 Introduction

1.1 What is Shape Optimisation?

Shape optimisation techniques are an attempt to automatically find the optimal geometric shape for an engineering component. Shape optimisation programs integrate optimisation algorithms, geometric modelling and engineering analysis algorithms into an automated computer-aided design process.

Typically, shape optimisation applications are classified by the behaviour of the component that is to be optimised. Structural shape optimisation and aerodynamic shape optimisation are the two problems that have received much attention by researchers. Work has also been reported in areas such as acoustics [Fisher 1995] [Soize & Michelucci 2000] [Bangtsson *et al.* 2003], magneto-statics [Kasper 1993] and manufacturing cost minimisation [Barton 2002] [Chang & Tang 2001].

Structural optimisation [Vanderplaats 1993] [Hsu 1994] [Sobieszczanski-Sobieski 1986] [Haftka & Grandhi 1986] seeks to find the optimal shape for a component which is subjected to some external loading. Often it is the weight of the component that is to be minimised, whilst also ensuring that the maximum stress remains within the yield stress of the component's material. Sometimes it is the maximum displacement that is to be minimised, when the component is subjected to a load with a constraint on weight. Structural optimisation can be further sub-divided into topology optimisation, size optimisation and structural shape optimisation.

Topology optimisation [Bulman *et al.* 2001], as its name suggests, looks to find the best topology for a structure. Often this is to find a topology for a truss structure. Size optimisation seeks to find the best value for dimensions of a component, for instance thickness or diameter, where the overall two-dimensional shape is fixed. Shape optimisation has come to mean finding the optimal shape given a particular topology. However, throughout this thesis, shape optimisation will be used to refer to all of topology optimisation, size optimisation and shape optimisation as it is the

author's opinion that topology and size are merely aspects of a component's shape. Indeed some researchers are now looking to integrate topology and shape optimisation [Bremicker *et al.* 1991] [Cappello & Mancuso 2003].

The typical aim of aerodynamic optimisation [Jameson *et al.* 1998] is to find a shape that minimises drag in a given flow whilst, perhaps, remaining within a specified lift constraint [Quagliarella & Cioppa 1995] [Burgreen *et al.* 1994]. Examples have also been described where other aerodynamic quantities, such as pitching moment and pressure distributions, are optimised [Fillipone 1995]. Many applications of aerodynamic shape optimisation are in the aircraft industry, both for the design of aircraft components [Doorly *et al.* 1996b] [Eleshaky & Baysal 1991] and aeroengines [Burguburu & le Pape 03] [Rogalsky *et al.* 1999a] [Song *et al.* 2002], although Fillipone optimises aerofoil sections for a wind turbine.

Most of the work to date has described shape representations for single criterion optimisation, although many researchers are interested in multi-criteria problems [Quagliarella & Vicini 2000] [Seller *et al.* 1996] [Vicini & Quagliarella 2000] [Lesieutre *et al.* 1998]. There are many practical design situations where there are a number of objectives, for instance, an aircraft wing must have an optimised aerodynamic shape, as well as an optimised structural shape. [Fugsland & Madsen 1999] describe the use of multi-criteria optimisation of wind turbine rotors.

1.2 Search Algorithms as Part of the Design Process

Engineering design is a process which aims to create artefacts that meet a particular need. Design specifications can be formulated which describe the requirements for the product. The design process can be seen as a decision making process in which these specifications are transformed into sufficient information for the creation and use of the artefact, throughout its lifecycle, from manufacture, through use to, possibly, its decommissioning. [Gero 1990] characterises design activity as 'a goal-oriented, constrained, decision-making, exploration and learning activity'.

[Gero 1990] identifies three classifications of design: *routine*, *innovative* and *creative*. He defines *routine* design as 'that design which takes place within a well-defined state space of potential designs'. This state space of potential designs is considerably smaller than the space of all possible solutions. *Routine* design looks to vary the values of variables in existing 'prototypes'. *Innovative* design similarly takes place within a well-defined state space of potential designs, but 'designs produced are outside the routine or 'normal' space', produced by 'manipulating the ranges of values for variables'. He defines *creative* design as 'that design which uses new variables producing new types and as a result extending or moving the state space of potential designs'.

Design can be seen as the *search* for a suitable or optimal design [Gero 1996] [Renner & Ekárt 2003] within a state space of potential designs. A search problem consists of a goal state, a search space and a search process. For design, the goal state is a design which, perhaps optimally, matches the requirements defined by the specifications. The search space is the set of all those designs that can be formed from all possible values of the design parameters. Optimisation algorithms, and in this thesis particularly Evolutionary Algorithms, are one possible search process by which the goal state can be found from the search space.

1.3 Shape Optimisation as Part of the Design Process

An important area of design research concerns the process of generating the geometric form for a component, given a desired function or behaviour for that component [Roy *et al.* 2001] [Shapiro & Voelcker 1989]. It would be helpful if computer tools could be developed which could take a desired function and, from this, produce a geometry that would exhibit such a behaviour. However, specifying function in a way that can be used to generate form has proved to be difficult [Roy & Bharadwaj 2002] and so the design of form is still regarded largely as a creative process undertaken by imaginative humans.

Recently, 'features' have been proposed as an approach to integrating function and form [Shah 1991]. Clearly, geometry often plays a uniquely important role in the representation of an engineering component. It therefore seems reasonable to group together aspects of a component's geometry into features and to attach information about the component's function, or possibly manufacturing process, to these features. There are, however, many different definitions of what constitutes a feature and equally many different approaches to generating form given the requirement for a particular set of features. Features only go a short distance toward the automatic generation of form from function.

The automatic generation of form, given a desired function, therefore seems problematic. However, the inverse of the geometric design process, namely the determination of the physical behaviour of a component given a particular geometry, is becoming increasingly easy for engineers and designers. Previously, engineers relied on empirical models, or analytic solutions to the equations governing the behaviour of components. This was restricted to a limited number of shapes and behaviours. The development of computational tools such as the finite element method [Desai & Kundu 2001] and computational fluid dynamics [Jameson 2001] has greatly increased the range of phenomena and shapes that can be analysed.

In the design process, such tools are typically used to assess a prototype design in order to find where the design is deficient or needs changing. Following the analysis, the designer either accepts that the design is adequate or changes the design and, possibly, undertakes a further analysis on the new design. This interactive process continues until an adequate design is found. Shape optimisation is an attempt to automate part of this process. Rather than a human designer changing the geometry of the design in response to the analysis, a computer program is used to make the changes in order to find an optimal geometry. [Papalambros 2002] reviews the current state-of-the-art for optimisation in the design process.

1.4 Evolutionary Algorithms in Design

Evolutionary Algorithms (EAs), such as the Genetic Algorithm [Holland 1975] [Goldberg 1989] [Davis 1991], Evolution Strategies [Rechenberg 1973] [Schwefel 1981] [Bäck *et al.* 1991] [Bäck 1996], Genetic Programming [Cramer 1985] [Koza 1990] [Koza 1992] and Evolutionary Programming [Fogel *et al.* 1966] [Fogel 1995] [Sebald & Fogel 1994], are search techniques which are inspired by an abstract model of how evolution takes place in biology. They are adaptive stochastic search techniques. A brief overview of the concepts and processes common to all EAs is given here to ensure the clarity of the following sections, however, for a more detailed treatment of EAs, the reader is referred to Section 2.3.4.2.

In an EA a ***population*** **of** ***individuals*** is maintained, where each ***individual*** represents a candidate solution. Each ***individual*** has a ***genotype,*** which is a structure that can be decoded to form the candidate solution. The genotype consists of a set or string of ***genes***. A gene has a number of possible values that are its ***alleles***. In order to test the ***fitness*** of an individual, it is necessary to transform the genotype to a ***phenotype***. Often this process is trivial, but this might not necessarily be the case. Mimicking natural selection, ***parents*** from the population are selected with some bias towards the better (fitter) solutions. From these parent solutions, ***offspring*** solutions are generated in various ways, by using ***operators***, which recombine or change the genes. Operators are usually chosen so that that the offspring inherit some of the attributes of their parents. These are then evaluated, placed in the population, and can subsequently be chosen as parents themselves. Often at this point, some of the least fit individuals are ***culled*** (i.e. removed) from the population. This process repeats a number of times generating subsequent ***generations***. This pseudo-Darwinian selection and breeding is intended to result in those properties that promote greater fitness being transmitted throughout the population. Selection of the fittest should result in increasingly good solutions appearing.

Evolutionary algorithms have a number of desirable properties, over other optimisation algorithms (see Chapter 2), for use in design:

- No derivatives need to be calculated. EAs are therefore easily integrated with any form of evaluation routine that may be required.

- They can deal with noisy landscapes. Since EAs do not use gradient information and make no assumptions about the smoothness of the landscape, they can cope with problems where small changes in variables can result in relatively large changes in the objective function, due to discretisation errors, for example. Such phenomena can often be observed in shape optimisation problems.

- They can cope with discontinuities in the landscape. EAs can optimise even where the objective function changes discontinuously with design variables.

- Discrete variables can be used. If appropriate operators are used, then EAs can deal with problems where the variables are not continuous [Deb & Goyal 1997]. An example of when this might be useful is where, for instance, the number of holes in a component is variable, as well as the sizes of those holes.

- EAs are (potentially) global optimisers. Although care must be taken to initialise the population correctly and to set EA parameters such as mutation rate and population size appropriately [Goldberg 1999], EAs can avoid merely finding a local optimum and can search through a large part of the search space for the global optimum.

- It is easy to deal with constraints with an EA. Some classical optimisation techniques have to be restricted to convex search spaces or must make special provision for dealing with constraints. With EAs these difficulties can be avoided with simple strategies, such as penalising fitness relative to the extent to which the constraints are violated, and thereby evolve away from these parts of the search space. Unfeasible individuals might just not be allowed to breed. Alternatively, individuals can be repaired to satisfy the constraints.

- EAs can use problem specific operators. EAs are not restricted to simple operators for moving around the search space. Whereas classical optimisation

techniques can only move within a local neighbourhood of the current point in the search space, EA operators can be designed using knowledge that the designer has about the nature of the problem. This can allow the EA to move about the search space in a more 'intelligent' way.

These advantages have led to a large body of research being undertaken with EAs in design. Some of this work is to apply EAs to problems to which classical techniques could not be applied because gradients could not be calculated or where the landscape is unsuitable. Alternatively, they have been used in order to search for global optima through larger search spaces, where, for instance, some of the design variables are discrete. This allows for the extension of the optimisation paradigm further into the design process.

The principal disadvantage of genetic algorithms is the need, in general, for a large number of function evaluations. One way in which this problem can be ameliorated is by using parallelisation. Because EAs deal with a population of solutions, they can easily be adapted so that a number of evaluations can take place on different processors (for shape optimisation a large proportion of the computation takes place in evaluation). [Cantu-Paz 1997] and [Nowostawski & Poli 1999] cover the use of parallelism for EAs.

[Renner & Ekárt 2003] gives a recent review of the use of genetic algorithms in computer-aided design. [Alander 1994] provides a bibliography of genetic algorithms in computer-aided design although there has been a considerable volume of research in this area since its compilation. [Giannakoglou 2002] gives a recent review of the use of stochastic optimisation techniques for aerodynamic optimisation concentrating primarily on EAs.

[Winter *et al.* 1995], [Gen & Cheng 1997], [Bentley 1999] [Parmee *et al.* 1993] and [Parmee 1993] give examples of the use EAs in design.

1.4.1 Routine Design with Evolutionary Algorithms

Evolutionary Algorithms lend themselves well to parametric or routine design [Gero 1990], where the structure of the design, and the variables allowing variation of the design within that structure, are well defined. The design process can be seen as the determination of values for those variables such that some measure of the design's utility (fitness) is optimised. The variables can be directly encoded as genes, with values for those variables being alleles for those genes. The genes together form the chromosome. The search space is thus determined by the range of values that the genes can take.

The majority of work done with Evolutionary Algorithms in design falls under this category. The variables to be used, and the ranges of those variables, are predefined and the EA is required to find values for those variables to optimise some objective. [Bentley 1999] refers to this as *Evolutionary Design Optimisation*, and a number of articles on this topic are included in his book. [Eby *et al.* 1999a] [Eby *et al.* 1999b] look to optimise a flywheel. [Robinson *et al.* 1999] describe the use of EAs in the design of satellite booms and load cells.

The number of papers describing work in this area is large and a comprehensive survey is not given here. However, some examples are [Husbands *et al.* 1996] who use genetic algorithms to design an aircraft wingbox. [Annicchiarico & Cerrolaza 1998] use genetic algorithms to optimise a truss structure. [Deb & Goyal 1997] use examples of the design of a gear train, a spring, a hydrostatic thrust bearing and a welded beam, and show the ability of a genetic algorithm to cope with mixed discrete and continuous variables. [Mäkinen *et al.* 1999] investigate a parallel genetic algorithm for the multi-disciplinary shape optimisation of aerofoils for both aerodynamic and electromagnetic (radar cross section) behaviour. [Quagliarella & Cioppa 1994] [Quagliarella & Cioppa 1995] [Oyamaa *et al.* 2001] [Doorly *et al.* 1996b] [Obayashi & Takanashi 1995] [Quagliarella & Vicini 2001] all look at the aerodynamic optimisation of transonic aerofoils. [Winter *et al.* 1995] contains a

number of examples of routine design with genetic algorithms. [Chen 2001] looks to optimise a structure for crash-worthiness.

Further details on EAs in shape optimisation specifically, for both structural and aerodynamic behaviour, are given in Section 2.3.4.3 and Section 2.3.4.4.

1.4.2 Creative Design with Evolutionary Algorithms

Whether computers can be creative is a contentious issue. [Boden 1991] discusses at some length creativity and computers. Reviews of her book in [Haase 1995] [Lustig 1995] [Perkins 1995] [Ram *et al.* 1995] [Schank & Foster 1995] [Turner 1995] and her reply [Boden 1995] provide an interesting discussion of the topic, along with Boden's later paper [Boden 1998].

Boden classifies creativity into three different types: *combinatorial*, *exploratory* and *transformational* [Boden 1998]. *Combinatorial* creativity involves the juxtaposition of familiar ideas or structures in novel ways. *Exploratory* creativity involves the search through a structured conceptual space. This, she says, can produce novel and unexpected structures, but they, clearly, 'satisfy the canons of the thinking-style concerned'. The parallels between this and Gero's *routine* design [Gero 1990] are apparent.

Transformational creativity involves 'the transformation of some (one or more) dimension of the space, so that new structures can be generated which could not have arisen before'. Again, this matches well with Gero's *creative* design [Gero 1990]. The line between exploratory and transformational creativity is somewhat unclear, as recognised by Boden: 'exploration of the space can include minimal 'tweaking' of fairly superficial constraints. The distinction between a tweak and a transform is to some extent a matter of judgement'.

Boden suggests that creativity involves 'going beyond the bounds of a representation' [Boden 1991], so that a novel solution is generated which could not

have been given by the representation. She does not think that a computer is capable of this kind of creativity.

[Bentley & Corne 2001a] give a number of descriptions of creativity as a process that have been given by various sources. These include:

(a) In the context of design, [Rosenman 1997] states that 'the lesser the knowledge about existing relationships between the requirements and the form to satisfy those requirements, the more a design problem tends towards creative design'.

This is a description which [Bentley 1999] finds useful. It does not explicitly define what creativity is, but rather presents design on a continuum from *creative* to *routine* based on the *a priori* knowledge available for transforming the requirements into a design.

(b) A second definition given by [Bentley & Corne 2001a] is that creativity is 'exploring a search space in an innovative and efficient way'.

With this definition, the creativity depends on how innovative or efficient the search is. This, though, is essentially the same as Gero's *routine* design and it would seem difficult to determine the boundary between when a search is being creative and not. It certainly conflicts with Boden's definition of creativity as 'going beyond the bounds of a representation', since all the designs produced would be set out by the representation that defines the search space. It does, though, match with her *exploratory creativity.*

(c) A third definition from [Gero 1996] is that creativity involves 'exploring alternative search spaces'.

Many of the systems for 'creative design' make use of a changing representation in a number of ways throughout the search. In some of the more advanced EAs, the number of variables can be changed, the coding can change, redundancy can be incorporated into the genotype, and various high-level structures can be evolved and

reused [Schoenauer 1996]. Thus the representation can be evolved along with the design and, arguably, alternative search spaces are explored.

[Bentley 2000] strongly advocates the use of representations that are based on components, rather than parameterisations, in order to allow evolution to design more creatively. The number and type of components can be evolved, along with the way they are arranged. Embryogenies can be used, in which the genotype to phenotype mapping can be more or less complex [Angeline 1995] [Kumar & Bentley 1999] [Kumar & Bentley 2003b]. This may allow complex structures to be 'grown' in the phenotype. This is discussed in more detail in Section 8.2.2.

These approaches are more creative than traditional uses of EAs with simple parametric representations, in the sense of (a) above. Such approaches show promise for generating form with less prior knowledge about the nature of the form to be generated.

They would not, however, be seen as creative given Boden's definition of creativity as 'going beyond the bounds of a representation'. All of the possible designs that can be generated in these 'creative systems' can be viewed as a single 'super-search-space'. The system searches through this space. It is certainly true that this new search space is potentially larger and more general. It may also be more easily searched. Solutions with complex structures may also be produced. This search space is, though, implicitly defined when the designer sets up the search - the system cannot generate a solution that lies outside this space. The efficacy of using search for design depends on defining a search space with relevant size and generality and a suitable way of searching the space. The use of evolving and varying representations may be a useful tool for doing this, but there may be other methods of achieving the same end result with similar 'creativity'.

(d) A fourth definition from [Goldberg 1999] is 'transferring useful information from other domains'. He distinguishes between *innovation*, which involves discovery

within a discipline, and *creativity*, which involves use of knowledge from outwith that discipline.

It seems that, until what it is to be *creative* can be defined, it will be impossible to determine whether computers can undertake *creative design*. [Bentley & Corne 2001b] contains a number of articles on *Creative Evolutionary Systems* that describe their use on a wide range of domains, from art and music, architecture, circuit design to antenna design.

[Bentley 1999] states that research into creative evolutionary design is concerned with the preliminary stages of the design process and can be categorized as *conceptual evolutionary design* and *generative evolutionary design*. These are discussed in the following two sections.

1.4.2.1 Conceptual Design with Evolutionary Algorithms

Conceptual design [Hsu & Liu 2000] [Hsu & Wonn 1998] [Wang *et al.* 2002b] takes place early in the design process [Renner & Ekárt 2003]. It commences with a high-level description of the requirements and then moves to a high-level description of a solution [M^{c}Neill *et al.* 1998].

With appropriate design of operators, EAs are able to search through large and complex search spaces. This has led to some researchers investigating the use of EAs for conceptual design. The motivation of this research is to allow a computer to undertake some of the creativity that takes place in the conceptual stages of design.

One of the important requirements for a system that is able to do this is that there is a very general shape representation available. [Husbands *et al.* 1996] describe a system using superquadrics which could generate a number of interesting shapes. [Bentley 1996] [Bentley & Wakefield 1996] developed a system based on 'clipped stretched cubes' that could be evolved by a genetic algorithm. This has the limitation that curved shapes cannot be generated, but does produce a shape representation that was very general and could be applied to the conceptual design of a large variety of

components. Since the aim of this work was conceptual design, only fast evaluation of the designs was undertaken measuring such aspects of the parts' behaviour as stability, extent, surface area, presence of flat surface, optical behaviour (for evolving prisms) and a particle-flow simulator (for evolving streamlined shapes). From an aerodynamicist's point of view, these evaluations would seem relatively simplistic and inaccurate. They do, however, have the merit of speed of execution and easy integration with the shape representation. To some extent this work can be seen as addressing very similar objectives as the work in this thesis, namely the use of EAs to extend the use of computers in the generation of form from function. However, he approaches this problem by extending preliminary conceptual design using search (with an EA) with simple evaluation of behaviour. In contrast the work in this thesis looks to extend detailed shape optimisation with analysis methods used by engineers towards conceptual design.

[Parmee *et al.* 2001] introduce an interactive evolutionary design system, which is aimed at supporting the decision-making processes during conceptual design. The software is intended for multi-disciplinary design where there may be multiple, uncertain and ill defined objectives. There are a number of modules (for example defining preferences among multiple objectives), built around a cluster-oriented genetic algorithm. An example is given for the conceptual design of an aircraft. [Parmee 2002] discusses how evolutionary computing can be used in the preliminary stages of design, where various problems might be encountered, such as criteria that are either qualitative or quantitative, or variables that may be continuous or integer. The usefulness of interactivity between the system and the designer is also discussed. This work is reported in detail in [Cvetkovic 2000] along with [Parmee 1996] [Cvetkovic & Parmee 1999a] [Cvetkovic & Parmee 1999b].

[Rasheed *et al.* 1997] use a genetic algorithm to optimise the conceptual design of a supersonic transport aircraft and a supersonic missile inlet. Variables were used for conceptual design parameters such as exhaust nozzle radius, engine size, wing area, wing aspect ratio.

As discussed in Section 1.2, [Gero 1990] defined *innovative* design as taking place within a well-defined state space of potential designs, but 'designs produced are outside the routine or 'normal' space', produced by 'manipulating the ranges of values for variables'. The use of EAs that can move outside the existing search space through the run and thus move towards potentially advantageous designs could be helpful during early stages of design, before a detailed design is settled upon. [Beck & Parmee 1999] describe a system that uses a multi-population genetic algorithm that allows the ranges of the genes to change through the run. A different approach is given by [Gero & Kazakov 2000] in which a crossover operator (see Section 2.3.4.2) is able produce designs that are outside the original design space.

1.4.2.2 Generative Design with Evolutionary Algorithms

One approach to searching through the very large space of possibilities during the early stages of design is to use a *generative* representation. [Hornby & Pollack 2001] define a generative system as being a system 'where the genotype is a program for constructing the final design' (rather than directly describing the design). [Schoenauer 1996] and [Bentley 2000] suggest that such approaches offer greater scalability, by allowing hierarchical, recursive and self-similar structures to be evolved.

[Hornby 2003] shows how systems using generative representations can better search large design spaces, since they can capture some of the properties of the structure of the search space and thus reuse components of the designs. Examples are given of generative design of voxel-based structures, neural networks and controllers for robots. [Hornby & Pollack 2001] use a representation based on Lindenmayer systems to evolve tables. [Funes & Pollack 1999] evolve Lego structures based on a generative tree-based representation.

Morphogenic evolutionary computation [Angeline 1995], in which a complex development process is used to generate the phenotype from the genotype, offers a number of potential advantages for design, such as better evolvability, and the ability

to generate solutions with complex structures from relatively simple genotypes. [Kumar & Bentley 1999] discuss various types of embryogenies and their advantages. Section 8.2.2 provides more detail on the current state-of-the-art in morphogenic evolutionary computation.

1.5 Motivation of the Work in this Thesis

A number of difficulties are encountered in setting up a shape optimisation algorithm to generate a useful shape for an engineering component:

- a well-defined specification of the desired function is required,
- a valid set of possible shapes through which to search must be defined,
- an analysis technique for accurately assessing how well potential shapes meet the desired function must be provided,
- an algorithm must be supplied that can effectively and efficiently search through the set of possible shapes.

Each of these points is easier to address when the set of shapes is relatively small and all shapes are kept similar to an initially defined starting shape. Most successful applications of shape optimisation have, therefore, relied on restricting the set of possible shapes.

The research reported in this thesis was motivated by the desire to extend the applicability of shape optimisation in the design process. It was hoped that shape optimisation techniques could be used so that the generation of geometric form, for a specified function, becomes more automated. Inevitably, this would mean that the set of shapes through which to search has to be large and general, and so choosing an appropriate geometric representation, analysis technique and optimisation algorithm becomes more problematic.

The work in this thesis can be seen as attempting to extend shape optimisation from Gero's [Gero 1990] *routine* design, to more *innovative* design. The work described here has, therefore, investigated shape representations, such as voxel models and spline models, which are able to represent a large generality of shapes. It has addressed the problem of accurately analysing a large range of shapes. It has also looked at modern stochastic global optimisation algorithms, such as genetic algorithms, which can avoid being trapped in local optima, since, as the size of the search space of shapes increases, the likelihood that the problem becomes multi-modal is increased. Following these investigations, the conclusion was drawn that a common framework for the integrated representation of both geometry and physical behaviour would be helpful.

1.6 Aims & Objectives of Work Described in this Thesis

This thesis presents the argument that a framework for shape optimisation, with a common representation of both geometry and physical behaviour, would allow the development of novel and efficient new algorithms better suited to the semi-automatic generation of an engineering component's geometry, given a certain desired behaviour.

The principal, overarching aim of this work was:

To determine whether shape optimisation can be extended, such that it can be used to increase the automation of the process of shape synthesis for engineering design.

In order to pursue this aim, research was undertaken into the following objectives:

(a) To determine whether evolutionary algorithms, along with novel shape representations which are able to represent a large generality of shapes, would enable more automation of the process of determining form from function, and to identify any obstacles that might be encountered with such an approach.

(b) To identify a computational framework which could provide an integrated representation of both component geometry and physical behaviour.

(c) To determine whether a morphogenetic evolutionary algorithm, using the identified integrated representation of geometry and physical behaviour, shows any potential to increase the automation of the process of shape generation for engineering design.

1.7 Chapter Outline

Chapter 2 reviews common approaches to shape optimisation from the literature.

Objective (a) is investigated in Chapters 3, 4 and 5, which describe shape optimisation work done by the author using novel shape representations and optimisation algorithms. Chapter 3 gives details of aerofoil shape optimisation work using various shape representations and analysis methods. In Chapter 4 an adapted genetic algorithm is used to optimise a voxel shape representation for structural optimisation. In Chapter 5 genetic programming is used to optimise a CSG solid model.

Investigations into Objective (a) are concluded in Chapter 6, which, following the review of optimisation techniques in Chapter 2 and the work described in Chapters 3, 4 and 5, argues that a computational framework which can provide an integrated representation of both component geometry and physical behaviour is desirable. This is done by analysing the various approaches described in Chapters 2 to 5 in terms of the methods that they use to search through the space of possible shapes.

Chapter 7 addresses Objective (b) and describes a possible framework that could be used for shape optimisation, based on Chain models, which use cell complexes and chains from algebraic topology. It also describes how existing techniques could be implemented in such a framework.

Chapter 8 addresses Objective (c) and describes an approach to shape optimisation that adds a morphogenetic stage to an evolutionary algorithm for structural optimisation. In this approach the evolutionary algorithm evolves genes which modulate the way that a cellular shape 'grows' in response to the stress on it. This is given as an example of a novel algorithm that can be implemented in the new framework from Chapter 6.

Conclusions are then drawn in Chapter 9.

2 Review of Shape Optimisation Techniques

2.1 Summary

In this chapter the numerous approaches to shape optimisation for engineering components are reviewed. The purpose of this review is not merely to list and describe the techniques, but to classify them in terms of the various methods they use for the optimiser, geometric model and analysis.

2.2 Typical Structure of a Shape Optimisation Method

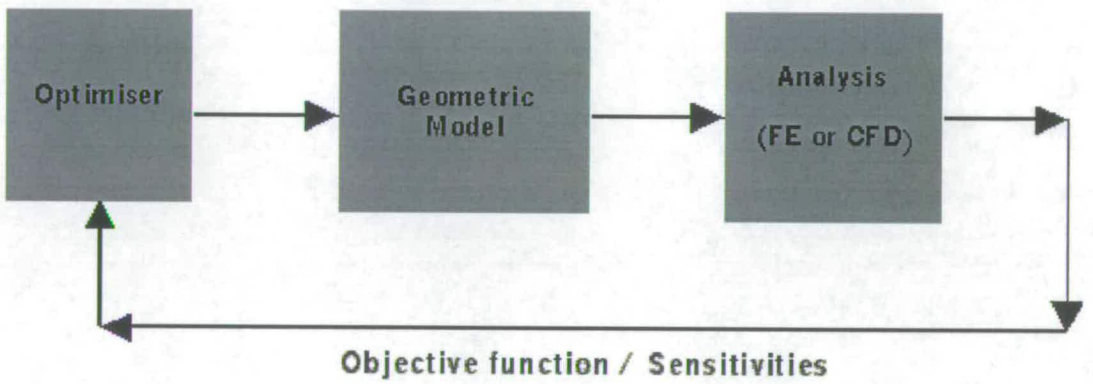


Figure 2-1 **Typical Structure of a Shape Optimisation Application**

Most approaches that have been used for shape optimisation can be split into three distinct sections [Hsu 1994]. These are the optimiser, geometric model and analysis modules. The optimiser changes some variables that affect how the geometric model is built. The geometric model is then passed to an analysis module where, typically, the geometry is discretised and the physical behaviour of the shape is approximated. From this analysis module, the objective function and possible constraints (for instance, maximum stress or deflection) are calculated. Based on the result of this

analysis, the optimiser again changes the variables over which it is optimising. This loop continues until some termination criteria are met.

This can be expressed more formally. A representation of the geometry is chosen so that there are a number of variables that can be changed in order to modify the shape. We can formulate the shape optimisation problem as:

$$\begin{aligned} &\text{minimise} && f(\boldsymbol{x}), \\ &\text{subject to} && g_e(\boldsymbol{x}) <= 0, \\ & && g_i(\boldsymbol{x}) <= 0, \end{aligned}$$

where $\boldsymbol{x}$ is the vector of design variables, $f(\boldsymbol{x})$ is the objective function and $g_e(\boldsymbol{x})<=0$ and $g_i(\boldsymbol{x})<=0$ are the constraints. Examples of possible objectives to be minimised are weight, volume, deflection or aerodynamic drag. $g_e(\boldsymbol{x})<=0$ are ***explicit*** constraints which can be described explicitly as a function of the design variables. Typically, these are upper and lower bounds on the design variables. $g_i(\boldsymbol{x})<=0$ are ***implicit*** constraints which depend on the design variables but cannot be expressed explicitly as a function of them. Examples of such constraints are stress, displacement or resonant frequency. These constraints must be evaluated using some computational model of the relevant physics.

In the following sections the various different approaches to each of these three modules in shape optimisation are discussed.

2.3 Optimiser

The optimiser is used to vary the shape design variables ($\boldsymbol{x}$) that the geometric model uses to build the shape. The optimiser alters these variables in order to find the best set of values for the shape design variables.

The geometric model and analysis combined can be thought of as a function mapping the vector of variables (in $\mathfrak{R}^n$ where n is the number of design variables) onto a value

for the objective (in $\mathfrak{R}^m$ where m is the number of objectives). This obvious similarity to the problem of optimising an analytic function led to the large body of techniques used for function optimisation being used for shape optimisation.

In the following sections, firstly some terminology is introduced. A review of optimisation techniques is then given. The field of optimisation is extremely large and so it is not possible to give a comprehensive survey of all classes of techniques. Rather, what follows is a survey of some of the major classes of optimisation techniques that have been applied to shape optimisation problems. These have been divided into classical deterministic techniques and stochastic techniques. A brief description of each technique is given so that an understanding of the algorithm and data structures used can be gained. The strengths and weaknesses of each method are discussed.

2.3.1 What is Optimisation?

Optimisation problems have three basic elements:

- An ***objective function*** which is to be minimised or maximised.
- A set of ***variables*** that affect the value of the objective function.
- A set of ***constraints*** that restrain the values that the variables can take.

Optimisation aims to find a set of values for the variables which minimises (or maximises) the objective function whilst respecting the constraints.

2.3.1.1 Some Terminology

Usually, optimisation techniques are used to minimise the objective function rather than to maximise the objective function. This does not imply a loss of generality since the maximisation of an objective function $f(\boldsymbol{x})$ is equivalent to the minimisation of $-f(\boldsymbol{x})$. It is therefore always easy to transform a maximisation problem into a minimisation problem.

For most optimisation problems there is only one objective function. When there is more than one objective function the problem is referred to as a ***multi-objective*** optimisation problem. Where there is no objective function and the problem is merely to find a set of values for the variables, which respects all the constraints, this is called a ***feasibility*** or ***satisfaction*** problem.

A problem in which there is only one variable is ***univariate***. Problems with more than one variable are ***multivariate***. Where the variables take values that are real numbers the problem is ***continuous***. Where some variables are real numbers and some integers the problem is a ***mixed integer*** problem, and the problem is described as ***discrete***. When the variables take integer values, but in permutations with each other, the problem is ***combinatorial***.

Problems that are subject to constraints are ***constrained***. Those not subject to constraints are ***unconstrained***. It is often useful to distinguish between those constraints that directly constrain the variables (i.e. x_1>15), as ***explicit constraints***, and those constraints that restrict the value that some response of the system other than the objective function can take, as ***implicit constraints***. An example of an implicit constraint in structural optimisation would be to keep the maximum stress below some value. The maximum stress obviously depends on the variables but not in an explicit way.

2.3.1.2 Search Spaces and Landscapes

The set of all possible combinations of values for the variables is referred to as the ***search space***. Each point in the search space is associated with a value for the objective function. When the variables are continuous, it is possible to consider the value of the objective function at each point in the search space as a height and visualise a ***landscape*** as in Figure 2-2 below.

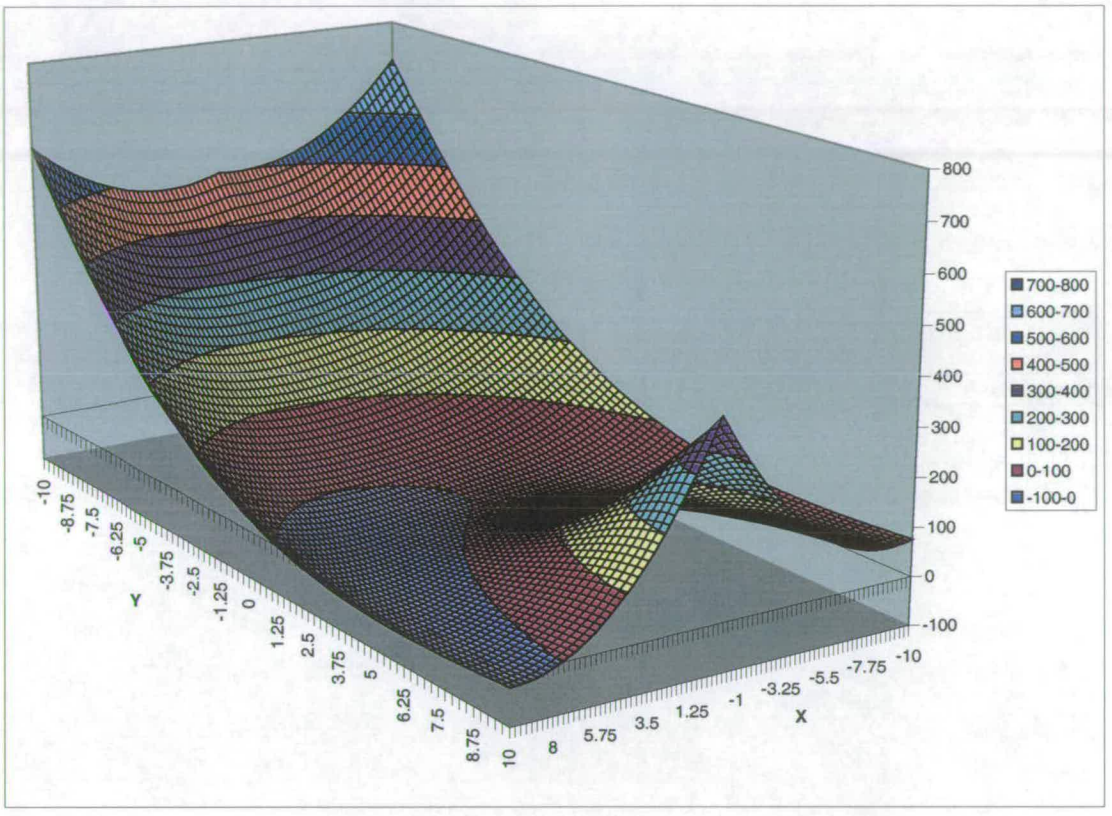


Figure 2-2 Example of Landscape

$$F(x,y) = 4(((10 - y)^2 + x^2)^{0.5} - 10)^2 + 0.5(((10 + y)^2 + x^2)^{0.5} - 10)^2 - 5x - 5y$$

In the example shown in Figure 2-2 there are two variables x and y, each constrained to have values between –10 and 10. There is a minimum near (8.6, 4.5).

Maxima and Minima

Figure 2-3 shows an unconstrained problem with a single continuous variable. The global minimum is the point in the search space that has the lowest value for the objective function. All other points in the search space have higher (or at best the same) value for the objective function.

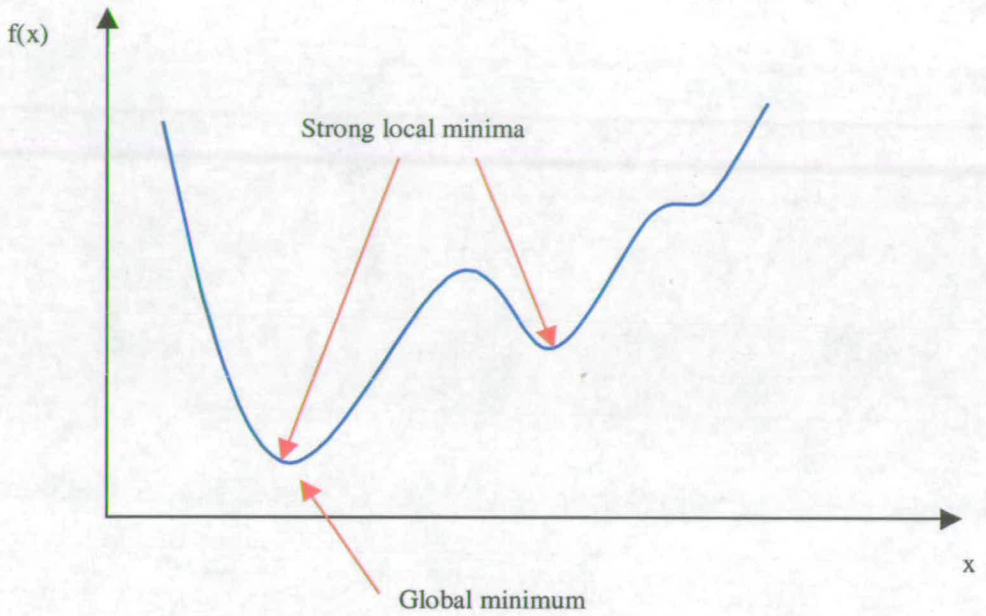


Figure 2-3 Unconstrained Objective Function Minima

A strong local minimum is a point that has the lowest value for the objective function in its neighbourhood. More rigorously this can be expressed by saying that there is a distance δ from the local minimum within which all points have a higher value for the objective function. When the objective function and its first derivative are continuous, it can be seen that at a local minima the gradient of the objective function is zero.

Figure 2-4 shows the same function, but now with a constraint on the variable. Now the global minimum lies against the constraint and the gradient at the global minimum is not zero. This is frequently observed in real optimisation problems.

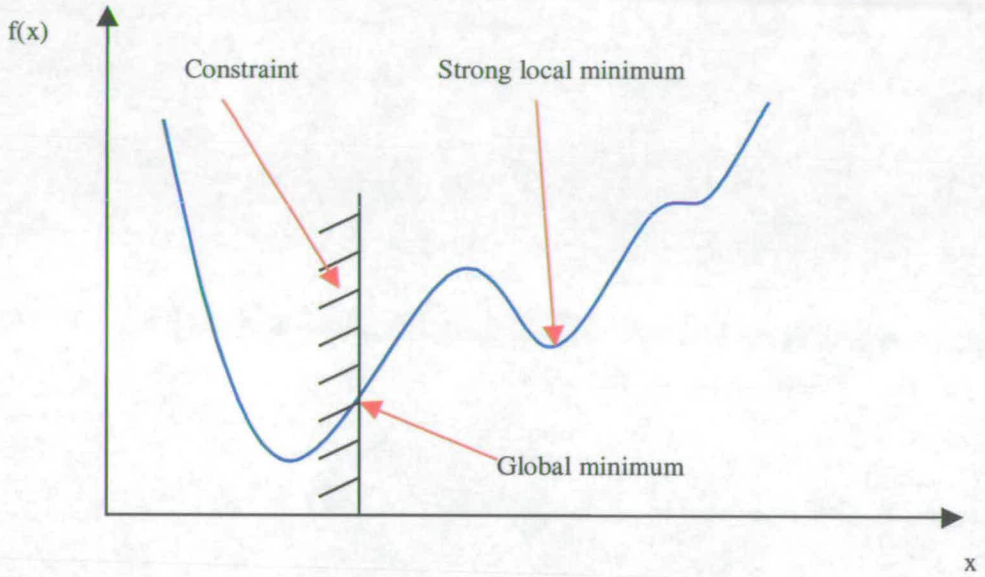


Figure 2-4 Constrained Objective Function Minima

2.3.2 Classical Deterministic Optimisation

The majority of research done into shape optimisation has used optimisation techniques from mathematical programming. These deterministic methods can be classified into second order, first order or zeroth order techniques depending on the order of the derivatives of the objective function that are required. Zeroth order methods require only the calculation of the objective function itself. First order methods require calculation of the objective function and its first derivatives (often called sensitivities) over the shape variables. Second order techniques also require second derivatives.

These techniques are primarily designed to work with continuous variables. Function minimisation is assumed throughout. Bold variables such as $\boldsymbol{x}$ refer to vectors and will naturally be used in the discussion of multi-dimensional problems.

Usually, these deterministic optimisers can find optima with fewer design evaluations than the stochastic methods described later. This is often important in engineering problems, where the time taken to perform one design evaluation is often many orders of magnitude greater than the time taken to produce candidate designs. However, such optimisers can often have difficulties in dealing with local optima, discrete design variables and noise which, for instance, can be generated when small changes in the design variables cause changes in finite element mesh topology.

All the major techniques described in this section are ***local*** methods. They will move to a minimum local to the point from which the method is started. When the problem is multi-modal (i.e. has many peaks and valleys) they cannot be guaranteed to find the global optimum and will instead find only local optima. It is possible to run these algorithms from several different initial positions. This can sometimes result in the global optimum being found. However, the stochastic methods described later in this section are usually much more effective in finding global optima as they can sample different areas of the search space and 'jump' out of local minima.

The following sections review some of the major algorithms from classical optimisation. Firstly, some zeroth order methods are described. Secondly, gradient based techniques are discussed. Finally, second order methods such as Newton-Raphson are reviewed.

2.3.2.1 Zeroth Order Methods

This section overviews the most common and important classical techniques that only require function evaluations and do not need gradient information. Such techniques are often referred to as ***direct search methods***.

Direct search methods are commonly used when:

- it is not possible to differentiate the function, or the function is subject to random error;
- the evaluation of derivatives of the function is very expensive and/or complex;

Hsu, in his review of structural shape optimisation techniques [Hsu 94], concluded that zeroth order methods have many advantages for three-dimensional shape optimisation, since often the above conditions are encountered. Often the determination of derivatives, which are typically calculated either by using finite differences or by using an analytic method, is expensive (see Section 2.3.3.5). Zeroth order algorithms, which do not require objective function gradients to be provided, therefore hold some advantages for shape optimisation. The following sections review some of the major zeroth order optimisation techniques.

Hooke and Jeeves

In the early 1960s Hooke and Jeeves developed a widely used direct search method [Hooke & Jeeves 1961] [Lewis *et al.* 2000]. This method makes moves along one dimension at a time. The Hooke and Jeeves method uses information gathered in previous function evaluations in order to determine the direction in which future moves might be profitable.

Starting from an initial base point, ***exploratory*** and ***pattern*** moves are undertaken. Exploratory moves perturb the each of the variables in turn, moving to the new point in the search space if improvements are made. If after perturbing all the variables no improved point is found, then a further set of exploratory moves are undertaken with the size of the perturbation halved. This continues until some minimum size of perturbation is reached. If after the exploratory move an improved point is found, then a pattern move is undertaken.

Pattern moves are used to speed up the search process by making larger moves in those directions that have previously found to be good. Pattern moves are repeated until no further improvement is found, at which point there is a return to exploratory moves.

This algorithm can work well for functions where the surface is well behaved and the dimensionality is fairly low (< 10). It is also easy to incorporate constraints by making a move a failure if it breaks a constraint.

[Keane 1994] compares the Hooke and Jeeves algorithm to a number of other optimisation techniques for a problem where the vibrational response of a truss structure of rods is to be optimised.

Nelder and Mead's Down-Hill Simplex method

Hooke and Jeeves' method attempted to make use of information about past function evaluations to decide where future exploration should take place in the search space. [Spendley *et al.* 1962] used similar ideas to produce a method based around the regular simplex. Their ideas were extended and refined by problems [Nelder and Mead 1965] in the mid 1960s to produce the downhill simplex method (unrelated to the simplex method of linear programming). Thus is a relatively straightforward multi-dimension search method that works well for low dimensional problems (up to 5 or 6), but becomes inefficient for larger problems [Nelder and Mead 1965].

This method is based around the movement, through stretching and contracting, of a non-regular simplex. An n dimensional simplex consists of $n+1$ vertices and all their connecting line segments and faces. In two dimensions a simplex is a triangle, in three a tetrahedron. Function evaluations are made at the position of the vertices in the search space.

[Duvigneau & Visonneau 2001] describe an application of the Nelder-Mead simplex algorithm to the shape optimisation of airfoils in incompressible, turbulent flows. [Rogalsky *et al.* 1999a] describes the application of the Nelder-Mead simplex

algorithm to the optimisation of turbine fan blades and compares the technique with other approaches. This method is robust and needs no derivatives and hence is easy to implement, however its convergence to the optimal solution can be slow, needing many potentially costly evaluations.

2.3.2.2 Powell's Direction Set Method

Many techniques for the optimisation of a multi-dimensional function $f(\boldsymbol{x})$ have the following basic framework:

(a) Choose a starting point, $\boldsymbol{x}_1$, and a direction $\boldsymbol{d}$.

(b) Find the minimum, $\boldsymbol{x}_2$, of $f(\boldsymbol{x})$ along direction $\boldsymbol{d}$ from $\boldsymbol{x}_1$ using a 1-dimensional minimisation technique.

(c) Stop if termination criteria met otherwise choose a new direction $\boldsymbol{d}_1$, replace $\boldsymbol{d}$ by $\boldsymbol{d}_1$ and $\boldsymbol{x}_1$ by $\boldsymbol{x}_2$, repeat step (b).

There are numerous optimisation algorithms which follow this form, each varying in the way in which the search direction and the line search is chosen.

A simple algorithm takes the set of unit vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_n$ of the n-dimensional space as a set of directions and then, starting from where the last minimisation reached, minimises in each direction in turn. This is repeated until no further improvement is found. This simple algorithm is easy to implement, but is often inefficient where the function has many long narrow valleys that are not parallel to one of the search directions. In this case the method can take a very large number of small minimisation steps, cycling through the direction set many times.

To avoid this problem a number of algorithms try to produce a better set of search directions which either lie along the valley or which avoid interfering with each other (so that the minimisation in one direction is not undone by the minimisation in the next direction). Such methods make use of the concept of ***conjugate directions***. At a particular point for two line minimisation directions $\boldsymbol{p}$ and $\boldsymbol{q}$ to be non-interfering,

changes in the gradient of $f(\boldsymbol{x})$ along $\boldsymbol{q}$ must be perpendicular to $\boldsymbol{p}$. The condition for two vectors, $\boldsymbol{p}$ and $\boldsymbol{q}$, to be conjugate in this sense for a quadratic function $f(\boldsymbol{x})$ is that $\boldsymbol{p}.\boldsymbol{A}.\boldsymbol{q} = 0$, where **A** is the Hessian matrix of $f(\boldsymbol{x})$, $[A_{ij}] = \partial^2 f/\partial x_i \partial x_j$

A method is needed in order to build such a set of conjugate directions. [Powell 1964] describes a method for creating n mutually conjugate directions during the optimisation run. [Brent 1973] extends these methods.

[Lesieutre *et al.* 1998] make use of Powell's direction set method for the multi-disciplinary optimisation of missile fin planforms.

2.3.3 Gradient Descent Methods

The methods described in this section use gradient information (directional derivatives) of the objective function in order to choose search directions. They are useful when the derivatives are defined, can be calculated and are not too computationally expensive to calculate. They can converge quickly to the minimum for problems when the objective function can locally be adequately approximated by a quadratic.

2.3.3.1 Method of Steepest Descent

If gradients can be calculated then an obvious direction in which to search is the direction in which the gradient is steepest. The direction of steepest descent is $-\nabla f(\boldsymbol{x})$ (the vector of partial first derivatives for the function at the point). Similar to the direct search methods described previously, a line search is undertaken in the chosen direction until the minimum is found in that direction. The gradient is then calculated at this point and the new steepest descent direction chosen as the new search direction. This is repeated until a termination criterion is met.

This algorithm is simple to implement but it suffers from similar difficulties to those described in Section 2.3.2.2 with interfering directions as is shown in Figure 2-5. This can cause the algorithm to take a large number of iterations when the landscape has long narrow valleys.

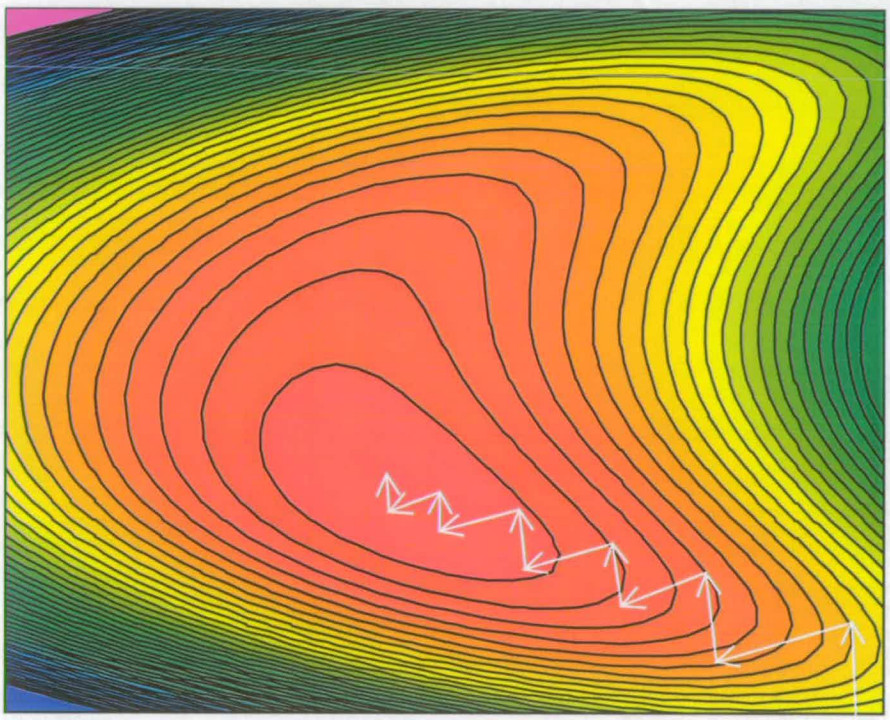


Figure 2-5 **Poor Performance of Steepest Descent Method in a Valley**

2.3.3.2 Conjugate Gradient Methods

Just as Powell's Direction Set method avoided interfering search directions by using conjugate directions, conjugate gradient methods look to use conjugate directions in order to avoid some of the difficulties encountered with the Steepest Descent methods.

At each iteration, this method chooses a direction based on the direction of steepest descent but which is conjugate to the previous direction and hence partially (in some informal sense) to all the previous directions searched. The most commonly used methods for choosing these conjugate directions are variations of the original Fletcher-Reeves method [Fletcher and Reeves 1964] [Press *et al.* 1993].

For a quadratic n-dimensional function, it can be shown that, with the Fletcher-Reeves method, the minimum will be found in at most n iterations of the algorithm. However, in most real applications the function will only be approximately quadratic and so the algorithm may need further iterations.

Conjugate gradient algorithms can perform well. However, their performance can deteriorate when the objective function is poorly approximated by a quadratic or if the calculated gradients are inaccurate.

2.3.3.3 The Newton-Raphson method

The Newton-Raphson method addresses the problem of interfering directions with basic steepest descent method by approximating the function at each iteration by a quadratic function and then moving in a direction toward the turning point of that quadratic.

At each iteration $f(\boldsymbol{x})$ and its first and second derivatives are calculated at the current point $\boldsymbol{x}_c$. A quadratic function $y(\boldsymbol{x})$ is found that matches these values at $\boldsymbol{x}_c$. This gives:

$$y(\boldsymbol{x}) = \tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_c).\boldsymbol{G}_c.(\boldsymbol{x} - \boldsymbol{x}_c) + (\boldsymbol{x} - \boldsymbol{x}_c).\boldsymbol{g}_c + f(\boldsymbol{x}_c)$$

where $\boldsymbol{G}_c$ is the Hessian matrix (matrix of 2nd derivatives) of $f(\boldsymbol{x})$ at $\boldsymbol{x}_c$ and $\boldsymbol{g}_c$ is the gradient vector $\nabla f(\boldsymbol{x})$ computed at $\boldsymbol{x}_c$. It is straightforward to show [Walsh 1975] that the minimum value of the quadratic $y(\boldsymbol{x})$, $\boldsymbol{x}_m$, is given by:

$$\boldsymbol{x}_m = \boldsymbol{x}_c - \boldsymbol{G}_c^{-1}\,\boldsymbol{g}_c$$

A 1D minimisation is then undertaken in the direction toward $\boldsymbol{x}_m$.

2.3.3.4 Variable Metric (quasi-Newton) Methods

Very often it is impossible or very costly to compute the inverse Hessian matrix required for the Newton-Raphson method. Quasi-Newton methods iteratively construct a sequence of positive definite symmetric matrices $\boldsymbol{H}_i$ that become better and better approximations to the inverse Hessian $\boldsymbol{G}^{-1}$.

The Davidon-Fletcher-Powell algorithm [Acton 1970] uses an updating formula that can be proved to converge $\boldsymbol{H}_i$ to $\boldsymbol{G}^{-1}$ in approximately n steps for an n-dimensional problem [Walsh 1975]. The algorithm starts as steepest descent and converges to Newton-Raphson whilst avoiding the worst problems of both. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [Acton 1970] uses a more complex update formula that is generally slightly superior to that given above [Walsh 1975].

Quasi-Newton methods are probably the most widely used algorithms for minimisation of functions that can be reasonably approximated by quadratic forms and whose gradients can be calculated at arbitrary points. In these circumstances they are highly efficient and accurate algorithms. However, if these assumptions do not hold their performance is much impaired.

2.3.3.5 Gradient Based Methods for Shape Optimisation

There is a vast literature on structural optimisation using first order optimisation techniques for linear elastic problems. [Vanderplaats 1993] writing in 1993 described the previous thirty years of structural optimisation using finite element analysis (see Section 2.5).

Early techniques used finite differences to calculate these gradients. These are formed by perturbing the design variables and recalculating the objective function and constraints. Typically this involved recalculation of the finite elements in order to calculate the constraint sensitivities. n additional function evaluations are required to calculate the forward difference approximations for each of the n design variables.

Later it was observed that the simultaneous equations solved during the finite element method:

$$\mathbf{Ku} = \mathbf{P} \qquad \text{(Equation 2.1)}$$

where: $\mathbf{K}$ is the stiffness matrix

$\mathbf{u}$ is the vector of node displacements

$\mathbf{P}$ is the vector of node loads

could be differentiated using the chain-rule with respect to the design variables to give:

$$\frac{\partial \mathbf{K}}{\partial X_i}\mathbf{u} + \mathbf{K}\frac{\partial \mathbf{u}}{\partial X_i} = \frac{\partial \mathbf{P}}{\partial X_i} \qquad \text{(Equation 2.2)}$$

and so:

$$\frac{\partial \mathbf{u}}{\partial X_i} = \mathbf{K}^{-1}\left\{\frac{\partial \mathbf{P}}{\partial X_i} - \frac{\partial \mathbf{K}}{\partial X_i}\mathbf{u}\right\} \qquad \text{(Equation 2.3)}$$

where X_i is the i-th design variable. K^{-1} has already been calculated in order to solve Equation 2.1 and so only the terms in the bracket need to be evaluated. If the loads are assumed to be independent of the design variables then only $\partial K/\partial X_i$ need be calculated. Analytic methods can be used to determine this term for some types of variables (for instance thickness) however in general analytic methods are not available and the semi-analytic method is used and $\partial K/\partial X_i$ found using finite differences.

These methods can be more efficient than using finite differences to calculate $\partial f/\partial X_i$ (where $f(X)$ is the objective function) and $\partial u/\partial X_i$ (which is needed to calculate the sensitivity of the constraints). However they rely on specialised analysis code and so make integration with standard analysis packages very difficult. Even with analytic methods, the number of available design variables is still limited.

[Hicks *et al.* 1974] were the first to use first order optimisation techniques for aerodynamic optimisation, using the method of Feasible Directions (based on the conjugate gradient method). Since then a large amount of research has been done in applying these techniques for various aerodynamic shape design problems. This work has focused on the efficient calculation of gradients [Anderson & Venkatakrishnan 1999] [Burgreen *et al.* 1994] [Burgreen & Baysal 1994] [Burguburu & le Pape 03] [Sadrehaghighi *et al.* 1995] [Reuther *et al.* 1999] and the handling of constraints.

Common to both aerodynamic and structural optimisation with gradient based optimisers is the need to efficiently determine the sensitivities of the behaviour under consideration to the design variables. Methods of deriving sensitivities for systems governed by systems of partial differential equations are described in some detail in

[Lewis 97]. For shape optimisation, function evaluations are typically expensive and so calculation of finite differences where there are a large number of design variables can be very computationally expensive. In addition, it can be difficult to determine the appropriate amount to perturb the variables and the accuracy of the approximation can be poor.

The Newton-Raphson method makes use of the second derivative of the objective function with respect to the variables. Where these derivatives can be easily calculated then these methods can be effective. Often though this is not the case and second order methods can be computationally expensive. [Ariana & Ta'asanb 1999] provide a detailed study into the nature of the Hessian for aerodynamic optimisation problems. [Novruzi & Roche 1995] compare Newton's method with the quasi-Newton method for an electromagnetic shaping problem. They find that while the number of iterations are fewer for Newton's method than for the Quasi-Newton method the total computational effort (since the second derivatives must be calculated) is between 1.2 and 3 times greater depending on the number of variables.

The principal advantage of gradient based shape optimisation is quick convergence to the local optimum, often resulting in a significant increase in the efficiency of the shape with relatively few evaluations of the objective function. They are, however, only local optimisers and so only find shapes in the neighbourhood of the given initial shape. They also require that the landscape is relatively smooth, having no discontinuities or noise. This can limit their robustness for the generality of problems. Finally, the calculation of gradients can be difficult or time consuming especially as the number of design variables increases. In summary, these are powerful techniques and are useful if the gradients can be calculated efficiently and they can be started in the vicinity of the optimal shape.

2.3.4 Stochastic Techniques

As discussed in Section 2.3.2, classical optimisation techniques are often efficient at finding local optima. However, in many shape optimisation problems the landscape

is multi-modal, the calculation of gradients is expensive and the landscape is not smooth. To address these problems, the last twenty years has seen a great deal of research into stochastic optimisers where various random elements are introduced in order to move out of local optima. The use of stochastic optimisation techniques, such as evolutionary algorithms [Holland 75] [Michalawicz 1992] and simulated annealing [Kirkpatrick 1983], in shape optimisation [Chapman *et al.* 1994] has been a popular area of research.

Stochastic optimisers are zeroth order optimisers and so require no gradient information and rely only on function values. Such techniques are also able to cope with problems with discrete variables or mixtures of continuous and discrete variables, which makes them considerably more flexible when applying them to engineering design problems. Their principal disadvantage is that convergence to the optimum can be slow requiring a large number of objective function evaluations.

2.3.4.1 Simulated Annealing

The method of simulated annealing [Kirkpatrick *et al.* 1983], was the first stochastic technique to become popular for practical optimisation. The algorithm uses an analogy of the annealing of solids. Annealing is the process in which a solid is heated to a maximum temperature at which all molecules of the solid randomly arrange themselves in the liquid phase, followed by a gradual cooling. As long as the maximum temperature is high enough and the cooling process gradual enough, the molecules settle into a very stable minimum energy lattice.

At each temperature T, as long as the solid is allowed to reach thermal equilibrium, the probability of being in an energy state E is given by the Boltzmann distribution:

$$P(E) = 1/Z(T) \,.\, exp(-E/kT)$$

Where Z is a normalisation function and k is the Boltzmann constant. In the 1950s Metropolis [Metropolis *et al.* 1953] developed stochastic simulations of thermodynamic systems in which systems were assumed to move from energy state

E_1 to a possible new value E_2 with probability $exp[-(E_2 - E_1)/kT]$. If $E_2 < E_1$ this probability is greater than 1 and so was assigned a probability of 1, i.e. certainty – the system always moved to a lower energy state but could probabilistically move to a higher one.

In the early 1980s Kirkpatrick and colleagues [Kirkpatrick *et al.* 1983] applied this principle to combinatorial optimisation. The simulated annealing method requires the following:

- a description of a possible solution to the problem,
- one or more operators to make random changes to the current solution to produce new possible solutions,
- an objective function, C, (analogous to energy) to be minimised,
- a control parameter T (analogous to temperature) and an annealing schedule which controls how T is reduced. New solutions will be accepted or rejected according to a Boltzmann like probability distribution.

The basic algorithm works like this:

(a) Initialise T. Generate an initial solution, S_c. Find the cost of this solution, C_c.

(b) Use an operator to randomly generate a new solution S_n from S_c. Calculate the cost of this new solution C_n.

(c) If $(C_n\text{-}C_c) < 0$, i.e. better solution found, then $S_c = S_n$. Otherwise generate a random number between 0 and 1, *random*. If $exp(-(C_n - C_c)/T) > random$, then $S_c = S_n$.

(d) If annealing schedule dictates, reduce T.

(e) Unless stopping criteria met, return to step (b).

As can be seen in step (c), moves which reduce the objective (i.e. good moves which improve the solution) are always accepted. Moves which increase the objective function can be accepted with a probability that depends on the size of the reduction in quality and the temperature T. Initially, T, is set high and so starts off high so poor moves are often accepted. Over time the temperature is reduced and so the probability of accepting a poor move is reduced. An annealing schedule describes how the temperature reduces over time. Correctly designing this schedule for a particular problem is important to ensure the efficiency of this method.

Typically the algorithm is stopped once T has fallen below some threshold such that the algorithm has degenerated into random search, or no improvement has been made for some time.

As well as handling combinatorial problems, simulated annealing can be applied to problems with continuous variables [M^{c}Ilhagga et al 1996]. Simulated annealing can also be hybridised with a suitable local classical technique by performing some sort of local search or gradient descent on each move.

The advantages of simulated annealing are that they can be applied to problems with non-smooth and discontinuous landscapes, they do not get caught in local optima and they do not require gradients of the objective function. These properties allow [Hasançebi & Erbatur 2002] to successfully apply simulated annealing to the simultaneous topology and shape optimisation of a truss structure.

[Reddy & Cagan 1995] apply simulated annealing to truss size and topology optimisation. They use shape grammars to modify the truss sizes and topology. These grammars define operators that allow one truss structure can be changed into another. The operator to use and the size of the 'move' are determined by the simulated annealing algorithm.

[Lin & Chen 2000] apply simulated annealing to structural problems with a non-convex design space. They also hybridised the method with local and random search algorithms.

A disadvantage of simulated annealing is that it can need a large number of function evaluations to converge to the optimum in comparison to classical gradient techniques. For shape optimisation problems objective function evaluations are generally expensive. Therefore, [Leite & Topping 1999] look to parallelise simulated annealing in order to extend its applicability for structural optimisation.

[Dibakar & Mruthyunjaya 1999] apply simulated annealing to the problem of determining the kinematic dimensions of a mechanism for a manipulator, so that its workspace is as close to the desired workspace as possible. This is an interesting application of stochastic optimisation to an area of design that could not be tackled with classical optimisation techniques.

[Kasper 1993] compares simulated annealing with evolution strategies for the shape optimisation in magneto-statics using the finite element method. They use the example of weight minimisation of a lifting magnet to prove the efficacy of the methods.

2.3.4.2 Evolutionary Computing

Evolutionary Algorithms (EAs) are adaptive stochastic search techniques. They are inspired by an abstract model of how evolution takes place in biology. In EAs a population of candidate solutions is maintained. 'Parents' from this population are selected with some bias towards the better solutions. From these parent solutions, offspring solutions are generated in various ways. These are then evaluated, placed in the population, and can subsequently themselves be chosen as parents. The technique has attracted a great deal of interest because it has been shown to be highly robust and to perform well without recourse to fragile domain specific heuristics.

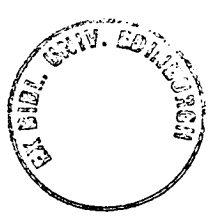
The best known of these techniques is the Genetic Algorithm (GA) [Holland 1975] [Goldberg 1989] [Davis 1991]. However, there are also the closely related Evolution Strategies [Rechenberg 1973] [Schwefel 1981] [Bäck *et al.* 1991] [Bäck 1996], Genetic Programming [Cramer 1985] [Koza 1990] [Koza 1992] and Evolutionary Programming [Fogel *et al.* 1966] [Fogel 1995] [Sebald & Fogel 1994]. There are

numerous variations on these algorithms and only a general introduction to this class of techniques is given here. Whereas classical optimisation techniques are typically applied as they are without alteration, EAs are a powerful and flexible framework in which optimisation algorithms can be developed.

There are common concepts and processes used in all evolutionary algorithms. An EA uses a ***population*** **of** ***individuals***. Each individual contains a structure, the ***genotype***, which can be decoded to form a candidate solution. Members of the population are assigned some fitness according to their performance that is assessed by some ***evaluator***. Individuals are ***selected*** (usually stochastically) with some bias towards fitter individuals. Stochastic ***operators*** are applied to selected individuals to produce new candidate solutions. Operators are usually chosen so that that the offspring inherit some of the attributes of their parents. The offspring are then evaluated and placed in the population. Often at this point some of the least fit individuals are ***culled*** (i.e. removed) from the population. This process repeats a number of times generating subsequent ***generations***. This pseudo-Darwinian selection and breeding is intended to result in those properties that promote greater fitness being transmitted throughout the population. Selection of the fittest should result in increasingly good solutions appearing.

Each individual contains a genotype that encodes the solution that the individual represents. These genotypes typically consist of strings of numbers and/or characters that are subsequently interpreted as a solution to the problem. The string of numbers or characters is often termed the ***chromosome***. Each element of the string is referred to as a ***gene*** and represents some aspect of the solution. A gene has a number of possible values that are its ***alleles***.

In order to assess the individual's fitness, it is necessary to decode the genotype into a ***phenotype***, which can be evaluated. The complexity of this transformation from genotype to phenotype can vary depending on the representation (encoding) chosen. For many applications the encoding might be direct. For instance, the chromosome contains a string of values for variables that can be directly evaluated. There are,



however, possibilities for more complicated mappings from genotype to phenotype, where the values of the genes need to be converted to an evaluatable solution through a more complicated process. Such a strategy might be used to 'grow' a solution from some initial conditions using rules of development. The genes might encode for the initial conditions or modify the rules of development (see Chapter 8)

The choice of ***population size*** is important when applying an EA to a problem [Goldberg 1999]. Too small a population can result in the population prematurely converging to a sub-optimal solution. However, excessively large population sizes can result in too many evaluations required.

There are a number of ways of producing an ***initial population***. Often a population is generated by choosing values for the genes at random from the range of possible values. This results in a random initial population. Alternatively, some possibly good solutions can be ***seeded*** into a random population. An initial population might entirely contain potentially good solutions.

The ***operators,*** crossover and mutation, developed by Holland [Holland 1975], are widely used. In simple crossover a random a crossover point along the chromosome

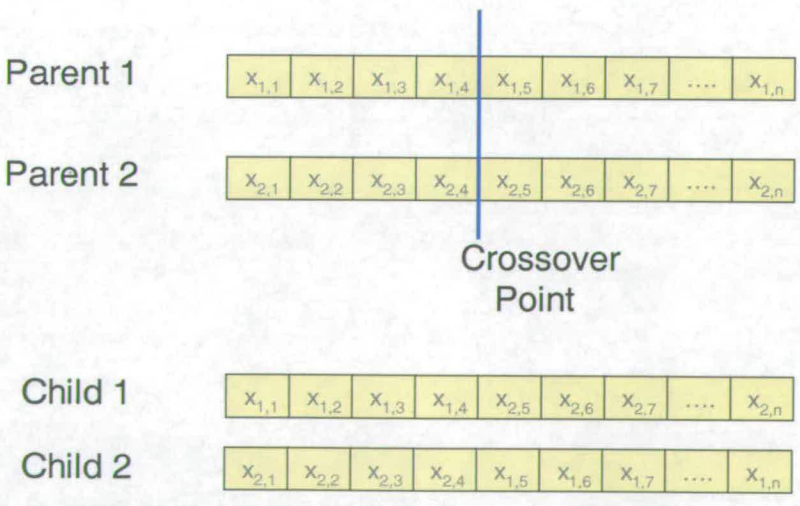


Figure 2-6 Simple One-Point Crossover

is chosen. Two new chromosomes are created by swapping over the sections lying after the crossover point (see Figure 2-6).

Crossover can be seen to rearrange existing genes, but does not create new genes. ***Mutation*** changes the value of a gene to some other possible value. The way in which this is done depends on the representation used. For example, if a bit-string representation is being used a mutation might involve a bit flip. In a representation consisting of a string of real numbers, a mutation might involve a small random move from the current value. Often mutation operators involve the change of only one gene. However, there may be cases where a mutation involving a change in a number of genes is desirable.

The typical application of operators when breeding new individuals proceeds as follows. Crossover (with some high probability) is applied one of the two new chromosomes formed is chosen at random. Following crossover, genes on the resulting chromosome undergo mutation with a low probability. This probability is chosen so that there is approximately one mutation per chromosome. The resulting chromosome is then taken for the new child individual.

The operators described here are those which are traditionally used with EAs. However, many modern users of EAs make use of other operators. For instance, for some problems, crossover is not useful as it merely acts to disrupt good solutions and algorithms using only mutation operators can be more efficient. Problem specific mutation operators are frequently developed so that mutation makes 'sensible' changes. These mutation operators may be guided by heuristics or make use of local search methods or even traditional optimisation techniques. Indeed, designing the encoding and its set of operators together for a problem is often successful.

Some method of ***selecting*** which individuals from the population to breed is needed. Below are given four of the most widely used selection methods.

Breeding Pool selection looks to make the expected number of offspring for an individual proportional to its relative fitness. The relative fitness of member i is calculated as follows:

$$Rel(f_i) = f_i / \Sigma f_i$$

The expected number of offspring for each individual is then calculated by $rel(f_i) \times N$ to the nearest integer, where N is the population size. The appropriate number of copies of each individual are then placed into a 'breeding pool'. Individuals are then chosen for breeding at random from this pool. The current population is replaced with the new population that is formed in this way. Thus the fitter individuals are more likely to contribute towards the next generation.

Some selection strategies only replace a proportion of the current population at each generation. It may be possible for the fittest individual not to be chosen for breeding with some (often small) probability and so not make it through to the next generation or for the operators to disrupt the best solution to a worse solution. To avoid this many selection schemes employ an ***elitist strategy*** in which the best solution in a generation is always allowed to pass into the next generation.

Roulette wheel selection is based around the analogy of a roulette wheel. The proportion of the roulette wheel assigned to each individual depends on its relative fitness, $rel(f_i)$. Individuals are selected by generating a random number between 0 and 1 (analogous to spinning the roulette wheel). The individual is chosen by moving through the population an individual at a time until the cumulative relative fitness of an individual is greater than the random value. The current individual is chosen for breeding.

It can be seen that the probability of an individual being selected is proportional to its relative fitness. Although every individual has some chance of breeding, there is a considerable bias towards better individuals based on their fitness. One problem that can often be found with breeding pool and roulette wheel selection is that at the start

of an optimisation run, even though all individuals are poor, some of the better individuals might be relatively much better than the others. This can lead to too many individuals being generated from the better, but still poor, individuals and so the population can converge too quickly.

Ranking schemes can overcome the problems mentioned above. The population is ranked according to the fitness values of its members. Selection is then performed using a pre-determined probability distribution function dependant on rank rather than fitness. This is typically a simple linear function. This can ensure that at the start of a run no individual is selected too often just because its relative fitness is high. Also at the end of a run, when fitnesses are often very similar, selection can favour better individuals.

In ***tournament selection*** n individuals are chosen at random and the fittest of these is selected for breeding. Typically, a value for n of two is chosen. Often this method is used with a steady state algorithm where only one individual is bred at a time.

Evolutionary algorithms have been successfully applied to a wide range of optimisation problems, often by adapting standard algorithms to produce algorithms which are well suited to the problem both in terms of parameter settings and operators. It is also relatively trivial to hybridise EAs with local search and classical techniques, by for instance, applying gradient descent to each new offspring. This can produce efficient robust algorithms [M^{c}Ilhagga *et al.* 1996].

A more detailed discussion of EAs in design is given in Section 1.4.

2.3.4.3 Evolutionary Algorithms in Structural Optimisation

There have been a wide variety of ways that EAs have been applied to structural optimisation. These have varied in the aspect of the structure that is to be optimised, for example, topology, layout, size or shape. The shape representations used have also been varied with, for example, parametric solid models, splines or cellular representations all used (this is covered in more detail in Section 2.4). They have also

varied in the types of behaviour that is to be optimised and hence analysis method used (this is covered in more detail in 2.5).

[Adeli & Cheng 1993] covers general applications of genetic algorithms to structural design. [Adeli & Cheng 1994a] investigates constrained optimisation of space frames, and introduces a Lagrangian Multiplier approach to cope with the constraints. [Adeli & Cheng 1994b] extend this work to use parallel computers.

A number of researchers have considered the optimisation of truss structures with genetic algorithms where various parameters are optimised such as cross-section and size [Leite & Topping 1998] [Jenkins 1992] [Jenkins 1997]. [Chapman et al. 1994] look to use genetic algorithms for what they term 'preparametric' design. They use a cellular shape representation with genes determining whether rectangular cells are filled with material or not. [Annicchiarico & Cerrolaza 1998] use finite element analysis to analyse the elastic behaviour of a 2-dimensional truss structure. They later in [Annicchiarico & Cerrolaza 2001] describe work done using β-spline surfaces to represent geometry for 3-dimensional shape optimisation again using finite elements to analyse the shapes' performance. [Cappello & Mancuso 2003] use genetic algorithms for the combined topology and shape optimisation of trusses and plates using finite element analysis.

[Schoenauer 1995] uses three shape representations for the topology optimisation of a cantilever plate. These are a 'natural' bit-array representation, a 'holes' representation in which rectangular holes can be introduced into the design, and a Voronoi representation. Analysis is done with the finite element method. The 'holes' and Voronoi representation are found to outperform the bit-array representation.

[Coello & Christiansen 2000] concern themselves with the multiobjective optimisation of a truss structure. They look to simultaneously minimise the structure's weight and maximum deflection. They review a number of ways that genetic algorithms have been used for multiobjective optimisation and implement a

weighted-sum of objectives method. [Coello 1999] reviews the use of evolutionary algorithms for multiobjective optimisation.

[Robinson *et al.* 1999] use EAs in the design of a truss structure for a satellite booms and for the design of a load cell. [Keane 1994] compares the performance of a GA with other techniques for the vibrational optimisation of a satellite boom.

[Deb 1997] [Deb & Goyal 1998] [Deb & Goyal 1997] describe a system, GeneAS, based on genetic algorithms with mixed variables for mechanical component design. They use examples of the design of a pressure vessel, gear train, a spring, a hydrostatic thrust bearing and a welded beam, and show the ability of a genetic algorithm to cope with mixed discrete and continuous variables.

The design of laminates was investigated in [LeRiche *et al.* 1995]. They developed a segregated genetic algorithm that uses separate interbreeding populations of solutions. The fitness function used for each population is different. The penalty for failing to meet a particular constraint differs for each population. The find that the segregated genetic algorithm allows solutions to be found which satisfy all constraints whilst minimising weight.

[Cerrolaza *et al.* 2000] use β-splines for 2-dimensional optimisation of plates. They use the boundary element method to analyse the Von Mises stress and attempt successfully to minimise weight.

[Eby *et al.* 1999a] [Eby *et al.* 1999b] optimise a flywheel with an 'injection island genetic algorithm' where the specific energy density of the flywheel is to be maximised. In this approach the variables are the depth of the flywheel at various radial distances. The injection island genetic algorithm allows various sub-populations to breed each with different resolution of the representation. Thus one population can use a coarse resolution with a correspondingly inexpensive analysis, whilst another has a finer representation with a more expensive analysis. Good individuals from a coarse population can be 'injected' into a finer population. This allows a quick exploration of a large part of the search space with the coarse

representation. Promising areas of the search space can then be explored with increased accuracy in populations with a finer representation.

Motivated by a similar desire to use a cheap evaluation in the initial stages of an optimisation run [Gage *et al.* 1995] use a variable complexity shape representation. As the run progresses the shape representation complexity can be increased. They apply this method to the structural optimisation of trusses and aerodynamic optimisation of low-speed wings.

2.3.4.4 Evolutionary Algorithms in Aerodynamic Optimisation

A number of researchers have described the integration of a genetic algorithm with computational fluid dynamics for transonic aerofoil shape optimisation [Quagliarella & Cioppa 1994] [Quagliarella & Cioppa 1995] [Oyamaa *et al.* 2001] [Doorly *et al.* 1996b]. [Obayashi & Takanashi 1995] described the use of a genetic algorithm to find a shape for an aerofoil that meets a specified pressure distribution.

[Quagliarella & Vicini 2001] describe the use of genetic algorithms for the design of configurations of multiple aerofoils. As well as optimising multiple aerofoils they also optimise at two design points, one with the aerofoils in a high lift configuration and one in a low drag cruising configuration [Quagliarella & Vicini 2000] [Vicini & Quagliarella 2000].

[Doorly *et al.* 1996a] use genetic algorithms for coupled aerodynamic-structural design. As previously discussed, one of the disadvantages of genetic algorithms is the number of function evaluations that can be required. Analysis using computational fluid dynamics can be very time-consuming and so parallelisation of genetic algorithms for aerodynamic optimisation problems has been investigated by [Marco & Lanteri 2000] [Doorly & Peiró 1997] [Doorly 1995].

[Rogalsky *et al.* 1999b] compares a genetic algorithm with the downhill simplex method described in Section 2.3.2.1 and simulated annealing. [Vicini & Quagliarella 1999] hybridise a genetic algorithm with a gradient based optimisation technique that

they apply to multidisciplinary optimisation of both aerofoil profile and wing planform.

[Mäkinen *et al.* 1999] investigate a parallel genetic algorithm for the multi-disciplinary shape optimisation of aerofoils, for both aerodynamic and electromagnetic (radar cross section) behaviour.

2.4 Geometric Model

Shape optimisation uses a geometric modeller in order to represent the shape. The area of geometric modelling is very large and, at some point, most forms of geometric model have been used for shape optimisation. In this section the intention is to describe some of the geometric modellers that have been used for shape optimisation and to explain the data structures they use.

Shape optimisation proceeds by using some elements of the geometric model as design variables. Thus, by changing the variables and rebuilding the model, the shape can be changed. This approach is natural, however the family of shapes that the range of variables described is often poorly understood. This is developed further in Chapter 6.

Early two dimensional shape optimisation work used the co-ordinates of nodes of the finite element mesh as the design variables [Zienkiewicz & Campbell 1973]. The boundary nodes (i.e. those on the perimeter) were moved by the optimiser. The shape representation used was therefore cellular. This seemed a natural representation and had the advantage of needing only one representation for both the geometry and analysis. Unfortunately, it proved impossible to ensure that a smooth boundary shape was maintained. Additionally, as the nodes were moved, the elements in the mesh become increasingly skewed and hence the results from the finite element analysis became less accurate. Ensuring that the mesh is sufficiently accurate is also a theme that is developed in Chapter 6.

Following this, the design model was separated from the analysis model in order to ensure smoothness of the boundary of the shapes generated. One approach then taken was to use polynomials to describe the boundary of the shapes [Kristensen & Madsen 1976], [Bhavikatti & Ramakrishnan 1980], [Pedersen & Laursen 1982]. The variables were chosen to be the coefficients of the polynomials. There were a number of problems that were found with this approach. Firstly, low order polynomials such as quadratics, cubics and quartics could only represent a limited family of shapes. This problem can be reduced by increasing the order of the polynomial. However, this causes other problems. In order to represent a shape feature with small radius of curvature requires high order polynomials and these can cause oscillatory 'ripples' away from the feature. Additionally, polynomials do not allow local control of the shape; changes in one of the variables (a coefficient of one of the polynomial) causes a change in the boundary of the whole shape. The ways in which the representation affects the family of shapes that the optimiser searches through, and the implications that this has on the optimiser, are discussed in Chapter 6.

Spline curves offer many of the advantages of polynomials, namely boundary smoothness and a useful separation of geometric model and analysis model. They also remove some of the problems encountered when using polynomials. The boundary does not oscillate because the splines are formed from low order polynomial pieces. Splines also offer good local control when enough control points are used. These properties of splines that make them highly useful for shape optimisation are the same reasons that make them so widely used in computer-aided design.

There are different ways of assigning variables when using spline curves for shape optimisation. A commonly used and natural assignment is to use the co-ordinates of the control points as the variables. This allows a very general family of shapes to be used. However, for some applications, a more restricted family of shapes is desired. For these cases the 'path' of the control points can be restricted to some line or curve and the variable used is the position of the control point along this path.

Alternatively, the control points can be restricted to lie within a particular area [Giannakoglou 2002].

Non-uniform Rational B-Splines (NURBS) [Hearn & Baker 1994] are an extension to b-splines allowing the exact representation of conic sections. NURBS have been used by [Schramm & Pilkey 1995] for structural optimisation. Conic patches were used by [Widmann & Sheppard 1993] as a shape representation for shape optimisation which allows the number of design variables to increase during the optimisation.

Most modern commercial CAD packages are built upon Boundary Representation (B-rep) solid modellers. Examples of such solid modellers are ACIS and Parasolid. These CAD packages allow the user to build 'parametric' models. As the user builds the model, some of the dimensions and positions of the component can be specified as variable. Then, once the model is built, the user can change any variable and the modeller will rebuild the model with this new value for the variable. The modeller does this by essentially undoing all the steps, back until the point in the model build when the variable was used, and then rebuilds the model from there. Such parametric models are very useful for shape optimisation as they provide an easy 'family of shapes' through which an optimiser can search. However, as Shapiro & Vossler note [Shapiro & Vossler 1995], the concept of parametric modelling is far from well defined. As anyone who has used these techniques will testify, these parametric models are can often result in non-intuitive shapes being generated. [Raghothama & Shapiro 2002] suggest a way in which parametric families of shapes might be better defined.

[Tavakkoli & Dhande 1991] and [Widmann & Sheppard 1994] use the intrinsic geometrical properties of curves such as their curvature to define shapes. Variables are used to define the curvature at particular arc lengths. The curvature then varies linearly between these points. From this the shape of the curve can be constructed in Cartesian space. Tavakkoli and Dhande apply this to the configuration optimisation of a truss structure. Widmann and Sheppard use the finite element method to analyse the shape generated for structural optimisation. [Kodiyalam *et al.* 1992] use a constructive solid geometry (CSG) approach to structural shape optimisation.

2.5 Analysis Methods

For each set of values for the design variables, a shape is built in the geometric model. It is then necessary to evaluate this shape to assess its relevant physical behaviour (for example its weight, volume, displacement or stress under a given load). It is then possible to assign a value for the objective function and to establish whether constraints have been violated. An analysis module is used to do this.

Engineers have developed many computational techniques for modelling the physical behaviour of models. In principle almost all of these methods could be used for shape optimisation. This section is not intended to provide a comprehensive review of all possible modelling techniques, rather it is a survey of the main techniques used for shape optimisation.

The principal criteria for choosing an analysis technique are that the analysis be as computationally inexpensive as possible, that it is relatively easy to convert the geometric model into the analysis model, that the analysis method is capable of modelling the behaviour of interest and that the method is sufficiently accurate. These considerations are discussed in much more depth in Chapter 6.

For shape optimisation the most often-used analysis method is the finite element method [Desai & Kundu 2001]. It is capable of modelling a large number of physical behaviours that can be described with partial differential equations, from elastic deformation, resonant frequencies to thermal analyses and others.

The first step in using the finite element analysis for shape optimisation is to take the geometric model and to generate a mesh of elements. The quality of the mesh (i.e. the number, shape and distribution of the elements) is very important in ensuring that the analysis is accurate. Early uses of the finite element method for shape optimisation relied only on moving the node positions as the geometry changed. This removes the need for a possibly time-consuming remeshing, but the accuracy of the solution can be poor as the element shapes become more distorted. This effect can, to some extent, be mitigated by using higher order elements. Now, though, it is usual to remesh each shape. It is also becoming increasingly common to use adaptive meshing techniques to ensure solutions are sufficiently accurate. These techniques are becoming standard in commercial finite element programs.

The boundary element method is another analysis method that has been frequently used for shape optimisation. The range of physical behaviours that this method can analyse is more restricted than for the finite element method. It is unable to model phenomena such as buckling or calculate mode shapes. However, only the boundary of the geometry needs to be meshed, rather than the whole geometry as in the finite element method. It is therefore much easier to produce the mesh. [Cerrolaza *et al.* 2000] [Meric 1999] [Sandgren & Wu 1988] [Schramm & Pilkey 1994] [Yamazaki *et al.* 1993] [Yamazaki *et al.* 1994] all describe the use of boundary elements for shape optimisation. [Makerle 2003] provides a bibliography of topology and shape optimisation with both boundary elements and finite elements.

For aerodynamic optimisation problems there are a number of analysis methods available. [Jameson 2001] gives a recent review of the techniques for aerodynamic analysis and design. These methods include finite volume, vortex panel, finite elements and finite difference methods. He also reviews some of the methods of mesh generation. With aerodynamic shape optimisation, as with structural optimisation, the determination of gradients can be time consuming.

3 A Study on Aerofoil Optimisation using a Genetic Algorithm

3.1 Summary

This chapter reports on a study made on aerofoil shape optimisation using a genetic algorithm. Fluid analysis was undertaken with a vortex panel method that was written for this problem. Three shape representations were used, an aerofoil parameterisation, a four Bézier curve representation and a four Bézier curve representation with the constraint that the tangent vectors are equal for both Bézier curves at the joins between curves (C^1 continuity). A number of optimisation runs were undertaken with various fitness measures (to maximise lift coefficient, minimise drag coefficient and minimise drag-lift ratio). The genetic algorithm used tournament selection, single point crossover and a floating point mutation.

The method worked well with the parameterised aerofoil representation, although a large number of evaluations were required. However, some difficulties were encountered. The analysis method used was quick but made certain assumptions about the flow regime (it could not predict stall) and so there were difficulties in ensuring that only shapes were generated for which these assumptions were valid. This was possible with the aerofoil parameterisation. However, with the more general Bézier spline representations, it proved very difficult to ensure that only shapes that could be accurately analysed were generated. The algorithm therefore produced unrealistic results.

3.2 Introduction

Computational fluid dynamics (CFD) is a tool to model fluid flows. The ready availability of powerful computers and the improved user-friendliness of commercial CFD programs has meant that CFD is becoming increasingly used by engineers to analyse the performance of engineering systems in fluid flows. Typically CFD is

used to 'virtually prototype' a system to test its behaviour before any physical prototype is produced. This allows changes to be made to the design, based on information generated in the analysis, more cheaply. [Jameson 2001] offers a comprehensive review of computational algorithms for aerodynamic analysis and design.

The aeronautics industry makes heavy use of CFD in the design of aircraft and aeroengines. Often for such applications, even small improvements in systems' performance are very valuable. Therefore, they look to make use of optimisation techniques, in conjunction with CFD analysis, in order to improve the system performance. The design of aerofoils for wing profiles and turbine blades is one typical use of such aerodynamic shape optimisation techniques.

Aerofoil design methods can be split into two different categories: inverse optimisation and direct optimisation. Often the engineer will be able to specify a desired pressure distribution for the aerofoil, so that it has required lift, drag and other aerodynamic properties, such as pitching moment. Inverse aerofoil design techniques use this pressure distribution on the aerofoil surface and then calculate the corresponding geometry. However, the corresponding geometry may not be a valid shape (i.e. top and bottom surfaces may cross), the required pressure distribution may imply a flow which is in some other way undesirable, or the required pressure distribution might be difficult to determine.

In a direct optimisation method the shape is parameterised into design variables (e.g. shape parameters or co-ordinates of spline curve control points). Candidate solutions are evaluated and some form of optimisation technique, numerical or stochastic, is used to search for an optimal set of values for the design variables. Direct optimisation is a more powerful technique since it allows a search through a much greater design space. However it is also much more computationally expensive since the flow needs to be solved for a large number of shapes.

As discussed in Chapter 2, there is a large range of optimisation techniques. Most of these have been applied to aerodynamic shape optimisation. However, predominantly hill-climbing algorithms have been used. These have the disadvantage of having difficulty distinguishing global optima from local optima. They may therefore often converge on a local optimum and so find a globally sub-optimal solution. These methods have difficulty where the search space is highly non-linear or discontinuous and where design parameters are discrete. Gradient methods or second order Newton (or quasi-Newton methods) have been used (see Section 2.3.3.5 for examples of gradient based techniques for aerodynamic optimisation). These require gradient information that is often found by perturbing each of the design variables by a small amount and resolving the flow and approximating the gradients with forward differences. This requires an extra solution of the flow for each design variable. Also, for this method to be accurate the flow has to be solved to a high degree of accuracy.

The work described in this chapter used a direct approach of optimisation. In this work a genetic algorithm has been applied as the optimisation procedure. As discussed in Chapter 1, one of the underlying motivations of the work in this thesis was to increase the automation of the design process using shape optimisation. Increasing the generality of space of shapes through which to search would therefore be important, as would the ability to find the global optimum.

3.2.1 Aims

- To determine whether genetic algorithms, along with a vortex panel fluid analysis, are effective for optimisation of a large range of aerofoil profiles.

3.3 Implementation

3.3.1 Shape Representations

A program has been written which allows three representations:

- an aerofoil parameterisation (illustrated in Figure 3-1),
- a four Bézier curve representation (see Figure 3-2) with the end points of the curves joining (C^0 continuity),
- a four Bézier curve representation with the additional constraint that the tangent vectors at end points of the curves should be equal (C^1 continuity) [Hearn & Baker 1994] except at the trailing edge.

3.3.1.1 Aerofoil Parameterisation

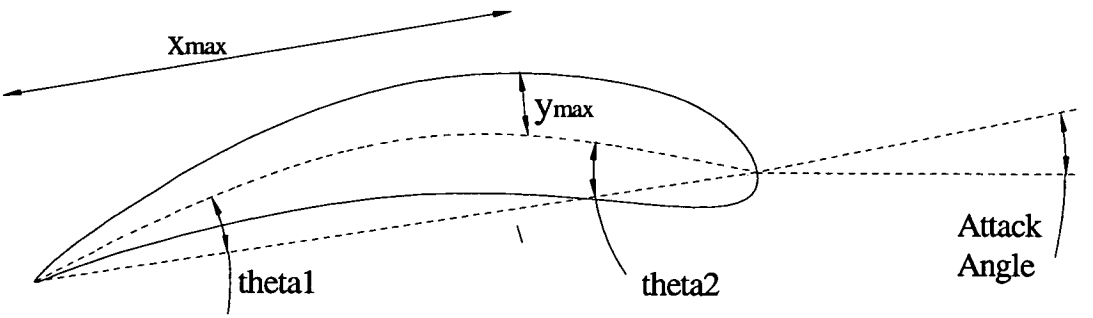


Figure 3-1 Aerofoil Parameterisation

With the parameterised representation the shape is constructed using a thickness envelope, wrapped around a mean camber line. The mean camber line is defined as lying midway between the top and bottom surfaces of the aerofoil and intersecting the trailing and leading edges. Given these fixed two points, a particular camber line is constructed given the angle it makes with the chord line at the trailing and leading edges. The thickness envelope is a function of the maximum thickness of the aerofoil and the position of the maximum thickness along the chord length. The whole aerofoil is then rotated about its trailing edge by an attack angle.

The aerofoil parameterisation used values for the angle between the chord and camber line at trailing and leading edges, maximum thickness, position of maximum thickness along the chord length and attack angle as the alleles of genes within each candidate shape's chromosome.

3.3.1.2 Bézier Representation

A cubic Bézier curve has the following equation [Hearn & Baker 1994]

$$\mathbf{p}(u) = (1-u)^2 \mathbf{x}_1 + 3u(1-u)^2 \mathbf{x}_2 + 3u^2(1-u)\mathbf{x}_3 + u\mathbf{x}_4$$

where u varies from 0 to 1 and x_1 to x_4 are the control points. The Bézier curve has some useful properties. It passes through the first control point, x_1, when $u = 0$. It also passes through the fourth control point, x_4, when $u = 1$. When $u = 0$ (i.e. at the

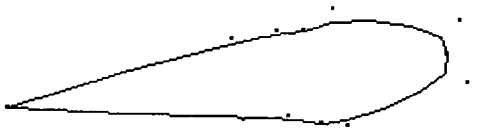


Figure 3-2 Bézier Representation for Aerofoil

first control point, x_1) the tangent to the curve is in the direction of the second control point, x_2. Similarly, when $u = 1$ (i.e. at the second control point, x_2) the tangent to the curve is in the direction of the third control point, x_3.

To represent the aerofoil four Bézier curves were used. The constraint was added that the end points of adjacent Bézier curves be coincident. In other words, C^0 continuity between the Bézier curves was imposed.

3.3.1.3 Smooth Bézier Representation

The Bézier representation from the previous section was modified to impose the additional constraint that the tangent vectors at the end points of the curves should be equal (C^1 continuity) [Hearn & Baker 1994] except at the trailing edge. This was done by making the control points at the junction between two Bézier curves lie halfway between the adjacent control points from the adjacent curves.

3.3.2 Evaluator

The evaluator used to calculate lift and drag coefficients was based on the vortex panel method. It was implemented based on the method descried in [Kuethe & Chow 1986]. This method solves the potential flow and so assumes that the flow is steady, inviscid and incompressible. It is also unable to predict separation of the boundary layer and so is not accurate in conditions when the aerofoil would stall.

The primary advantage of this method was that it takes much less time to solve the flow around the candidate aerofoil than a steady Euler or Navier-Stokes CFD code. [Jameson 2001] reviews the current literature on the computational costs of the various mathematical models for solving flows around aerofoils. On a Pentium 3 (300MHz) PC the vortex panel method, with 48 panels, was found to take approximately 0.017 seconds. In comparison, a finite volume CFD solution of Euler flow was found to take approximately 5 seconds.

The evaluator required co-ordinates of forty-eight points around the aerofoil. The method was run with a free air velocity of 80 ms^{-1} corresponding to a Reynolds number of about $5x10^6$.

3.3.3 Fitness Measure

With all optimisations the objective is to minimise some measure of fitness. In this work three measures of fitness were used.

The first fitness measure was to maximise C_l where:

$$C_l = \frac{f_l}{\rho A \frac{V^2}{2}}$$

where: C_l is the lift coefficient

f_l is the lift force (N)

ρ is the fluid density (kg m^{-3})

V is the air speed (m s^{-1})

A is the aerofoil area (m^2).

This measure was useful primarily to validate the analysis and optimisation code. An aerofoil with maximal lift would have the largest camber, thickness and attack angle allowed (since the analysis was unable to predict stall). Therefore if analysis and optimisation code was correct then these values should be set to their upper constraints following optimisation.

The second fitness measure was to minimise C_d where:

$$C_d = \frac{f_d}{\rho A \frac{V^2}{2}}$$

where: C_d is the lift coefficient

f_d is the drag force (N).

The third fitness measure was to maximise C_l / C_d where C_d is the drag coefficient of the aerofoil and C_l is the lift coefficient. This is a realistic design criterion for many aerofoil applications.

3.3.4 The Genetic Algorithm

A genetic algorithm (GA) was written for this application. The following sections detail the implementation of this algorithm.

3.3.4.1 Chromosomes

For all three shape representations a chromosome of real numbered genes was used. Bounds were placed on these values constraining the values for these variables to lie between a minimum and maximum value.

For the parameterised aerofoil representation there were five genes:

- maximum thickness from camber line,
- position along chord length of maximum thickness,
- camber angle at trailing edge,
- camber angle at leading edge,
- attack angle.

For the unsmoothed Bézier representation there were 24 genes each corresponding to a co-ordinate for one of the twelve control points. There were twelve control points since there were four control points for each of the four Bézier curves making sixteen control points, however four of these control points were on two of the Bézier curves.

It was decided to use using polar co-ordinates for the position of the control points. This was because it was easy to ensure that reasonable aerofoil shapes were generated (i.e. without creating a 'crossing' shape), by placing bounds on the control points' angular co-ordinate so that the adjacent control points moved successively 'around a circle'. Therefore, for each control point, there was a gene to represent the radius and one to represent angle. All angles ran from $-\pi$ to π radians.

For the smoothed Bézier representation there were six fewer genes since the position of the control points between Bézier curves (except at the trailing edge) was set by placing it halfway between adjacent control points.

3.3.4.2 Initialising the Population

For the aerofoil parameterisation the initial population was formed by setting each gene randomly within the bounds for that variable, with a uniform distribution.

For the Bézier representation, a gene's value was set by deviating from an example value for a specified aerofoil profile. The size of this deviation was set randomly, with uniform distribution, and the maximum size of the deviation was set by the user.

Details on the bounds selected, and reasons for the selection of these bounds, for each experiment undertaken, is given with the results for the experiment in Appendix A.

3.3.4.3 Selection

Tournament selection was used in this GA. A user-specified proportion of the population to be bred every generation was chosen. Thus a number of individuals to breed could be calculated. These individuals were selected by repeatedly choosing two individuals from the population at random. The individual with the best fitness was then allowed to breed by placing it in a list of 'parents'.

3.3.4.4 Crossover and Mutation Operators

Each of the individuals in the parents list was taken in turn and crossover applied. This was done by selecting at random another of the parents and using single point crossover. A crossover point is chosen at random. A child individual is then generated by taking genes from the first parent up to the crossover point and from the second parent after the crossover point.

Mutation was then applied to the child individuals. There was a user defined mutation rate and mutation amplitude. Each gene was taken in turn and a random number (from 0 to 1) generated. If this random number was less than the mutation rate then mutation was applied to this gene. The size of this mutation was in the range:

*[-0.5 * mutationAmplitude * geneRange, 0.5 * mutationAmplitude * geneRange]*

with uniform distribution where *geneRange* is the difference between the value of the gene's upper and lower bounds.

Following mutation all the child individuals were then placed into the population and the population then sorted in order of fitness. The population was then returned to its original size by culling the least fit individuals.

3.4 Results

A number of different experiments were undertaken. Full details of these experiments can be found in Appendix A. The following sections give a brief synopsis of the results of the various experiments undertaken. Each experiment was repeated 10 times.

3.4.1 Aerofoil Parameterisation

Experiment A

The objective of this run was to maximise lift coefficient (i.e. minimise $-C_l$). This run was used primarily to validate the analysis and optimisation code. As was expected, the genetic algorithm consistently produced an aerofoil with the largest camber, thickness and attack angle allowed within the parameter bounds, with a value of C_l of 2.54.

Experiment B

The objective of this run was to minimise the drag coefficient. Again, this run was used primarily to validate the analysis and optimisation code. It was anticipated that the optimum aerofoil for low drag coefficient would have low or zero camber and would be at an attack angle very close to zero. However, the aerofoils generated by the genetic algorithm had a high camber and negative attack angle. The value of drag coefficient was negative which was clearly incorrect. There was clearly some problem with the optimisation algorithm or fluid analysis.

Further investigation of the landscape of this problem around the generated aerofoil was undertaken as detailed in Appendix A. This indicated that the vortex panel analysis produced unrealistic results when attack angles were negative and camber angles were high.

This highlighted a problem that was to be frequently encountered with the vortex panel fluid analysis. It worked well for most shapes, but in some areas of the search space the values for lift and drag it returned were incorrect. It was also difficult to predict in which areas it performed poorly.

Experiment C

This experiment was a repeat of Experiment B, attempting to minimise drag, but restricted the bounds on the camber angles to 10° in order to avoid the problems encountered with false values for drag coefficient being generated for aerofoils with negative attack angle and high camber angles. With this restriction, the genetic algorithm produced an aerofoil shape with low camber and low attack angle. The calculated drag coefficient of 0.019, averaged over the ten runs, was realistic.

Experiment D

This experiment looked to maximise the lift/drag ratio (i.e. minimise C_d / C_l). As this was the first experiment that attempted to optimise the lift/drag ratio, it was decided to firstly attempt a simplified problem in which the attack angle was constrained to be 0°. It should also be noted that the camber angles were not restricted to be below 10°, but were allowed up to 30°.

The aerofoils that the genetic algorithm generated matched well with what was expected, with a reasonably thin aerofoil with relatively high camber. The lift-drag ratio of 26.8 was realistic. It was found that runs converged to two slightly different aerofoils, although with similar fitnesses. It was suspected, therefore, that this genetic algorithm was, perhaps, prematurely converging to a sub-optimal solution.

Experiment E

This was a repeat of Experiment D with a larger population size of 400. This was to check that Experiment D had not prematurely converged to a sub-optimal solution.

Contrary to what was expected, for some of the aerofoils that the genetic algorithm generated were not similar to the results found in Experiment D. For these aerofoils lift-drag ratio calculated was unrealistic.

Since it was suspected that the vortex panel method was producing inaccurate results in some parts of the search space, as was the case in Experiment B, an investigation of the landscape around the generated aerofoil was undertaken (as detailed in Appendix A). From this investigation, it was apparent that the accurate calculation of both lift and drag was not possible for values for the position of maximum thickness above about 85%. More pertinently to the problem encountered on this run, it could be seen that although the lift was calculated accurately at low values of thickness (between 2% and 3% of chord length), the calculation of drag was not.

This problem was not encountered when a smaller population size was used because this inaccuracy only occurs in a small part of the search space (it relies on the other parameters, such as camber angles, being in certain ranges). With a large population this area of the search space is more likely to be encountered either when the initial population was formed or during the optimisation.

Experiment F

This was a repeat of Experiment E with the lower bound on aerofoil thickness raised from 2% to 3%, in order to avoid the problems encountered with faulty fluid analysis at small thicknesses. This aerofoil generated by the genetic algorithm closely matched the solutions found in Experiment D with a lift/drag ratio of 26.7, averaged over the ten runs.

Experiment G

This run repeated Experiment F but allowed the attack angle to vary from a lower bound of -4° to an upper bound of 4°. The solution again produced unrealistic results. This was due to the same problem encountered in Experiment B, where the vortex panel was unable to calculate drag correctly for aerofoils with large camber angles and negative attack angles.

Experiment H

This run repeated of Experiment G, but with the maximum camber angles restricted to 10°. This run again produced realistic solutions. It should, however, be noted that this solution was considerably less fit than the aerofoil profile found in Experiment F. The aerofoils produced had a lift/drag ratio of 11.8, averaged over the ten runs, compared to 26.7 for Experiment F, despite the fact that this run had a considerably larger search space (attack angle was not included in Experiment F). This is discussed further in Section 3.5

3.4.2 Bézier Representation

Experiment I

The population was initialised by perturbing the control points from those of a given aerofoil profile within a set of specified bounds. The best individual had an unrealistic fitness of 8.57e-5 (C_l/C_d of 11700). Numerous runs were undertaken with various initial base aerofoil profiles, each produced similarly unrealistic results to these.

3.4.3 Smooth Bézier Representation

Experiment J

It was thought that one possible cause of the analysis problems encountered in Experiment I was the presence of 'kinks' in the aerofoil shapes. Therefore, a second Bézier representation was tried in which C_1 continuity was imposed between the Bézier curves except at the trailing edge. The population was initialised by perturbing the control points from those of a given aerofoil profile within a set of specified bounds. The best individual had an unrealistic fitness of 5.55e-8 (C_l/C_d of 1.8e6). Again numerous runs were undertaken with various initial base aerofoil profiles, each produced similarly unrealistic results to these.

3.5 Discussion

3.5.1 The Genetic Algorithm

The genetic algorithm performed well on this problem. For problems where the vortex panel analysis was able to accurately model the flow the optimiser produced aerofoils which had geometries which looked as though they optimised the criteria specified and had realistic lift and drag coefficients. In Experiments A, B, C, E, G and H, the GA consistently found very similar solutions on each run, suggesting that the algorithm was indeed finding the global optimum. In the case of Experiments D and F, different runs converged to two slightly different aerofoils, but each had very similar fitnesses.

The number of evaluations used was large. It is difficult to say how much larger the number of evaluations used by the GA was above a traditional optimisation algorithm without implementing such an algorithm. This was not a problem with the vortex panel method since with this method evaluation was relatively quick. If a more computationally expensive method was used, as discussed in Section 3.5.3, then this might be a problem.

3.5.2 The Shape Representations

3.5.2.1 Aerofoil Parameterisation

The aerofoil parameterisation was found to work well. The representation produces a set of shapes through which the GA seemed to be able to search well.

There were some problems encountered where the analysis method could not accurately assess the shapes. However, by using bounds on the parameters it was possible to control the search space over which the optimisation was undertaken. Those areas of the search space that could not be modelled effectively could therefore be avoided.

This did, however, throw up an interesting difficulty. In Experiment F an aerofoil was found with a lift-drag ratio of 26.7. This run used bounds on camber angles of 0° and 30° and with attack angle set at 0°. The optimal shape was found to have a high degree of camber. A similar optimisation was undertaken in Experiment H, but with attack angles allowed to vary from –4° to 4°. However, because the vortex panel analysis produced inaccurate results for shapes with high camber and negative attack angles, the camber angles were restricted to below 10°. Experiment H found an optimal shape with a lift-drag ratio of only 11.8. By adding attack angle as a variable, the intention had been to increase the size of the search space of shapes and thus possibly to find a better solution. This, though, had made available part of the search space that could not be analysed accurately. The subsequent constraint placed on camber angles made to avoid this area inadvertently also removed the optimal solution found in Experiment F. This is shown schematically in Figure 3-3.

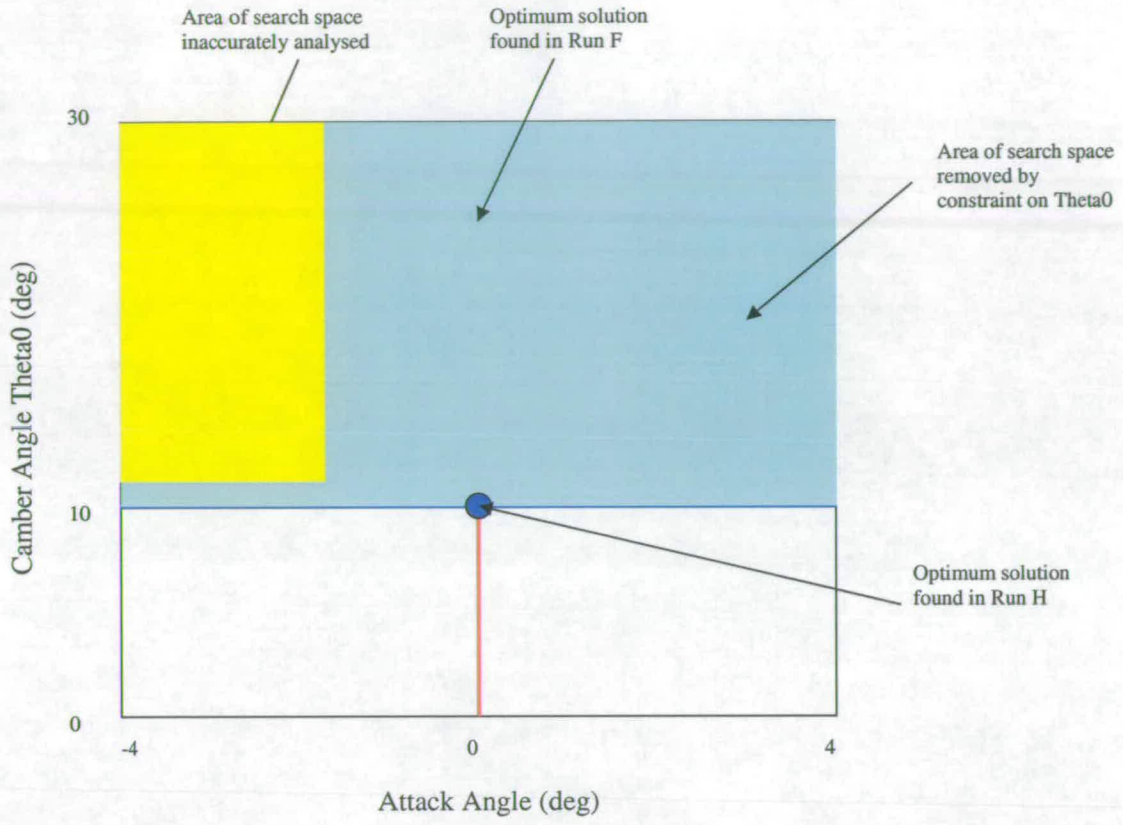


Figure 3-3 Addition of Constraint on Theta0 in Experiment H

The main reservation about this representation was that it was relatively simplistic. The range of potential shapes it could generate was fairly limited. For instance, it would not be able to produce the shapes of some of the modern aerofoil profiles. More complicated parameterisations are available, although they typically use more variables. [Giannakoglou 2002] reports on the use of some of these parameterisations with genetic algorithms. [Samareh 1999] surveys shape parameterisation techniques for aerodynamic optimisation problems.

3.5.2.2 Bézier Representations

The intention behind using a Bézier curve representation was to greatly increase the range of potential shapes that could be produced, so that the ability of the genetic algorithm to find a global optimum on a multi-modal landscape could be exploited.

However, this proved to be difficult since the vortex panel method analysis was unable to model accurately many of the shapes generated. For many shapes the analysis seemed accurate, but for others very high lift coefficients or low drag coefficients were assigned. If the analysis had assigned poor values for lift and drag for those shapes that it could not analyse accurately, this might not have been too much of a problem. The optimisation would then just be to find a shape which performed well and which could be analysed accurately. Instead, often shapes were assigned unrealistically good fitness and so the genetic algorithm would evolve towards those areas of the search space that could not be accurately assessed by the vortex panel method.

To compound this problem, unlike with the aerofoil parameterisation, it proved to be difficult to define where the analysis did not work. With the aerofoil parameterisation it was possible to place bounds on the variables in order to avoid areas of the search space that could not be analysed. For Bézier representations this proved to be impossible to do, since it was too difficult to characterise, in terms of the combinations of the control point co-ordinates, those areas of the search space which the vortex panel could not accurately analyse.

3.5.2.3 Further Comments on Shape Representations

The approach to using Bézier curves to represent the aerofoil shapes used in this study combined with the vortex panel method of fluid modelling was unsuccessful. There are a number of possible solutions to this. An analysis method that is better able to model these general shapes could be developed (as discussed in the following section). Alternatively, the positions of the control points could be more tightly constrained in some way so that only analysable shapes are generated. Indeed, Bézier

curves and splines have been frequently used successfully in the literature [Giannakoglou 2002].

Nevertheless, whatever shape representation and analysis code is used a number of considerations must be taken into account, when deciding on a representation to use:

- The representation should be selected so that the search space is likely to contain the optimal shape. Often, the designer does not know *a priori* the likely nature of the optimal shape. One possible solution to this is to make the search space as large as possible.

- The analysis code can analyse accurately all the shapes in the search space. The Bézier representation was found to fail on this point, because the vortex panel method could not analyse all of the shapes the Bézier representation could produce. Even in the case of the aerofoil parameterisation, where almost all of the search space was accurately analysed, the presence of even a small area of the search space where this was not the case, caused problems.

- It is desirable to keep the number of shapes in the search space as small as possible. Clearly, the computational cost of searching through the search space depends on the algorithm employed, the way it searches through space and the nature of the problem landscape. However, all other things being equal, the smaller a search space is, the less effort is required to search through it. Often this means having as few variables as possible. It should be noted that this conflicts with the first consideration above.

- The optimisation algorithm should find it easy to search through the space of potential shapes. As an example, consider increasing aerofoil camber to increase lift. In the aerofoil representation, increasing the camber angle changes the aerofoil in such a way as to produce an otherwise similar aerofoil with higher lift. An optimiser with an operator that changes the camber angle can easily explore the possibilities of increasing lift by increasing camber angle. With the naïve

Bézier representation used in this study, changing the aerofoil camber can only be done by moving a number of control points. If the optimiser has no operator for moving a number of control points at once, then the uncambered aerofoil and cambered aerofoil (but which are otherwise similar) are far apart in the search space and it is therefore difficult to move between them.

This topic is covered in more detail in Chapter 6.

3.5.3 The Evaluator

The vortex panel method worked effectively for most reasonably shaped aerofoil shapes. It makes assumptions about the type of flow being modelled, namely that the flow is steady, inviscid and incompressible. However, these assumptions are acceptable for the flow regime for which aerofoils were being designed in this study (Reynold's number of about $5x10^6$). It had the advantage of being a much quicker analysis method than available alternatives such as the finite volume method.

Problems were encountered, though, when this method was used to analyse more unconventional shapes. The vortex panel is not able to model boundary layer separation and so cannot predict stall. This inability resulted in poor results with the aerofoil parameterisation when shapes with high camber angles and negative attack angles were analysed. Similarly, for many of the shapes generated by the Bézier representation inaccurate results were generated.

If there was a requirement to evaluate more general candidate shapes a different evaluator is required. An Euler flow solver [Ferziger & Peric 1996] [Versteeg & Malalasekera 1995], is available which is able to accurately solve the flow around more general shapes than could be evaluated with the vortex panel flow solver. This evaluator has been written but at time of writing this thesis had not been fully integrated with the rest of the shape optimisation application.

The Euler solver assumes that the flow is compressible and inviscid. This solver takes the co-ordinates of thirty-two points around the candidate shape with a higher density of points at the trailing edge. A 32 x 32 'O' shaped grid is then produced up to a distance of about five chord lengths from the aerofoil. This is about the lowest resolution which can be used, whilst still retaining reasonably accurate results.

The primary disadvantage of this method is the time taken for each evaluation (approximately 5 seconds on a Pentium 3 PC). If similar numbers of evaluations were required as were used in the runs described in Section 3.4 (typically thousands) optimisation would take a considerable length of time. For Experiment F 8400 evaluations were undertaken, with the Euler solver this experiment would take approximately 12 hours.

The Euler solver iterates towards a solution. It is possible to start an evaluation using the solution of the flow for a previous similar aerofoil. This will reduce the time taken to converge. It would therefore be possible to produce a library of solutions for a range of shapes. The evaluator then selects as its starting conditions the solution from the most similar aerofoil in this library.

At early stages of the optimisation highly accurate solutions are not required since it is only necessary to rank aerofoils from good to bad. At first, therefore, the solution can be stopped when only weak convergence conditions have been achieved. This will reduce the evaluation times at the beginning of the optimisation. Throughout the optimisation, the solutions will be required to be increasingly converged so that the accuracy of the solution will be increased when fine-tuning of shapes is being made at the end of the optimisation. This will require the solutions of those shapes that survive during the early stages of the optimisation to be restarted so that increased convergence can be achieved. It may therefore be advantageous to include the solutions for all surviving shapes in the library of starting conditions.

For a more general evaluator it may be possible to use full Navier-Stokes solvers [Ferziger & Peric 1996] [Jameson 2001] [Versteeg & Malalasekera 1995] to evaluate shapes for shape optimisation [Nemec & Zingg 2001]. Use may also be made of unstructured grids, so that some restrictions imposed on candidate shapes to enable the structured grid to be used may be lifted.

3.6 Conclusions

A genetic algorithm was applied to the problem of optimising the shape of an aerofoil with the aims of investigating the appropriateness of various shape descriptions for aerofoil shape optimisation and investigating the efficiency of genetic algorithms for aerofoil shape optimisation

Three shape representations were used, an aerofoil parameterisation, a four Bézier curve representation and a four Bézier curve representation with C^1 continuity imposed at the joins between curves. The aerofoil parameterisation worked well as long as care was taken to ensure only shapes were generated that the vortex panel fluid analysis could analyse. Further work would be useful with more sophisticated aerofoil parameterisations (with more variables).

Less success was had with the Bézier curve representations. Many of the shapes that were produced could not be accurately analysed by the vortex panel method and the genetic algorithm therefore evolved towards unrealistic shapes. This highlighted the need to ensure that all the shapes in the search space set up by a shape representation can be analysed accurately.

Fluid analysis was undertaken with a vortex panel method that was written for this problem. This method had the advantage of being quick and able to model the flow around 'reasonable' aerofoil shapes but was unable to analyse more general shapes.

The genetic algorithm worked well with the parameterised aerofoil representation. The GA consistently found the same solution on similar runs, suggesting that the algorithm was indeed finding the global optimum. This was also checked by repeating some runs with much larger population sizes.

4 Voxel Based Genetic Algorithm Optimisation

4.1 Summary

A voxel-based shape representation, when integrated with an evolutionary algorithm, offers a number of potential advantages for shape optimisation. Topology need not be predefined, geometric constraints are easily imposed and, with adequate resolution, any shape can be approximated to arbitrary accuracy. However, lack of boundary smoothness, length of chromosome and inclusion of small holes in the final shape have been stated as problems with this representation. This chapter describes two experiments performed in an attempt to address some of these problems. Firstly, a design problem with only a small computational cost of evaluating candidate shapes was used as a test-bed for designing genetic operators for this shape representation. Secondly, these operators were refined for a design problem using a more costly finite element evaluation. It was concluded that the voxel representation can, with careful design of genetic operators, be useful in shape optimisation. However, since the boundary of a voxel model is necessarily not smooth, difficulties were encountered in ensuring that the finite element analysis produced accurate results.

4.2 Introduction

4.2.1 Voxel Shape Representation

The work described in this chapter involved investigating the possibility of replacing the usual boundary representation of the shape usually used for shape optimisation with a cellular representation. The cellular representation chosen in this work used voxels, which partition the design space into rectangular regions or boxes that are then assigned a binary full or empty value. This approach was motivated by a number of potential advantages [Smith 1995a]:

- any shape can be represented to an arbitrary accuracy by increasing resolution,
- it is easy to convert existing engineering solutions into voxels,
- they map naturally to the representations frequently used by genetic algorithms (GAs),
- domain knowledge can be readily incorporated,
- geometric constraints can easily be applied, and,
- the topology of candidate shapes is not predefined.

However, Watabe and Okino [Watabe & Okino 1993] state the following objections to voxels:

- the occurrence of small holes in the final shape,
- the long length of the chromosomes,
- the expectation that crossover operators would be ineffective, and,
- the lack of smoothness in the shapes' outlines.

4.2.2 Aims

Given the potential advantages of a voxel representation, it was considered worthwhile addressing these difficulties. Specifically, the aims of this work were:

- to determine the suitability of voxels as a geometric model for use in shape optimisation, and,
- to design suitable operators for a GA optimiser to use with such a representation.

4.2.3 Acknowledgement

The work described in this chapter was done in collaboration with Peter Baron, an MSc student, under the supervision of the author. It is included in this thesis as it was instrumental in the development of the arguments put forward in this thesis.

4.3 Experiments

Two experiments were devised in order to investigate the voxel representation. Firstly, a simplified beam design problem was formulated for which the cost of evaluation would be small. Using this problem as a test-bed, a number of operators were designed. Secondly, an annulus design problem was tackled using a finite element analysis. The computation cost of evaluation in this case was thus much greater. The usefulness of the operators designed in the first experiment could then be evaluated with a more difficult design problem.

4.3.1 Simplified Beam Design

A prototypical mechanical engineering problem is that of optimising a beam to support various loads with a minimal amount of material. Evaluation of the candidate cross-sections was made using bending theory for symmetrical beams, considering only normal stresses [Gere and Timoshenko 1984]. This is an oversimplified model, but is sufficient to test whether the potential problems with a voxel representation outlined above do pose a problem in practice. The maximum stress constraint imposed by the physics model used in these experiments is summarised below.

$$\left|\frac{My}{I}\right| < \sigma_{max} \qquad \textit{for all voxels}$$

where: σ_{max} is the maximum stress allowed within any given area (voxel);

M is the bending moment;

y is the distance of the voxel from the neutral axis of the shape;

I is the second moment of area of the candidate cross-section.

The neutral axis of a shape is defined as a horizontal line that passes through the centre of mass of the shape. As a voxel representation uses areas which are all of uniform size and density, the centre of mass can be found by taking the average of the positions of all occupied voxels. The second moment of area is approximated in the discrete representation by summing the moments of each voxel, that is:

$$I = \sum_{i=0}^{n} ay_i^2$$

where a is the area of a voxel.

In the real world, the solution to this problem would correspond to an I-beam, but that also requires a web to connect the two flanges of the beam together. In a design based on a full calculation with shear stress, the web would be necessary to counteract this additional stress. However, as shear stress is not represented in this problem, a connectivity requirement in the form of a repair step was added, whereby all pixels must be connected to a seed voxel in the centre top edge of the beam. In addition, a straight web was enforced before the connectivity repair step. This was found, in formative experiments, to prevent the formation of a crooked web (as the physics model used does not prevent this), and improve slightly the results obtained.

To try to ensure that the alterations and improvements made to the GA will also prove beneficial to the real-world problem, it was decided not to concentrate on fine-tuning any of the various parameters available, but rather to focus on the design and operation of various new operators. Therefore, parametric variations were restricted to an absolute minimum and were used only to determine the approximate values required to gain reasonable advantages from the new operators. Therefore in the following experiments, the following parameter settings remain constant unless mentioned otherwise:

Beam Dimensions	$= 0.05 \times 0.10$	m
Bending Moment	$= 13000$	Nm
Voxel Grid	$= 32 \times 64$	voxels
Maximum Stress Allowed	$= 2 \times 10^8$	Nm^{-2}

4.3.1.1 Experiments Using the Naïve Genetic Algorithm

The first set of experiments with a 2D representation treated the chromosome as a long one-dimensional binary string that wrapped around at the vertical edges onto new lines to form the two-dimensional cross-section. Standard two-point crossover (p_c = 0.35) and bitwise mutation (p_m = 0.001) were used in conjunction with a generation GA with a population of size 20. GENITOR-style rank-based selection [Whitley 1989] was used throughout. From the above, the fitness function, F, used was of the following form:

$$F = V + \frac{S}{1000\,\sigma_{max}} + max\{(S - \sigma_{max}), 0\}$$

where: V was the count of active voxels (proportional to weight),

S the maximum stress of any voxel,

σ_{max} the value of the maximum stress constraint,

k the constraint penalty multiplier (set to 5×10^{-5} according to the results of formative experiments).

With this particular optimisation problem, the difficulty lay not in getting a valid solution, but in getting a near optimal-mass solution. The first experiments were relatively unsuccessful in this regard: the results after 2000 generations were full of small holes and had extremely uneven inner edges. This can be seen in the typical end-of-run results shown in Figure 4-1 (the numbers represent the fitness values of each individual).

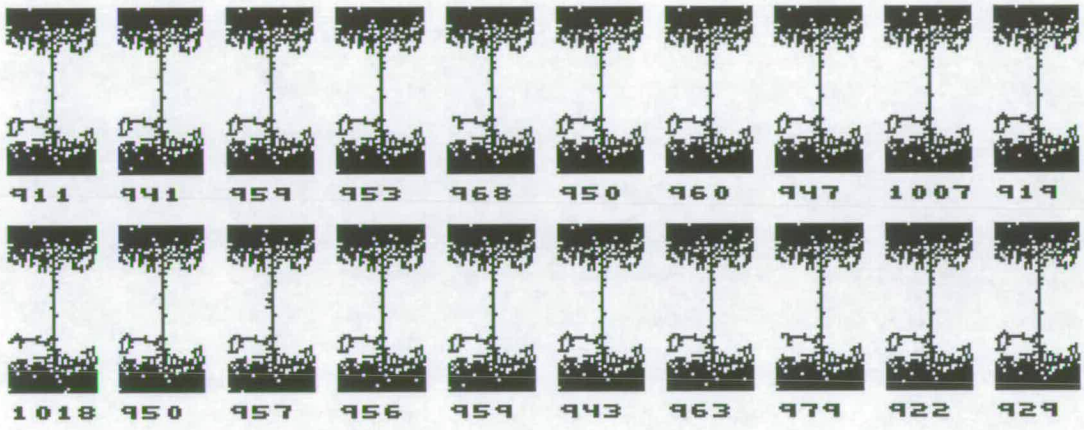


Figure 4-1 Typical End Population (with fitnesses)

The stresses were concentrated at the vertical extremes of the beam, so the material in the middle contributes less towards the beam's ability to withstand the load, and therefore as we are trying to minimise the mass of the beam, the material is more usefully employed at the extremes of the beam. The GA, even in this simple standard form, rapidly removed material from the middle of the cross-section, and in the later stages of the experiments was observed to be moving material from low stress areas into high stress areas where holes were left near the extremities.

However, this first naïve GA approach took an extremely large number of evaluations in order to make significant progress, and this is not acceptable as later experiments would have a greatly increased evaluation time due to the integration of the finite element package. The rate of improvement was also seen to decrease as the run continued, levelling off to almost none at all by the end of the run. This means that the GA was not finding any further improvements to the chromosome, and as the results are visibly poor, it indicates a general weakness in the operators being applied.

Attention was therefore concentrated towards improving the GA operators, in order to achieve greater benefits during the early search period, and to produce better quality final results.

4.3.1.2 The Smoothing Mutation Operator

The smoothing operator experiments were an attempt to address directly some of the weaknesses of the voxel representation by devising a new specialised operator, which should aid the search by reducing the number of small holes and ragged edges produced by the GA. The new operator was intended to be capable of easy expansion from two-dimensions to n-dimensions, so that it would continue to be useful in the case of higher dimensional problems using the voxel representation.

This operator selects an area with both random position and size ranging from 2 pixels to 1/4 of the dimensions of the grid. The most common value for the pixels in the area selected was then found and written to all of the pixels in that area (see Figure 4-2).

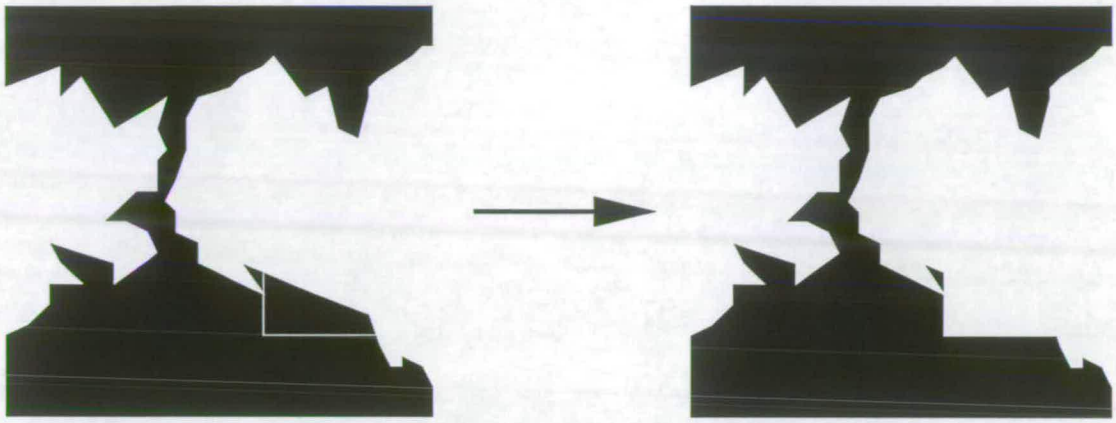


Figure 4-2 **The Smoothing Operator**

The GA parameters used were the same as before and the new operator was applied in addition to the previous mutation and cross-over operators – application of this operator to 60% of the chromosomes in the population was found, in formative experiments, to give the best results. The GA configuration was otherwise unchanged, though the number of generations was limited to 1500 in this case.

Figure 4-3 **Typical End Population for GA with Smoothing Operator after 1500 Generations (fitnesses are shown)**

Comparing Figure 4-3 which displays some typical end-of-run population members with earlier results (shown in Figure 4-1), shows just how effective this domain specific approach to operator design has been, especially at eliminating isolated holes and reducing ragged edges.

4.3.1.3 UNBLOX: An N-dimensional Crossover Operator.

The two-point crossover operator that had been used up to this point treated the chromosome as a one-dimensional string of bits and therefore suffered from a problem with linkage; voxels that are adjacent in a two-dimensional grid are not necessarily adjacent in the one-dimensional string. This separation increases the possibility that useful building blocks (areas of the grid that contribute to a higher overall fitness evaluation) will be disrupted during the crossover procedure.

[Cartwright & Harris 1993] describe the use of the UNBLOX crossover operator, which was specifically designed to overcome these limitations with conventional two-point crossover. This operator swaps a rectangular area of the grid instead of the sub-string swapped by two-point crossover. If the area overlaps an edge of the grid then it is made to 'wrap-around' to the opposite side. The size and location of the area to be swapped are both selected at random, and in this implementation the area was restricted to a minimum size of two voxels per dimension in order that the operator would always have some effect when applied.

The crossover operators were used with the standard probability of 0.3 per chromosome and no changes were made to the standard algorithm or to any of the other parameter settings described earlier. The graph in Figure 4-4 shows the results of three experiments using each of three crossover operators: the UNBLOX operator, standard two-point crossover and uniform crossovers [Goldberg 89]. The average fitness, over ten trials, of the best individual in the population is plotted against the generation.

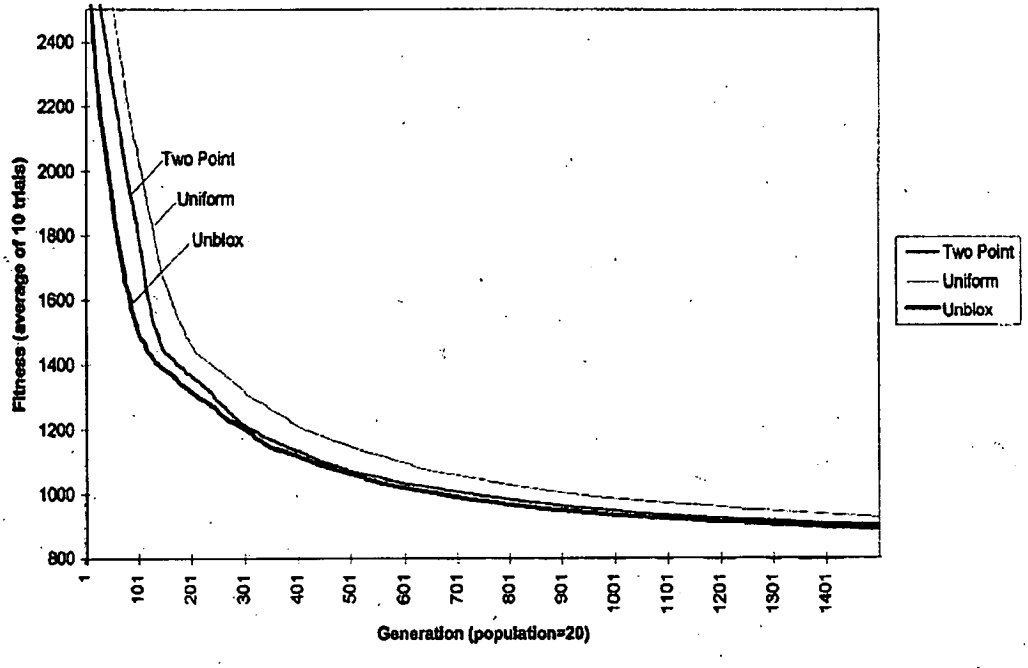


Figure 4-4 Effectiveness of Various Crossover Operators

The results confirm that the UNBLOX operator does indeed perform better than either the two-point crossover or the uniform crossover techniques on this problem. The rate of descent of the UNBLOX line is quicker, indicating that the population converged to good solutions faster with this approach than with the other operators, and the eventual end result after 1500 generations had a slightly better fitness value than those produced by the other techniques.

4.3.1.4 Two Dimensional Mutation Operators

A new mutation operator was designed which scrambles the contents of a randomly selected rectangular area of the voxel grid. It is referred to here as the 'two dimensional' operator. This operator can be easily modified to work in n-dimensions, and affects a relatively small area of the chromosome intensively in the selected rectangular area, in the same way as for the smoothing mutation operator. A second, somewhat altered, version of this mutation operator was also designed and tested in these experiments called the 'two-by-two' area mutation operator. This operator uses a fixed mutation square of two by two voxels and was designed to be applied only if at least one voxel in the mutation area is already active. It was observed that most of the modifications need to be made to the surface or interior of the evolving shape and that very little benefit will result from flipping isolated voxels in the middle of the void areas. The choice of a fixed two by two area was motivated by the observation that most of the irregularities on the surfaces would fit into such an area and that with only sixteen permutations possible (four binary bits), the probability of mutating a poor quality area into a more fit variation would be reasonably high.

The new operators were again applied, in addition to the original bitwise mutation operator, with a probability of 0.25 per chromosome of being applied. After each application, there was a decreased probability of the same operator being applied again, with the probability of a further application being decreased to one half of its previous value each time. The experiments were performed ten times for each of the three alternative mutation combinations, over a period of fifteen hundred generations.

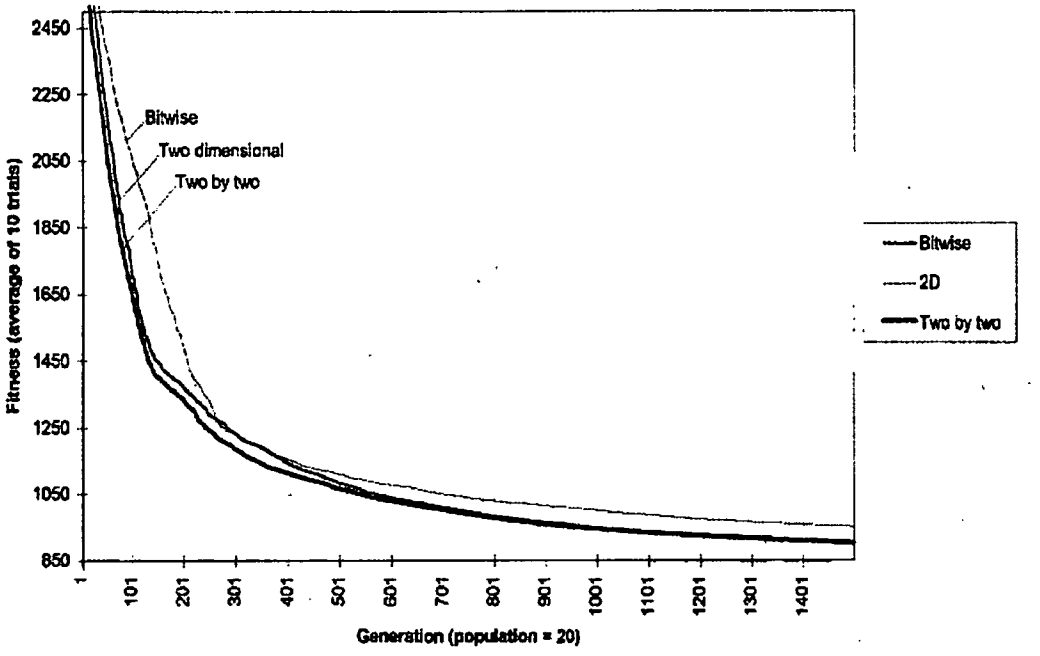


Figure 4-5 Effectiveness of Various Mutation Operators

The graph in Figure 4-5 shows the results of three experiments: 2D and bitwise mutations, two-by-two and bitwise mutations and bitwise mutation alone. The average fitness, over ten trials, of the best individual in the population is plotted against the generation.

The addition of the 'two dimensional operator' generally results in better performance than the bitwise operator alone, though the two lines do meet between generations 300 to 400. The steeper descent of the two dimensional operator line indicates that early performance was especially improved, and the final result after fifteen hundred generations is significantly better than previously. The 'two-by-two' operator offers a similar rate of improvement during the early stages of the trial, a slightly better performance between generations 100 to 600 and finally converges with the 'two dimensional' operator's line at about generation 1000. This seems to

indicate that although offering early benefits to the optimisation, it is not better than the 'two dimensional' operator in the long run.

In conclusion, two new mutation operators were designed with the particular intention of directly addressing the perceived problems with the prior optimisations. Both of the new operators were found more effective than the previous uninformed bitwise mutation, producing benefits to both the rate of early improvement and the final quality of solution generated.

In the absence of any other clearly distinguishing features, the 'two-by-two' operator will be used during the further experiments as it offers a speed advantage over the two dimensional mutation operator outlined above.

4.3.1.5 Conclusions on the Beam Design Problem

The results have shown that although a naïve GA does indeed suffer from the problems suggested by [Watabe & Okino 1993], a small selection of operators informed only by domain knowledge about the representation, will effectively solve each of these difficulties.

The final system uses a normal bitwise mutation operator in addition to the two new mutation operators, smoothing, and 'two-by-two'. The smoothing operator rapidly cuts away unwanted areas of material during the early stages of the optimisation and can help to smooth ragged edges and fill small holes later. The two-by-two mutation operator is highly effective at both smoothing off ragged edges and at filling in small holes in the material if they occur in undesirable places. Finally, the two-point crossover operator has been replaced by the n-dimensional UNBLOX operator.

4.3.2 Annulus Design Problem using Finite Element Analysis

The experiments undertaken with the simplified beam design problem outlined in Section 4.3.1 led to the design of effective GA operators for manipulation of 2D shapes. This section details further experiments undertaken to apply these operators to a more difficult design problem. The problem chosen was to design a jet-engine annulus. The finite element method was chosen as the analysis technique. Initially, for ease of implementation, the voxel shape description was directly used as the finite element mesh.

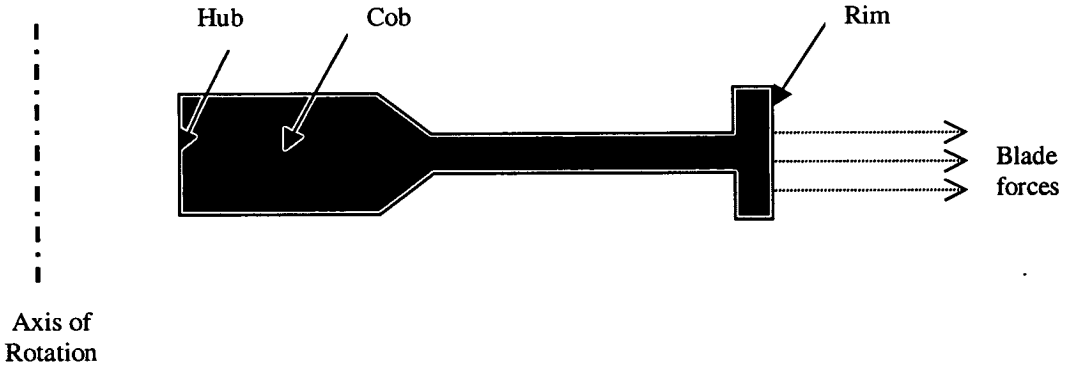


Figure 4-6 Annulus Axisymmetric Cross-section

4.3.2.1 The Annulus Design Problem.

The full original specification of this problem came from an industrial source and is taken from [Smith 1995b]. The problem is to design a jet-engine annulus. This part is subjected to loading due to rotation and due to the attachment of the turbine blades to its outer circumference. The part is axisymmetric around the axis of rotation, and

consequently it reduces to the two-dimensional shape optimisation problem shown as Figure 4-6.

The optimisation involved reducing the mass of the annulus whilst observing a series of four separate stress constraints at discrete locations in the annulus. The constraints relate to the hoop stresses at the inner and outer circumferences and the radial stresses along the centre line of the annulus. The stress constraints to be observed were, in descending order of importance:

Hub hoop stress	< 1330 MPa
Rim hoop stress	< 396 MPa
Inner radial stress	< 741 MPa
Outer radial stress	< 334 MPa

4.3.2.2 The Fitness Function

The GA fitness function was defined as an objective (weight of the annulus in kg, and a factor to minimise the *total* stress at the four test points, in MPa) plus a sum of penalty terms if one of the four stress constraints was broken. Therefore, the GA maximised:

$$F = \frac{\sum_i \sigma_{max(i)}}{\sum_i 1000\, S_i} - annulus\_weight - \sum_i \left(k\, i\, max\left\{\left(S_i - \sigma_{max(i)}\right), 0\right\}\right)$$

Constraint penalties were applied if any of the four constraints limits $\sigma_{max(i)}$ were exceeded by the stress, S_i, measured (in MPa), The constraints were ordered in importance by using $4 \times k$ for the most important, $3 \times k$ for the second most important, $2 \times k$ for the next and $1 \times k$ for the least important constraint.

4.3.2.3 Results from the basic system.

Again, a generational GA with a population of size 20 and GENITOR-style rank-based selection was used. The UNBLOX, smoothing mutation, and 2-by-2 mutation operators were applied sequentially with probabilities 0.3, 0.8, and 0.8 respectively (on the basis of formative experiments). A 62 by 27 voxel grid was used to represent the annulus and the constraint penalty, k, was set to 0.00005. The settings used for the annulus were:

Dimensions of design space	=	0.25 x 0.05	m
Radius of hole	=	0.10	m
Blade force	=	10×10^5	N rad^{-1}
Young's modulus	=	2.238×10^{11}	N m^{-2}
Material density	=	8.221×10^3	kg m^{-3}
Revolution speed	=	1571.0	rad s^{-1}

The basic system was first applied without further modifications to the annulus optimisation. However, the problem as specified was very tightly constrained, which meant that the attempts to solve this problem using random population initialisation violated all of the stress constraints by large amounts. Also, the rate of improvement in the population, when extrapolated beyond the time period allocated to the experiments indicated that a valid solution would not be found for some considerable number of generations still to come.

To circumvent this problem, the population was instead initialised with a selection of variations on the annulus design supplied with the original specification, which were modified further by an aggressive random mutation operator that added and removed small areas of material over the surface of the annulus design. This kind of intelligent initialisation was thought reasonable, as a user will often want to start the GA with

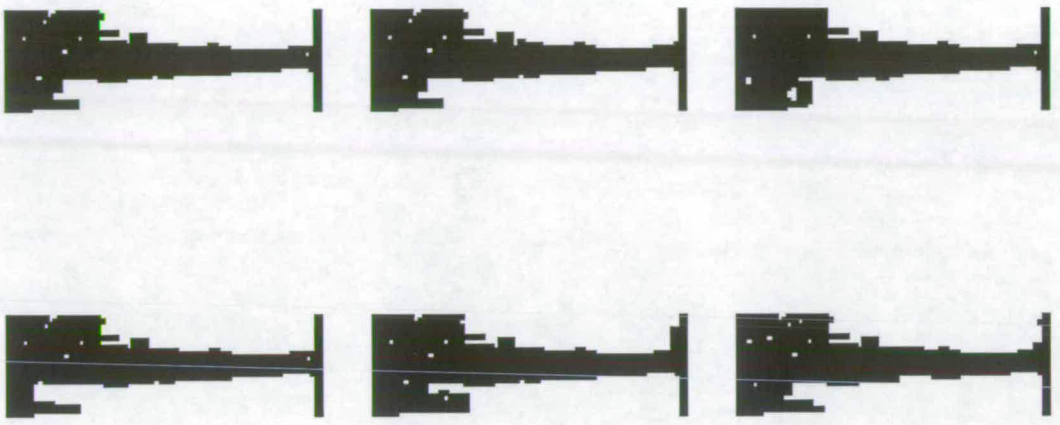


Figure 4-7 **Results of the Basic Annulus Optimisation after 75 Generations**

existing designs in order to see what improvements can be made. Even when a totally new shape is being designed, the user would normally have some expectation about the final form, which could easily be used to initialise the population. The intelligent initialisation approach meant that the initial population was not unreasonably far outside of the stress constraints, yet supplied the optimisation with sufficient variation that the population did not rapidly converge onto a single solution. Some of the results from this basic system can be seen in Figure 4-7 that shows six members of the population after seventy-five generations.

The results shown in Figure 4-7 were poor. The lack of symmetry around the horizontal axis and the uneven edges were just the most visible failings in this set of results. A second problem was the occurrence of large stresses at the corners of elements on the edge of the shape.

4.3.2.4 Improvements made to the system

4.3.2.4.1 Use of Symmetry

It was known that a solution to the annulus design problem should be symmetric about a radial axis. It was therefore decided to utilise this domain knowledge and thus reduce the search space of the problem. The central line of voxels along the axis of symmetry is not mirrored as it is now enforced by the GA to be always turned on – this also provides a guaranteed central line of elements for the stress measurements to be taken from.

4.3.2.4.2 Mesh Improvement

It was found in the initial experiments for the annulus design problem that directly using the voxel description of the geometry as the finite element mesh caused problems with high stresses caused by corners in the mesh. It was therefore decided to attempt to separate the geometry model and mesh. There were several possible approaches that could have been taken. An approach that was considered was to use interpolation splines to form a smoothed edge. The voxels would then act as a 'skeleton' and the spline as a 'skin'. A mesh generator could then produce a mesh whose density could then be independent of the voxel model. However, for this prototype system, it was decided simply to add triangular elements at the corners. Whilst this was a far less elegant solution it was much simpler to implement.

These new triangular elements were created by specifying connections between groups of three nodes in the element connection file. These triangular elements were added to the shape at all suitable 'steps', which were identified by convolving the voxels in the shape against a series of four matching template masks. If each square in the mask matched the value of the voxels surrounding an empty voxel then the appropriate triangular element was created in the 'step'. The convolution masks and the triangles which they caused to be inserted are shown in Figure 4-8.

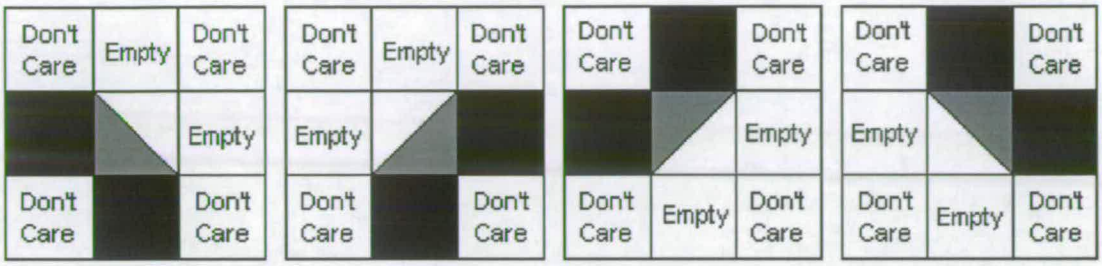


Figure 4-8 **Convolution Masks for Triangle Insertion Process**

4.3.2.4.3 Design of Operator to Remove Holes

The 'two-by-two' mutation operator (which can either fix holes or cause them to appear) was modified to only mutate areas where, as well as at least one voxel being turned on, at least one of the four voxels is also turned off. The result of this modification is that the two by two mutation operator can now only mutate at the boundaries of the shapes being formed, and consequently it should also help reduce the number of small protuberances.

4.3.2.5 Results of Improved System

The improved GA for annulus optimisation used the same settings as the basic system for all parameters, except that the chromosomal grid was set to 21 voxels high, which is mirrored due to the symmetry used to produce a voxel grid height of 41 voxels. The analysis was permitted to continue for 114 generations and this took approximately twenty-four hours in total. Some members of the final population created by the improved GA are shown in Figure 4-9. This displays three of the twenty individuals and shows a clear improvement in quality over the results generated previously. The small protuberances have been totally eliminated and only a few members of the population contain small holes. The rate at which a valid

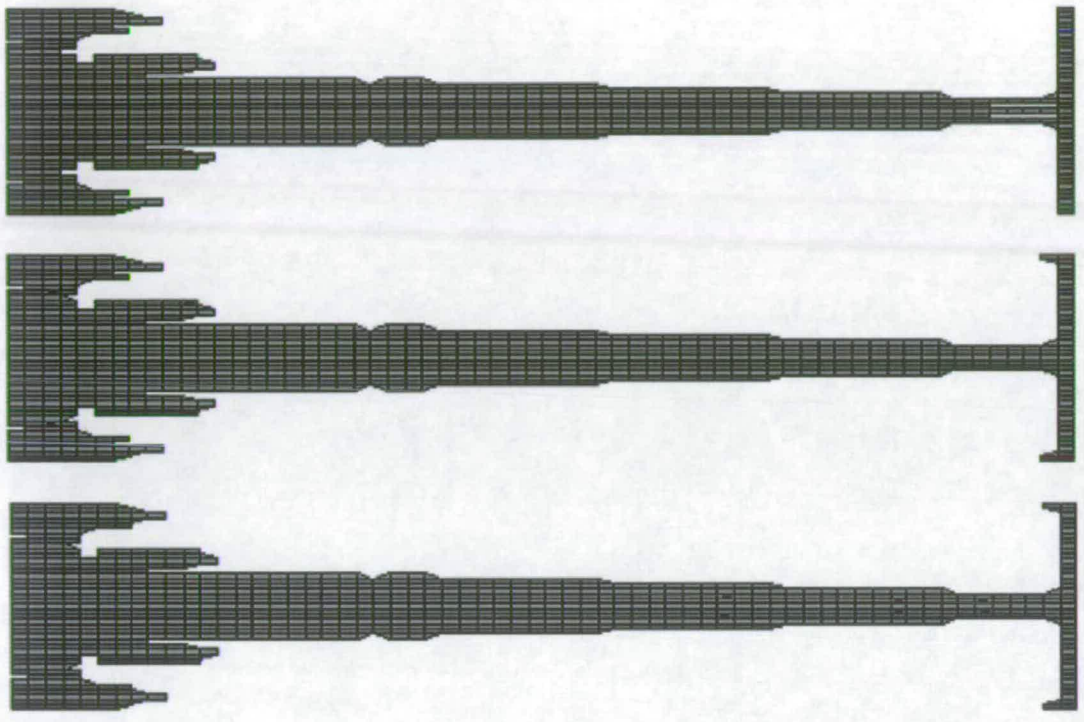


Figure 4-9 Final Annulus Cross-Sections from Improved GA

solution was found is considerably faster than the basic implementation, and once found, the GA continued to improve upon this solution even to the very last pass of this trial.

The annulus shapes produced can be seen to be unusual. It is proposed that the 'overhangs' present at the cob and the thinness of the neck are due to the inadequate specification used for the annulus and the method used to penalise constraint violation. Stress constraints were defined for four discrete points in the specification that was intended to be used with a parameterised shape description. This specification would be adequate for such a representation. However, with the voxel representation the optimiser was able to remove material with greater flexibility. At an optimal solution, one of the stress constraints is just inactive. Removing more material would then increase the stress to above the maximum value. However the

GA could improve the fitness value if, by adding material elsewhere, the position of high stress was moved from the point at which the constraint was assessed, as long as the amount of material added was less than that removed. If this explanation is correct, the problems do not lie with the voxel representation and could be solved by improving the specification and method of penalising constraint violation. This highlighted the necessity to express in a formal and unambiguous way what is the desired behaviour.

After using the finite element package to examine the solutions produced by this optimisation, it was possible to confirm that the use of the triangular elements to smooth the boundary worked as expected in reducing the amount of stress in the regions immediately surrounding a step. Figure 4-10 shows the stress values calculated by the finite element package for the voxels surrounding steps in two typical runs and clearly shows how the triangles permit the excess stress to be distributed more in a more evenly. Darker shades indicate higher stress.

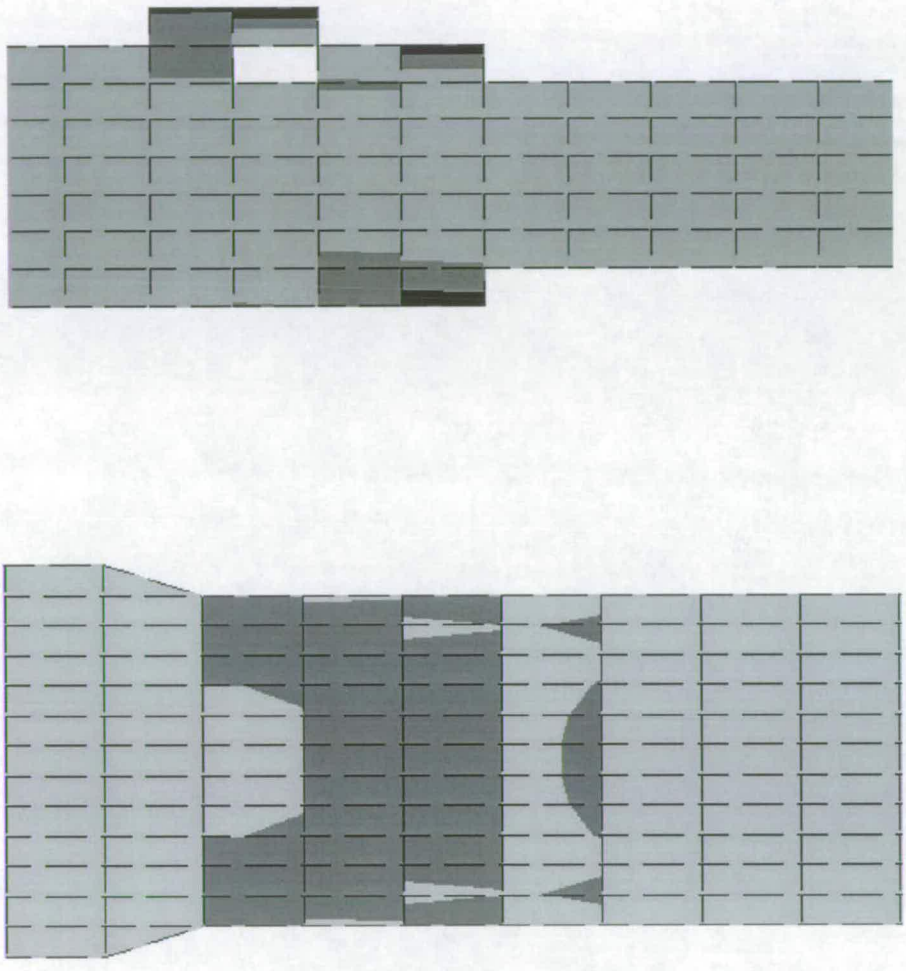


Figure 4-10 **Results without and with Smoothing Triangles**

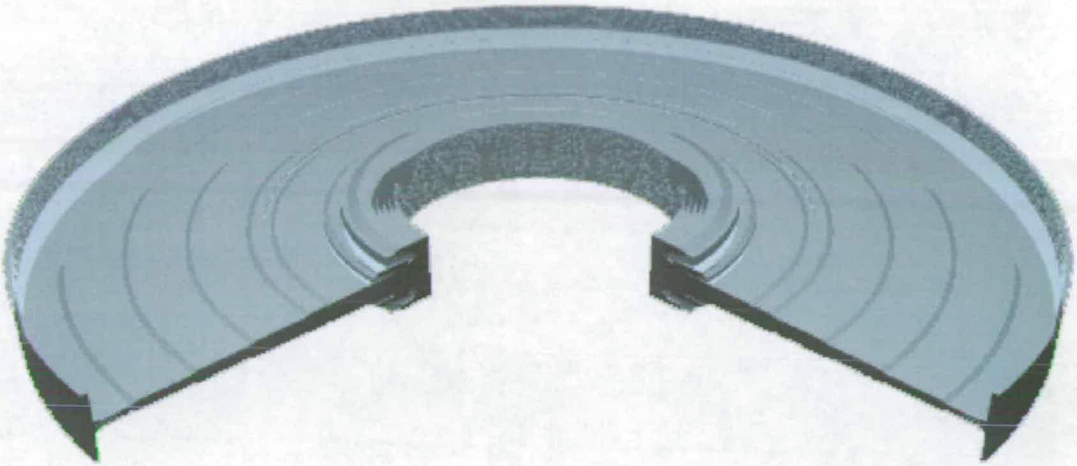


Figure 4-11 The Best Annulus Design from the Final Set of Experiments

4.3.2.6 Conclusions for Annulus Design Problem

It was found that the use of unmodified operators from the beam design problem was unsuccessful. However when the operators were modified, taking into account knowledge held about the annulus design problem, the results were more successful.

Difficulties were encountered in the direct use of the voxel shape representation as the finite element mesh. These were to some extent alleviated by the use of smoothing triangular elements. However, the full decoupling of the primary voxel-based shape description and finite element mesh would be desirable.

Due to the flexibility of the voxel representation in removing and adding material coupled with the GAs ability to exploit the whole search space, it was found that the specification of the problem needed to be more tightly defined.

4.4 Conclusions

Voxels were found to be a viable representation for shape optimisation with an evolutionary algorithm in 2D problems. They have a number of potential advantages over other representations such as parameterised boundary descriptions. It is easier to not predefine the topology (whilst it is possible to represent an arbitrary topology with a boundary representation, it is much more difficult to parameterise a b-rep model so that topology can be changed [Shapiro & Vossler 1995]), domain knowledge is easier to incorporate, geometric constraints can be easily applied and it is easier to convert existing solutions into such a description in order to 'seed' an initial population of shapes.

Experiments were undertaken on two design problems, a simplified beam design and a jet-engine annulus design using finite element analysis. During these experiments a number of difficulties inherent with this representation were addressed, primarily by use of specifically designed genetic algorithm operators which utilised domain knowledge held about the problems tackled. An n-dimensional crossover operator was used which provided linkage between adjacent rows of voxels and thus avoided the slow convergence found with a conventional crossover operator. An operator was designed to remove unwanted holes produced in candidate shapes and to smooth boundary edges.

The direct use of the voxels as the finite element mesh was found to be inadequate. Further work required involves the decoupling of the voxel representation and mesh.

The flexibility of the voxel representation along with the genetic algorithm's exploitation of the whole search space uncovered deficiencies in the specification, supplied by an industrial collaborator, used for the annulus design problem.

Finally, it should be noted that GA optimisations can easily be modified into interactive optimisation systems [Tuson *et al.* 1997] and in this case the computer would rely on an engineer's practical experience and knowledge of the problem domain to direct key choices in the optimisation process.

5 Application of Genetic Programming in a Solid Modeller

5.1 Summary

In this chapter an initial investigation into the use of genetic programming in a hybrid B-Rep / constructive solid geometry (CSG) solid modeller is described. In a CSG solid model the solid is represented with a tree structure of Boolean functions, such as 'union', 'intersection' and 'subtract', acting on a collection of primitive point sets [Mäntylä 1988] [Hearn & Baker 1994]. Genetic programming [Koza 1990] is a cousin of genetic algorithms, which acts on tree-structure chromosomes, rather than on the linear chromosomes traditionally used by genetic algorithms.

The ACIS 3D toolkit [Corney 1997] was used as the solid modeller. This toolkit has an interface based on the Scheme language. The genetic programming algorithm was implemented using this Scheme interface.

A test problem was used in which the aim was to regenerate a given test component. Genetic programming was used to manipulate a tree structure with Boolean operations at internal nodes and the primitive bodies making up the component as the leaves. The technique was found able to regenerate the test body.

It was hoped that this technique might be used to automate the difficult process of converting arbitrary B-Rep solid models into CSG models. However, these initial tests suggested that the computational time required to solve any realistic problem with this technique would be prohibitive.

5.2 Introduction

5.2.1 Genetic Programming

Genetic programming (GP) is an extension of genetic algorithms in which the chromosome is a tree structure (see Figure 5-1), rather than the linear chromosomes

used in traditional genetic algorithms (GAs). The technique was initially developed by [Cramer 1985] and has been extensively studied by [Koza 1990], [Koza 1992], [Koza 1994] and [Koza *et al.* 1999]. GP has been applied to a number of diverse areas including mechatronics design [Seo *et al.* 2003], modelling of waste treatment plants [Hong & Bhamidimarri 2003] and image classification [Agnelli *et al.* 2002]. [Ryan *et al.* 2003] demonstrates the current breadth of possible applications of GP.

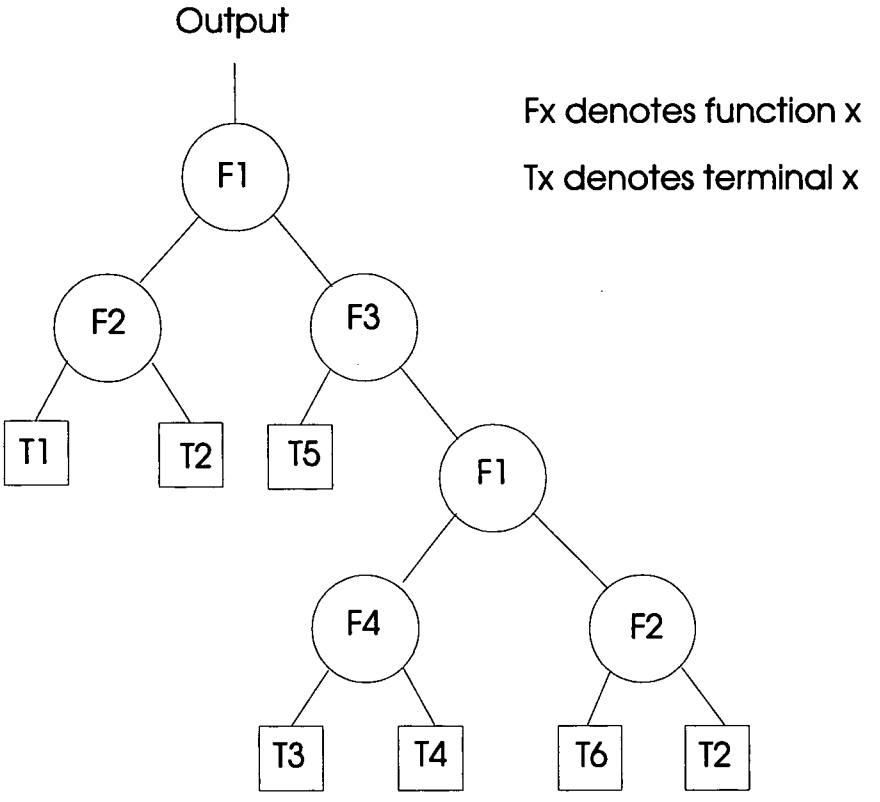


Figure 5-1 **A Genetic Programming Chromosome**

In a GP chromosome tree, each node is a function and takes n arguments (in Figure 5-1 n = 2). These arguments are either the outputs from other function nodes or terminals. Terminals are either inputs to the system or constants. When applying GP

to controller design, for example, the terminals would be sensor outputs. The set of available functions could include algebraic and trigonometric functions.

Initially a population of trees is produced at random. As in a GA, each chromosome is tested and assigned some fitness. Parents are selected with some bias towards the fittest trees. Crossover is undertaken by choosing at random a node or terminal on each parent and then two child trees are produced by replacing subtrees on each

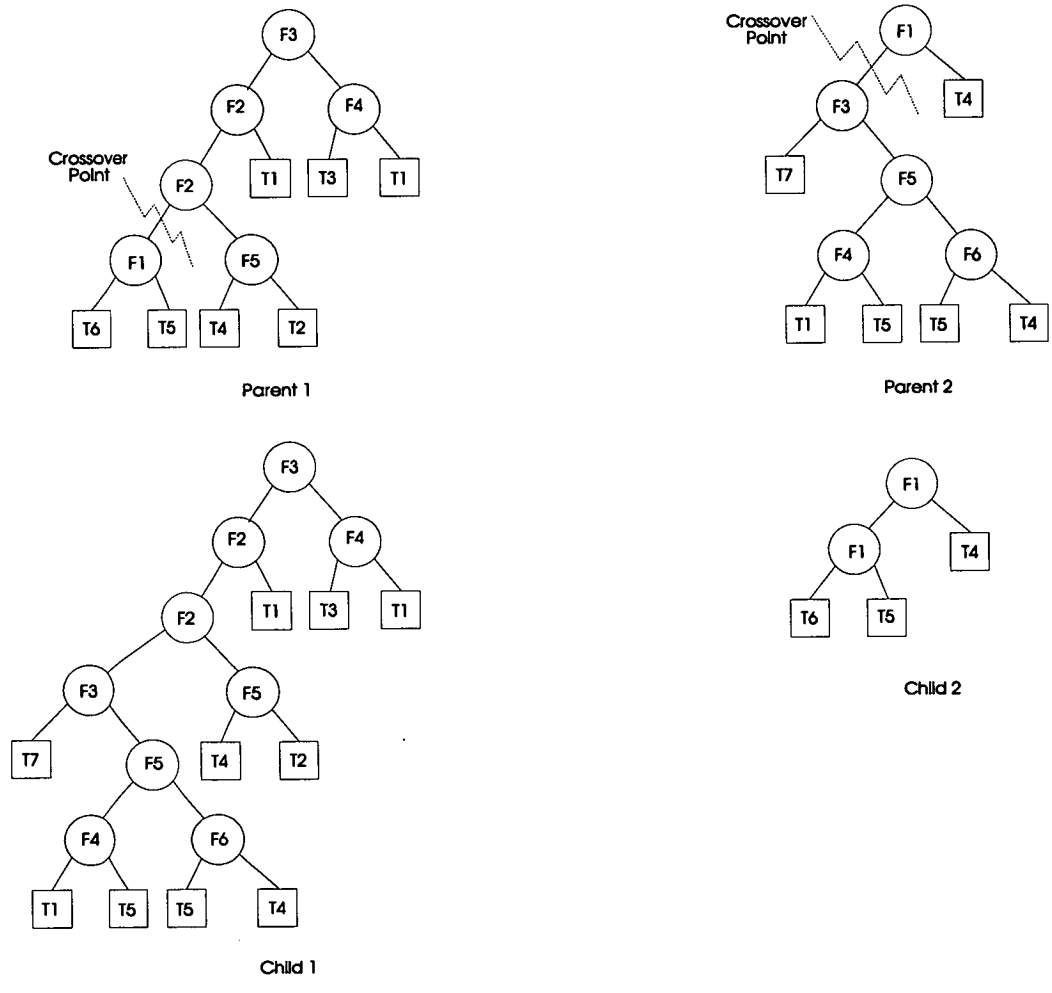


Figure 5-2 Crossover of GP Tree Chromosomes

parent with the subtree from the other parent (see Figure 5-2). Mutation takes place at random by changing a function to another function, or function (and subtree below) to a terminal. The child trees are then evaluated and added to the population.

5.2.2 The ACIS 3D Toolkit

ACIS [Corney & Lim 2001] is a B-Rep solid modeller that is widely used as the solid modelling engine in many CAD packages. For easy prototyping of applications based on ACIS, a development environment, the ACIS 3D toolkit, is available [Corney 1997]. This toolkit provides an interface to the data structures and algorithms in the ACIS solid modelling libraries via the Scheme language. Scheme is a language that is very similar to LISP, which is a language commonly used for genetic programming.

Although ACIS is a B-Rep solid modeller, since ACIS provides Boolean solid operations such as 'union', 'subtract' and 'intersection' and Scheme allows for manipulation of tree structures, it is possible to use the ACIS 3D Toolkit as a hybrid CSG/B-Rep solid modeller (see Figure 5-3). [Mäntylä 1988] gives an introduction to CSG solid modelling.

5.2.3 Potential Application of Technique to Practical Problems

If this technique were to prove successful, it might have a number of practical applications. This might involve the use of a finite element package to assign a fitness to candidate components. This would then allow components to be evolved that satisfy loading conditions and stress constraints.

The GP technique might also be applicable to the problem of translating B-rep solid models into CSG solid models [Raghothama & Shapiro 2000]. At present, there is no general method of translating a B-rep solid model into a CSG model. The GP optimisation technique is implemented using ACIS, a B-rep solid modeller. It is

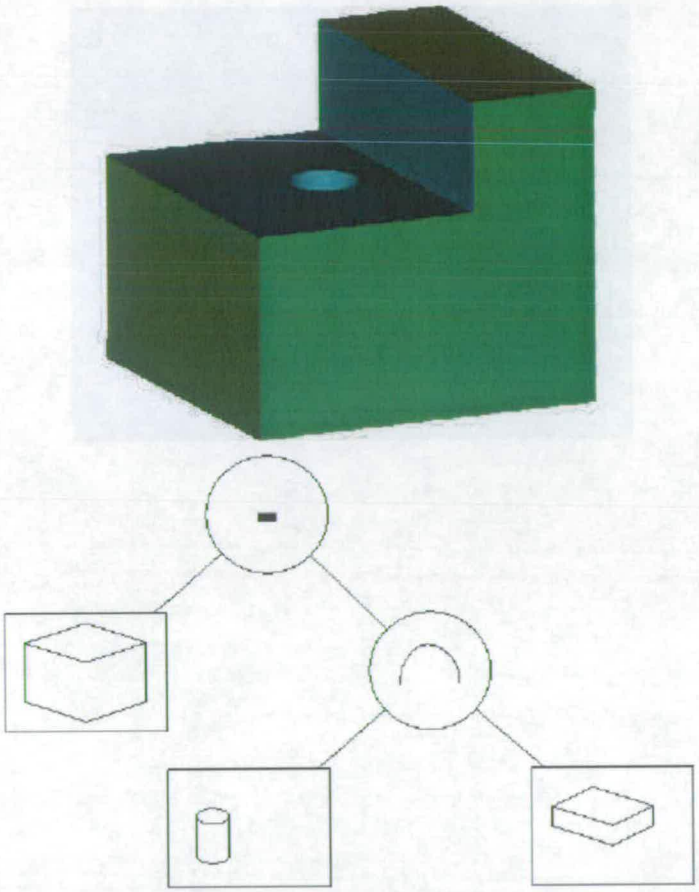


Figure 5-3 **Example of a CSG Tree and Solid Model**

therefore possible to bring an arbitrary B-rep solid model into ACIS. This solid could then be the target solid for the GP optimisation. If the dimensions, positions and orientations of the primitives used as the terminals in the chromosome trees were also optimised along with the tree structure then it may be possible to evolve trees

which, when evaluated, closely approximate the target body. It is then trivial to translate the chromosome tree into a CSG tree. A translation of the B-rep model to a CSG model would then have been achieved.

5.3 Implementation & Initial Test Problem

Genetic programming has been applied in a solid modelling context. This has been implemented in the ACIS 3D Toolkit in which the programming code is written in Scheme (a derivative of LISP - the most commonly used language for GP).

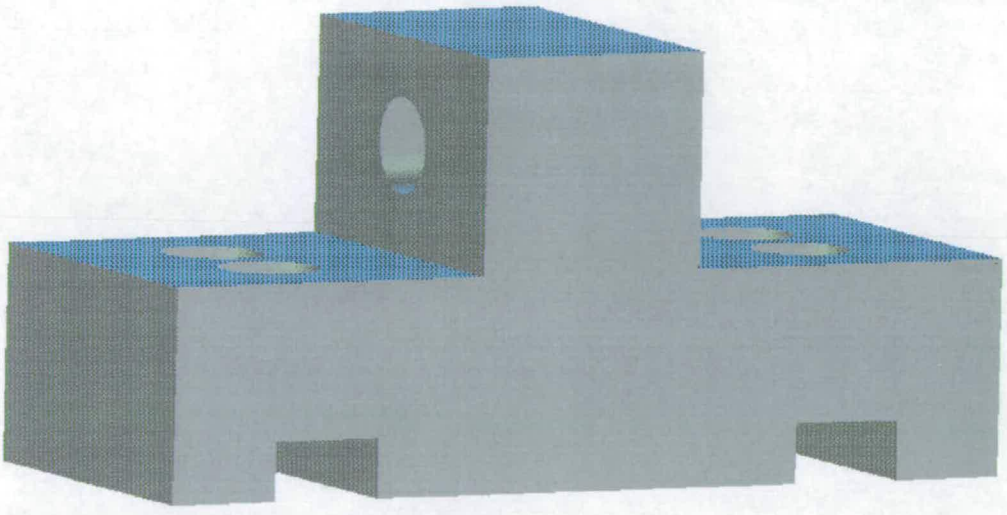


Figure 5-4 The Test Component

The function set consisted of the Boolean functions 'unite', 'subtract' and 'intersect'. Wrappers were added around the 'unite', 'subtract' and 'intersect' Boolean functions supplied with the ACIS 3D Toolkit so that errors are avoided when null bodies are created (for example when the intersect function is applied to two separate bodies). If a null body is created the wrapper returns the first body supplied as an argument to the function. An additional function 'Return-Body1' is also in the function set. This

function just returns the body that is its first argument. This allows parts of the tree to be recessive - they are transmitted with the chromosome but make no contribution to the final body.

The terminal set consisted of primitive blocks and cylinders of fixed dimensions. The aim of the optimisation was to create a tree structure using the Boolean functions and the terminal primitives which, when evaluated, corresponded to a predefined 3-dimensional solid. A test component (see Figure 5-4) was used as the target body. This component could be produced by subtracting the primitives from a blank that is also available in the terminal set.

Each candidate tree was assigned a fitness that was the difference between the volume of the union of the candidate body and the target body and the intersection between the two bodies. This corresponds to the sum of the volume in the target body but not in the candidate body and the volume in the candidate body not in the target body (demonstrated in Figure 5-5 in the 2-dimensional case).

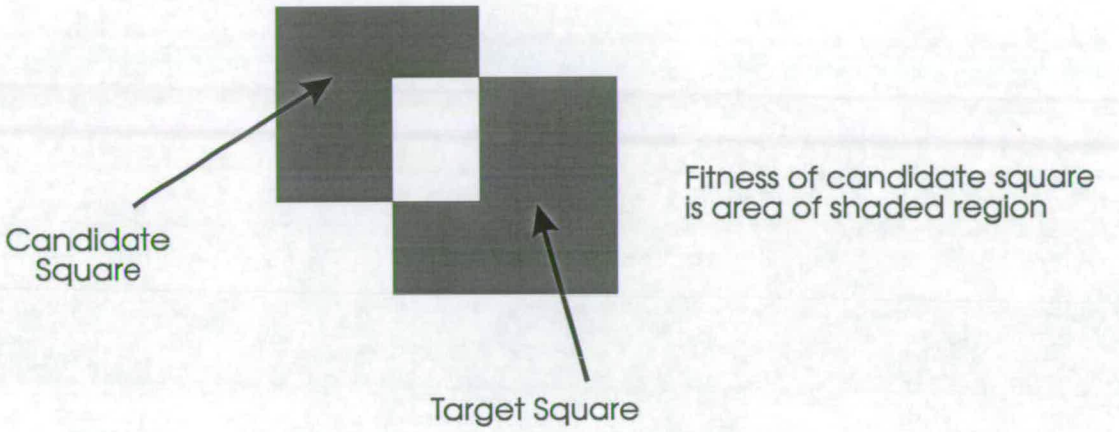


Figure 5-5 **2D Example of fitness calculation**

Genetic operators used are crossover and mutation as described in 5.2.1. In addition, ***compress*** and ***uncompress*** operators were available. The compress operator evaluates a subtree below a given point into a solid (see Figure 5-6). This solid is then placed in the terminal set. The subtree is then replaced by a reference to this terminal body. The uncompress operator reverses the compress operator by replacing a reference to a compressed terminal in a tree with the full subtree which evaluates to this terminal. The purpose of the compress operator is to allow frequently used 'good' subtrees to be compressed so that they cannot be disrupted by the crossover operator. [Angeline & Pollack 1992] successfully use similar operators in evolving programs to play Tic-Tac-Toe. [Koza 1994] describes Automatically Defined Functions (ADFs) which are similarly motivated by the desire to enable the reuse of sub-programs.

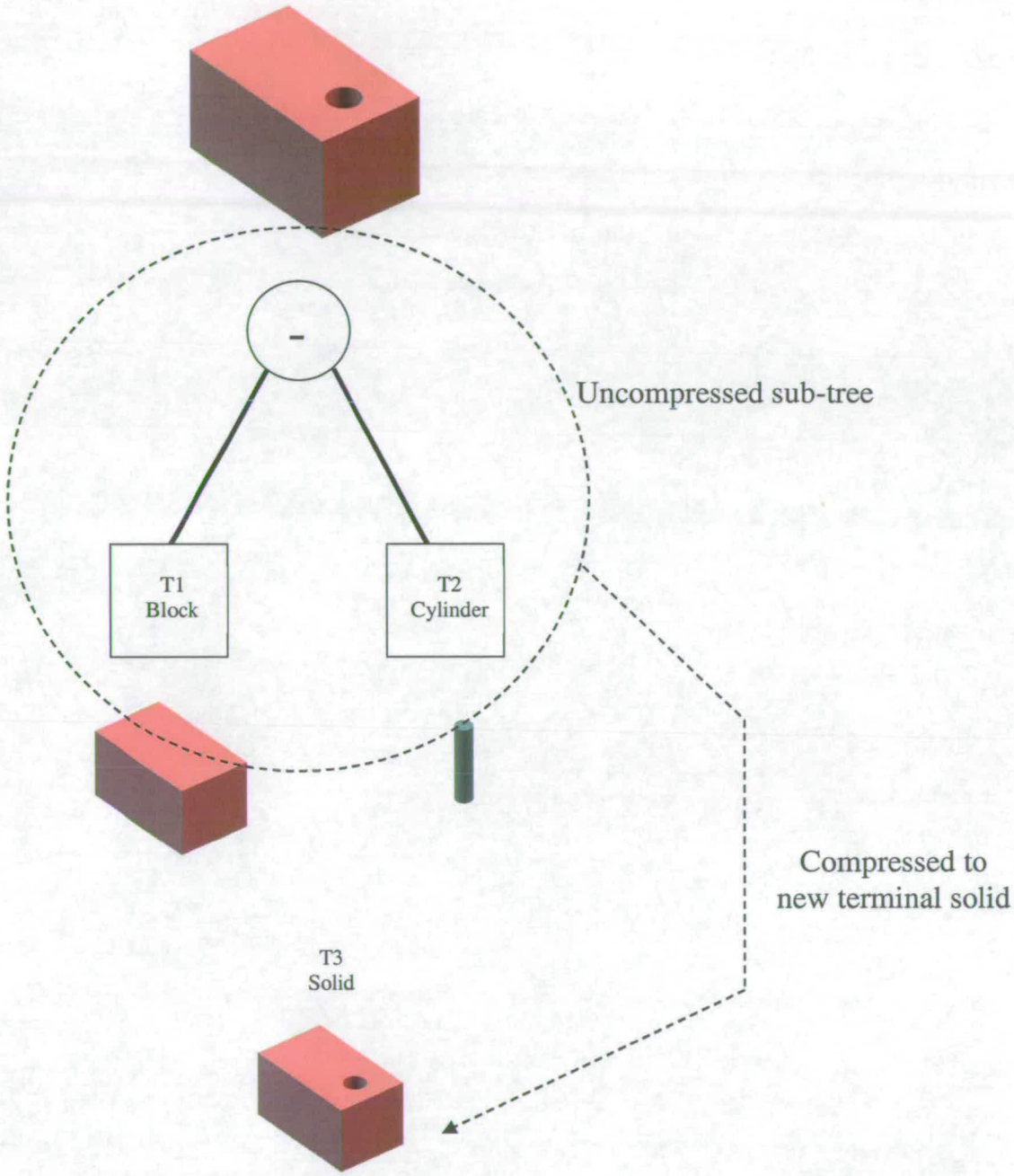


Figure 5-6 **Example of Compress Operator**

5.4 Results

With the terminal set included ten primitives - the blank and nine primitives to be subtracted - the GP scheme consistently found a tree that evaluated to the target in about 400 to 600 evaluations. This was with a population size of 100, rank based selection, crossover and mutation rate of about 10%. Since about one hour was required to undertake one run on a Sun Sparc10, it was not feasible to undertake a complete investigation of the optimisation parameters.

When the terminal set included fourteen primitives - the blank, nine primitives to be subtracted and four blocks to compose the main body of the component - then GP scheme was presented with two possible ways to build the component. These were to subtract blocks from the blank or to build the main body of the component by uniting the four blocks and subtracting the other primitives from this union. This slowed the GP scheme. Between 700 and 1000 evaluations were then required to produce a tree which, when evaluated, corresponded to the test component.

5.5 Discussion

Genetic programming proved to be able to generate the target component. It did better than randomly generating the trees which was not able to produce the test component. However, it required a large number of evaluations and therefore took a long time even for this simple problem. Therefore, it was concluded that the computational effort required to convert an arbitrary B-Rep solid model into a CSG tree for any realistic models would be prohibitive.

5.6 Conclusions

This chapter showed that genetic programming techniques could be applied to the generation of constructive solid geometry trees to optimise solid models. The approach was able, given a set of primitive solids, to evolve a CSG tree that evaluated to a test component constructed from those primitives. However, the computational cost of doing this suggested that the approach would not scale up to the problem of converting arbitrary B-rep models into CSG trees.

6 The Need for a Common Data Structure for Shape Optimisation

6.1 Summary

Building on the review of common approaches to shape optimisation given in Chapter 2, and the examples given in Chapters 3 to 5, it is argued in this chapter that a common data structure for shape representation and analysis would allow novel and efficient shape optimisation algorithms to be developed. Such a common data structure would increase the ability of shape optimisation to be used as a way of partly automating the process of geometry generation.

Firstly, the process of generating form from function is described and then the role that search and shape optimisation algorithms could play in this are discussed. An analysis of the shape optimisation process is then given. Finally, the argument is put forward that a common data structure for optimiser, shape representation and analysis would help in the realisation of search algorithms which are able to generate form for a desired function.

6.2 Form from Function

As briefly described in Section 1.2, an important area of design research concerns the process of generating the geometric form for a component given a desired function or behaviour for that component.

Modern current design practices use computers extensively. Product data management tools store all relevant information generated about a part or assembly throughout the design process. The part's geometry is generated, usually in three dimensions, using Computer-Aided Design (CAD) packages. Analysis of the component's physical behaviour is undertaken using virtual testing techniques such as finite element analysis or computational fluid dynamics.

In each of these applications, geometry clearly has an important role. Indeed, the CAD model of geometry is often seen as *the* model of the part. Analysis techniques take this model, transform it, and the results are stored with the CAD model. Manufacturing information is similarly generated and can be stored with the CAD model. Traditionally, engineering drawings are used as the primary method of storing and communicating information about mechanical parts. However, the synthesis of the geometry itself is primarily a *creative* process, in which human designers create appropriate geometries. CAD packages can be used to facilitate this process, but they only play a passive role. It would be helpful if computer tools could be developed which could take a desired function and from this produce a geometry, which would exhibit the required behaviour [Roy *et al.* 2001].

In contrast, computers are playing an increasingly important role in analysing the physical behaviour of a mechanical component given a specified geometry (i.e. performing the inverse of form from function, determining function for a given form). Previously, engineers relied on analytic solutions to the equations governing the behaviour of components. This was restricted to a limited number of shapes and behaviours. The development of computational tools such as the finite element method and computational fluid dynamics (CFD), have greatly increased the range of phenomena and shapes which can be analysed. This has enabled designers to check, quickly and inexpensively, whether a component behaves as expected. The designer can then enter into a further design iteration, changing the geometry to improve it or other aspects of the design. This interactive process continues until an adequate design is found.

6.3 Shape Optimisation as a Method of Partially Automating the Generation of Form from Function

In a limited number of application domains, it has proved to be possible to specify a desired behaviour and from that to deduce an appropriate geometry. For instance, for some aerodynamic problems it is possible to specify a desired pressure distribution over an aerofoil profile, and then to use an inverse technique to produce the required

geometry. However, even in the limited range of applications where such inverse techniques are available, they cannot cope with geometric constraints (for example to accommodate engine-mounting points) or rely on an idealisation of the physical behaviour.

In contrast to inverse approaches, which for most applications are not available, approaches that *search* for a shape that meets the desired specification might be much more widely applicable. The designer must in some way define a set of shapes through which to search, supply a method of assessing any of these shapes' ability to meet the desired specification and an algorithm to effectively search through the set of shapes is needed.

Shape optimisation might be one way in which this is done. It automates the process of changing the geometry in response to information about the components generated by the analysis. Rather than a human designer changing the geometry of the design in response to the analysis, a computer program is used to make the changes in order to find an optimal geometry.

However, applications of shape optimisation have tended to rely on keeping the set of shapes through which to search quite small. The shapes only vary parametrically in some relatively small way from an initial geometry supplied by a designer. Clearly, this defeats our purpose of semi-automatically generating geometries meeting a specified function, for which it is desirable that the space of shapes be large and general.

The following sections analyse the process of shape optimisation and try to identify where the difficulties lie in expanding the applicability of shape optimisation to the process of determining form from function.

6.4 Overview of The Shape Optimisation Process

Shape optimisation is used as a part of the process of designing a component. It concentrates particularly on the geometry of the part. An engineer when attempting

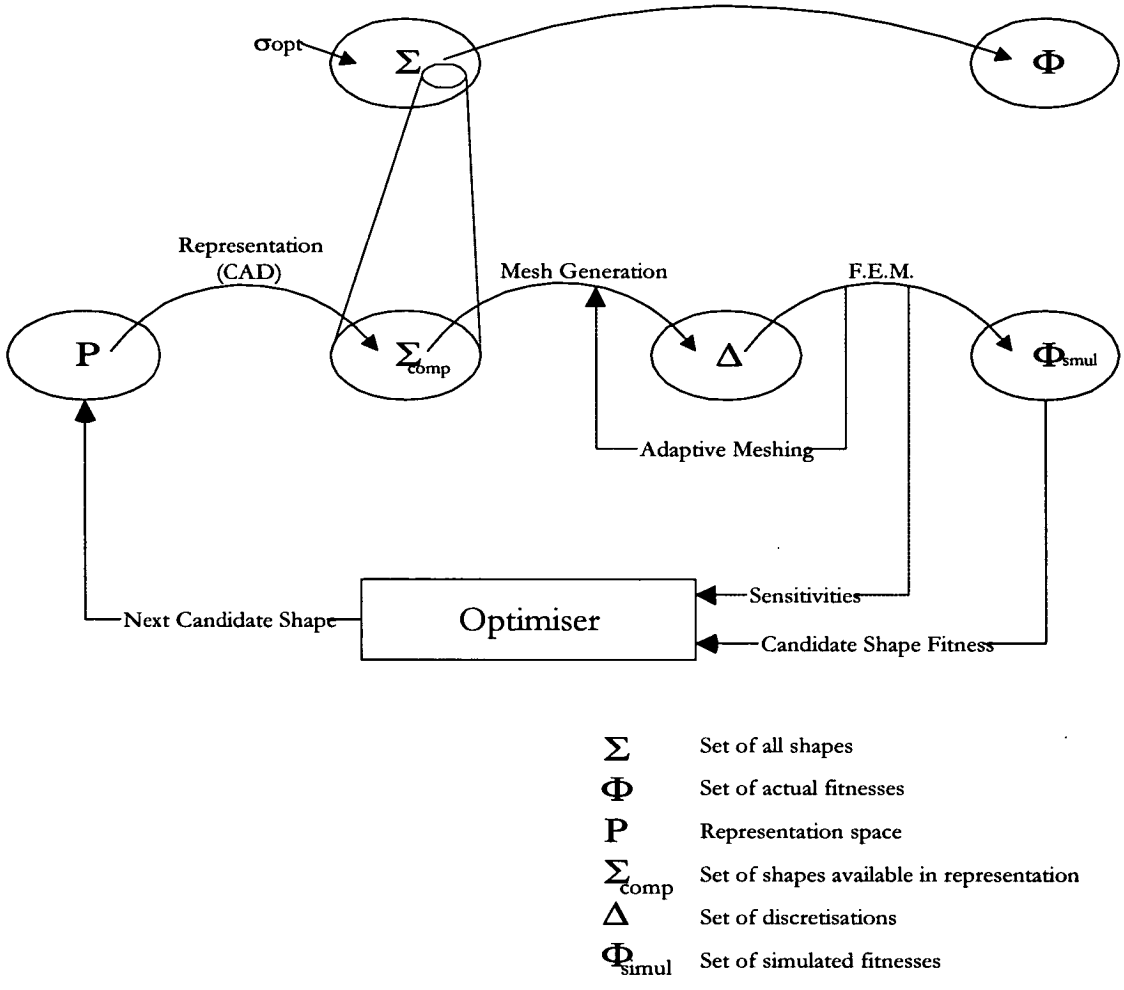


Figure 6-1 Searching for an Optimal Shape

to find a shape for a particular component is faced with an infinite set of possible shapes Σ (see Figure 6-1). Each shape within Σ can be assigned a 'fitness', a measure of its ability to do the job for which the component is intended. Often this is a single number, but may be a vector of numbers in the case of a multi-objective design. Each point in Σ, therefore, is mapped to a 'fitness' vector in the n-space Φ, where n is the number of objectives. Many of the shapes in Σ will violate some design constraint and will thus have a poor fitness in Φ. The designer wants to find the shape, σ_{opt}, in Σ which maps to the highest fitness in Φ.

To some extent finding σ_{opt} has been the concern of engineers for many years (or at least finding some sufficiently good σ). In shape optimisation we are primarily concerned with using computers to find σ_{opt}. We therefore need to translate Σ, the set of possible shapes, Φ, the set of fitnesses, and the mapping between them, onto a computer. We then need to automate the process of searching through the set of possible shapes. Sections 6.3 to 6.5 deal with each of these translation processes.

In Figure 6-1, the set of shapes through which the shape optimisation algorithm searches is shown as Σ_{comp}. Optimisation algorithms, however, do not optimise over sets of shapes but rather optimise the values of a number of variables. This is shown as the set P. Each member ρ of P represents a particular set of values for these variables. Typically, these variables take real number values although certain applications such as the one described in Chapter 4 can take Boolean values. A set of values, ρ, for variables in P is translated into a shape by a CAD package, where the shape is typically represented using a B-Rep model, spline curves or other geometric model.

Once a shape has been represented, it is necessary to analyse its behaviour. Finite element analysis is frequently used for structural analysis. Finite volume or vortex panel schemes for computational fluid dynamics are frequently used for fluid analyses. Both of these techniques require the shape (or the space around the shape) to be discretised. This is shown in Figure 6-1 as a mapping from Σ_{comp} to Δ, the set of discretisations.

The analysis method then takes a particular discretisation δ in Δ and calculates the physical behaviour. From this a value for the fitness of the shape ϕ_{simul} in Φ_{simul} can be calculated. Some shape optimisation techniques use specially developed analysis techniques in order also to provide the sensitivities of the shape to the design variables [Pourazady & Fu 1996].

The optimiser completes the loop. Given the fitness of the candidate shape (and possibly sensitivities) it chooses which point in P to try next. This loop continues until some termination criteria are met.

6.5 What Makes a Good Shape Representation?

Choosing an effective shape representation is critical to obtaining good results from the shape optimisation process. There are a number of potentially opposing factors that need to be addressed in formulating a good shape representation:

- The representation should be selected so that Σ_{comp} contains σ_{opt} i.e. the set of shapes available in the representation contains the optimal shape.
- The number of shapes in the search space (i.e. the size of Σ_{comp}) should be as small as possible so that the computational cost of searching through the space can be kept low.
- The analysis code can analyse faithfully all the shapes in Σ_{comp}.
- The optimisation algorithm should find it easy to search through the space of potential shapes.
- It should be possible to both efficiently store and compute the representation.

Each of these points is developed below.

The representation should be selected so that Σ_{comp} contains σ_{opt}. Only in a trivial case, where σ_{opt} is known from the outset, is the choice of representation easy. The designer must use knowledge he has of the problem to determine a class of shapes that might contain σ_{opt}. This is because there is no shape representation, implementable on a finite computer, which can model every possible shape in Σ. This is true on two levels. Firstly, any computer representation of shape can only operate up to a finite resolution. The designer must use some scheme for representing the shape, which has sufficient accuracy, in his judgement, to approximate the real

practical component shape, taking into account the degree of 'fine-tuning' to the optimal performance needed, as well as the manufacturing tolerances possible. Secondly, whichever shape representation (or combination of shape representations) is used, there will be some shapes in Σ that cannot be represented. For example, a B-Spline shape representation cannot exactly represent a conic [Hearn & Baker 1994]. A voxel representation cannot represent a shape with boundary smoothness. Non-uniform rational b-splines (NURBs) cannot be used to represent fractal shapes. In addition to this, the need to parameterise the shape representation, in order to produce a family of shapes, will further remove some of the shapes that could have been represented by a particular shape representation. Even when more exotic evolutionary algorithms are used, which can vary their representation during the search (see Sections 1.4.2 and 8.2.2), the set of shapes that can be generated is still smaller than the set of all possible shapes. The family of shapes to be searched, therefore, contains the designer's implicit view of what an optimal shape will be like. Often it is useful to choose a representation that is as general as possible in order to maximise the chance that it contains σ_{opt}.

This was demonstrated in Chapter 3 where the aerofoil parameterisation could be optimised but the aerofoils produced were probably sub-optimal because the representation had insufficient power to express the truly optimum shape. In Chapter 4, voxels produced a very general shape representation, with a large number of possible shapes. However, it was clear from the outset that the representation could not generate the optimal shape because such a shape would not have the 'stepped' boundary implied by the voxel representation. An *ad hoc* solution to this was to smooth the boundary with triangular elements, but nevertheless this was a weakness with representation.

The second point in the above list was that the number of shapes in the search space should be as small as possible so that the computational cost of searching through the space can be kept low. For most of the problems in shape optimisation the number of shapes in the search space can (simplistically) be stated as a^n where a is the accuracy

to which a variable should be resolved and n is the number of variables. For instance in the aerofoil parameterisation in Chapter 3, the thickness could vary from 3% to 8%. If the accuracy of this value was required within ±0.001%, a would be (8-3) / 0.001 = 5000. For the voxel representation with a 62 by 41 grid in Chapter 4, $a = 2$ (on or off) and $n = 2542$ and so the search space size is an enormous 2^{2542}. Since the search space size increases exponentially with the number of variables, it is sensible to have as few variables as possible. Clearly, the computational cost of searching through the search space depends on the algorithm employed, the way it searches through space and the nature of the problem landscape. However, all other things being equal, the smaller a search space is, the less effort is required to search through it.

From this perspective, it can be seen that one of the problems with the voxel representation was that the vast majority of shapes in the search space that was formed, were not reasonable shapes for the annulus. They contained numerous holes or were disjoint. The success of the genetic operators, which were designed to move the search away from such shapes, was because of this.

One approach that researchers have used to restrict the size of the search space is to start with a coarse shape representation in which a fairly broad but undetailed set of shapes can be searched initially. When a promising area of the search space is identified, the representation can be refined. [Kohli & Carey 1993] present such a shape refinement approach. One of the advantages of evolutionary algorithms is their ability to optimise with more complex representations than traditional optimisation techniques. This has been exploited, in shape optimisation, by several researchers [Eby *et al.* 1999b] [Gage *et al.* 1995] [Raich & Ghaboussi 2000] [Vekeria & Parmee 1997], others are detailed in Section 1.4.2. These approaches allow the evolution to modify not only the parameter values, but also the set of parameters themselves, so that the overall structure (or topology) of the parameterisation can also be evolved. It should be noted that, whilst variable complexity representations may allow a greater generality of shapes to be present in Σ_{comp}, and they provide a potentially efficient

method of searching through this space of possible shapes, they cannot represent every shape in Σ for the reasons given above.

The first two points in the above list clearly oppose each other. Where it is not clear what the optimal shape is like, it would be helpful to have as large a search space as possible in order to increase the chance that the search space contains the optimum shape. However, this needs to be set against the computational cost of having to search through an unduly large search space. Clearly, some compromise needs to be found.

Modern CAD packages now allow designers to parameterise their models. This has proved to be a powerful tool allowing engineers to produce 'families' of parts by changing a number of design variables. When a variable is changed, the model is rebuilt with this new value for the variable. There are, however, problems with this approach. Variables can position aspects of the geometry, for instance a hole, relative to some other part of the geometry, for instance an edge. If during a parameter update the edge disappears or is merged with another edge then it becomes impossible to place the hole. Updates of variables in the CAD model can have unexpected effects. [Shapiro & Vossler 1995] demonstrate these problems in a number of leading CAD packages; these problems are still evident in modern packages. They argue that this is because the concept of parameterised families of CAD models is mathematically ill defined.

From a practical point of view, it can often be difficult to assess the broad range of shapes that can be generated within a single parameterised model. Where a parameterised part has a large number of variables, it can become impossible to predict the interactions of all the variables with each other. This can make it difficult to ensure that it is possible to analyse accurately all the shapes in the search space Σ_{comp}.

This was demonstrated in Chapter 3 where the vortex panel method could not accurately model flows around most of the shapes generated by the Bézier curve

representation. Even with the more tightly controlled aerofoil parameterisation, there were some shapes that could not be accurately analysed. It might be argued that this merely implies that a better analysis method be chosen or that some better discretisation method be used. This will mitigate the problems, however, no analysis method can model all possible shapes and so some consideration must be made as to how to ensure that all shapes that can be generated can be analysed. Other issues to be considered for a successful analysis are given in Section 6.6.

Integration is an important consideration in dealing with shape representations because very often it is likely that use will be made of third party software for evaluation purposes. Examples are the use of finite element packages to help estimate likely stress distributions, fluid flows or thermal transfer characteristics, or analysis tools based on other methods, such as the boundary element method, or bespoke methods for manufacturing cost estimating. Sometimes it is necessary to communicate with several different software modules to perform multi-criteria evaluations.

[Samareh 1999] surveys shape parameterisation techniques for aerodynamic optimisation problems.

6.6 Ensuring Sufficiently Accurate Analysis

Usually the analysis of a particular shape is undertaken using a computer simulation, for example using the finite element method, boundary element method or computational fluid dynamics. It is necessary that shapes in Σ_{comp} are discretised for use with these methods (or for CFD the space around the shape). Each shape, σ, in Σ_{comp}, therefore maps to a discretisation, δ, in Δ. The analysis code then uses the discretisation to produce a value for the fitness of the shape.

For a computer-aided shape optimisation to be successful it is necessary that the mapping from a shape, σ, to its corresponding discretisation, δ, and so to ϕ_{simul}, be faithful to the mapping of the shape to its actual fitness, ϕ. This can be accomplished

with a combination of two techniques. Firstly, adaptive meshing techniques can be used. This can be viewed as a search for the most suitable discretisation in Δ which remains a valid discretisation of the candidate shape σ. This search is directed by the analysis code. Adaptive meshing techniques allow for a more robust mapping from the candidate shape, σ, to simulated fitness, ϕ_{simul}. Secondly, as discussed in the previous section, the representation used could be defined so that only shapes that can be adequately analysed can be produced. In other words, for all shapes in Σ_{comp} the corresponding ϕ_{simul} adequately approximates ϕ.

It is important to choose an appropriate analysis method. It is obvious that the analysis method should model the phenomena that are important for the aspect of the design that is being optimised. However, at some level all analysis methods make use of mathematical models that necessarily use some simplifications. For instance, models of elasticity make the simplification of assuming homogeneous material. The vortex panel method used in Chapter 3 assumed the flow was inviscid and incompressible.

One consideration that becomes more apparent when using analysis for shape optimisation than for other aspects in design, is that the simplifications used in the analysis should be valid for all of the shapes being considered in the search space. For instance, when designing aerofoils it is reasonable to assume that an efficient shape will be smooth and streamlined and so there will be no separation and viscous effects will be minimised. Therefore, for manual design purposes it would be reasonable to use a CFD method that made these assumptions for the limited number of shapes under consideration. However, for shape optimisation to be effective using the same method, these simplifications should be valid for all the shapes in the search space. It is not trivial to ensure this, after all, if the behaviour of the shapes could be determined beforehand then the analysis would not be required. The break down of the simplifications used in the analysis was the cause of the problems with the Bézier representations in Chapter 3.

The break down in these simplifications can cause the optimiser to move towards solutions for which the simulation gives a good score, but which in practice are poor. It may be possible to recognise when a simulation is inaccurate and either discard that result (this would be equivalent to adding a constraint on the confidence in the simulation result) or, alternatively, add a penalty to the objective function.

[Ellman *et al.* 1993] and [Gelsey *et al.* 1998] address similar concerns about ensuring that the model being used is appropriate for shape optimisation. [Gelsey *et al.* 1998] look to estimate by how much a model's assumptions are being violated. This information can then be supplied to the search algorithm as a 'model constraint function'. This can be used to penalise those designs that cannot be modelled accurately. [Ellman *et al.* 1993] state the philosophy behind the work they describe is that 'artefact performance models should be chosen in the light of the design decisions they are required to support'. The aim is to use cheaply evaluated models, where appropriate, to allow quick exploration of the search space and more expensive models when these are needed. They introduce the concept of 'Gradient Magnitude Model Selection' which is intended to be used with hillclimbing optimisation algorithms. Models are selected based on the requirements of the hillclimber. They illustrate their approach on a yacht design problem.

For some problems the objective function is non-smooth, i.e. small changes in the variables can result in 'jumps' in the objective function. This may be due to discretisation errors that are inevitable when using some types of simulation. These can be caused when, for instance, a change in a variable causes a change in the mesh topology of an unstructured mesh and thus a sudden, possibly small, change in the objective function. If this 'noise' is large, it may make some optimisation algorithms unsuitable for the problem. It may be possible to reduce this noise by increasing the accuracy of the simulation, but at the cost of increasing the time taken to undertake the simulation. Similarly, it may be computationally expensive and/or difficult to compute the derivatives of the objective function accurately. The need for adaptive mesh refinement for shape optimisation applications motivates the work described in

[Banichuk *et al.* 1995], [Canales *et al.* 1994], [Kodiyalam & Thanedar 1993] and [Kodiyalam & Parthasarathy 1993].

6.7 Speeding up the Analysis

The time taken to evaluate the objective function can often be great. There are a number of ways that this difficulty can be overcome.

One method is to use an optimisation technique that only uses a small number of objective function evaluations. This, however, will probably reduce the confidence with which the global optimum is found. Those algorithms which require very few objective function evaluations will often only find local optima and since the evaluation time is great it is not possible to start the algorithm from a large number of points in the search space.

At the beginning of an optimisation when the approximate region of the optimum in the search space is being sought, it may be possible to undertake a simulation which is less accurate (and thus less time-consuming) than the full simulation. Over time, as the optimiser approaches the optimum, the accuracy of the simulation can be increased. A related approach might identify those solutions that can be seen to be obviously bad due to some prior knowledge the engineer has about the problem. These bad solutions need not be fully evaluated and can be given a 'bad' score. Whilst both these approaches might save simulation time it may be difficult to implement them with some optimisation techniques, where continuity of the search space is required.

[Eby *et al.* 1999a] [Eby *et al.* 1999b] optimise a flywheel with an *injection island genetic algorithm* (iiGA) based on similar ideas, this is described in more detail in Section 2.3.4.3. In their iiGA various sub-populations are used each with different resolution of the representation, with fit individuals able to move from lower to higher populations. This allows lower sub-populations to search the search space extensively, with a quick, low accuracy evaluation. Higher populations search a smaller, fitter, part of the search space more intensively with a more expensive,

higher accuracy analysis. Extensions of these ideas are reported in [Hu & Goodman 2002] [Hu *et al.* 2003] in which the concept of 'Hierarchical Fair Competition' is introduced. This approach is motivated by the desire to avoid premature convergence to local optima. Again, a hierarchy of populations is used with a number of levels. A larger number of low-level populations are used than higher populations. Solutions migrate from lower to higher populations when its fitness is sufficiently high to pass an 'admission threshold'. Solutions thus pass from lower levels, through intermediate levels up to higher levels. By altering the relative effort apportioned to lower or higher level populations, the relative amounts of 'exploration' and 'exploitation' can be changed.

[Vekeria & Parmee 1997] also make use of an iiGA, along with the use of a Dynamic Shape Refinement, with which the resolution of the representation can be varied. They report significant improvements in both reduction of computational expense and in the quality of the design, over a single level representation.

[Younsi *et al.* 1996] use multi-mesh for structural shape optimisation in three dimensions. The system uses several meshing levels along with error estimation, in order to reduce the computational cost of the finite element analysis.

[Rasheed & Hirsh 2000] introduce an interesting approach that attempts to speed up a genetic algorithm when evaluation is expensive, making use of informed operators. Instead of just generating one random mutation, a number of mutations are made, the solutions thus generated are ranked using a less expensive reduced model, and the best is then kept as the result of the mutation. The authors report that this can significantly speed-up the GA optimiser in several engineering domains.

It may be possible to parallelise the optimisation algorithm and/or the simulation so that a number of computers/processors are used at the same time. Some optimisation algorithms readily lend themselves to parallelisation. It is very easy, for example, to use a genetic algorithm in which a number of solutions are simultaneously evaluated on separate processors. It may also be possible to run a number of local optimisers

from different starting positions in the search space on a number of computers. Any parallelisation of the problem, however, makes the optimisation harder to implement.

[Wang *et al.* 2002a] investigate the use of hierarchical parallel evolutionary algorithms for aerospace optimisation. In this approach, several sub-populations are simultaneously evolved. Some populations are considered to be 'low' in the hierarchy. These are run with coarse and quick CFD evaluation and are primarily intended to explore the search space. At certain intervals, good solutions from these lower populations are migrated to higher level populations. These higher populations use more computationally expensive, but higher fidelity, CFD analysis and are intended to refine the shape.

It may be possible to separate the problem into independent sub-problems, for which a full simulation is unnecessary. The global solution can then be found by combining the optima of the sub-problems.

Some approaches use response surfaces. Rather than computing the objective function for each point required by the optimiser, the response of a smaller number of shapes is calculated and a model of the response surface (landscape) is built by extrapolation. Often polynomials are used to model the response surface. The optimiser is then used to find the optimum point on this surface. This method can be useful where full simulations are very computationally expensive, but clearly, they require that the method chosen to model the landscape is appropriate to the particular problem. [Otto *et al.* 1996] describe a so-called surrogate approach to the optimisation of multi-element aerofoils. [Ratle 2001] use kriging to approximate the landscape for evolutionary optimisation. [Seller *et al.* 1994] use response surfaces to enable multi-disciplinary optimisation. [Liu & Batill 2000] use an artificial neural network to approximate landscapes for multi-disciplinary optimisation. [El-Beltagy & Keane 2001] describe the use of a Guassian Processes approximation model to provide an approximation to the results that would be given by detailed analysis code. They apply this to the structural optimisation of a satellite boom.

6.8 Searching Effectively for the Optimal Shape

Once a set of shapes through which to search, and an analysis routine that can faithfully calculate the objective function, have been defined, then an optimisation algorithm is required which can search for the best shape. The optimisation algorithm can be thought of as sampling the search space, and from the information acquired about the objective function at the sample points, it chooses where next to sample. Each optimisation algorithm differs in the way in which it chooses which points to sample and in the way that it then chooses subsequent points in the search space.

The performance of an optimisation algorithm can be judged in a number of ways. All other things being equal, an algorithm that uses a smaller number of evaluations is better. Also, the ability of the algorithm to robustly find an optimum for a class of problems is also useful. An algorithm might prove very efficient at a particular problem instance, but might be ineffective for problems that are apparently similar. Robustness is therefore an important performance measure.

Traditional optimisers rely on local 'move' operators and only have one active sample point and so are susceptible to being caught in local optima. They may, however, quickly find these local optima. Optimisers such as genetic algorithms have a number of sample points active in a 'population'. They also can make use of 'move' operators that may move out of the locality of active points and so are more likely to find the global optimum.

For an optimisation algorithm to be effective, it is necessary that the algorithm, representation and nature of the problem be well matched. In other words, the move operators must be helpful in searching through search space.

6.9 A Representation for Geometry and Physical Behaviour

The previous sections have identified a number of issues related to undertaking effective shape optimisation, namely representing a family of shapes, ensuring an accurate analysis and searching effectively. All of these issues can be seen as

difficulties encountered at the interface between each of the modules (optimiser, representation and analysis) used for shape optimisation.

Producing a family of shapes through which to search is a process that happens at the interface of the optimisation module and the geometric model. Transferring a vector of usually real numbers as variables into a geometric model is a process that is mathematically ill defined as Shapiro and Vossler noted [Shapiro & Vossler 1995]. It is difficult to predict the range of shapes that a parameterisation will generate for most non-trivial multivariate parameterisations [Hoffmann & Kim 2001]. There is no well-defined process of transformation which can be guaranteed to be robust, although [Raghothama & Shapiro 2002] present a method based on *topological categories* which may be useful for the systematic generation of part shapes.

Ensuring an accurate analysis is a problem encountered at the interface between the geometric model and analysis model. [Palmer & Shapiro 1994] argue that the separation of the geometric model and physical model of behaviour is a considerable block on the development of computational tools for the synthesis of geometry with a given physical behaviour. Geometric models can be used for the calculation of various spatial properties of a component such as volume, surface area and interference. They can also be used to provide the spatial data for use by analysis techniques, but are not themselves able to represent the information (i.e. stresses, pressures and heat) which are generated. Geometric models are not able to represent such spatially distributed properties because they are not part of the mathematical model which solid models represent. Geometric modelling and physical modelling are therefore distinct from each other. Bridging this gap is an active area of research as can be seen with the attempts to move to *meshfree* analysis [Botkin *et al.* 2002] [Grindeanu *et al.* 2002] [Lu & Chen 2002].

Producing an optimisation algorithm that can effectively search for the optimal shape, is a problem that involves all of the three shape optimisation modules. The optimiser in a shape optimisation application is aiming to change the shape of a component in order to improve its performance. Clearly, for almost all components

the way a shape should be changed depends on the current performance of the component: where stresses are too high material should be added, where stresses are low material can be removed. However, the optimisers generally deal only with vectors of real numbers – there is no way to make use of the information on the current spatial performance of the component. This becomes most clear when considering the information flow in a typical shape optimisation application. Almost all of the computing time is spent in analysing the component, but the only information returned to the optimiser is a value for the objective (i.e. mass, volume, or displacement) and whether or not the constraints are violated (i.e. maximum Von Mises Stress or displacement). All the information about where in the component the stress is high is discarded because the optimiser has no mechanism for using it.

The aim of this thesis was to try to establish whether shape optimisation could be used to semi-automate the process of moving from function to form. At present, as has been discussed, the shape optimisation process is far from automatic. There is inevitably a good deal of *ad hoc* integration between geometric model and analysis model. To address these problems what is needed is a computer language in which shape optimisation applications can be built. This language would be able to deal with the process of shape optimisation from optimisation, geometry, physics and discretisation to a numerical solution in an automated and consistent way. Such a language needs computer representations that can represent geometry, discretisation and physical behaviour in a common representation. This would also offer opportunities to develop algorithms that could make use of the information on the shape's current physical behaviour in order to modify the shape, and thus search the space of possible shapes more effectively.

7 Chain Models

7.1 Summary

The previous chapter argued that a common data structure is desirable for shape optimisation. However, what should this data structure be? In this chapter, Chain models, first proposed by Palmer and Shapiro [Palmer & Shapiro 1994], are put forward as such a data structure. Chain models are first described, and then other applications in which Chain models have been used are given to demonstrate their wide applicability. Then it is shown how existing shape optimisation techniques could be implemented within a Chain model framework. Finally, there is a discussion describing how Chain models could enable the development of novel and effective shape optimisation algorithms.

7.2 Chain Models

Chain models were first proposed by Palmer and Shapiro [Palmer & Shapiro 1994]. They argued that the lack of a unified computational model of geometry and physical behaviour has led to a large range of inconsistent and incompatible CAD and engineering analysis tools. This has limited the extent to which CAD tools have been used in practice. They say 'it seems clear that formalisation of the relationship between form and function is a prerequisite to taking full advantage of computers in automating design and analysis of engineering systems'. They present Chain models as a unified computational model of physical behaviour that links explicitly and consistently geometric and physical representations. Chain models use the algebraic-topological concepts of ***cells***, ***cell complexes***, ***chains*** and ***operations*** on them to model these physical systems.

For full details on Chain Models the reader is referred to [Palmer & Shapiro 1994]. The following sections attempt to précis the important concepts from this paper and to show how Chain models might be applied to design in general and shape optimisation in particular.

The principal aim of Chain models is to model physical objects. Common to most models of a physical object is that they consist of distributions in time and space of various physical quantities (mass, energy, force, momentum, charge, velocity, temperature etc.). Therefore [Palmer & Shapiro 1994] offer the following two definitions:

Definition 1 A ***system*** (or object) is a set of quantities $\{Q\}$ distributed in space and time.

Definition 2 A ***physical*** system (object) is a system that satisfies some ***physical laws***, which are constraints on the values of these distributions.

Thus, a physical system has a state that can be characterised as a distribution of relevant quantities at a particular point in time. The distribution of these quantities is constrained by physical laws. Often it is necessary to refer to all objects that exhibit the same ***physical behaviour*** and so the following definition is given:

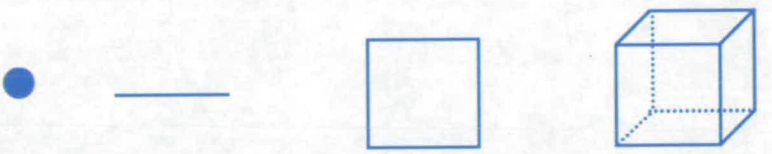
Definition 3 A ***physical behaviour*** is the class of all physical objects (or systems) satisfying a given set of physical laws. A given physical object is said to ***exhibit*** a behaviour if it is in the class.

7.2.1 Cells and Complexes

Chain models look to represent these distributions in Euclidean space, E^n. Therefore, some way of distinguishing regions of space is required. Chain models use the concepts of cells and cell-complexes from algebraic topology for this purpose.

Definition 4 An ***n-cell*** c is a set that is homeomorphic to a closed unit n-ball B^n. The closed unit n-ball is a subset of $\Re^n$: $B^n = \{x \in \Re^n \mid \|x\|_2 \leq 1\}$. An n-cell c is said to have dimension n.

n-simplices up to n=3



n-cubes up to n=3

Figure 7-1 **n-Simplices and n-Cubes**

An n-cell is therefore topologically equivalent to an n-ball. Some function h defines the cell c: $c = \{x \mid h(x) \in B^n\}$ so that h can be seen as a representation of c.

Definition 5 The ***boundary*** of an n-cell c is the set $\partial(c) = \{x \mid \|h(x)\|_2 = 1\}$, where h is a homeomorphism defining c.

Figure 7-1 shows some possible cell types: ***n-simplices*** and ***n-cubes***, up to $n = 3$. Cells define simple regions of space. To represent more complicated regions we need to define cell complexes:

Definition 6 A ***cell complex*** K is a set of cells that satisfy the following properties:

1. The boundary of each n-cell c is a finite union of (n - 1)-cells in K: $\partial(c) = \cup_j c_j$
2. The intersection of any two cells c_i, c_j in K is either empty, or is a unique cell in K.

Definition 7 A cell complex K is said to ***decompose*** a region R if R is equal to the union of the cells of K.

A cell complex can therefore be used to decompose a possibly complicated domain into a number of simpler cells. This is clearly an approach that is frequently used in geometric modelling and analysis tools such as finite elements, finite differences and finite volumes. The complex consists of all the 2-cells (triangles or quadrilaterals), 1-cells (edges) forming the boundary of the 2-cells, and the 0-cells (nodes) forming the boundary of the 1-cells. Figure 7-2 shows a simplicial 2-complex and Figure 7-3 shows a cubical 2-complex.

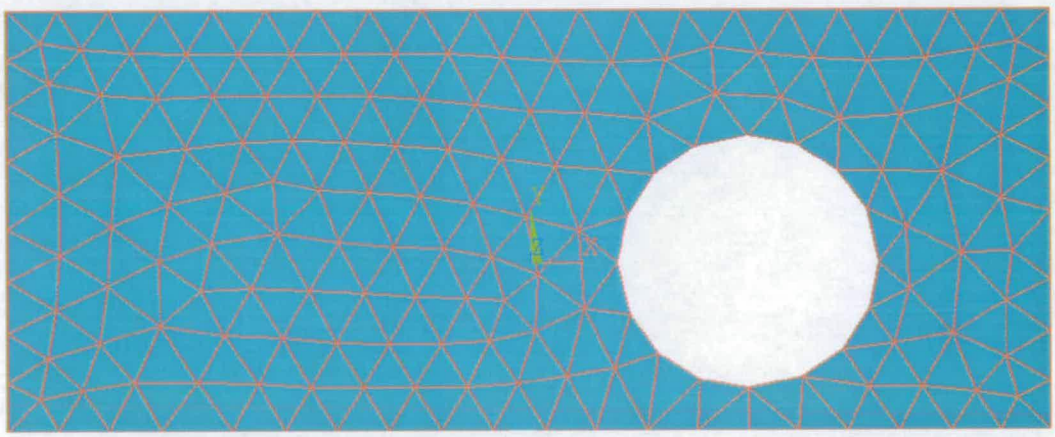


Figure 7-2 **A Simplicial 2-Complex**

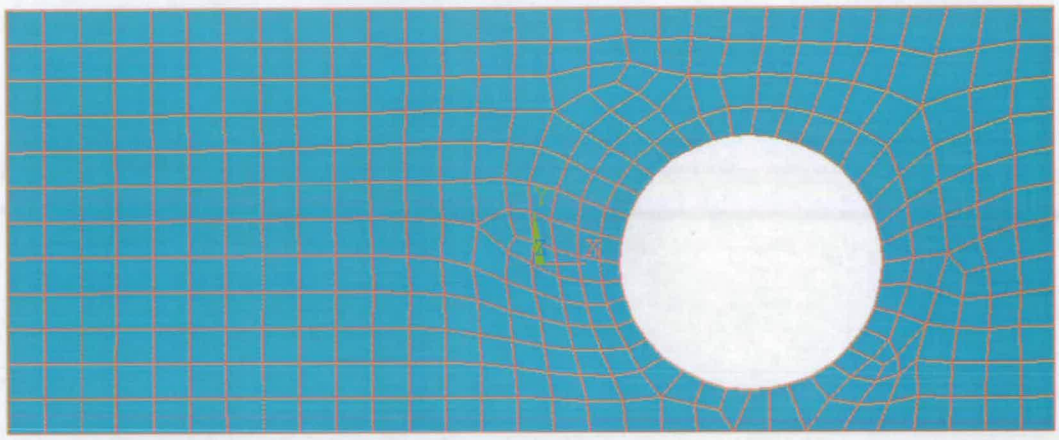


Figure 7-3 **A Cubical 2-Complex**

Definition 8 The ***faces*** of an n-cell $c \in K$ are the (n - 1)-cells in K comprising its boundary. If f is a face of c, then c is a coface of f.

Definition 9 An ***oriented*** cell is a pair $c = (u, o)$, where u is an (unoriented) cell, and $o \in \{1,-1\}$.

Definition 9 allows for the relative orientation between a cell and one of its faces to be defined. Given an oriented cell $c_i = (u_i , o_i)$ and one of its faces a faces $c_j = (u_j , o_j)$ the relative orientation is defined as $\sigma(c_i , c_j) = o_i o_j$ and thus takes values –1 or 1.

7.2.2 Chains

Cells and cell complexes provide the means to decompose space into simple regions. Definitions of faces and orientation provide sufficient structure to represent the relationship between these regions. We now need some way of representing physical quantities within these regions. Chain models use ***chains*** defined over cell complexes for this purpose.

Definition 10 A ***p-chain*** *ch* defined over a complex K, and a vector space G, is a formal sum $\sum_{c_i \in p-cells(K)} g_i c_i$ of p-cells of K with coefficients $g_i \in G$. We use the notation $ch(c_i)$ for the value of the coefficient associated with the cell c_i in *ch*.

Thus a chain associates with every p-cell in the complex K an element of G. G could be any abelian group, but in Chain models G is typically the integers, real numbers or polynomials, or vectors or tensors of these. For example, we might represent the mass associated with cells in a 3-complex K with a 3-chain where the elements of G are scalar real numbers. Another example might be to represent the displacement of 0-cells as a 0-chain where the elements of G are vectors of real numbers.

Chains have useful computational properties for the purpose of calculation. For example, suppose we have a dimension p, a cell complex K and a group G, and G is a field, then the set of all p-chains over K, $Ch(K,G,p)$ form a vector space and so vector operators can be used on chains. (A field is defined as a set of elements that satisfy the field axioms for both addition and multiplication, namely:

commutativity	$a + b = b + a$	$a\,b = b\,a$
associativity	$(a + b) + c = a + (b + c)$	$(a\,b)\,c = a\,(b\,c)$
distributivity	$a\,(b + c) = ab + ac$	$(a + b)\,c = ac + bc$
identity	$a + 0 = a$	$a\,.\,1 = a$
inverse	$a + (-a) = 0$	$a\,.\,a^{-1} = 1 = a^{-1}\,.\,a \quad (a \neq 0)$

The real numbers and complex numbers are examples of fields, however the integers are not).

Chains support two operators:

Definition 11 The ***boundary*** $\partial(ch)$ of a p-chain $ch \in Ch(K,G,p)$ is a (p - 1)-chain defined as follows: $\partial(ch) = \Sigma_i g_i c_i$, where $g_i = \sum_{cf \in cofaces(c_i)} \sigma(cf, c_i) ch(cf)$.

Performing the boundary operator on a p-chain *ch* produces a (p - 1)-chain over each (p - 1)-cell, c_i in the complex K, by taking each coface of c_i and summing the oriented values of the p-chain *ch* on these cofaces. This is shown in Figure 7-4

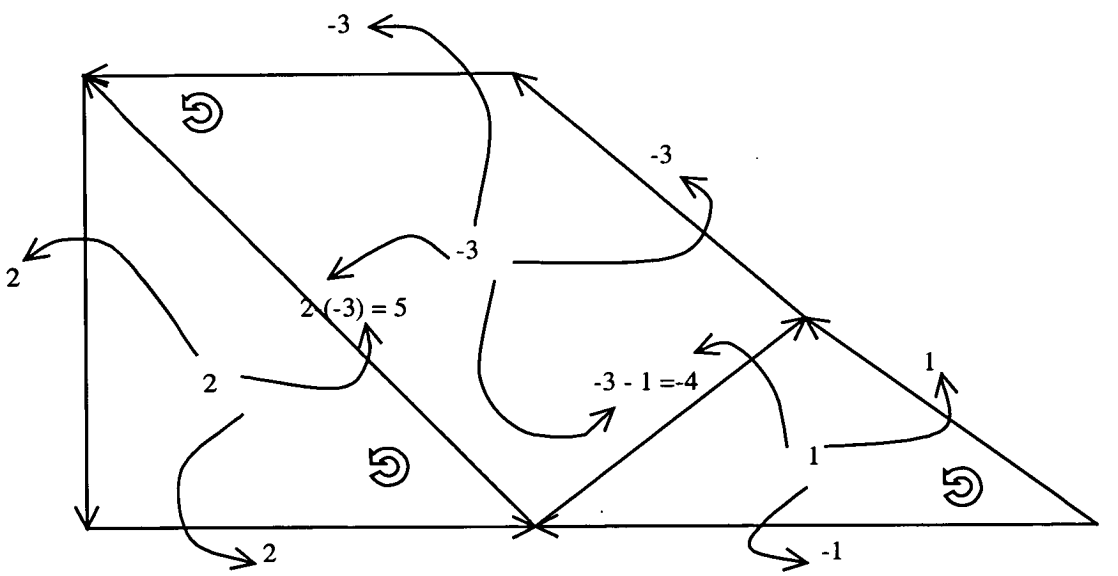


Figure 7-4 **The Boundary Operator Applied to a 2-Chain**

Definition 12 The ***coboundary*** $\delta(ch)$ of a p-chain $ch \in Ch(K,G,p)$ is a (p + 1)-chain defined as follows: $\delta(ch) = \Sigma_i g_i c_i$, where $g_i = \sum_{f \in faces(c_i)} \sigma(f, c_i) ch(f)$.

Similarly to applying the boundary operator, performing the coboundary operator on a p-chain *ch* produces a (p + 1)-chain over each (p + 1)-cell, c_i in the complex *K*, by taking each face of c_i and summing the oriented values of the p-chain *ch* on these faces as shown in Figure 7-5.

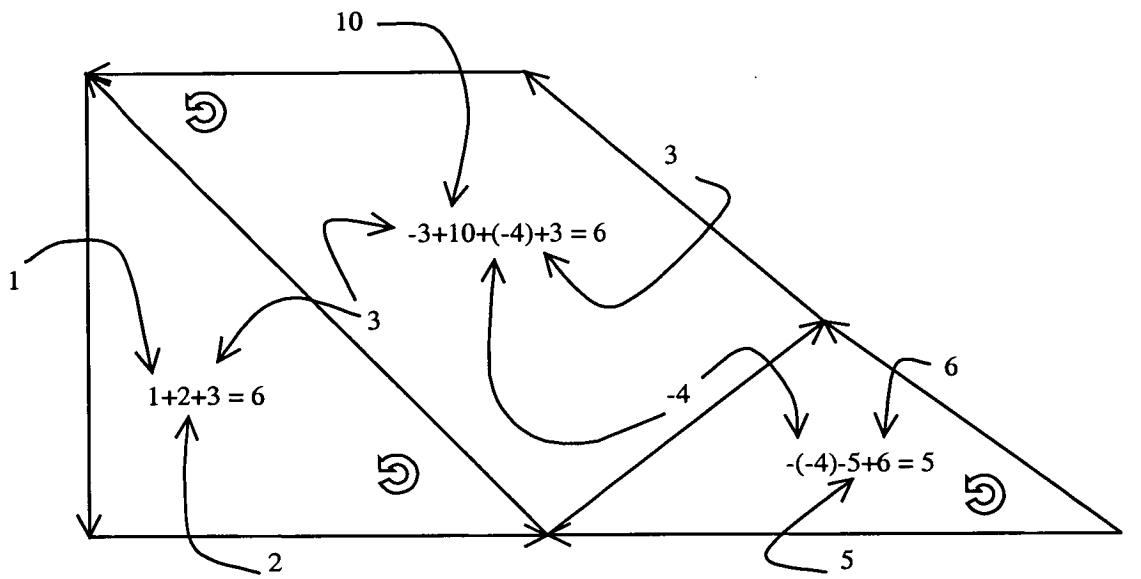


Figure 7-5 The Coboundary Operator Applied to a 1-Chain

The last definition given by Palmer and Shapiro allows any function which can be applied to the elements of *G* to be extended to chains by applying the function cell by cell to the elements of *G* associated with each cell.

Definition 13 Given a function $f: G \rightarrow S$, and a chain $ch = \sum_{c_i \in p-cells(K)} g_i c_i$, we define

$$f(ch) = \sum_{c_i \in p-cells(K)} f(g_i) c_i$$

They finish by defining ***equality*** between two chains ch_1 and ch_2 to be true when they are both defined over the same complex, K, and elements, G, and $ch_1(c_i) = ch_2(c_i)$ for each cell c_i in K.

7.3 Using Chain Models

The above thirteen definitions have defined cells, cell complexes, chains and various useful operators (boundary, coboundary, addition, multiplication by a scalar, function application and equality). These are useful tools for representing physical objects. The following sections show how these concepts might be applied to geometry and physical behaviour, with which we are interested for shape optimisation.

7.3.1 Chain models of Geometry

Chain models can be used to represent geometry and to calculate various properties of the geometry. Any solid S can be represented by a 3-cell complex that decomposes the solid. It can be shown that any 2-cell in this complex either has one or two 3-cell cofaces in K. Also the 3-cells can be oriented consistently so that for any two adjacent 3-cells a and b sharing a 2-cell face c $\sigma(a,c) = -\sigma(a,c)$.

With each 'full' 3-cell within the solid, an integer of 1 is associated. 0 is associated with those 3-cells that are 'empty' (in this case since K decomposes S there will be no 'empty' cells). In this way S can be defined as a 3-chain $\Sigma_i a_i c_i$ where $a_i = 1$ and c_i are the 3-cells in K. Applying the boundary operator to this chain produces a 2-chain whereby each 2-cell, c, is associated with $\Sigma_j \sigma(c_j,c)$ where c_j is a coface of c (there will be two cofaces). Internal 2-cells will have two 3-cell cofaces (which will be coherently oriented) and so will have a coefficient of 0. Whereas external 2-cells will

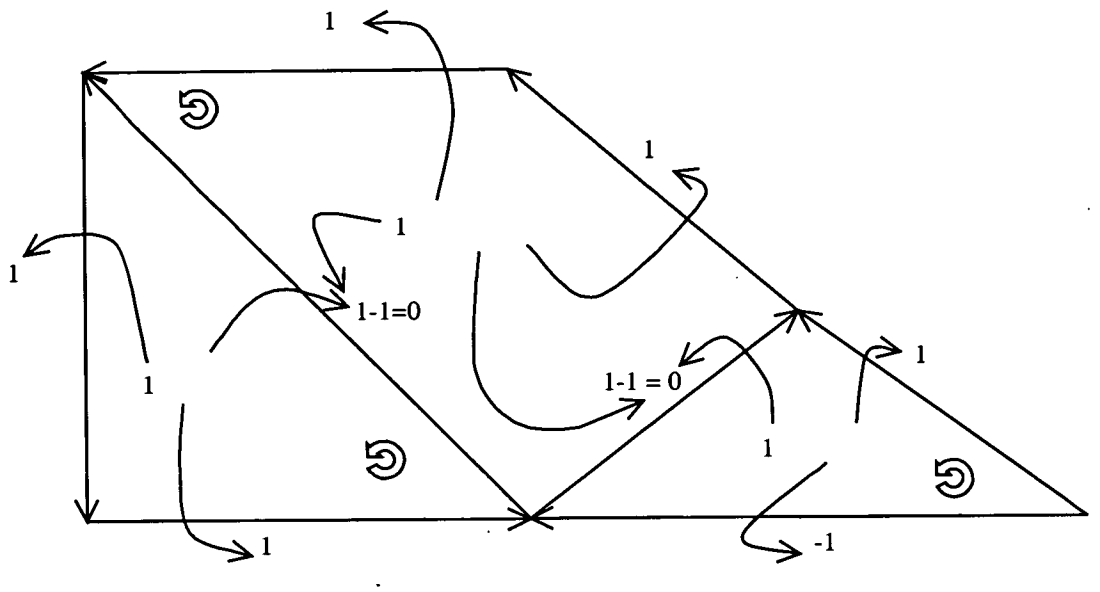


Figure 7-6 A Chain Model Representation for Geometry in 2 Dimensions

have only one 3-cell coface and so the coefficient will be non-zero. Thus, using this Chain model, the calculation of a solid's boundary is done by applying the boundary operator. Figure 7-6 demonstrates this, but for a two dimensional shape.

It is possible to represent geometry in a variety of ways with Chain models. A two dimensional surface might be modelled with a triangular mesh of faces. This requires an appropriate 2-complex. A 0-chain could be used to represent the corners of the triangles and a 2-chain could be used to represent the triangles' surface normals. Alternatively the surface might be represented with b-spline patches with a 2-chain used to represent the spline polynomials. In fact, any representation of geometry then could be used by a b-rep model could be implemented in a Chain model.

7.3.2 Chain models of Physical Behaviour

7.3.2.1 Physical Behaviour

As well as using Chain models to model geometry, we are also interested in using them to model physical behaviour. [Palmer & Shapiro 1994] give the following three definitions which are reformulations of definitions 1 to 3 in terms of Chain models.

Definition 14 A ***system*** (a distribution of quantities in time and space) is a set of chains, Q, defined over a cell complex K time and space.

Definition 15 A ***physical system*** is a pair $S = (Q, C)$, where Q is a system satisfying a set C of ***chain constraints*** on Q.

Definition 16 A ***physical behaviour*** $PB(C)$ is the set (equivalence class) of all physical systems (Q, C_i) that satisfy C, i.e. $PB(C) = \{(Q, C_i) \mid C_i \Rightarrow C\}$.

In other words, we have defined a system as a set of chains representing a distribution of physical quantities defined over a cell complex. A physical system is a system along with a set of constraints on its chains. A physical behaviour is the set of all physical systems that satisfy a particular set of chain constraints.

7.3.2.2 Constraint Elements

So what is the nature of these chain constraints? Clearly, there are many possible constraints we might put on a chain. However, if we are interested in constraints which model physical laws and which are easy to compute, the nature of the constraints we need is restricted. Specifically, Palmer and Shapiro state the following conditions:

- Constraints can be imposed only on chain coefficients associated with incident cells or adjacent cells
- All cells in the decomposition of space are similar in the sense of being able to 'implement' the specified constraints.

They go on to state that most physical laws can be placed into two categories, both of which act locally and so allow constraints to be defined consistently with the first condition (i.e. that they are imposed on chain coefficients associated with incident or adjacent cells). These categories of physical laws are:

1. Structural laws (conservation, balance, equilibrium), which are based on topological invariants and can be expressed using operations of boundary and coboundary; clearly, these operations constrain incident cells.

2. Constitutive laws (such as Ohm's and Hooke's), that represent phenomenological (macro) constraints corresponding to material properties; these are obtained and defined by local measurements.

They go on to describe ***constraint elements*** which are used as an abstract specification of the relationship between chains. For a particular physical law (balance or conservation, for example), a constraint element can be produced which specifies the constraints on chains in order to meet that law.

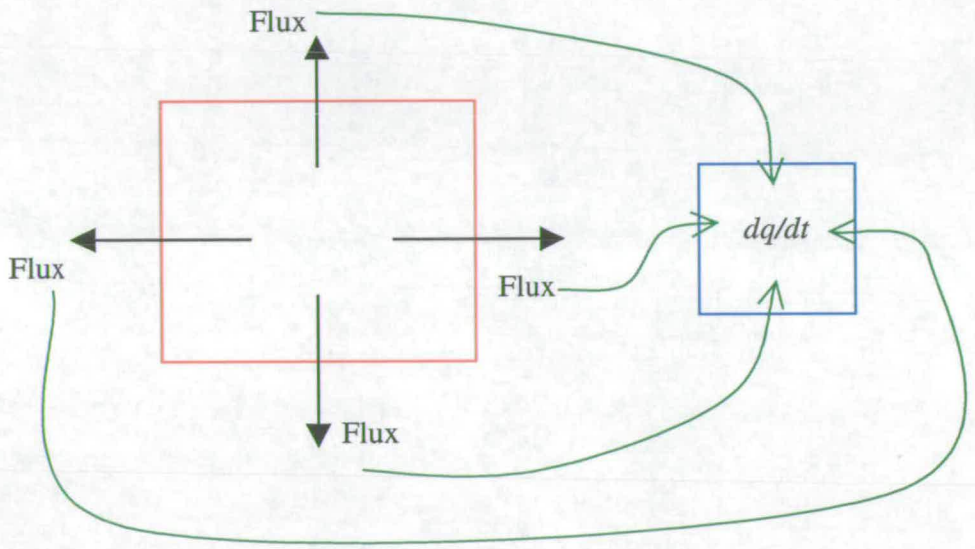


Figure 7-7 A Constraint Element for Conservation
after [Palmer & Shapiro 1994]

Figure 7-7 and Figure 7-8 illustrate constraint elements for the structural laws, conservation and balance respectively. For the conservation constraint element the constraint has the form $\frac{d}{dt}2 - chain = \delta(1 - chain)$. For the balance constraint element the constraint has the form $bf = -\delta(sf)$, where bf is a 2-chain representing the body force and sf is a 1-chain representing force through the 1-cells.

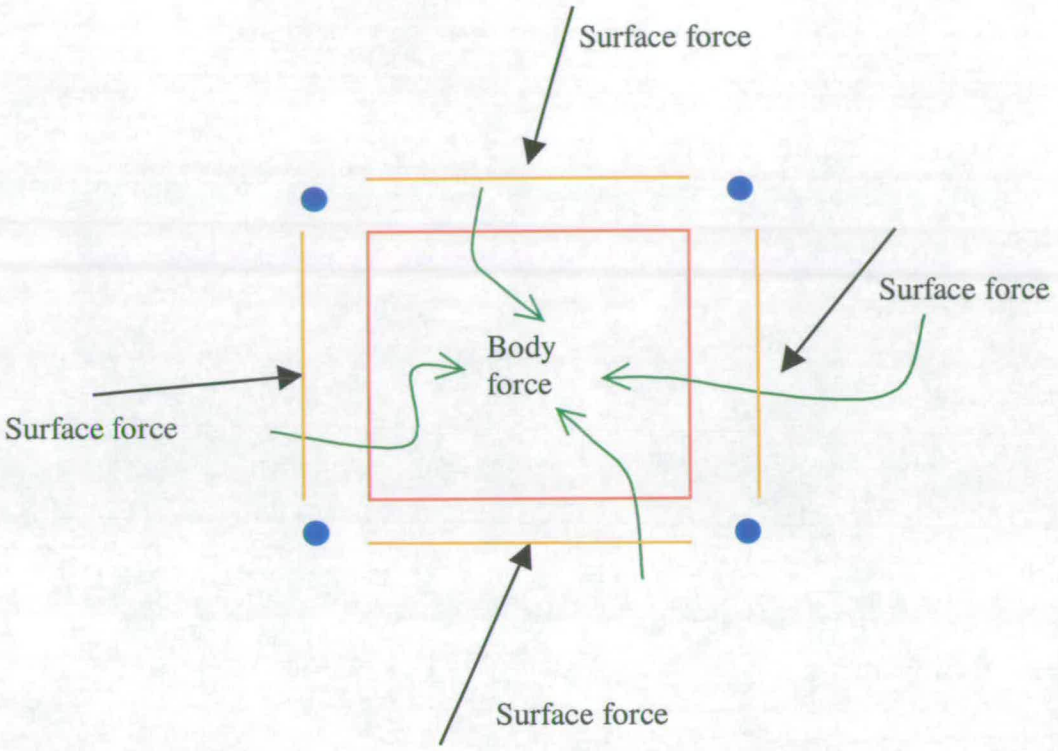


Figure 7-8 A Constraint Element for Balance
after [Palmer & Shapiro 1994]

It can be seen that these constraints rely only on the boundary and coboundary operators and could be applied to any cell type. Constraint elements for constitutive laws are more difficult to derive as they also depend on the cell type. Palmer and Shapiro derive, at some length, a chain constraint for elasticity, which relates force through a set of faces to the deformation of a cell. Such a chain constraint makes use of both structural and constitutive chain constraints.

7.3.2.3 Physical Elements

Definition 17 A ***physical element*** is

1. The cell complex K generated by a single n-cell of a particular cell type (e.g, simplex, cube).
2. A set of p-chains on K that represent physical state.
3. A set of constraints elements defining the constraints on the p-chains of K.

Physical elements are a computational model of a physical behaviour. They define the behaviour in a single n-cell of a system. Given a region decomposed into a complex, the physical element can be applied to each n-cell in the complex and so model the behaviour of the region. As [Palmer & Shapiro 1994] state:

> 'Physical elements can be viewed as 'object oriented' components for building computational models of physical systems. They are object oriented in the sense that 1) physical behaviour is defined in terms of a few definitions (like the notion of 'class' in object oriented programming) which may then be 'instantiated' by applying to regions of space, 2) they interact through predefined, well defined interfaces, which simplifies working with them - previously defined models of physics (say of fluid flow) need not be redefined when a new physical element (say of elasticity) is introduced. In fact, once each of these types of element has been defined, we may represent systems that contain interactions between elastic solids and fluids without introducing additional elements.'

7.3.3 Implementations of Chain models

[Palmer 1995] describes CHAINS; a computer language for modelling physical systems based on chain models. As an example of how the language can be used to

build models of physical systems, a program producing a finite element solution to plane strain is developed.

[Egli & Stewart 2000] describe the development of an application programming interface (API) for building chain models of physical systems. They illustrate the use of their API with a diverse range of examples such as tissue modelling with a mass-spring network, a simulation of a waving flag and a Cattmull-Clark sub-division for the recursive refinement of surfaces.

Although the use of Chain models in these applications does not enable the development of analysis code that would not otherwise be possible, it does raise the semantic level at which an analysis problem can be described.

If you consider a traditional implementation of the finite element method for structural analysis, a human has had to take the basic concepts of distributions such displacement, stress and strain, and represent these using a general purpose programming language such as FORTRAN or C using the data structures (such as floating point numbers) available in such a language. An execution path has then to be created to manipulate these primitives in a way that is consistent with the finite element method. The knowledge about how the simulation is undertaken is thus only implicit in the code. Hence, when later the code needs to be integrated with other code (perhaps to combine the code with a finite volume model of fluid dynamics), it is very difficult to reuse the code and much rewriting is needed.

In contrast, a language such as CHAINS, which makes use of Chain models, is able to explicitly deal with concepts such as, for example, physical quantities, equilibrium and conservation of mass. For CHAINS these form the data structures that the language manipulates. Thus a simulation built in such a language documents itself – the code shows explicitly which physical quantities are to be manipulated and how this is to be done in terms of high-level algebraic topological operations. It is the computer's job to 'compile' this code into a form that is executable by a computer. For a traditional implementation of the finite element method in FORTRAN, say, in

essence a human programmer is required for the first step in this compilation process from physical model to code based on FORTRAN data structures (the FORTRAN compiler deals with the rest of the compilation into machine code). The need for this is removed in CHAINS.

7.4 Chain Models in Design

In addition to the analysis and simulation applications of Chain models described in previous sections, [Palmer & Shapiro 1994] describe some other possible uses of Chain models in the design process. These are formal function specification and shape synthesis. They illustrate both of these applications using a bracket design problem taken from [Shapiro & Voelcker 1989]. Since their discussion of this use of Chain models is highly pertinent to the argument put forward by this thesis, namely that Chain models would be highly useful for shape optimisation, much of the following sections are drawn from their paper.

7.4.1 Function Specification

Typically the design specification for a bracket would relate how the bracket must interface with other parts, the forces to which it will be subjected, maximum deflections etc. (see Figure 7-9). The bracket has three holes with given diameters through which they interface with other parts, applied loads or constrained movement on those nodes and a physical behaviour relating the deflections to the applied loads [Palmer & Shapiro 1994] argue that while such specifications are common, they are rarely stated in a formal way which would enable them to be amenable to representation on a computer. Chain models provide such a formalism for defining a design specification.

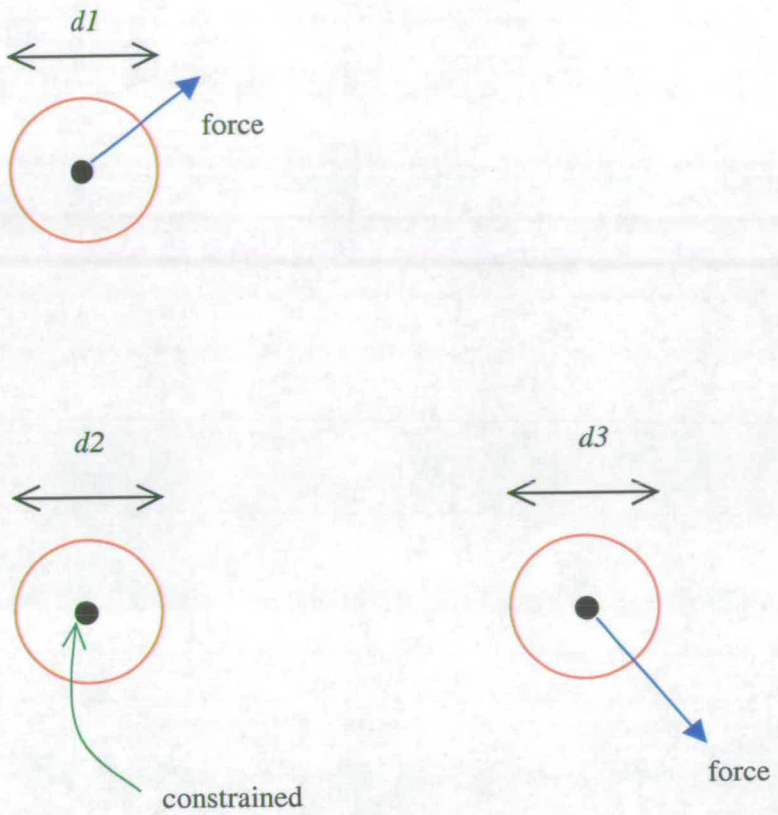


Figure 7-9 **A Sample Specification for a Bracket**

after [Palmer & Shapiro 1994]

[Palmer & Shapiro 1994] give a possible chain model specification, shown in Figure 7-10. Each hole is represented with a 0-cell. 0-chains are defined for positions (x), deflections (u) and forces (f). 1-cells are formed with pairs of the 0-cells forming a cell complex. 1-chains are formed using the coboundary operator to form the 1-chains: relative position $dx = \delta(x)$, relative displacement $du = \delta(u)$ and relative force $df = \delta(f)$. Once these cells and chains are defined, constraints can be set on the values that these chains can take. For instance, values for required forces or displacements can be defined or bounded. These types of constraints are similar to loading

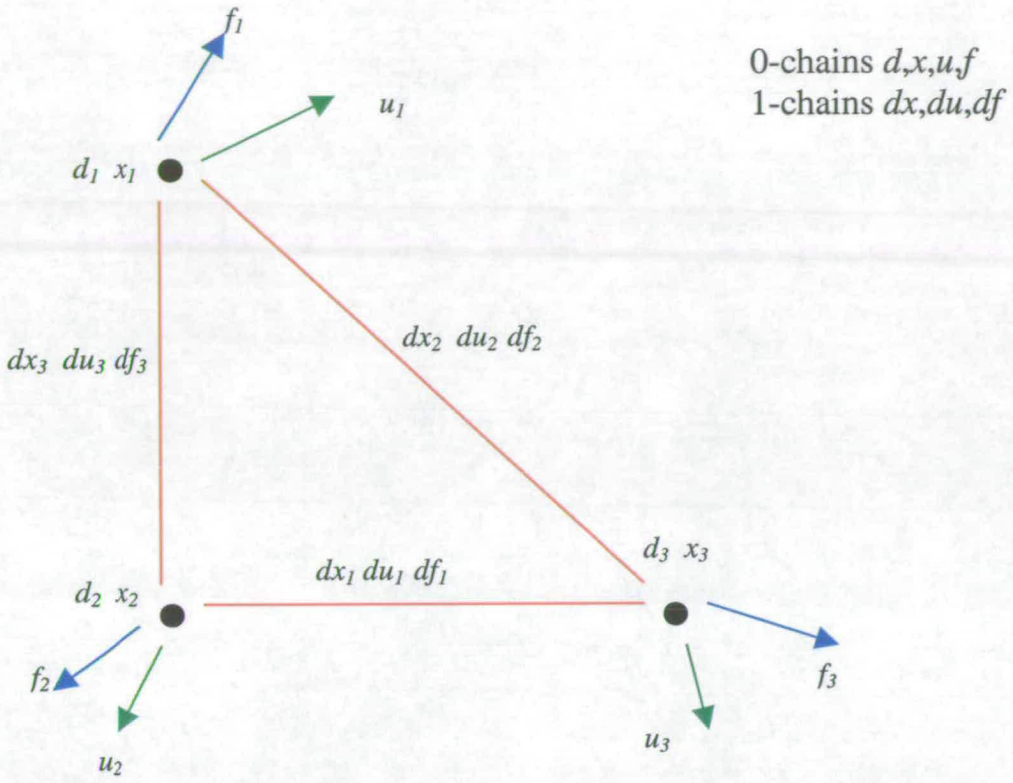


Figure 7-10 **A Chain Model Specification for a Bracket**
after [Palmer & Shapiro 1994]

conditions applied to finite element models and similarly should be well-posed, so that the problem is not over-constrained (i.e. either displacements or forces are defined but not both).

In addition to such common constraints, [Palmer & Shapiro 1994] argue that the chain model of function specification can support much more general constraints. Constraints can be placed on the relationship between chains such as force (*f*), relative force (*df*), displacements (*u*) and relative displacements (*du*). For instance, *du.df* can be constrained to be below some bound.

7.4.2 Shape Synthesis

The previous section showed how a chain model could be used to produce a formal design specification for the bracket discussed. It specifies displacements and forces but does not model the geometry. It is still necessary to generate a geometry for the bracket. This can be seen as a systematic transformation from the specification chain model to a more detailed chain model which specifies the geometry embedded in space that is still consistent with the specification.

The specification cell complex needs first to be transformed. The holes will need to be transformed from the 0-cells into a 1-complex representing the bracket's boundary around the hole. The chains also need to be transformed from 0-chains on the node to 1-chains. Figure 7-11 and Figure 7-12 show how this might be done.

The bracket's geometry will be represented by a 2-complex, the boundary of which is already partially defined by the 1-complexes representing the holes. The geometry synthesis problem is now to find a 2-complex embedded in space, with appropriate chains representing the physical behaviour, defined over it. If the bracket is to be made of an elastic material, cubical elastic physical elements might be used. The chain model formed will satisfy the design specification, geometry and will be physically realisable.

The question remains how this 2-complex should be found. [Palmer & Shapiro 1994] discuss the approach given by [Shimada & Gossard 1992] whereby the holes are connected by 'support regions'. The size of these regions is proportional to the forces applied. Alternatively, the whole design area around then holes might be filled as described in [Bremicker *et al.* 1991] and shape optimisation undertaken.

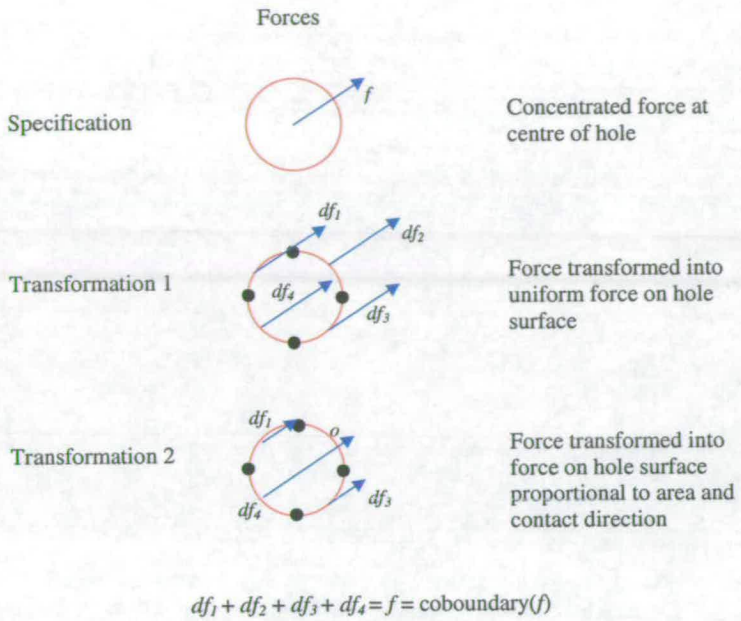


Figure 7-11 **Transformation of Abstract Chain Model Specification of Force to a Spatially Embedded Chain Model**
after [Palmer & Shapiro 1994]

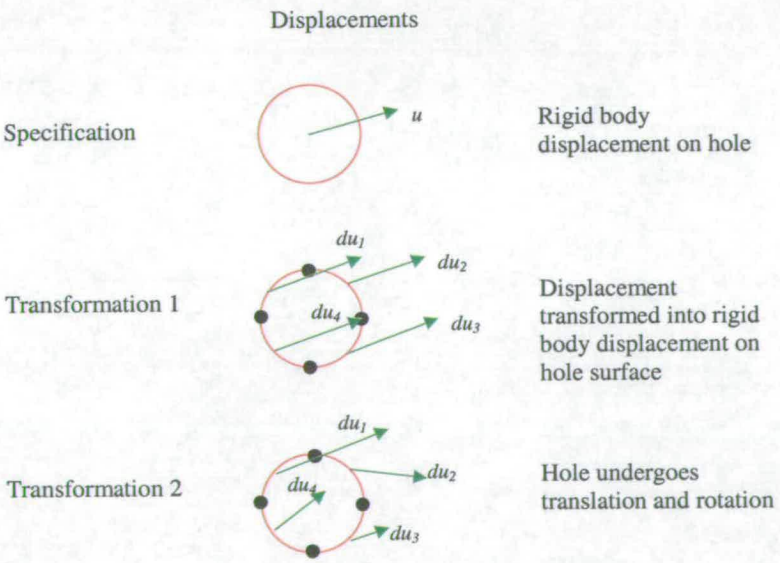


Figure 7-12 **Transformation of Abstract Chain Model Specification of Displacement to a Spatially Embedded Chain Model**
after [Palmer & Shapiro 1994]

7.5 Chain Models and Shape Optimisation

Chain models and shape optimisation have complementary roles. In chain models, we have a have a formal way of specifying desired behaviour, of representing geometry and of calculating behaviour. Referring back to Section 6.9 it can be seen that these were some of the properties that were required for a common data structure for shape optimisation. Therefore, Chain models could possibly be a method of formalising the process of undertaking shape optimisation.

In Section 7.4.2 Chain models were shown to be capable of constituting a formal design specification and the process of synthesising geometry could be seen as a systematic transformation of this specification chain model to a chain model embedded in space which represents both the geometry and the physical behaviour of the component. Shape optimisation is one possible tool that could be used to facilitate this transformation of chain models.

How might shape optimisation and chain models be integrated into a formal geometry synthesis process?

The use of Chain models by themselves does not solve the problem of producing sensible geometries, decomposing that geometry into a reasonable mesh and ensuring an accurate analysis. As discussed in Section 7.3.3 languages for physical analysis (such as CHAINS) based on Chain models allow physical simulations to be written at a high semantic level in a formal and consistent manner and thus enable greater code reuse and interoperability. Similarly, a language for shape optimisation based on the Chain model formalism could allow shape optimisation algorithms to be written in which the geometry definition, discretisation and subsequent analysis of physical behaviour could be developed at a high level in a consistent way.

Chains would also allow a formal way to define a specification and objective for the optimisation. Shape optimisation can be reformulated then as the systematic transformation of the chain model specification into an optimal chain model fully embedded in space which satisfies the constraint elements necessary for the physical

behaviour of interest – or indeed behaviours of interest. Chain models would make the specification and analysis for multi-disciplinary optimisation much easier to implement. This would be possible without the difficulties of integrating various incompatible analysis routines.

7.5.1 Opportunity for novel shape optimisation techniques in a Chain Model Framework

It is possible to formulate any of the current approaches to shape optimisation in the Chain model framework. The Chain model framework would provide a useful formalisation of the process in terms of transformations of Chain models from specification to full geometry. However, one of the primary motivations behind looking for a common data structure for shape optimisation was the desire to develop novel shape optimisation algorithms that could be useful in the design process. Therefore, this section looks at the possibilities for novel algorithms for shape optimisation, which are facilitated by Chain models.

The main opportunity that Chain models offer for the development of new algorithms is that they allow access to information about the spatial performance of a shape. They make it possible for the shape to be changed in response to the behaviour of the component calculated in the analysis.

As discussed in the previous section, shape optimisation can be viewed as the systematic transformation of the chain model specification into an optimal chain model fully embedded in space. Since optimisation is a process of searching for a solution, this transformation will be the result of a repeated loop. The specification Chain model is transformed into a partial geometry definition in some way. This partial geometry Chain model is converted to some full geometry Chain model. This geometry model might itself be set up so that analysis can take place (i.e. it has appropriate chains defined for the desired analysis) or a further transformation might be needed. The results of this analysis are used to change the geometry shape. With a traditional optimisation technique, this will involve calculating an objective function

and whether constraints are violated and reporting this information to the optimiser which will vary parameters used to build the geometry Chain model. With Chain models, though, there is the possibility of using more of the information in the analysis Chain model to change the geometry model. The loop changing geometry, undertaking an analysis and subsequently changing the geometry can be repeated numerous times until some termination criteria is met.

How might the information present in the analysis Chain model be used to change the shape? It is perhaps easiest to demonstrate with an example, the bracket design problem described in Section 7.4.1. This is not intended to be a description of a necessarily good algorithm for the design of this bracket. Rather it is to show the types of possibilities that become available when using Chain models.

Figure 7-10 shows a Chain model specification for the bracket, with perhaps constraints on the deflection of the holes, a stress constraint and a desire to minimise weight. Figure 7-11 and Figure 7-12 show how this specification might be transformed into a partial definition of the bracket geometry for the holes. We now need to move from this partial definition of geometry to a 2-chain fully embedded in space with Chains defined on it to model elasticity, with minimum weight and meeting the constraints on deflection and stress.

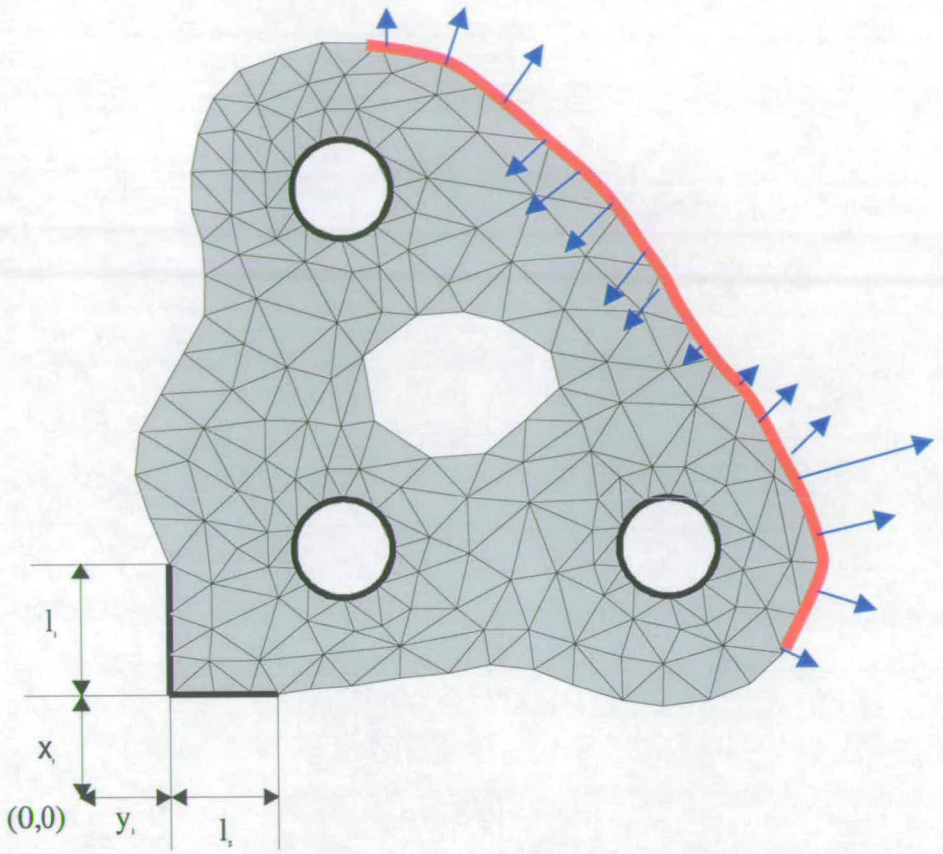


Figure 7-13 **Example of Candidate Bracket with Parameterised Corner and 'Grown' Mesh**

Figure 7-13 shows one way that this might be achieved. Two additional features are added to the partial geometry, a 'corner' to the bottom left hand side and a spline curve to the upper right. The full geometry is represented by the simplicial Chains, these are formed by meshing the area partially enclosed by the partial geometry model. Loads and displacements can be transferred from the holes to the appropriate nodes and an analysis done. The geometry can be changed in one of three ways:

(a) Similar to a traditional shape optimisation the corner dimensions and position are under the control of a hill-climbing optimiser. The optimiser can change the corner dimensions and then the area is remeshed and so on.

(b) The simplicial cells can be removed if the stress on them is below some threshold value and a further analysis undertaken and so on. This will remove material from those areas in which it is being underused.

(c) The spline control points can be moved in or out depending on the stress of the cells with which it is adjacent. The area can be remeshed and another analysis done and so on. Thus, that section can become thicker or thinner as necessary depending on the results of the analysis.

All three of these geometry-changing methods might be used at once. For example, (a) might be used as an outer loop, (b) as an intermediate loop and (c) as an inner loop. Again, it should be stressed that this is not intended to be an example of a 'good' algorithm for this problem. Instead, it demonstrates some of the ways in which analysis information might be used to influence the geometry using the Chain model framework.

Another use for Chain models in shape optimisation might be to develop further the 'object-oriented' nature of physical elements so that they might act more like 'intelligent agents'. For instance, an instantiation of a physical element might deduce that the approximation that it represents might be inaccurate in its current configuration and so change itself. For instance, an instantiation of a physical element equivalent to a solid finite element might transform itself into a shell element when its size in the third dimension drops below some value. Similarly, it might be possible that Chains can be made 'intelligent' so that they can automatically change themselves in response to their internal condition or the environment around them. For example, the spline in Figure 7-13 might be made 'intelligent' so that it can change its shape in response to its environment.

8 A Morphogenetic Approach to Shape Optimisation

8.1 Summary

This chapter describes a novel approach to shape optimisation inspired by the way in which biology produces load-bearing structures. It is programmed using the Chain model framework [Palmer & Shapiro 1994] described in the previous chapter. It demonstrates how novel algorithms can be developed in this framework which can access all of the information generated by the analysis software and use this information to change the shape of the component to be optimised.

This approach uses a cellular shape representation, where cells from the Chain model framework are given some of the abilities of biological cells. They can divide, move, pass messages to neighbours and die. They are also allowed to respond to the stress upon them. This allows them to die where stress is low and in this way remove redundant material.

Each cell is programmed as an independent agent in the Swarm programming language. The behaviour of the cells is influenced by various parameters that can be thought of as analogues of biological genes. A genetic algorithm is used to evolve these genes towards producing good shapes.

The approach was applied to the problem of finding shapes for bicycle frames, arches and a cantilever beam. Realistic high performance shapes were produced demonstrating the possible usefulness of such an approach.

8.2 Introduction

8.2.1 Overview and Motivation

In the approaches to using genetic algorithms for shape optimisation described in previous chapters, the genes explicitly encode for the components' shape. For

example, a gene represents a dimension in a parameterised CAD model or defines whether a voxel is filled or not. For these approaches, the generation of the shape from the chromosome is trivial. In this chapter, a morphogenetic approach is used so that the genes do not explicitly encode for the shape, but instead change the behaviour of the cells and hence influence the shape generated during a growth stage in which the cells can respond to the calculated stress on them.

As described in Chapter 6, the primary motivation behind this work was to attempt to make more use of the information generated by the finite element analysis. The analysis produces large amounts of data on the local performance of a component at particular positions. In contrast to most approaches to shape optimisation, this approach looks to make use of this information so that the shape optimisation algorithms can change the shape in response to this information, in order to improve the component's performance.

A second motivation was the desire to establish whether the concept of 'intelligent cells' discussed in Section 7.5.1 could be used in a useful way. This, therefore, determined that control of the shape generation should be 'bottom-up'. Cells respond to their internal state and to the environment to generate the shape and are not controlled by a centralised algorithm.

This work did not attempt to produce a definitive algorithm for shape or topological optimisation using morphogenesis. Instead, it was intended as a study into whether such an approach shows any promise for shape optimisation.

8.2.2 Morphogenic Evolutionary Computation

[Angeline 1995] defines morphogenic evolutionary computation (MEC) (*morphogenic* being a synonym of morphogenetic) as 'evolutionary computations that distinguish between the representation that is evolved and the representation that is evaluated by the fitness function'. Typically, MEC uses evolutionary algorithms, along with a growth stage, to produce a solution to a problem. Just as biological embryology takes a genotype and, through a complex development process, produces

a physical phenotype, morphogenic evolutionary computation uses a more complicated mapping from genotype to phenotype than those typically used in evolutionary computation. An evolutionary algorithm is used to manipulate the genotype. The genotype is then developed into a phenotype by some non-trivial process. The phenotype is then evaluated.

Angeline reviews the relatively few pieces of research undertaken which use a morphogenetic development stage in evolutionary computation [Angeline 1995]. He introduces a formal description of morphogenic evolutionary computation and describes its potential advantages over standard evolutionary computation.

The advantages he identified were firstly *evolvability*. Using morphogenesis, it may be possible to make reproduction operators more effective at moving through the search space to good solutions. Secondly, there was the possibility of producing solutions with large structures with relatively simple genomes.

He identified three types of development used for morphogenic evolutionary computation: ***translative***, ***generative*** and ***adaptive*** development functions. Translative development is essentially a fairly trivial mapping from a genome to a larger structure. Generative development involves a recursive function whereby repeated application of some growth rules produces the required structure. Examples given of generative development functions were Lindemayer (L-) systems and production rules. Adaptive development functions may be recursive like generative development, but they also involve some adaptation of how the development takes place during evolution. The work described in this chapter does not neatly fit into any of these categories, since in this work the growth of the structure itself is influenced by the structure's performance during the growth stage.

[Bentley & Kumar 1999] classify embryogenies (an ***embryogeny*** being the growth process by which a genotype becomes a phenotype) for evolutionary algorithms into three types, ***external***, ***explicit*** and ***implicit***. In external embryogenies, the growth process is generally hand-coded. The embryogeny is not itself evolved, but instead is

fixed. Parameters are evolved which feed into the embryogeny, which then generates the phenotype. Dawkins' Biomorphs [Dawkins 1986] [Dawkins 1987] can be seen as using an external embryogeny.

For explicit embryogenies, the steps in the growth process are given by explicit instructions, which are evolved. These instructions are executed to form the phenotype. Such embryogenies might make use of tree structures of instructions and be amenable to evolution using genetic programming.

[Bentley & Kumar 1999] explain implicit embryogenies as having no explicit set rules defining the embryogeny. Instead, they are closer to natural embryogenies that use 'indirect chains of interacting rules'. These rules are evolved and, through their application to the elements of the growing solution, form the phenotype.

[Bentley & Kumar 1999] give a list of advantages of a morphogenic stage, namely:

- **Reduction of the search space**. A relatively small number of genes can be used to generate a phenotype with a much larger (more complex) structure.
- **Better enumeration of the search space**. The genotype space can be designed to be easier to search through than the phenotype space, making the search more efficient. This can be compared with some of the ideas developed in Section 6.8.
- **More complex solutions in the solution space**. The use of growth rules can allow for the generation of more complex phenotype structures than could otherwise be evolved.
- **Repetition**. The use of a morphogenic stage, if properly designed, might allow for the use of repeating structures, exploiting symmetry and segmentation.
- **Adaptation**. Phenotypes can be grown so that they can meet constraints, change in varying conditions, and correct malfunctions in the design. Of the advantages given, the work described in this chapter perhaps looks to exploit this advantage the most.

They give two potential disadvantages of morphogenic evolutionary computation:

- **They can be hard to design.**
- **They can be hard to evolve.** Bentley and Kumar cite pleiotropy (one gene causing numerous phenotypical traits) and the disruption of child solutions as potential problems.

Morphogenic evolutionary computation has been applied in a number of application areas. [Jakobi 1995] [Jakobi 1996] describes its use for the generation of Artificial Neural Networks (ANNs) for robot controllers. His approach would be termed *generative* by Angeline and *implicit* by Bentley and Kumar. [Broughton *et al.* 1999] describe an explicit embryogeny, using genetic programming and Lindenmeyer (L-) systems, to evolve three-dimensional structures. [Hornby & Pollack 2001] [Hornby 2003] describe the use of an evolutionary algorithm that evolves L-systems. These L-systems are used as a generative encoding to produce voxel-based objects. The system is used to evolve table designs. [Bentley & Kumar 1999] compare an external embryogeny, an explicit embryogeny and an implicit embryogeny on a problem in which tessellating tiles are to be evolved. They found that their implicit embryogeny performed best. [Kumar & Bentley 2003b] attempt, with success, to evolve the shapes of letters of the alphabet, finding an implicit embryogeny to scale well. [Eggenberger 1996] uses development processes such as cell division and cell differentiation, to create neural control structures for real-world agents using an artificial evolutionary system. [de Garis 1994] uses an implicit embryogeny using cellular automata to grow very large neural nets.

8.2.3 Artificial Life and Structural Analysis and Optimisation

[Hajela 1998] reviews some of the recent applications of artificial life to structural analysis and design. This included evolutionary algorithms for optimisation. An approach to using cellular automata (CAs) for structural analysis was put forward. The motivation for this was that specialised massively parallel hardware for

implementing CAs might allow for very quick analysis of large structural problems. The rules in the CAs were evolved using a GA with a fitness depending on how well the CAs were able to approximate stress results from a finite element analysis [Hajela & Kim 2001].

[Kita & Toyoda 2000] use cellular automata for topology optimisation. The design domain is decomposed into a rectangular grid of cellular automata. Cell thicknesses are used as the design variables. These thickness variables are updated based on local rules and on the stress calculated by a finite element analysis.

[Taura & Nagasaka 1999] use a morphogenic approach to 'growing' shapes. They use an unusual shape representation. A unit sphere is used on which points are placed. A free form object is then formed around the centre of this sphere. The density of points in a particular area of the sphere determines how far the free form object is pulled in that direction. Initially only a few points are on the sphere, but rules are used to generate new points or remove points on the sphere which are analogous to cell division or cell death. A genetic algorithm is used to evolve these rules for particular design applications.

An interesting approach to producing triangular meshes for a two-dimensional domain is described in [Saitou & Jakiela 1994]. Each element is thought of analogously to a biological cell. It can grow outwards or reproduce on one of its sides to fill the domain with a high quality mesh. The cell has an internal state and variables describing how far it is from other cells or the boundary. The behaviour of the cell is determined by rules that are determined by a classifier system evolved by a genetic algorithm. The fitness given by the quality of the mesh, in terms of element size and shape and the proportion of the desired area which is meshed. The method could produce good meshes for constant shape. However, it was hoped that once the rules were found for meshing a particular shape then it would be possible to reuse these rules for different shapes. However, it seemed that there was some sensitivity of the rules to the specific shape. [Langham & Grant 1999] describe a similar application in which rules are evolved for mesh generation.

8.2.4 Models of Biological Cellular Development

Some work has been done in biology on computational modelling of development. [Agarwal 1994] and [Agarwal 1995] describes a 'Cell Programming Language' which is used to model phenomena exhibited by interactions between cells. [Fleischer 1995] has developed a simulator of biological multi-cellular development. Chemical, mechanical influence and cell lineage factors and other biologically feasible mechanisms can influence cell development. This was applied to synthetic biology in order to explore questions of pattern formation and morphogenesis, artificial evolution of neural networks and computer graphics modelling. The work described for modelling the development of cellular structures was much more detailed and biologically realistic than the one described in this chapter. However, as is discussed in Section 8.6.3, such modelling could be the basis of further investigation following on from the work reported in this chapter.

[Kumar & Bentley 2003a] [Kumar & Bentley 2003c] describe the development of an 'Evolutionary Development System' (EDS) which attempts to model aspects of the biological process of development. The system represents cells, embryos, genes, cell cytoplasm, cell walls, proteins, receptors, transcription factors and cis-regulatory regions. The use a genetic algorithm to evolve the genes. In [Kumar & Bentley 2003c] individuals are grown within this biologically plausible model and assigned a fitness according to how well they achieve a particular shape. Experiments were undertaken comparing different modes of cell division, where the shape to be generated was a straight line and a sphere. They concluded that different methods of oriented cell division do affect the final developed solution.

8.2.5 Related Work on Shape & Topology Optimisation

[Baumgartner & Mattheck 1994] and [Mattheck *et al.* 1994] note that living biological load bearing structures, such as bones and trees, seem to have developed mechanisms for growing and changing their shape in order to adapt to the conditions found in their environment. Such mechanisms have evolved in order that organisms

may meet the need for simultaneously lightweight and reliable structures. He states that the *axiom of constant stress* is the principal rule that determines this growth. In other words, the growth that takes place tends to equalise the stress throughout the biological load carrying structure. Thus, material grows where there are stress concentrations that might cause failure and less material is used where underloaded, reducing the weight of the structure.

Clearly for many engineered mechanical components, reliability and minimum weight are desirable. These are the aims of many shape optimisation techniques. Mattheck therefore attempts to mimic such mechanisms for the generation of shapes for mechanical components. He describes two complementary techniques inspired by this biological analogy, which they term *Soft Kill Option* (SKO) and *Computer Aided Optimisation* (CAO).

SKO simulates the adaptive mineralisation of bones, whereby increased mineralisation takes place where the stress is high. This leads to the distribution of stiff, high strength matrices which are well-adapted to the particular loading that is experienced. SKO takes this process and looks to apply it to find optimal topologies for complex loading situations. SKO takes an initially rectangular finite element mesh covering the whole of the proposed design area with a constant Young's Modulus. The stresses due to the loading are then determined using the finite element method. The Young's Modulus of each element is then changed as a function of the stress. Areas of higher stress are given higher Young's Modulus; areas of lower stress are given lower Young's Modulus. The function used is not given in the paper. This is repeated several times until a clear distinction between areas of high and low Young's Modulus is achieved. The shape of the structure is then determined by calculating an isoline of the Young's Modulus. Areas of high Young's modulus are inside the structure, whilst areas of low modulus are outside.

The second method, *Computer Aided Optimisation* (CAO), simulates surface swelling of the structure according to the stress distribution, similar to the way that trees grow. This leads to a more homogenised distribution of stresses on the surface.

Again, the whole of the design area is meshed with rectangular finite elements. The stresses are calculated for the load case required. The next step involves replacing the mechanical loading with a fictitious temperature field, where the temperature is determined from the stress calculated previously. High stresses result in high temperatures and vice versa. This thermoelastic problem is then solved with the heat expansion coefficient set at zero for everywhere except the surface layer. Thus, a fictitious 'thermal' surface expansion takes place based on the magnitude of the stresses calculated. The expanded shape is then used in a further iteration of this process. This loop is continued until a constant surface stress is achieved.

[Mattheck *et al.* 1994] describe the use of SKO to determine topology firstly and then CAO to produce homogenous surface stresses of optimisation of a cantilever beam.

The CAO method is not specifically aimed at the typical shape optimisation problem of minimising volume whilst remaining within a stress constraint. Instead, CAO reduces stress concentrations on the surface in order to increase reliability. [Chen & Tsai 1993] extend the simulated biological growth approach so that it can be used for two different design procedures: minimising volume subject to a maximum stress constraint and minimising maximum stress subject to an area constraint.

A very similar approach to the SKO called the Hard Kill Method (HKM) is described in [Bulman *et al.* 2001]. In this method, rather than the element's Young's modulus being varied linearly with the Von Mises stress on the element, the Young's modulus is reduced to a very small value if its stress is below some value. Thus, an element that is being underused is 'killed' (although it actually remains in the mesh with a low Young's modulus).

A similar approach, Evolutionary Shape Optimisation (ESO), is a technique for topology optimisation developed by Xie and Steven [Xie & Steven 1997] [Querin *et al.* 2000]. Despite its name, ESO does not use evolutionary algorithms; instead, 'evolutionary' refers to the gradual removal of material to achieve an optimal design.

The basic process that is used in ESO is to start with an initial rectangular finite element mesh. An analysis is then undertaken. Elements are then removed depending on this analysis. A further analysis is then done and further elements are removed. This repeats, gradually improving the topology.

The decision on which elements to remove depends on the criteria that is being optimised. Typically in topology optimisation, the objective is to minimise compliance of a structure for a given volume of material. With this objective, ESO removes material where the stress is low. ESO has been applied to a number of other problems including buckling [Rong *et al.* 2001] and frequency response optimisation [Zhao *et al.* 1998] [Xie & Steven 1996] The ESO method has the advantage that it is easy to integrate with standard finite element packages. [Tanskanen 2002] provides a theoretical study of ESO.

A disadvantage of SKO, HKM and ESO is that they generate shapes with an unsmooth boundary. There have been a number of approaches to producing smooth geometry from the mesh. [Chen *et al.* 2002] describes an approach to ESO, which they call Nodal Evolutionary Shape Optimisation (NESO). In this approach rather than elements being removed, nodes are allowed to migrate from low stress areas into high stress areas. Remeshing is undertaken when element shapes become invalid and boundary smoothness is maintained.

The techniques described above all look to change the shape based on information generated by the analysis routine. These partially addressed the aim of this work to utilise the information generated by the FE analysis. However, they are essentially 'one-shot' methods in that only one of the possible trajectories through the space of possible shapes is taken, from an initial shape in which the whole of the design area is filled to a better shape.

The work described in this chapter seeks to make use of a growth stage, in which the shape is taken through a sensible trajectory through the space of possible shapes informed by the analysis. However, it seeks to avoid the 'one-shot' nature of the

described techniques by moderating the behaviour of the growth by a number of factors. These factors are controlled by a genetic algorithm and the shape space is thus searched through a number of different trajectories.

8.2.6 Aims

- To determine whether a morphogenetic evolutionary algorithm shows any promise as an effective approach to shape optimisation.

- To determine whether the information generated in the finite element analysis on a shape's performance can be used to influence the shapes generated by a morphogenetic genetic algorithm approach to shape optimisation.

- To establish whether the use of 'intelligent cells', where finite element are endowed with simple behaviours, can result in emergent behaviour which produces efficient shapes.

8.3 Implementation

8.3.1 The Design of the Algorithm Used

It was decided not to mimic biological morphogenesis too closely. This was for a number of reasons. Firstly, to imitate the complexity of biological morphogenesis seemed too computationally expensive. Secondly, the determination of the details of organism growth is still an area of much research for developmental biologists [Solé *et al.* 1999] and an area in which the author has only limited knowledge. Thirdly, it is necessary that the loads can be applied to the component throughout the shape generation process. This would not be possible if the initial shape was 'small'. Finally, from a pragmatic point of view it seemed that certain engineering rules-of-thumb might be useful. Namely, that material should be removed from low stress areas where material was not being efficiently used and that material should be added in areas where stress is too high.

The purpose of the morphogenetic stage was to provide a method of producing a set of shapes that followed a sensible trajectory through the space of possible shapes, which was informed by the results of the analysis undertaken. Bearing this in mind, it was necessary to design the morphogenetic stage so that it was able to respond to information generated about the shapes' performance and, in general, move towards improved shapes. Many aspects of the morphogenetic stage should be fixed to enable this, however certain aspects should be variable so that by setting these variables at differing values the exact operation of this stage could be controlled. This would enable a large number of varying trajectories through the shape space to be followed. These variables would be under the control of a genetic algorithm.

It was decided that a cellular shape representation would be used. This was for a number of reasons. This representation would be easily implementable in the Chains framework described in Chapter 7. Secondly, biological morphogenesis is clearly cellular in nature. Thirdly, since the data from the finite element analysis would be available for each element, a cellular representation would make it easy to integrate this information. It was therefore decided to use a triangular cellular shape representation. This would then be used as the finite element mesh.

Finally, the nature of the development stage needed to be decided on. It was decided to copy some of the features of ESO since these would be simple to implement and had proven themselves to be successful for topology optimisation. During the growth stage, cells would be forced to die if their stress falls below some threshold (Von Mises) stress value. This threshold would be slowly increased throughout out the run until a constraint on area was achieved.

8.3.2 Overview of the Algorithm Used

The algorithm naturally split into two parts: the genetic algorithm and the shape development stage. The genetic algorithm was used to evolve values for some of the variables that were used in the growth stage. Thus, the genetic algorithm (GA) formed an outer loop, evolving individuals whose genes varied the way in which the

shape development stage produced the shape. When an individual in the GA's population is evaluated, the values for genes are passed into the cells and then the morphogenetic stage is allowed to run. Following this, the best fitness (typically $-\delta$, where δ is the maximum deflection) that is achieved through the growth stage is used as the individual's fitness. The GA is described in more detail in Section 8.3.8.4.

Below is given a brief overview of the mechanics of the algorithm used for the morphogenetic stage. Much greater detail on the actual implementation is given in Section 8.3.5.

Each cell was implemented as a two-dimensional triangular cell (2-cell) from the Chains framework. As discussed in the previous chapter, each 2-cell is bounded by edges (1-cells), which in turn are bounded by nodes (0-cells). The complex formed by all of these Chains cells is used as the shape representation. Each of these cells is implemented as an agent in the Swarm programming language described in Section 8.3.4. Only the triangular 2-cells are 'active' agents so that they are able to act independently. The 1-cell edges and 0-cell nodes could, however, act to check their status and act if they were currently invalid – this is explained further in Section 8.3.5.

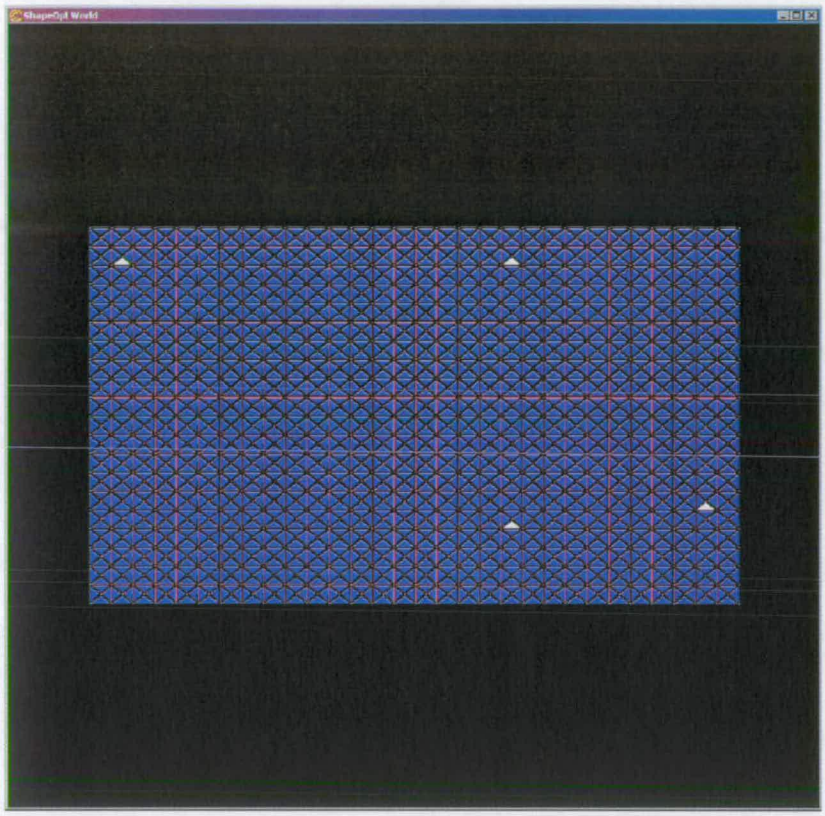


Figure 8-1 The Initial Mesh

As shown in Figure 8-1, an initial mesh of triangular cells was used which filled the whole proposed design area. Loads were applied to this mesh and a two dimensional static elastic analysis undertaken using the Ansys finite element package by importing the mesh via node and element files. Quadratic elements were used. Ansys returns the Von Mises stress and each element is informed of the Von Mises stress on it. Each element is then allowed to 'act'. Possible actions are to, *kill*, *divideSelf*, *smooth* or *boundarySmooth*. Once all the consequences of these actions have been completed, cells check their status to ensure that they still form a valid cell complex and any necessary actions are then undertaken to kill any 'hanging' edges, 'isolated' nodes or kill any cells which only contact each other through one node.

The basic execution path followed during the morphogenetic growth stage is shown in Figure 8-2.

Elements *die* if the stress on them is below some threshold value. This threshold value changes through time, starting quite low and is increased until the shape drops below a constraint on the area of the shape. This threshold stress is moderated by a number of factors such as position, whether the cell is at the edge of the shape and whether the death of an element would result in a change in the topology of the shape. The strength of these factors are under the influence of genes which are the same for each cell.

Elements *divideSelf* if the stress gradient across them is above some threshold value. This action was needed because the shape representation and the analysis mesh are the same. It is therefore necessary to try to ensure that the analysis produces sufficiently accurate results. Thus, the split action acts in a rudimentary way as an adaptive meshing routine, which is controlled from the 'bottom-up'. Cells split in a way that is similar to approaches used in Delaunay triangulation [Filipiak 1996]. The cell schedules itself to die along with any other cells with nodes within its circumcircle. A new node is placed at the centroid of the splitting cell and new cells are created using this node and those nodes surrounding the newly formed 'hole'. Section 8.3.8 describes this in some detail.

The *smooth* action is undertaken by every cell following any other actions. Again, this is to ensure that the mesh is sufficient to enable the finite element analysis to produce accurate results. The 'smooth' action takes each of the nodes of the cell and moves it towards the centroid of those nodes with which it shares an edge.

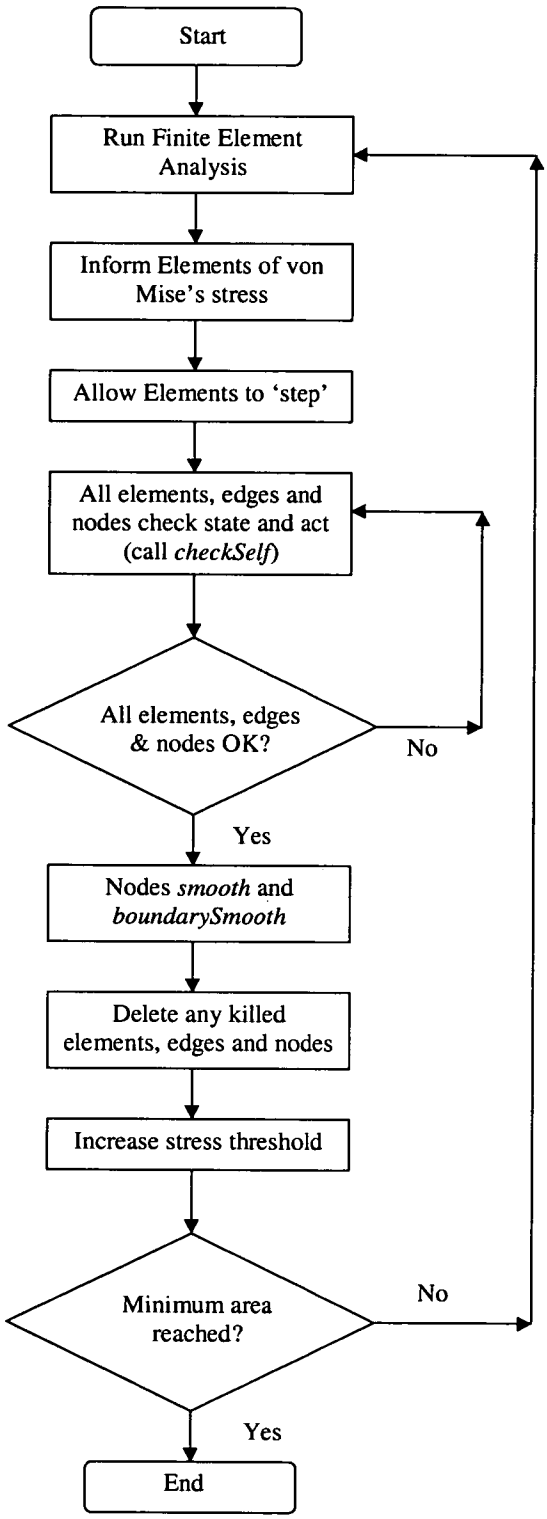


Figure 8-2 **Overview of the Morphogenetic Stage**

The *smoothBoundary* action aims to produce a smooth outer boundary for the shape. Like *smooth*, *smoothBoundary* is used following all other cell actions. Those cells that have an external edge move their external nodes towards the centroid of the two nodes that share a common edge to the node being moved.

This loop continues until a minimum area for the shape is reached.

8.3.3 Using the Chains Framework

The object structure of the program written was implemented so that it mapped onto the data structures discussed in the previous chapter. It should be noted however that due to time constraints it was not possible to code the finite element analysis in the Chains framework. Instead, Ansys was used to undertake the finite element analysis and the results read back into the cells.

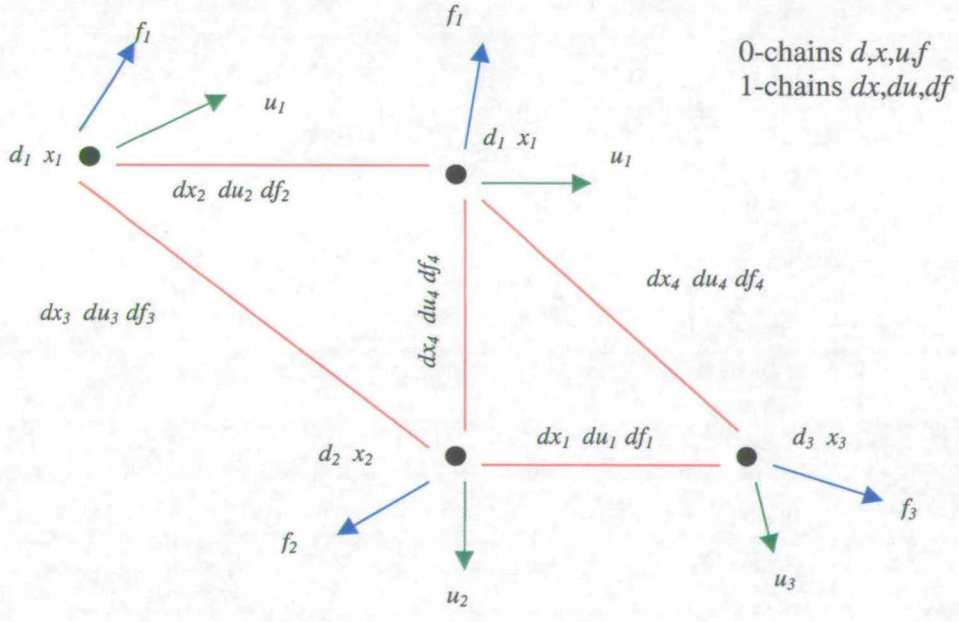


Figure 8-3 A Chain Model Specification for a Bicycle Frame

The cells in this chapter are an attempt to extend the physical elements described in 7.3.2.3 to give them 'initiative'. This ties in with the idea of intelligent physical elements described in last chapter.

Three problems are addressed in this chapter, the design of a bicycle frame, a load bearing arch and a cantilever beam. The process of shape and topology optimisation of each of these can be thought of transformation of a Chain model specification as shown in Figure 8-3 to a Chain model fully embedded in space.

8.3.4 Swarm

It was decided to implement the morphogenetic shape optimisation software in 'Swarm' [Burkhart 1994] [Minar *et al.* 1996]. Swarm is a software package for multi-agent simulation of complex adaptive systems. It was developed by the Santa Fe Institute. It has been used by researchers from a large number of disciplines, including ecology [Booth 1997] [Pepper & Smuts 1999], politics [Johnson 1998], biology [Kreft *et al.* 1998], economics [Luna & Perrone 2001] [Luna and Stefansson 2000] and manufacturing [Krothapalli & Deshmukh 1997] to implement a large variety of agent based models.

Swarm provides libraries to implement simulations of collections of concurrently interacting agents in a discrete-event simulation. Swarm provides the necessary machinery to effect those actions at the appropriate time. Along with the scheduling libraries, Swarm provides a number of other libraries of components for building models, controlling experiments on those models, and for displaying and analysing data generated by the experiments.

In recent years, there has been a great interest in the study of complexity. One approach to the study of complexity has been agent-based modelling using computer programs. This has been undertaken in a wide range of disciplines in which complex systems are encountered. The Santa Fe Institute is a research institute whose primary interests are in complexity theory. The motivation behind their development of Swarm was to provide a set of standard tools for undertaking computer simulations

of complex systems. Swarm programmers are thus able to concentrate primarily on the behaviour of their agents and the interactions between them, rather than implement complicated discrete-event machinery. This reduces duplication of programming effort, increases the quality of programs generated and allows for the publishing of models in a standard form allowing other researchers to reproduce results.

The forms of models that Swarm looks to support are multi-agent discrete-event simulations. The fundamental unit in such a simulation is an agent, which can execute events and can generate events that affect itself and other agents. Typically, a Swarm simulation will have a collection of many interacting agents. A discrete model of time is used rather than a continuous time model. Events take place at a single point in time.

At the core of a Swarm model is a ***swarm***. A swarm consists of a collection of agents and a schedule of actions to be executed by those agents. The swarm therefore constitutes a 'mini-simulation', complete with agents, a representation of time and the scheduling machinery to effect the actions on those agents.

An important feature of Swarm is the ability to produce hierarchical models. A swarm itself can be an agent and can thus be contained in a super-swarm. An example of this would be a model of a forest. At the top-level, the forest is modelled as a swarm of trees. Below this level, each tree could be modelled as a swarm containing a collection of the cells of that tree.

Object oriented (OO) languages naturally lend themselves to the agent based modelling implemented by Swarm. The Swarm libraries are written in Objective C. In OO programming 'classes' define the types of 'objects' that can be used. A class defines the behaviour of a type of object. It defines which instance variables are used to describe the state of an object of that type. It also describes the methods that can be executed by an object of that type. An object, then, is a particular instantiation of a class and has its own values for the state variables. In Swarm models, agents are

modelled as objects. Classes are thus used to define the generic behaviour of each type of agent.

Swarm models are typically written in the Objective C or Java programming languages. For this project, Objective C was used. In building a typical Swarm simulation, the agents must first be created. A class is built for each type of agent and then each agent is instantiated from the appropriate class. A swarm (often termed the 'model swarm') is then created in which the agents are placed. A schedule of actions is then needed to define which actions are to be executed by which agents at which point in time. Once this done the swarm can be executed.

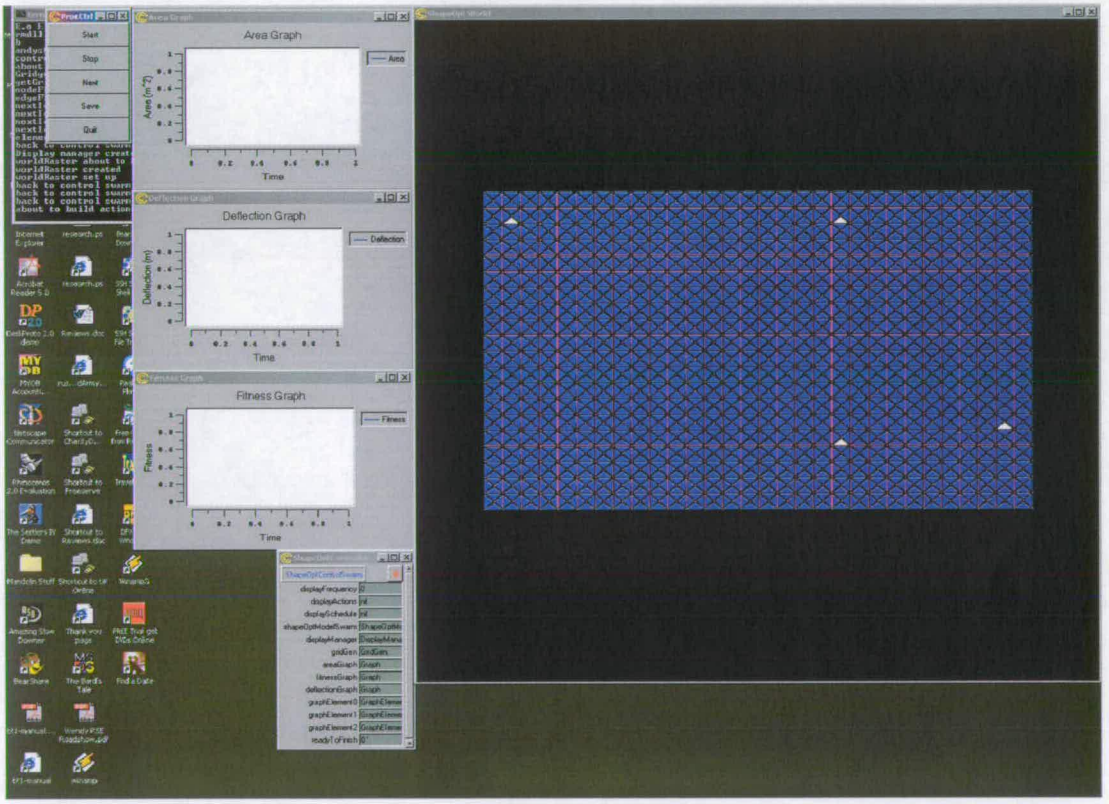


Figure 8-4 **The Swarm Application for Growth Stage at Start-Up**

The model swarm in essence acts as a self-contained simulated world. For this to be of use to the modeller it is necessary to monitor the model swarm in some way. Typically in a Swarm application, this is done by placing the models swarm in an observer swarm as a sub-swarm. Along with the model swarm, the observer swarm can contain various other agents who can acquire data from the model swarm via probes, store that information in a file or display it in graphs or other displays. Swarm provides objects for saving data, interfacing to statistical packages, displaying graphs and other graphical representations of the model.

Probes are an important feature of Swarm. When writing code in standard object oriented programming it is not necessary to be able to observe the inner state of an object, so long as the object behaves as required. For modelling purposes, though, this is very important. Swarm provides probes for this purpose. Probes can read or set any of the state variables of an object (agent) or call any of the methods of an object. As long as the object has been declared as a SwarmObject, then any object in Swarm is 'probable' without the need for any additional user code.

8.3.5 Details of Implementation

This section describes the implementation of the program as programmed in Swarm. Figure 8-5 shows the principal objects in the program. These objects are explained in further detail below.

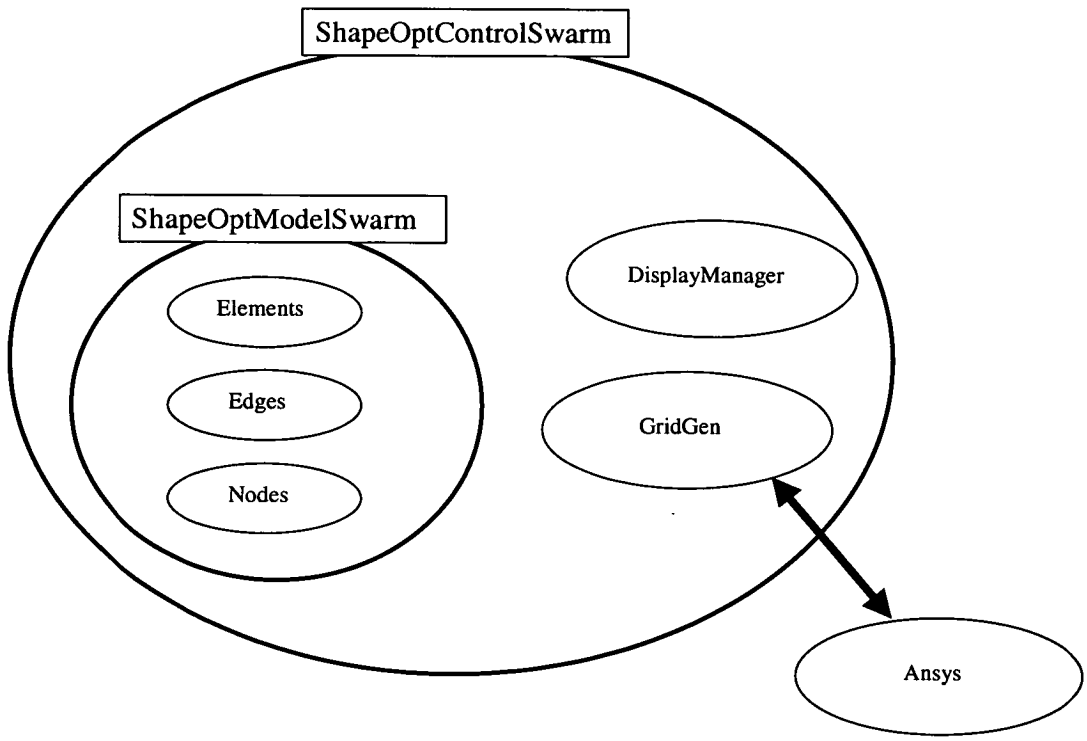


Figure 8-5 Overview of Program Structure

8.3.5.1 ShapeOptControlSwarm

The control swarm controls (*ShapeOptControlSwarm*) the operation of the program. It is responsible for creating the model swarm (*ShapeOptModelSwarm*). It owns an object *GridGen*, for generating the mesh, writing the mesh in the Ansys format and obtaining stress results from Ansys. It also owns *DisplayManager* that is responsible for displaying the current state of the model, along with graphs displaying fitness, area and deflection. The control swarm also manages the schedule of events for these objects.

8.3.5.2 ShapeOptModelSwarm

The model swarm owns the model of the component. For this application the shape representation is a cell complex. The model swarm therefore owns collections of the cells that form this complex and is responsible for the creation and destruction of these cells. The model swarm manages the schedule of events for the cells in this complex. The model swarm also keeps track on the current stress threshold below which the cells (elements) die.

8.3.5.3 Elements

All the elements in the mesh that form part of the shape representation are of this class. Elements are the primary 'agents' in this approach. An element maintains a list of the edges that form its faces. Below are given some of the principal methods implemented by the Element class, that are relevant to the basic execution flow given in Section 8.3.2 and illustrated in Figure 8-2.

Step

The *step* method on the Elements (2-cells) is the driving method for the growth of the shape.

Figure 8-6 shows the algorithm implemented by the Element *step* method. Firstly, *getCellStatus* is called to calculate the current status of an element (see below). If the element is not in a valid state then *kill* is called. Then *getStressThreshold* is called to determine the stress threshold for this element (see Section 8.3.6). Depending on the status of the element status and the stress on the element then a number of actions can take place. The three primary actions that an element can undertake are to *kill*, *killWithCheckForContact* or to *divideSelf*.

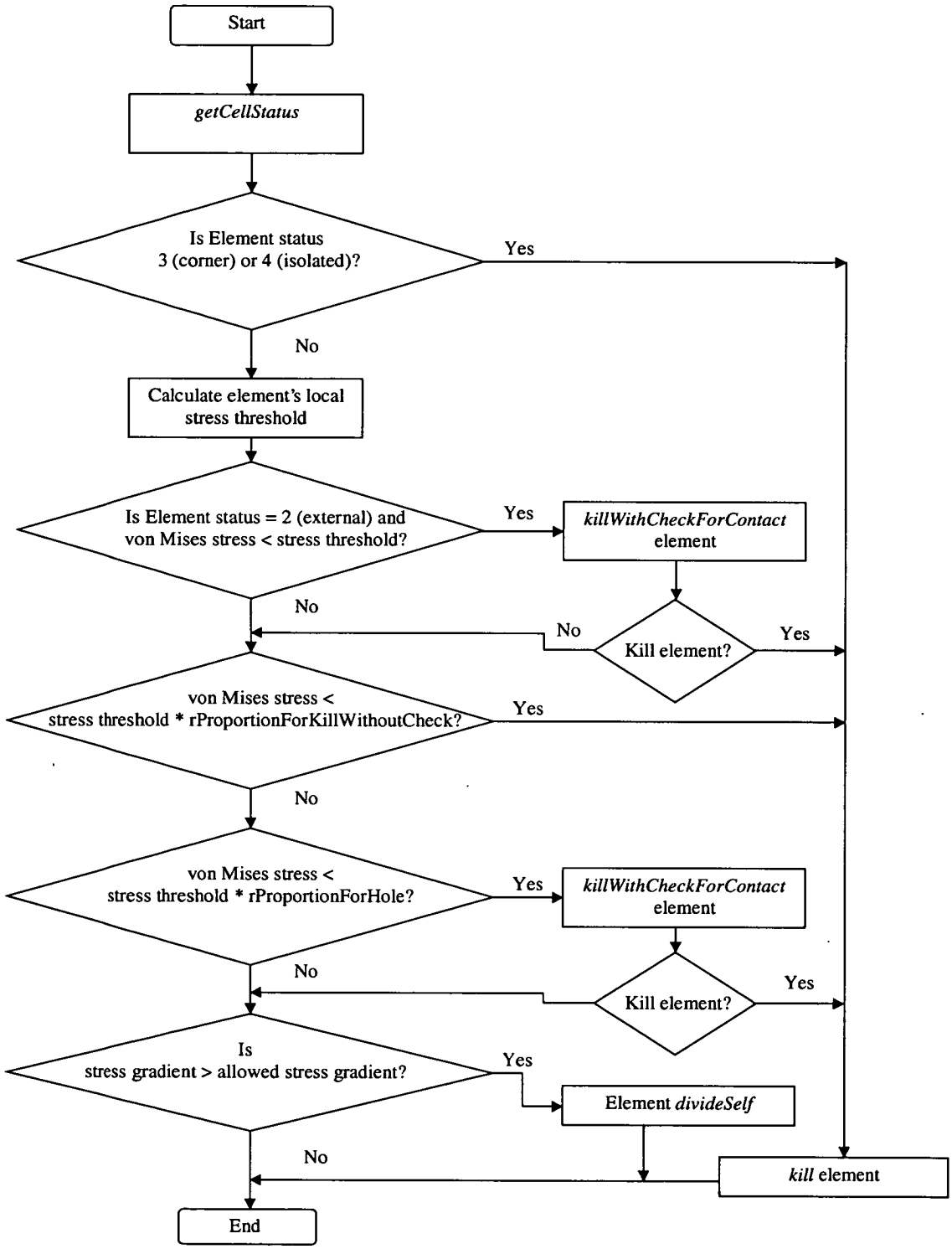


Figure 8-6 The Element *step* method

The basic logic is as follows. Firstly, elements execute the *kill* method when the stress on them drops below the threshold value. However, this should not happen if the death of the element would cause the topology of the shape to change (see Section 8.3.7). Therefore, before the *kill* method is executed, a check is made to establish whether this topology change takes place, using the *killWithCheckForContact* method. Secondly, it is necessary sometimes for an element to die which would cause the topology change, because that part of the shape is carrying so little load. When this is the case, the *kill* method can be called without the check being made. Thirdly, it is preferable that internal elements are less likely to die than external elements, so that the generation of holes does not become excessive. Internal elements, therefore, have a lower stress threshold than external elements. Again, a check is made to ensure that a change in topology would not be changed by the element's death. Finally, if no other action takes place, *divideSelf* is called if the stress gradient across the element is greater than some value (for all runs documented this value was 2).

The criteria for choosing the action that the *step* method calls are described below. The tests for these actions are done in the order given.

(a) If the element is external (status 1) and the stress of the element is less than the stress threshold then the action *killWithCheckForContact* is called.

(b) Else if the stress of the element is less than (*stressThreshold* * *rProportionForKillWithoutCheck*) then the cell is 'killed' without the check for contact being done. *rProportionForKillWithoutCheck* is variable under control of the genetic algorithm and typically takes a value of 0.01 to 0.6 and is usually less than the value of *rProportionForHole* described below. This allows elements to die that would change the topology if the stress on them is suitable low.

(c) Else if the stress of the element stress is less than (*stressThreshold* * *rProportionForHole*) then *killWithCheckForContact* is called. *rProportionForHole* is variable under control of the genetic algorithm and

typically takes a value of 0.05 to 0.8. *rProportionForHole* is less than 1 and so, in effect, the stress threshold for internal cells is lower than for external edges, ensuring that holes are only created where the stress is much lower than elsewhere.

(d) If the 'stress gradient' (determined by calling *getStressGradient*) is greater than *rSplitThreshold*, which typically takes the value of 2, then the *divideSelf* method is called. This is intended to increase the density of the mesh in those areas where the stress is changing rapidly and is a crude attempt at adaptive meshing (see Section 8.3.8).

getCellStatus

This method calculates the status of an element, which is determined as follows:

Status *0* if element is enclosed i.e. the element has no external edges in its face list. (Case A in Figure 8-7).

Status *1* if element is external i.e. the element has one external edge in its boundary list (Case B in Figure 8-7).

Status *2* if element is a 'corner' i.e. the element has two or more external edges in its boundary list (Case C in Figure 8-7).

Status *3* if element is isolated i.e. has three external edges. This will usually result in the element 'dying'.

Status *4* if element is ill-defined i.e. has fewer than three edges in its boundary list. This will usually result in the element dying.

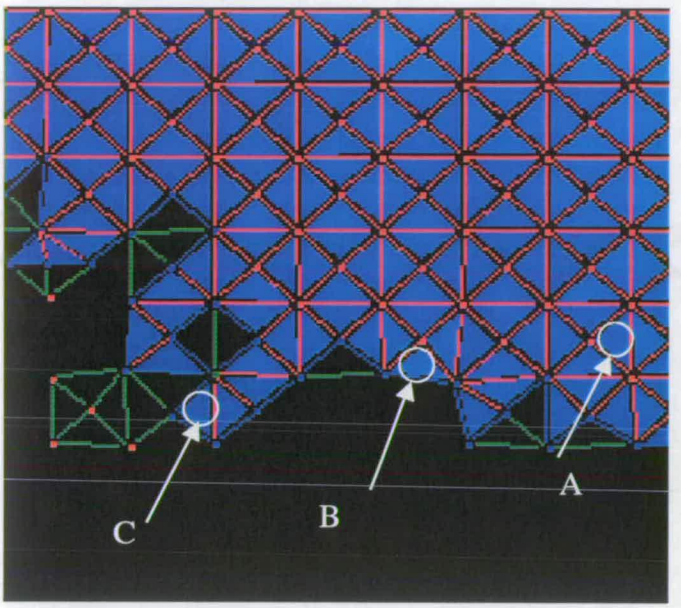


Figure 8-7 Illustration of Element Status

checkSelf

This method is used to undertake any action required to maintain the validity of an element. *getCellStatus* is called and if the element is a 'corner' (status 2), isolated (status 3) or ill-defined (status 4) then the *kill* method is called.

kill

This method begins the process of an element dying. This involves informing edges in the element's face list that the element is to be removed from their coface list, and informing the model swarm that this element is to be killed at the end of the time step.

killWithCheckForContact

This method checks whether this element dying would cause any node to change its status to 'contact'. If this does not happen then the *kill* method is called. See Section 8.3.7 for more details.

divideSelf

This method causes the current element to divide itself. This is done to increase the density of the mesh in this area. This is done by the addition of a node at the centroid of the current element and then undertaking a local Delaunay retriangulation. This is described in more detail in Section 8.3.8.

8.3.5.4 Edges

All the edges in the mesh, which form part of the shape representation, are of this class. An edge maintains a list of the nodes that form its faces and a list of the elements that form its cofaces. Below are listed are some of the methods that the Edge class implements:

getCellStatus

This method calculates the status of an edge, which is determined as follows:

Status *0* if edge is enclosed i.e. the edge has two elements in its coface list. (Case A in Figure 8-8 - these are the red edges on the display).

Status *1* if edge is external i.e. the edge has only one element in its coface list (Case B in Figure 8-8 - these are the blue edges on the display).

Status *2* if edge is isolated i.e. the edge has no elements in its coface list (Case C in Figure 8-8 - these are the green edges on the display).

Status *3* if edge is ill-defined i.e. has fewer than two nodes in its face list

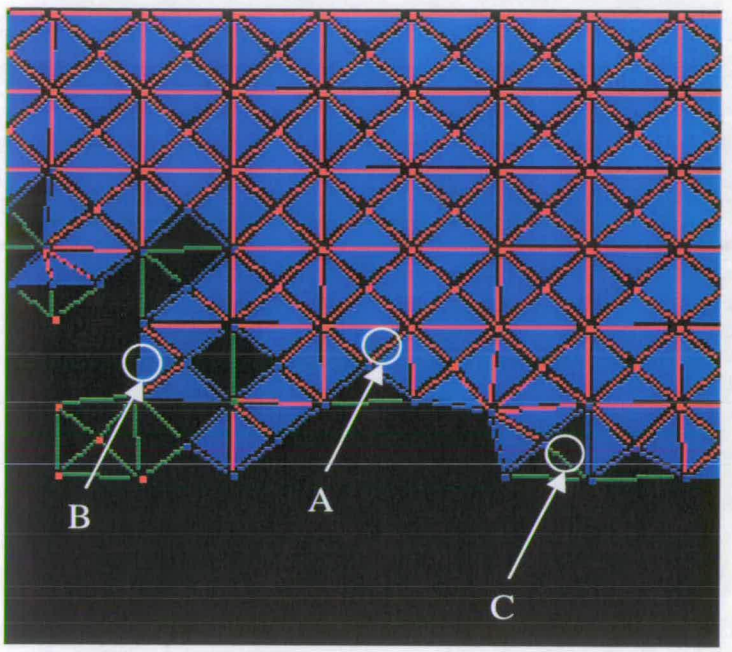


Figure 8-8 **Illustration of Edge Status**

checkSelf

This method is used to undertake any action required to maintain the validity of an edge. *getCellStatus* is called and if the edge is isolated (status 2) or ill-defined (status 3) then the *kill* method is called.

kill

This method begins the process of an edge dying. This involves informing elements in the coface list of the edge that the edge is to removed from their face list. The nodes in the face list of the edge are also informed that the edge is to removed from their coface list. The model swarm is then informed that this edge is to be killed at the end of the time step.

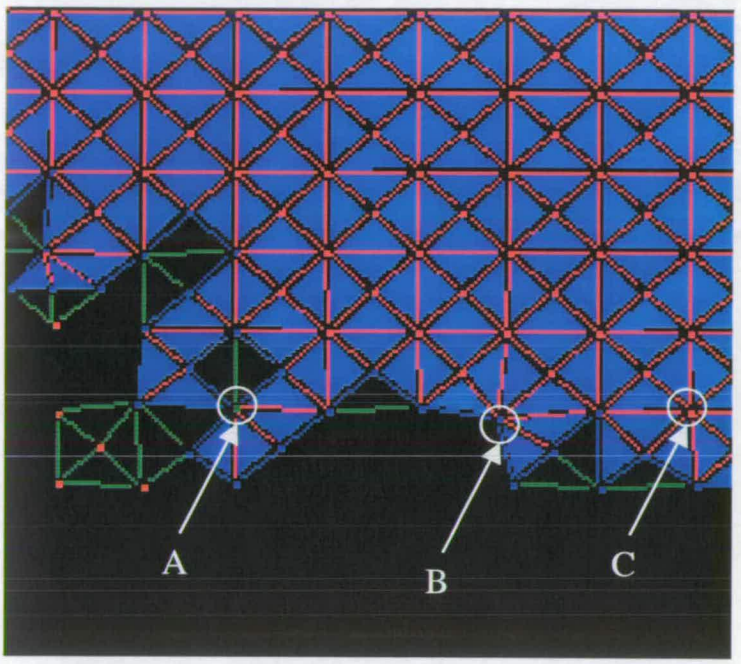


Figure 8-9 **Illustration of Node Status**

8.3.5.5 Nodes

All the nodes in the mesh that forms part of the shape representation are of this class. Nodes have chains for x position, y position and z position (for this two dimensional problem all z positions are set at 0). A node maintains a list of the edges that form its cofaces. Below are listed are some of the methods that the Node class implements:

getCellStatus

This method calculates the status of a node, which is determined as follows:

Status *0* if node is enclosed i.e. none of the edges in its coface list are external (Case C in Figure 8-9 - these are the red nodes on the display).

Status *1* if node is external i.e. two of the edges in its coface list are external (Case B in Figure 8-9 - these are the blue nodes on the display).

Status *2* if node is in 'contact' i.e. more than two of the edges in its coface are external (Case A in Figure 8-9 - these are the green nodes on the display). Since mesh topologies where a node is in 'contact' are not valid, a cell in this condition will normally die.

Status *3* if node is 'isolated' i.e. there are no edges in its coface list.

checkSelf

This method is used to undertake any action required to maintain the validity of a node. *getCellStatus* is called and if the node is in 'contact' (status 2) or 'isolated' (status 3) then the *kill* method is called.

kill

This method begins the process of a node dying. This involves informing edges in the coface list of the node that the node is to removed from their boundary list, and informing the model swarm that this node is to be killed at the end of the time step.

smooth

This method moves the node to the centroid of its adjacent nodes. It goes through each of the edges in the coface list of the node and from these edges 'gets' the other nodes which forms the face of these edges. The x and y position of the original node is then set at the centroid of these nodes.

boundarySmooth

The *boundarySmooth* method aims to produce a smooth outer boundary for the shape. Those cells which are not external (status 1) do nothing when this method is called. Otherwise, the method is similar to the *smooth* method, except only nodes attached to the two external edges are retrieved. It was also found that moving to the

centroid of these two nodes caused excessive smoothing and so the node was only moved part way towards this centroid i.e.

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \frac{1}{2}\left(\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}\right)$$

$$\begin{bmatrix} x_n' \\ y_n' \end{bmatrix} = (1-\alpha)\begin{bmatrix} x_n \\ y_n \end{bmatrix} + \alpha\begin{bmatrix} x_c \\ y_n \end{bmatrix}$$

where $\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ is the position of the first adjacent node

$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$ is the position of the second adjacent node

$\begin{bmatrix} x_c \\ y_c \end{bmatrix}$ is the position of the centroid of the two adjacent nodes

$\begin{bmatrix} x_n \\ y_n \end{bmatrix}$ is the original position of this node

$\begin{bmatrix} x_n' \\ y_n' \end{bmatrix}$ is the new position of this node

α is the proportion of the way to move the node from its original position to the centroid of the adjacent nodes. Experimentally a value of 1/40 for α was found to work well.

8.3.6 Calculating the Stress Threshold

In this approach to shape optimisation, the primary way in which the shape is generated is through the death of elements and the subsequent smoothing and splitting of those elements. Elements die when the stress on them drops below some threshold value. This threshold value depends on the current time, the status of the element (whether it is internal or external) and its position.

The global current stress threshold is calculated by the model swarm. Each time a new finite element analysis is undertaken, the current stress threshold is increased by a constant amount. This global stress threshold is reduced by each element according to its position. This is done by using four 'chemicals' deposited in the environment. The position, strength and spread of this chemical are under the control of the genetic algorithm. This gives the genetic algorithm the ability to change how the shape develops in different parts of the design area. This was a crude attempt to mimic chemotaxis in biological development [Gilbert 1994].

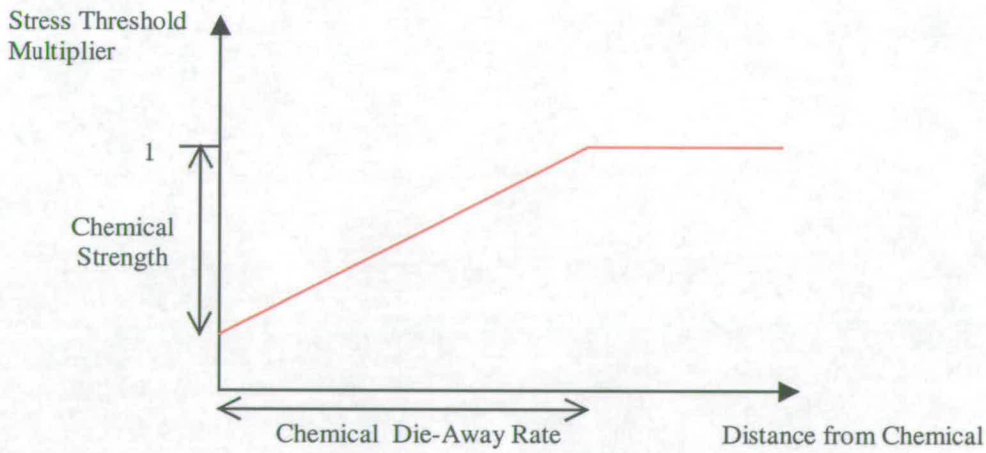


Figure 8-10 **Chemical's Effect on an Element's Stress Threshold as a Function of Distance from the Chemical**

Each element calculates the amount by which each chemical modulates its stress threshold as shown if Figure 8-10. Each chemical has a position, strength (between 0 and 1) and die away rate. The effect of the chemical is at a maximum at a distance of 0 from the chemical's position, with a stress threshold multiplier of (1 - chemical strength). This effect drops linearly from this maximum to zero at a distance of 'chemical die away rate'. An element's stress threshold multiplier for chemical 1 is therefore given by:

$$stressThresholdMultiplier_1 =$$

$$\begin{cases} (1 - chemStrength_1) + \dfrac{chemStrength_1 * distanceFromChemical_1}{chemDieAwayRate_1} & distanceFromChemical_1 < chemDieAwayRate_1 \\ 1 & otherwise \end{cases}$$

There are four such chemicals and the total *stressThresholdMultiplier* for each element is found by summing the four individual *stressThresholdMultipliers*, with a minimum of 0.

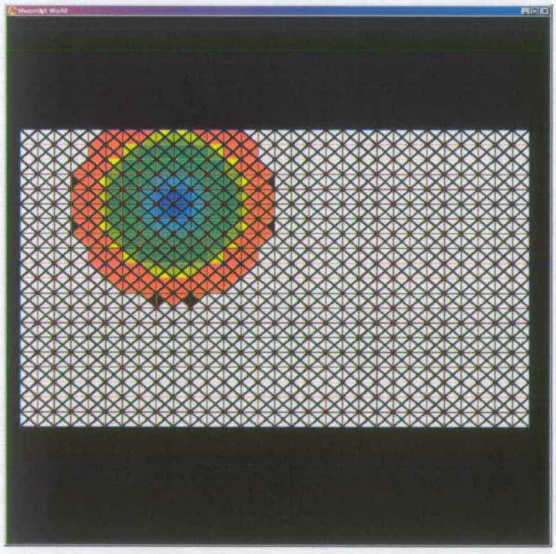


Figure 8-11 **Display of *stressThresholdMultiplier* with One Chemical Placed at (0.3,0.45), strength 1, rate 0.2**

8.3.7 Maintaining Similar Topology

This approach works by repeatedly increasing the stress threshold, calculating stresses, killing appropriate elements, increasing the stress threshold, calculating stresses, killing appropriate elements and so on. This continues until the area of the shape drops below some target value. Ideally, the stress threshold should only increase very slowly so that only a few elements die at any step, so that this process approximates a continuous dying of cells with a constant load applied. However, this would necessitate very many finite element analyses to be done with a large computational cost. The stress threshold therefore increases in fairly large discrete steps.

Early in the development of this approach, a problem was discovered. Often it was found that early in the run there would be generated a 'wide' load bearing section of the shape. Because the section is wide, the stress on it would be relatively low. Consequently, cells in this section would fairly soon drop below the stress threshold and die. Because the stress threshold increases in fairly large discrete steps, frequently the whole load bearing section would die even though it was usefully carrying a load and merely needed to be thinner.

To overcome this, in the normal round of cell deaths a check would be made to ensure that the death of a cell would not cause a 'contact' node (see Section 8.3.5.3) which would indicate the breakage of a load bearing section. Thus the section could become thinner, but would not be allowed to break.

Clearly, there needed to still be some mechanism for 'breaking' sections of the shape that were not contributing to the performance of the component. Therefore, if the stress on an element is below a threshold, typically between 5% to 50% of the normal stress threshold, then the cell can be killed without this check. The exact value of this threshold is variable and is given by multiplying the cells normal stress threshold by *rPropKillWithoutCheck* (a variable under control of the genetic algorithm).

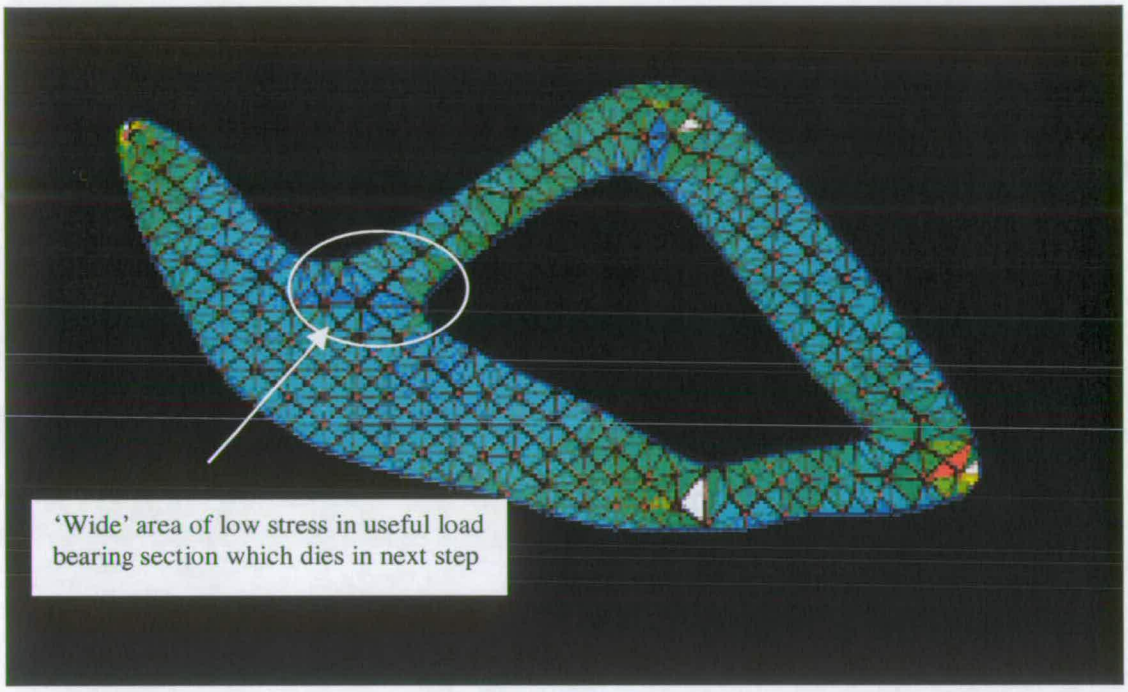


Figure 8-12 **Example of breakage of useful load bearing section**

8.3.8 Increasing Mesh Density

There is no separation between the shape representation and the finite element mesh in this approach. In order to ensure reasonable accuracy from the finite element analyses it was decided that some form of adaptive meshing was required, even though this would not change the boundary geometry of the shape generated.

Where the stress gradient over an element is above some threshold value an element triggers the *divideSelf* method. This method implements a node insertion and local Delaunay retriangulation based on the algorithm described by [Watson 1981] and [Filipiak 1996].

Delaunay triangulation is used widely for the generation of unstructured meshes for the finite element method and has been studied extensively in the literature [Schewchuk 1997]. Given a set of points on a plane, the Voronoi tessellation is formed from the set of Voronoi polygons. These polygons are the regions around each point, which are closer to that point than to any of the other points. This is shown in Figure 8-13. The Delaunay triangulation is the dual of the Voronoi tessellation.

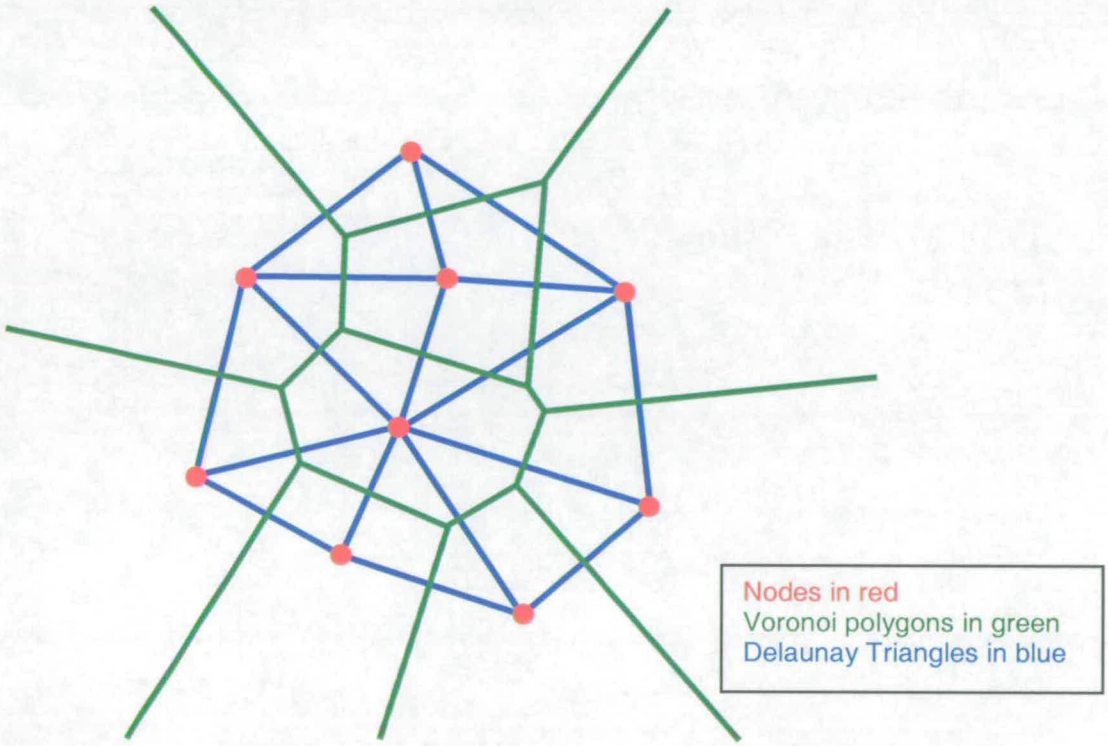


Figure 8-13 Example of Delaunay Triangulation

Delaunay triangulations have some useful properties:

- None of the points (nodes) are contained in the circumcircle of any triangle (the circumcircle of a triangle is the circle passing through all three of its vertices).
- In two dimensions only, given a set of points to triangulate, the Delaunay triangulation is the triangulation that maximises the minimum angle for all triangular elements. This is a very useful property for generating high quality finite elements.

The most widely used algorithm for Delaunay triangulation is the Bowyer-Watson algorithm [Watson 1981]. This method starts with an initial simple Delaunay triangulation (often a single triangle) and successively adds new points into this triangulation.

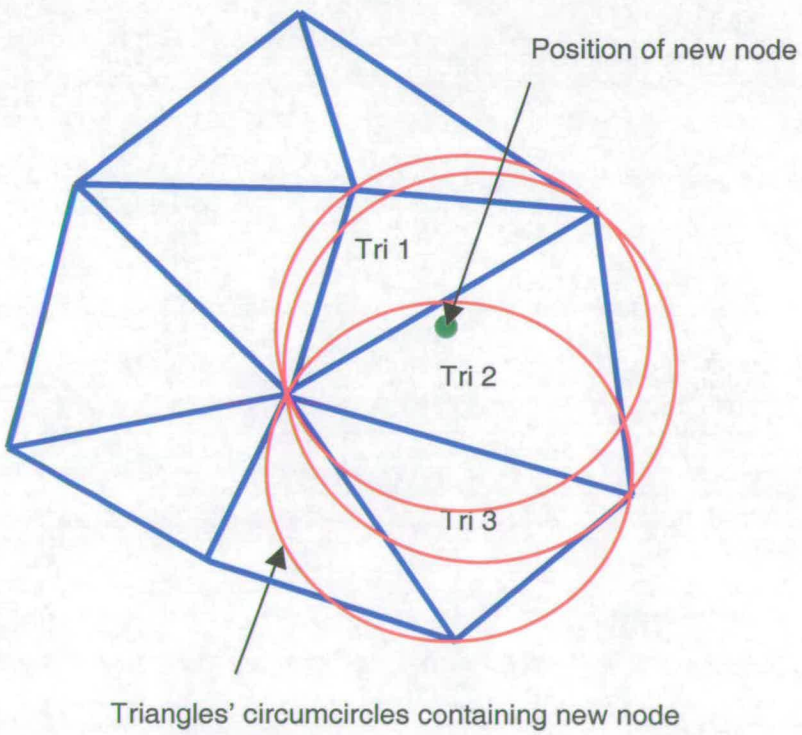


Figure 8-14 Addition of New Node and Calculation of Circumcircles

The algorithm repeats the following steps:

(a) Add a new point

(b) Find the existing triangles whose circumcircle contains the new point (see Figure 8-14).

(c) Delete these triangles, creating a convex cavity.

(d) Join the new point to all the vertices on the boundary of the cavity (Figure 8-15) and create appropriate elements.

It was decided that the element *divideSelf* method should implement this algorithm. It should be noted that there is no 'top-down' control in the implementation of this algorithm. The dividing element creates the new node, then it needs to determine the

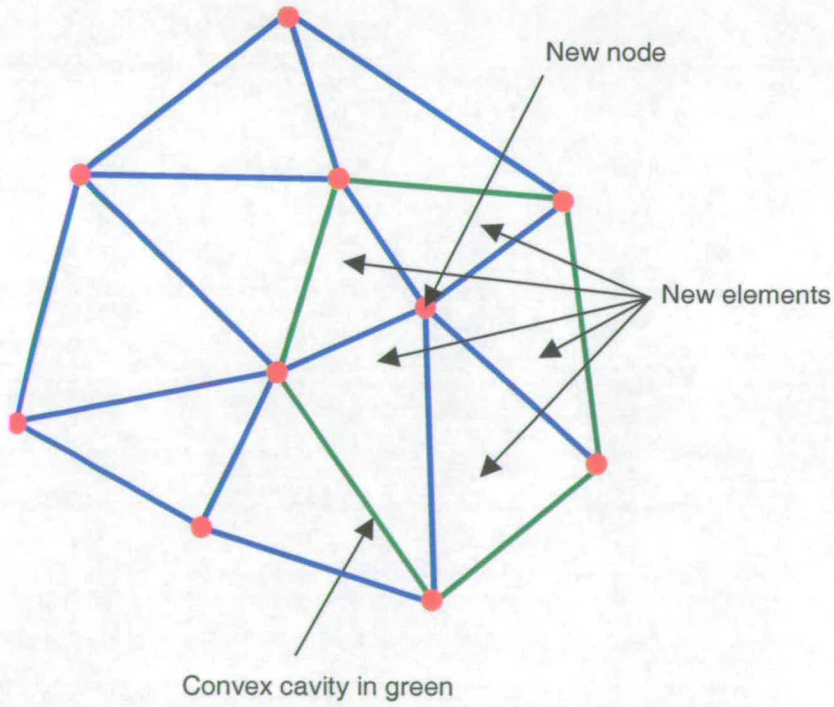


Figure 8-15 **New Elements Formed from Convex Cavity**

edges that constitute the convex cavity that the insertion of this node should cause, kill the elements and edges within this cavity and then kill itself. This takes place by calling the methods described below, which 'spread' out from the original element through the edges to adjacent elements. The methods needed to do this are in both the Element and Edge classes. These are listed below:

8.3.8.1 Explanation of Element Method *divideSelf*

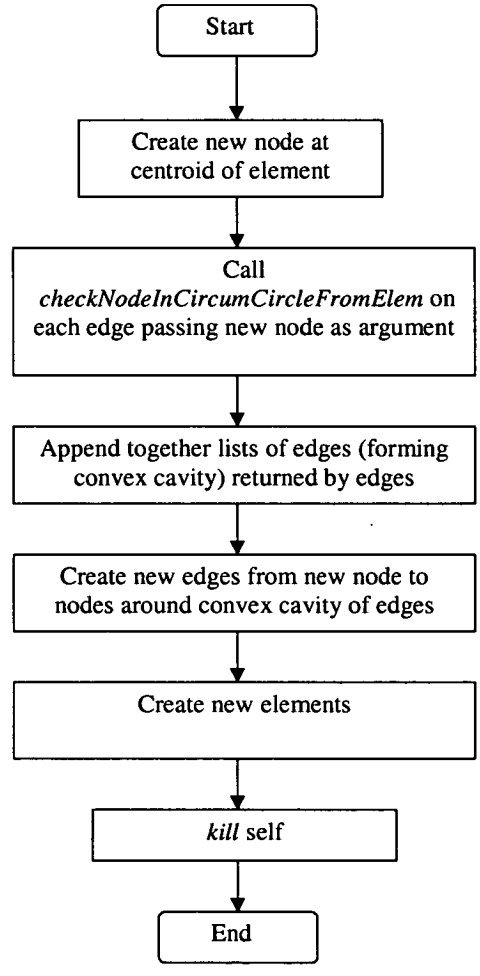


Figure 8-16 **The Element *divideSelf* method**

This method (see Figure 8-16) creates a new node at the centroid of the element. It then calls *checkNodeInCircumCircleFromElem* (described below) on each of the edges in its face list. The lists of edges returned by these three calls are then appended together to form a list of edges which make up the convex cavity in the mesh. New edges can be then formed by traversing around the list of edges so formed creating new edges between the new node and the nodes on the edges in the convex cavity. New elements can then be formed using these edges. The element then calls *kill* on itself.

8.3.8.2 Explanation of Edge Method *checkNodeInCircumcircleForElem: node from: element*

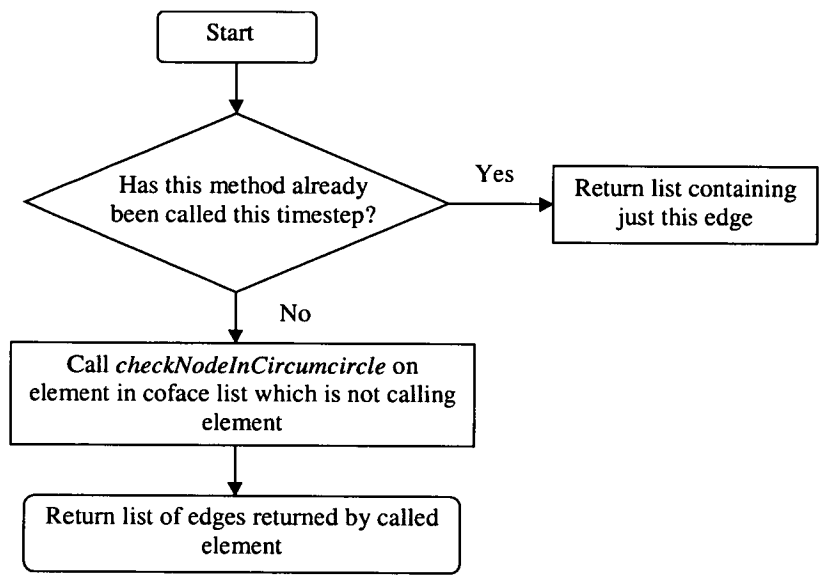


Figure 8-17 The Edge *checkNodeInCircumcircleForElem* method

This method (see Figure 8-17) is called by an element. It takes the new node and the calling element as arguments and returns a list of edges that should lie on the convex cavity. If the *checkNodeInCircumcircleForElem* has already been called for this edge then a list containing only this edge is returned. Otherwise, *checkNodeInCircumcircle* is called on the element in the coface list that is not the calling element. The list of edges (which are on the convex cavity) returned by this element are then returned by the edge.

8.3.8.3 Explanation of Element Method *checkNodeInCircumcircle: node from: edge*

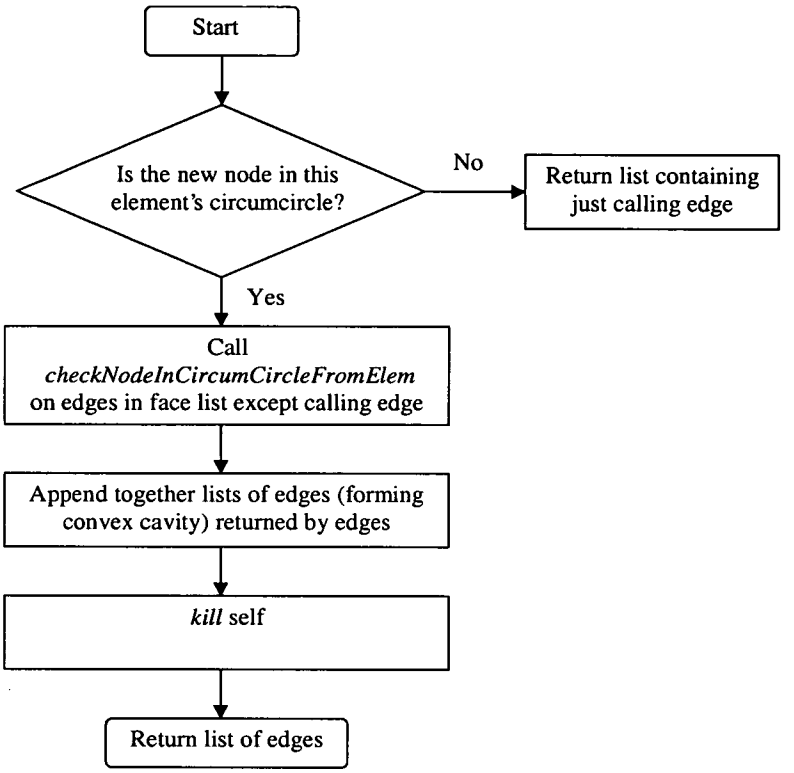


Figure 8-18 The Element *checkNodeInCircumcircle* method

This method is called by an edge. It takes the new node and the calling edge as arguments. If the node is not within the circumcircle of the element then a list containing only the calling edge is returned. If the node is within the circumcircle then *checkNodeInCircumCircleFromElem* is called on the two edges in the boundary list of the element other than the calling edge. The element calls *kill* on itself. The lists of edges returned by the two calls to *checkNodeInCircumCircleFromElem* on the edges are then appended together and returned.

8.3.8.4 Worked Example of Element Division

This section attempts to explain how element division works, by undertaking a worked example. Figure 8-19 shows a partial mesh before element 1 has divided. Elements are labelled 1 to 6 and edges *a* to *f*. Figure 8-20 shows the convex cavity formed by the division of element 1. Figure 8-21 shows the mesh following the completion of the division of element 1.

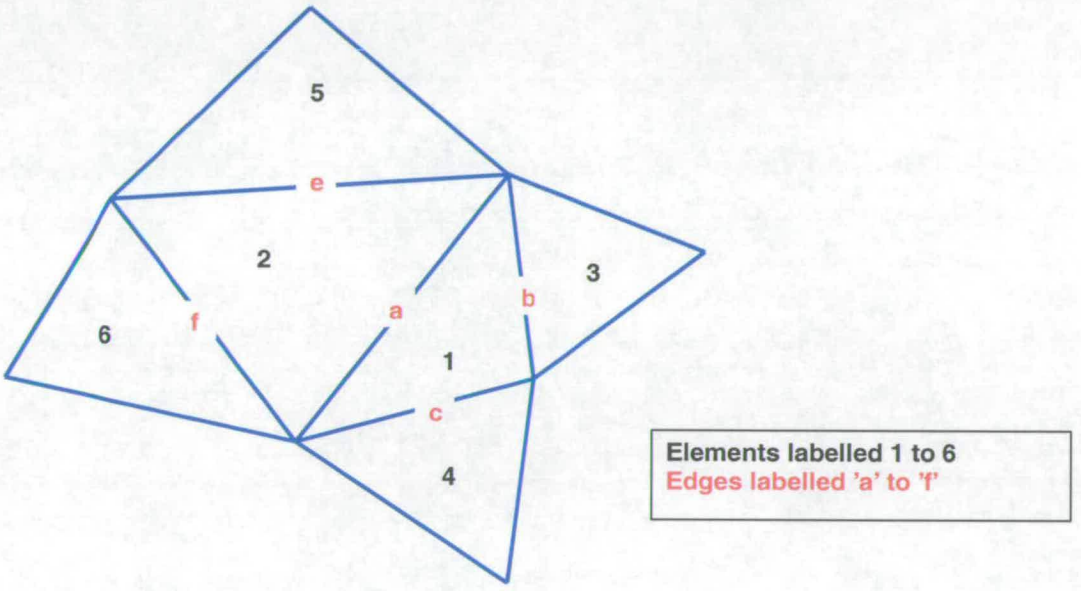


Figure 8-19 **Partial Mesh before Element Division**

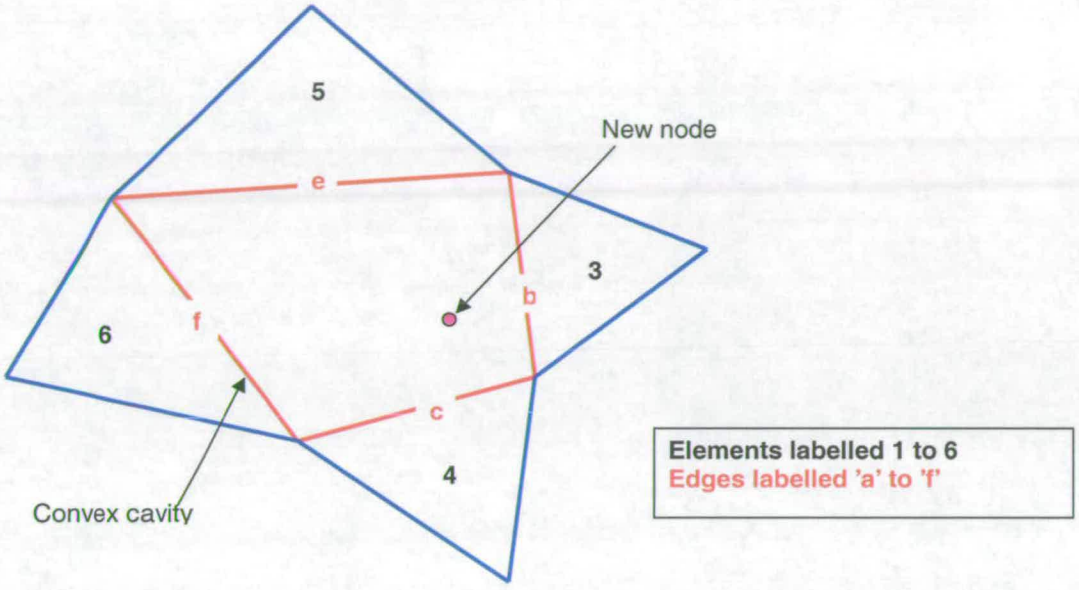


Figure 8-20 **Convex Cavity formed by Division of Element 1**

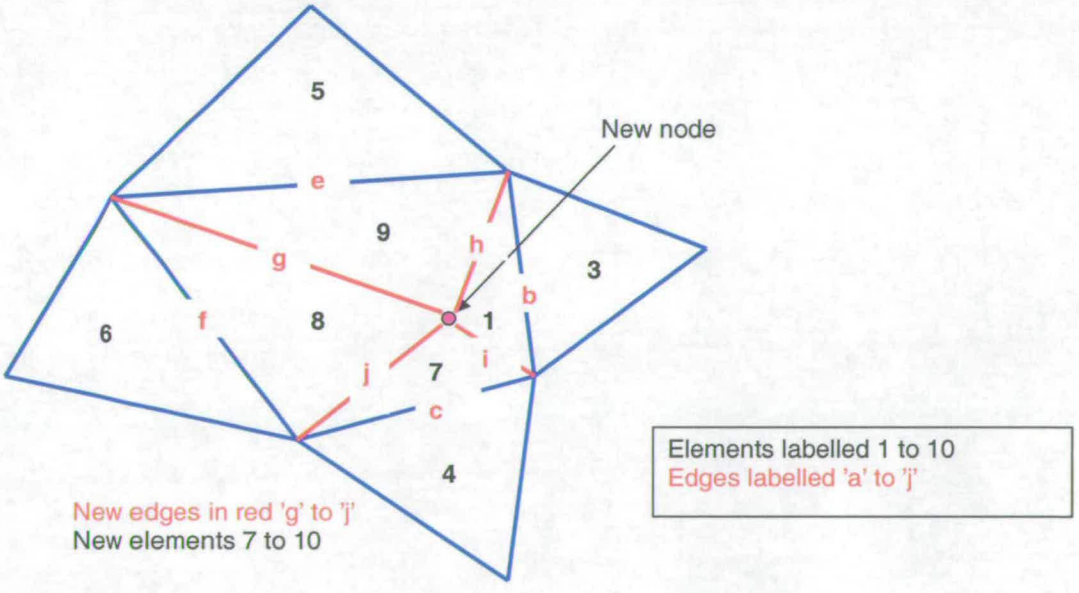


Figure 8-21 **Mesh following Division of Element 1**

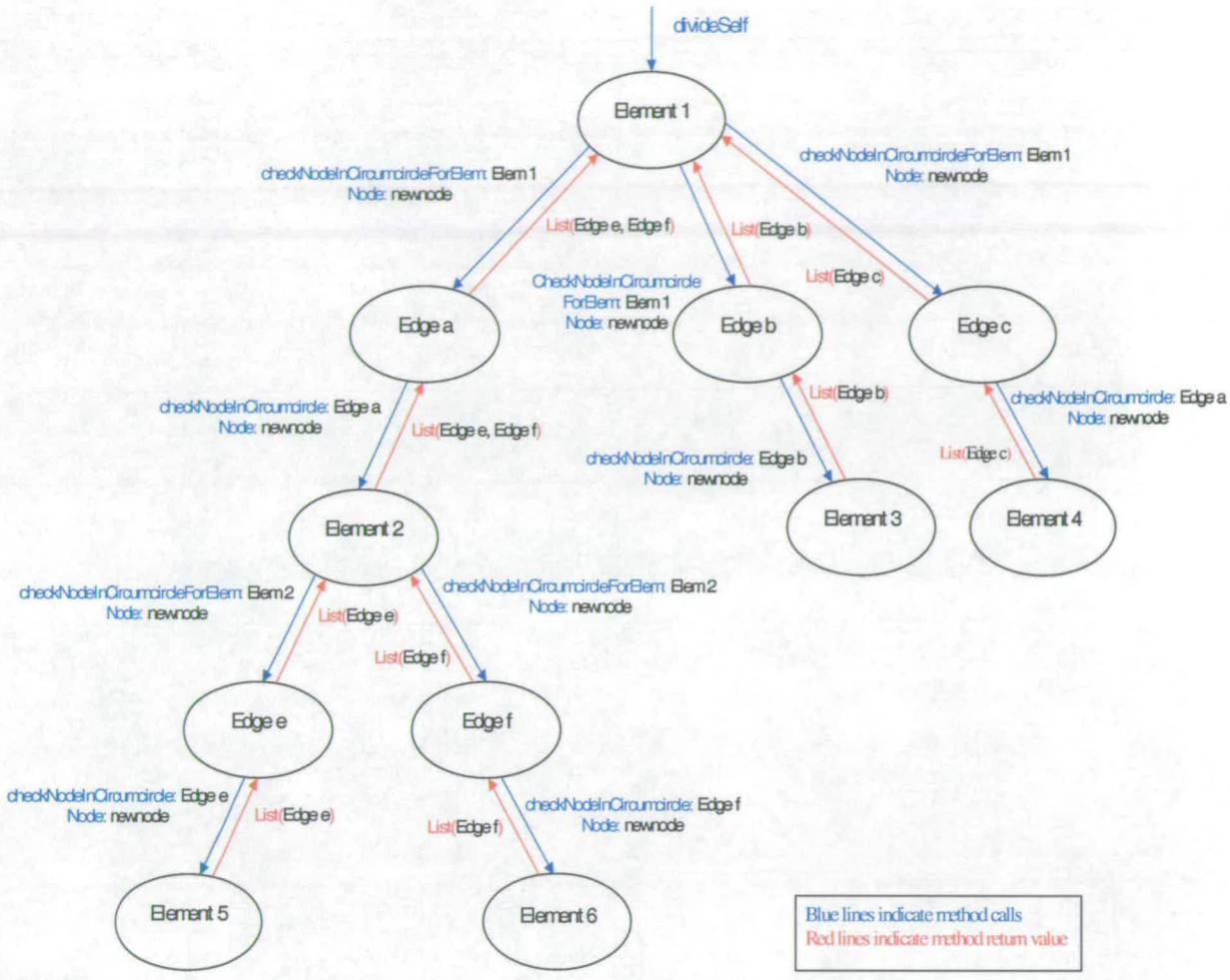


Figure 8-22 Method Calls and Return Values for Division of Element 1

Figure 8-22 shows the path of method calls following a call to Element 1 to *divideSelf*. Element 1 firstly creates a node, newnode at its centroid. The method *checkNodeInCircumcircleForElem* is called on Edges *a*, *b* and *c*, passing newnode and Element 1 as arguments. These edges call the method *checkNodeInCircumcircle* on Elements 2, 3 and 4 respectively.

For Elements 3 and 4, the new node does not lie within their circumcircle. No further action is therefore required and so Element 3 returns a list containing just Edge *b* back to Edge *b*, whilst Element 4 returns a list containing just Edge *c* back to Edge *c*. Edge *b* and Edge *c* return these lists to Element 1.

In the case of Element 2, the new node does lie within its circumcircle. Element 2 therefore executes its *kill* method. Element 2 also needs to establish whether any further elements need to be removed and to find those edges that will form the resulting convex cavity. Therefore, *checkNodeInCircumcircleForElem* is called on Edges *e* and *f*, passing newnode and Element 2 as arguments. These edges call the method *checkNodeInCircumcircle* on Elements 5 and 6 respectively.

For Elements 5 and 6, the new node does not lie within their circumcircle. No further action is therefore required and so Element 5 returns a list containing just Edge *e* back to Edge *e*, whilst Element 6 returns a list containing just Edge *f* back to Edge *f*. Edge *e* and Edge *f* return these lists to Element 2.

Element 2 appends these two lists together, to form a list containing both Edge *e* and Edge *f*. This list is returned to Edge *a*, which in turn returns it to Element 1.

Element 1 appends the lists returned from Edge *a*, Edge *b* and Edge *c* together, to form a list containing Edges *e*, *f*, *b* and *c*. This list now contains those edges that bound the convex cavity. New edges *g*, *h*, *i* and *j* can be then formed by traversing around the list of edges creating new edges between the new node and the nodes on the edges in the convex cavity. New elements 7 to 10 can then be formed using these edges. Element 1 then calls *kill* on itself.

8.3.9 The Genetic Algorithm

The genetic algorithm was written in Matlab. The vast majority of computing effort required for this approach was in the growth and finite element analysis of the mesh. The relative slowness of Matlab as an interpreted programming language was therefore not a problem. Each individual in the population has an array of real numbers as its chromosome. Each of these real numbers corresponds to a variable used in the shape growth simulation. In order to evaluate an individual, Matlab runs the Swarm model in batch mode, with the variables passed on the command line. The best fitness that the shape achieves is returned by Swarm. The following variables were manipulated by the GA:

propHole	*propKillWithoutCheck*	*rChemSens*	
rChem0Xpos	*rChem0Ypos*	*rChem0Str*	*rChem0Rate*
rChem1Xpos	*rChem1Ypos*	*rChem1Str*	*rChem1Rate*
rChem2Xpos	*rChem2Ypos*	*rChem2Str*	*rChem2Rate*
rChem3XPos	*rChem3Ypos*	*rChem3Str*	*rChem3Rate*

propHole determines the proportion of the global stress threshold below which an internal element will die. This can take any value from 0 to 1. Higher values for this variable result in more internal holes being produced in the shape.

propKillWithoutCheck determines the proportion of the global stress threshold below which an element will die without checking whether this would cause a change in the topology. This can take any value from 0 to 1. Lower values for this variable result in a shape with more trusses.

There are four chemicals that can be used to influence the generation of the shape as described in Section 8.3.6. Each of these is determined with four variables for *x-position* (i.e. *rChem0XPos*), *y-position* (i.e. *rChem0Ypos*), *strength* (i.e. *rChem0Str*) which takes a value from 0 to 1 and *dispersion rate* (i.e. *rChem0Rate*). There is also a variable *rChemSens* that varies the sensitivity of the shape to the chemicals.

The genetic algorithm was run with a rank based selection scheme. A fixed number of new individuals were generated per generation. Normalised Geometric Ranking [Joines & Houck 1994] was used. In this scheme the probability of selection is given by:

$$P[\text{Selecting the } i^{\text{th}} \text{ individual}] = q' \,.\, (1 - q)^{r-1}$$

where:

q is the probability of selecting the best individual

r the rank of the i^{th} individual where 1 is the best

P is the population size

q' $q / (1-q)^p$

Simple single point crossover was used. Non-uniform mutation [Michalewicz 1992] [Michalewicz & Schoenauer 1996] [Michalewicz & Fogel 2000] was applied at a rate of about one per individual. This applied a random disturbance to the value of a gene based on a uniform distribution. The width of this distribution was narrowed through the generations. The mutation works as follows:

- A gene, x_i, is chosen for randomly for mutation.
- $$x_i \rightarrow \begin{cases} x_i + \Delta(t, u(i) - x_i), & \text{if random binary number} = 0 \\ x_i + \Delta(t, x_i - l(i)), & \text{if random binary number} = 1 \end{cases}$$

  where t is the current generation

$u(i)$ is the upper bound for gene i

$l(i)$ is the lower bound for gene i

[Janikow & Michalewicz 1991] used the following function for Δ:

$$\Delta(t, y) = y \cdot r \cdot \left(1 - \frac{t}{T}\right)^b$$

where r is a random number from 0 to 1

t is the current generation

T is the maximal generation

b is a parameter determining the degree of non-uniformity

For the experiments shown in the results, b was set at 1.

8.4 Problems Addressed

Three problems were addressed. These were a frame for a bicycle, a cantilever beam and a load-bearing arch. The loadings and dimensions for these problems are shown in Figure 8-23, Figure 8-24 and Figure 8-25. The material was AISI 1020 steel with a Young's modulus of 200 GPa and a yield stress of 350 MPa. For each of the problems, an area constraint was chosen so that the stress on the components would be approximately 100 MPa. For the bicycle frames and arches this corresponded to an area of 0.07 m^2 and for the cantilever 0.14 m^2.

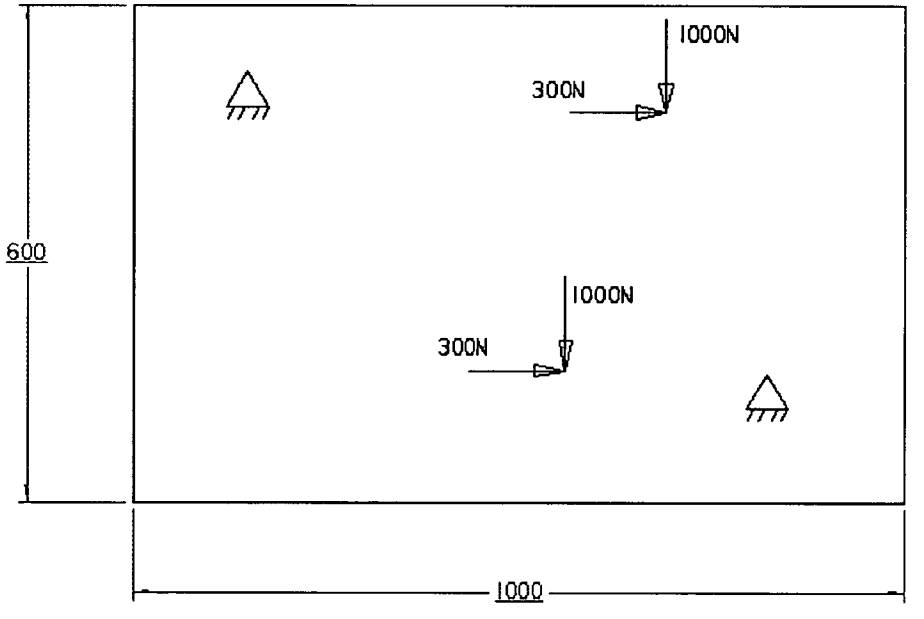


Figure 8-23 Load Case for Bicycle Frame

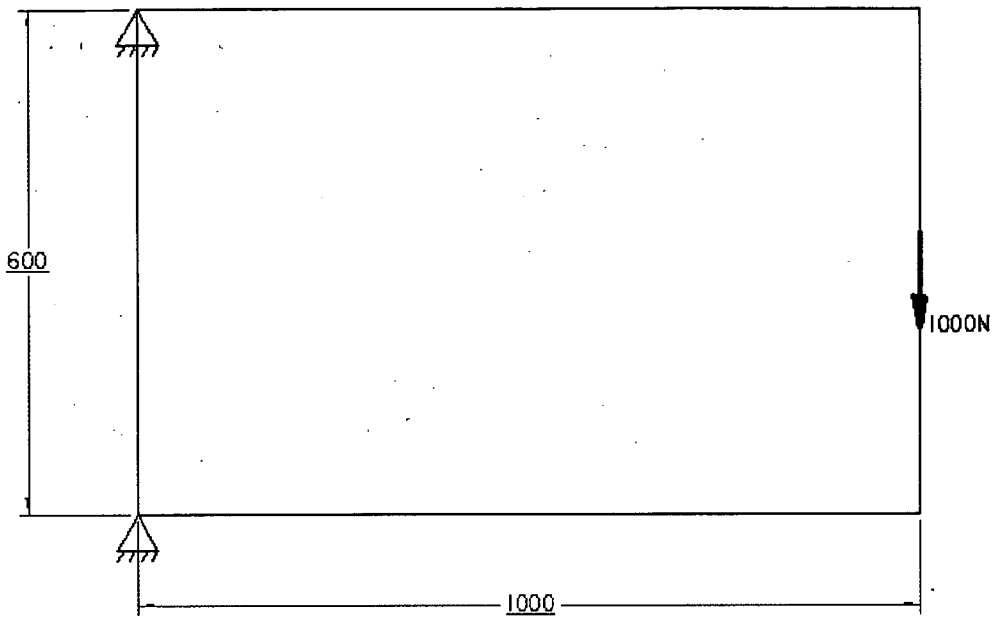


Figure 8-24 Load Case for Cantilever

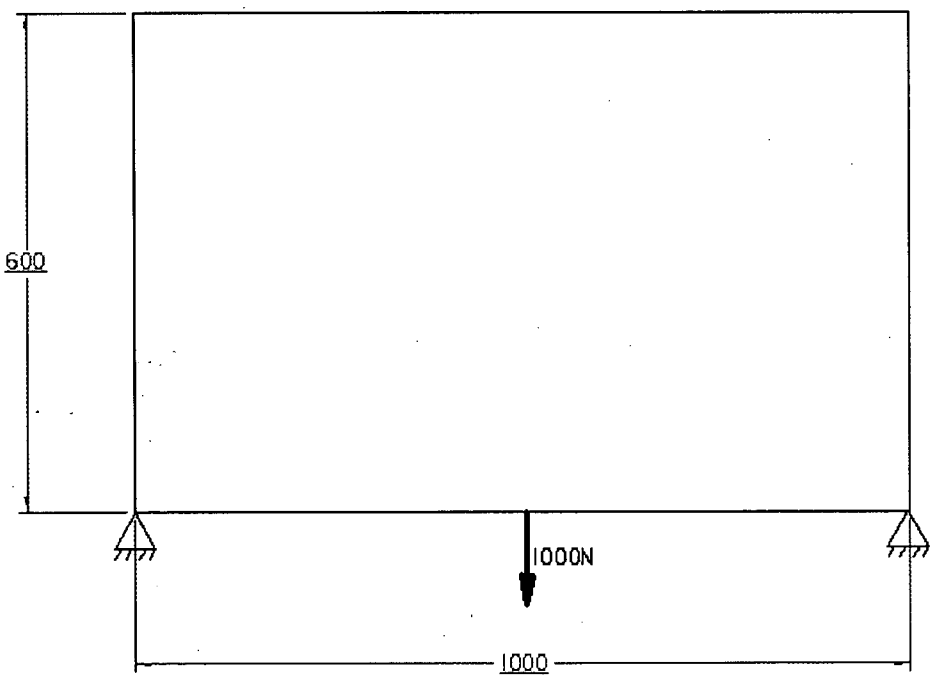


Figure 8-25 Load Case for Arch

8.5 Results

Various optimisation runs were undertaken with various genetic algorithm and growth stage parameters for each of the load cases. Many of these runs were undertaken during the simultaneous development of the both the approach and the software. Because of the time taken to do a run, it was not possible to undertake a systematic study on the best parameters to use. Therefore, in this section just one run is documented for each of the load cases. The aim of the work in this chapter (Section 8.2.6) was to determine whether a morphogenetic evolutionary algorithm shows any promise as an effective approach to shape optimisation. Therefore, the runs chosen were those which were indicative of the current status of the software.

Some of the members of the initial population are shown, demonstrating the diversity of shapes which the morphogenetic process can generate. The best individual generated following optimisation by the genetic algorithm is then shown.

8.5.1 Bicycle Frame

The load case for the bicycle frame was given in Section 8.4. The genetic algorithm was run with the following settings.

Fitness	- deflection	
Constraint	Area < 0.07 m^2	
Population size	20	
Number of new individuals per generation	5	
Number of mutations per generation	5	
Number of generations	20	
Parameter	**Min**	**Max**
propHole	0.01	0.8
propKillWithoutCheck	0.05	0.5
rChemSens	0	0.3
rChem0XPos	0	1
rChem0YPos	0	0.6
rChem0Str	0	1
rChem0Rate	0	0.6
rChem1XPos	0	1
rChem1YPos	0	0.6
rChem1Str	0	1
rChem1Rate	0	0.6
rChem2XPos	0	1
rChem2YPos	0	0.6
rChem2Str	0	1
rChem2Rate	0	0.6
rChem3XPos	0	1
rChem3YPos	0	0.6
rChem3Str	0	1
rChem3Rate	0	0.6

8.5.1.1 Initial Population

Below are shown some of the members of the initial population.

Bicycle Frame A

Fitness	-0.000928
Max Deflection (mm)	0.928
Parameter	**Value**
propHole	0.1416
propKillWithoutCheck	0.3715
rChemSens	0.2901
rChem0Xpos	0.3481
rChem0Ypos	0.5953
rChem0Str	0.9292
rChem0Rate	0.2674
rChem1Xpos	0.5858
rChem1Ypos	0.5986
rChem1Str	0.6996
rChem1Rate	0.3802
rChem2Xpos	0.5031
rChem2Ypos	0.0822
rChem2Str	0.0205
rChem2Rate	0.0872
rChem3Xpos	0.0213
rChem3Ypos	0.3946
rChem3Str	0.9576
rChem3Rate	0.2516

Figure 8–26 shows snapshots of the growth of bicycle frame A from the initial population. As can be seen the growth stage generates reasonable looking shapes even with parameters set at random. This bicycle frame had a low value for *propHole* and so few holes were generated through the development. *propKillWithoutCheck* took a moderate value and so it was reasonably easy for the topology to change through the development. The sensitivity to the chemicals is quite high and so the shape has been somewhat influenced by the chemicals.

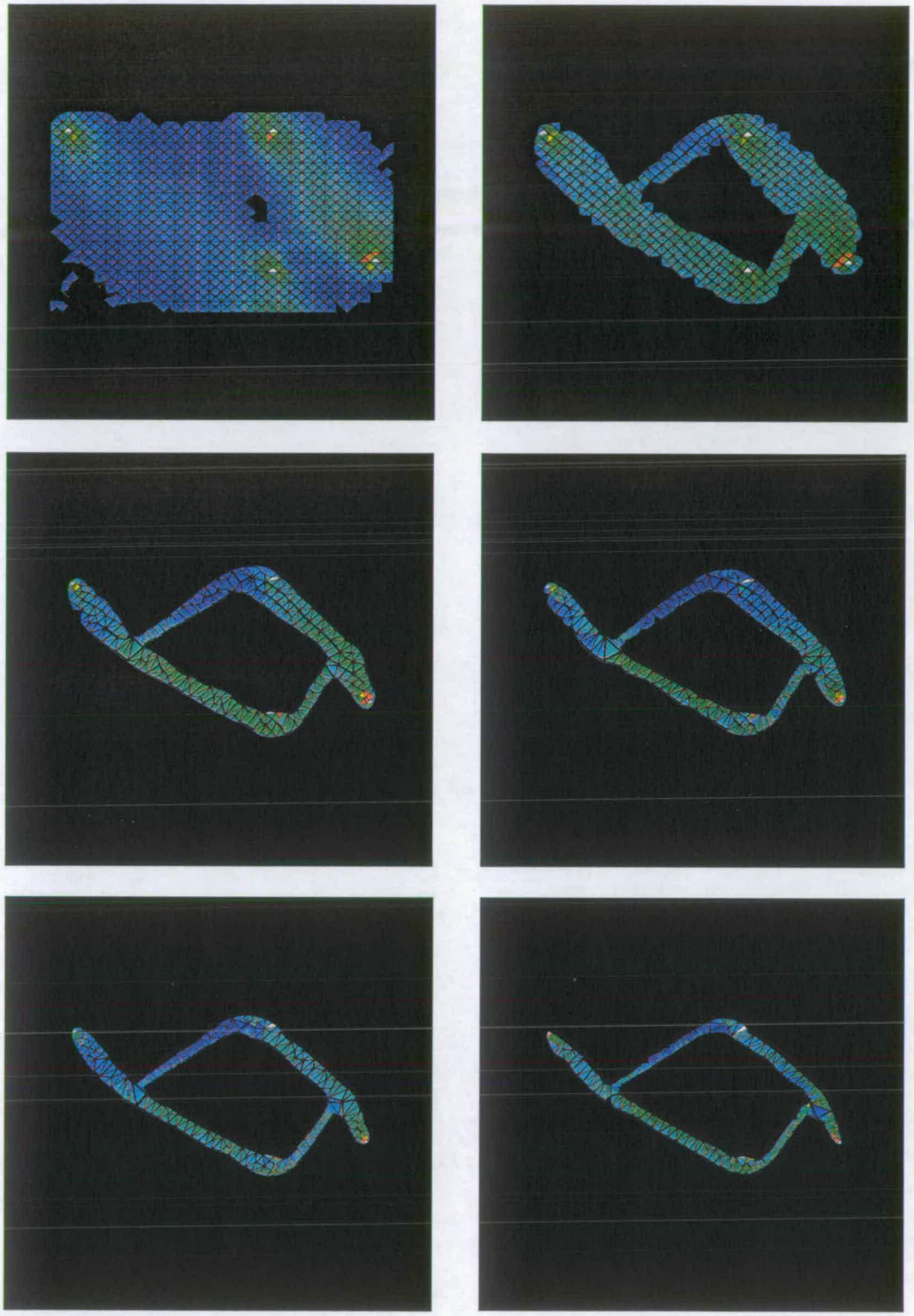


Figure 8–26 **Snapshots of Growth of Bicycle Frame A**

Bicycle Frame B

Fitness	-0.000634
Max Deflection (mm)	0.634
Parameter	**Value**
propHole	0.7577
propKillWithoutCheck	0.3515
rChemSens	0.0626
rChem0Xpos	0.9974
rChem0Ypos	0.5483
rChem0Str	0.5631
rChem0Rate	0.5343
rChem1Xpos	0.2918
rChem1Ypos	0.2094
rChem1Str	0.1678
rChem1Rate	0.2167
rChem2Xpos	0.4135
rChem2Ypos	0.3720
rChem2Str	0.5672
rChem2Rate	0.4138
rChem3Xpos	0.6431
rChem3Ypos	0.4486
rChem3Str	0.8057
rChem3Rate	0.3307

Figure 8-27 shows snapshots of the growth of bicycle frame B from the initial population. This bicycle had a high value for *propHole* and so many holes were generated through the development. p*ropKillWithoutCheck* took a moderate value and so it was reasonably easy for the topology to change through the development. Therefore, sections formed by the formation of the holes were easily broken. The sensitivity to the chemicals is low and so they have little effect. As can be seen, the shape has a considerably higher section from the seat post to head-set than bicycle frame A. The bottom bracket is also raised on a truss platform.

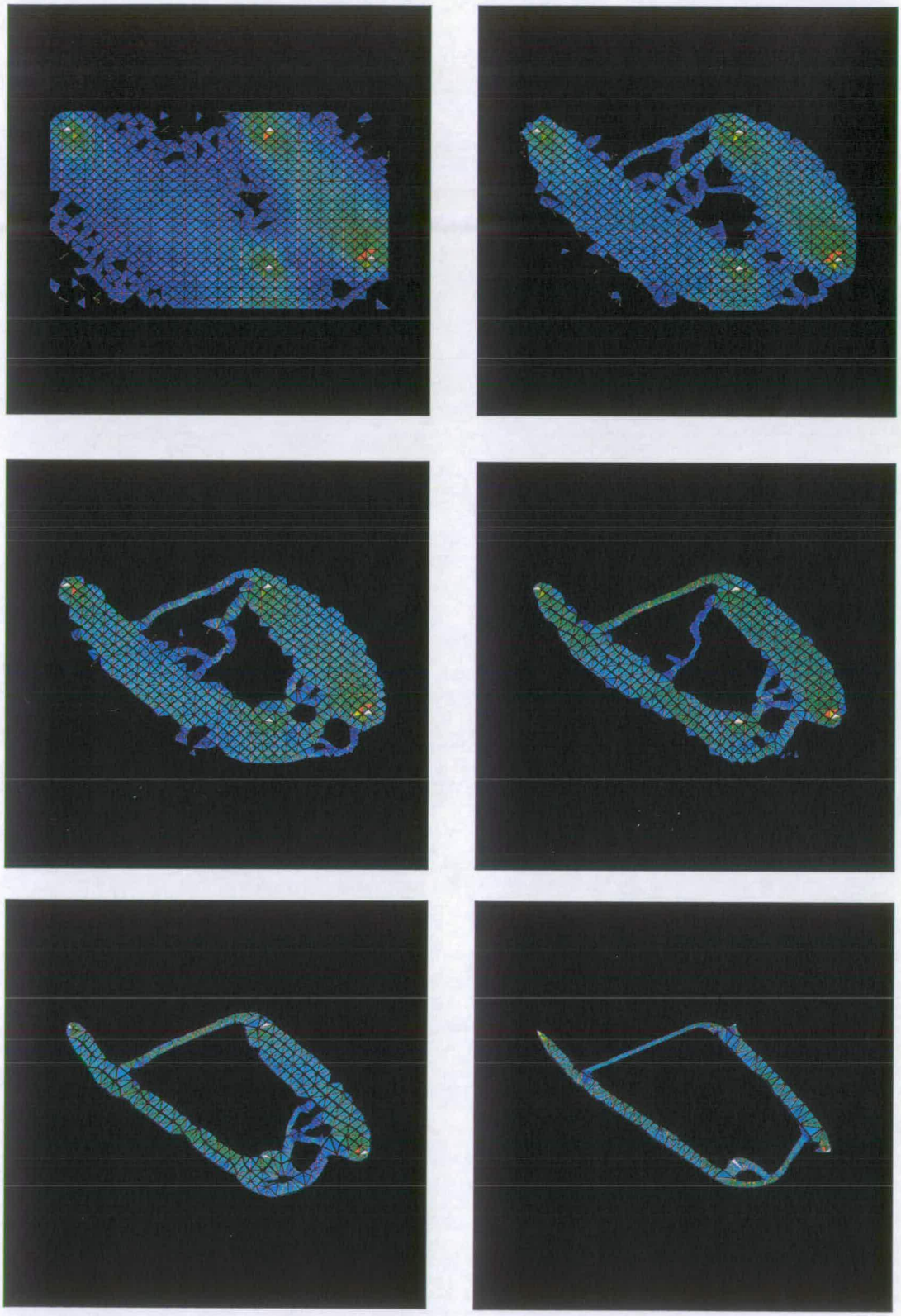


Figure 8-27 Snapshots of Growth of Bicycle Frame B

Bicycle Frame C

Fitness	-0.000653
Max Deflection (mm)	0.653
Parameter	**Value**
propHole	0.3010
propKillWithoutCheck	0.2032
rChemSens	0.0490
rChem0Xpos	0.7665
rChem0Ypos	0.1465
rChem0Str	0.8437
rChem0Rate	0.3422
rChem1Xpos	0.9759
rChem1Ypos	0.3794
rChem1Str	0.3865
rChem1Rate	0.0132
rChem2Xpos	0.2298
rChem2Ypos	0.2021
rChem2Str	0.7163
rChem2Rate	0.5181
rChem3Xpos	0.9264
rChem3Ypos	0.4283
rChem3Str	0.2774
rChem3Rate	0.1194

Figure 8-28 shows snapshots of the growth of bicycle frame C from the initial population. This bicycle frame had a fairly low value for *propHole* and so few holes were generated through the development. *propKillWithoutCheck* took a low value and so the topology remained similar through the development. The sensitivity to the chemicals is low and so they have little effect. As can be seen the shape has a considerably lower section from the seat post to the beam section from the head-set to the bottom bracket than bicycle frames A or B. The section from the bottom bracket to the back wheel is also fairly high.

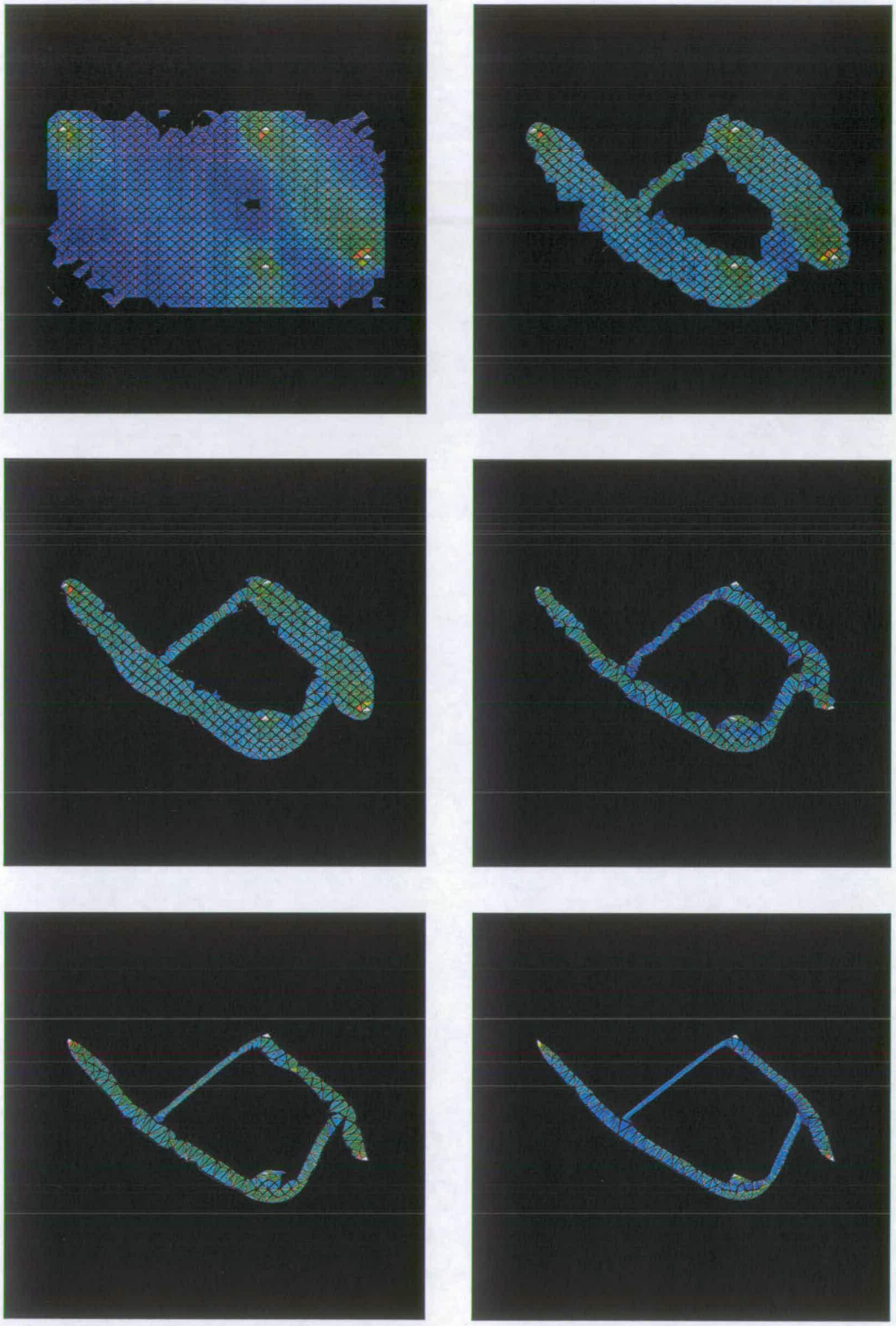


Figure 8-28 **Snapshots of Growth of Bicycle Frame C**

Bicycle Frame D

Fitness	-0.000952
Max Deflection (mm)	0.952
Parameter	**Value**
propHole	0.2631
propKillWithoutCheck	0.0892
rChemSens	0.2107
rChem0XPos	0.2805
rChem0YPos	0.5088
rChem0Str	0.2381
rChem0Rate	0.7271
rChem1XPos	0.5125
rChem1YPos	0.6488
rChem1Str	0.2180
rChem1Rate	0.2659
rChem0XPos	0.0854
rChem0YPos	0.5923
rChem0Str	0.3570
rChem0Rate	0.0979
rChem1XPos	0.3245
rChem1YPos	0.9597
rChem1Str	0.3320
rChem1Rate	0.6702

Figure 8-29 shows snapshots of the growth of bicycle frame D from the initial population. This bicycle frame had a fairly low value for *propHole* and so few holes were generated through the development. *propKillWithoutCheck* took a very low value and so the topology remained similar through the development. The main observation to be made about this bicycle frame is that the sensitivity to the chemicals was high, with *rChemSens* at 0.2107 - towards the upper bound of 0.3. This resulted in the section from the headset to the bottom bracket staying quite wide through the development of the shape.

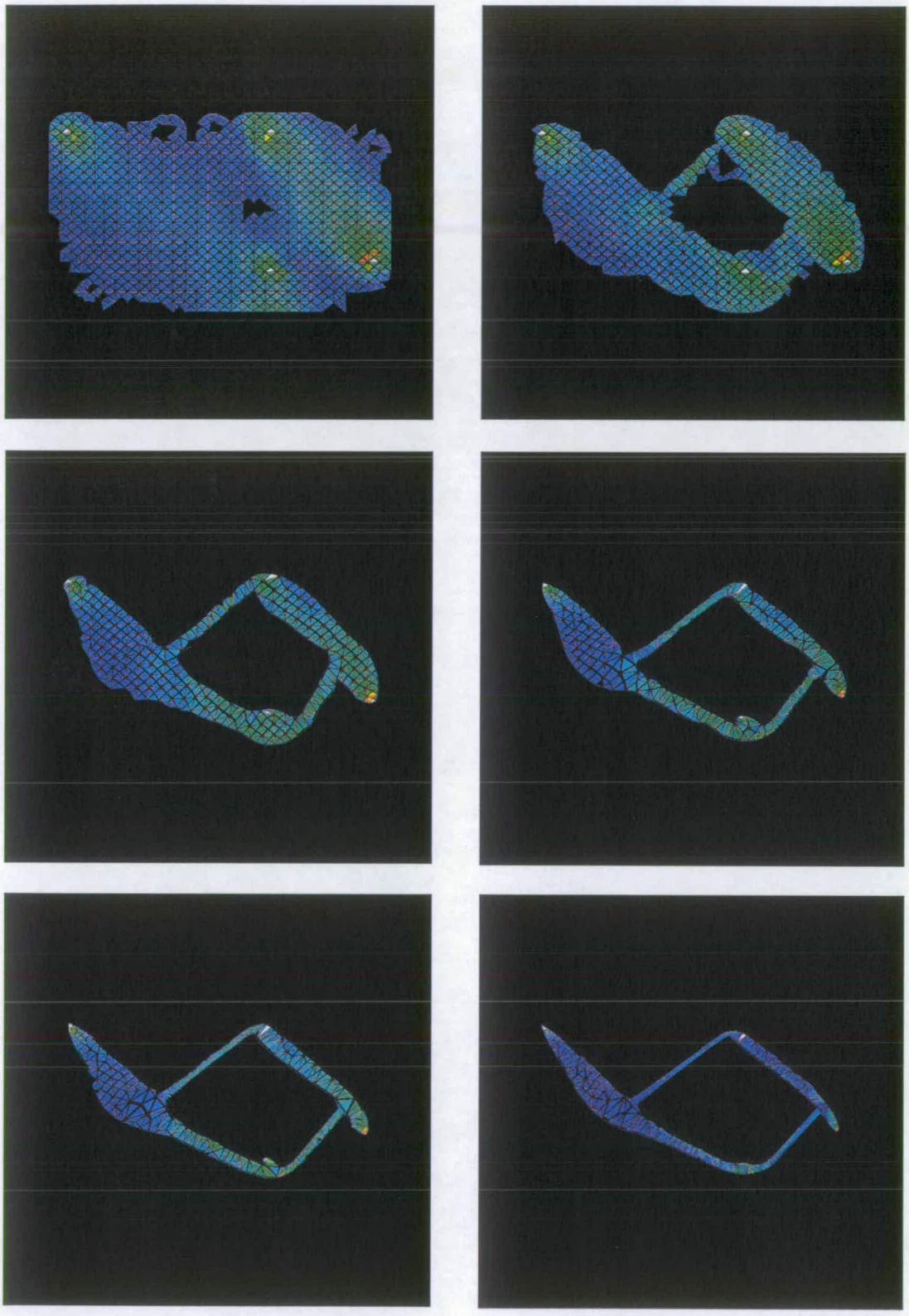


Figure 8-29 Snapshots of Growth of Bicycle Frame D

8.5.1.2 Results of GA Optimisation

This was the best individual found during the optimisation run.

Fitness	-0.000547
Max Deflection (mm)	0.547
Max Von Mises Stress (x10^6 Nm^2)	111
Mean Von Mises Stress (x10^6 Nm^2)	371
Area (m^2)	0.069
Parameter	**Value**
propHole	0.3805
propKillWithoutCheck	0.2901
rChemSens	0.1560
rChem0Xpos	0.4868
rChem0Ypos	0.5541
rChem0Str	0.9585
rChem0Rate	0.0197
rChem1Xpos	0.5118
rChem1Ypos	0.5177
rChem1Str	0.2334
rChem1Rate	0.2597
rChem0Xpos	0.7431
rChem0Ypos	0.2651
rChem0Str	0.5389
rChem0Rate	0.3657
rChem1Xpos	0.7714
rChem1Ypos	0.3471
rChem1Str	0.1151
rChem1Rate	0.5341

Figure 8-31 shows snapshots of the growth of the optimised bicycle frame. The maximum deflection was 0.547mm which represented about a 15% improvement over the best individual in the initial population (bicycle frame B). This bicycle frame had a value for *propHole* around the middle of the range of values for this parameter and so a reasonable number of holes were generated through the development. *propKillWithoutCheck* similarly took a midrange value. The sensitivity to the chemicals was also midrange and so the chemicals did have some influence over the development of the shape. Figure 8-30 shows the position of the chemicals

for this shape. As can be seen, they lie between the seat post, bottom bracket and back wheel. Consequently, cells in this area were more likely to be retained, which has resulted in the section from the bottom bracket up to the section from seat post to back wheel being more vertical than for other shapes. This may have increased the stiffness of the shape.

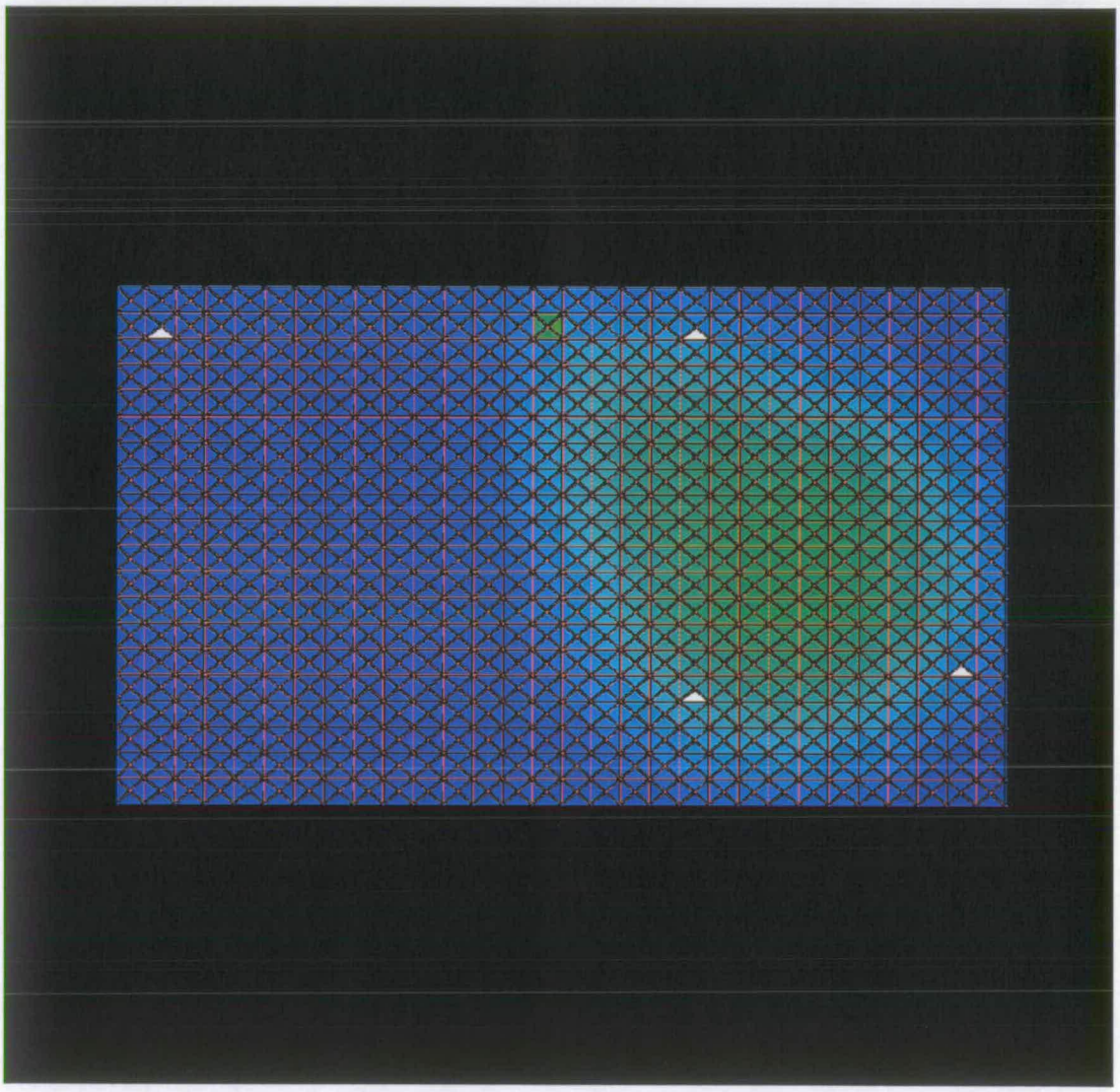


Figure 8-30 **Chemical Positions for the Optimised Bicycle Frame**

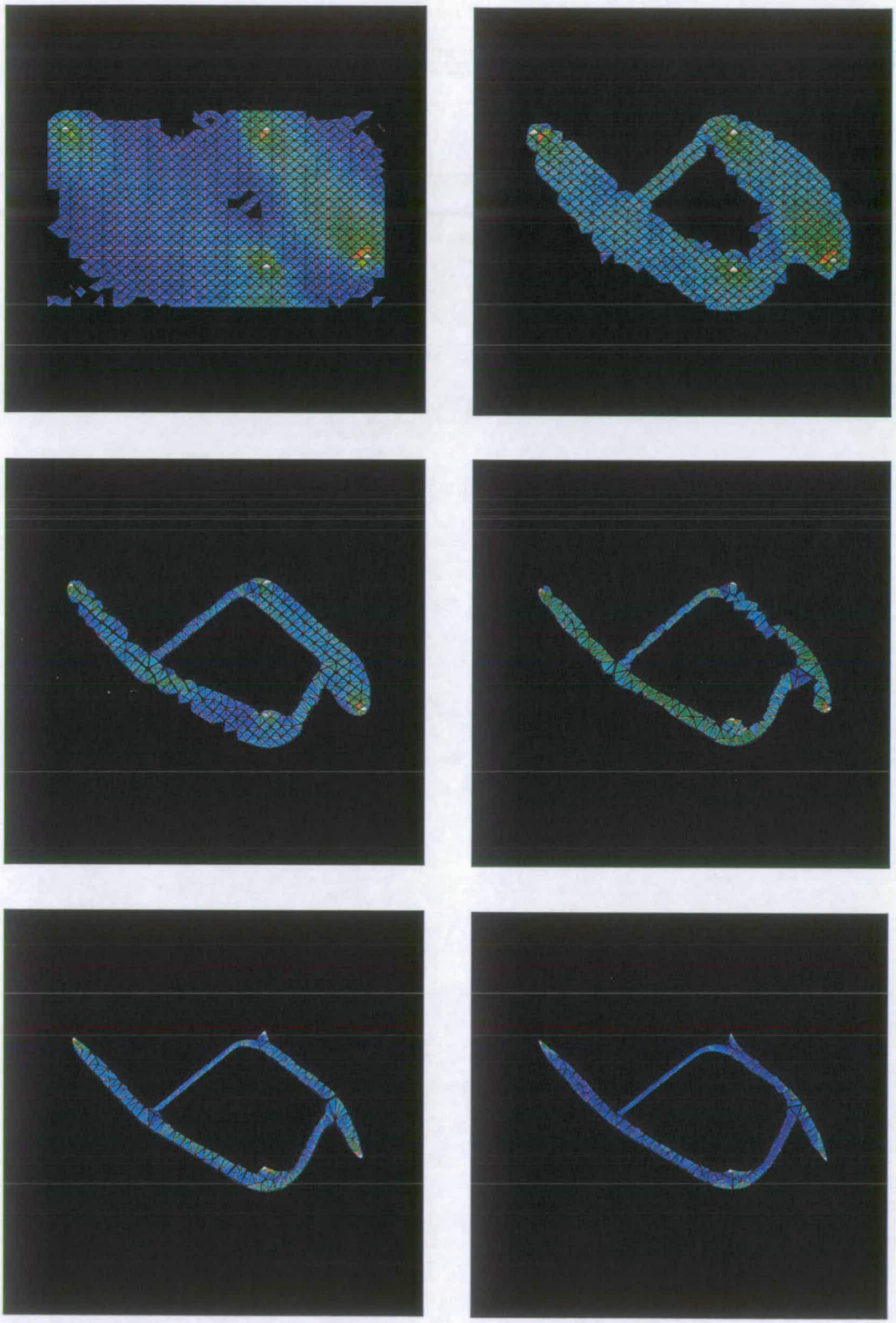


Figure 8-31 Snapshots of Growth of Optimised Bicycle Frame

8.5.2 Cantilever Beam

The load case for the cantilever beam was given in Section 8.4. The genetic algorithm was run with the following settings.

Fitness	-deflection	
Constraint	area < 0.14 m^2	
Population size	20	
Number of Crossovers per generation	5	
Number of Mutations per generation	5	
Number of Generations	20	
Parameter	**Min**	**Max**
propHole	0.01	0.8
propKillWithoutCheck	0.05	0.6
rChemSens	0	0.3
rChem0Xpos	0	1
rChem0Ypos	0	0.6
rChem0Str	0	1
rChem0Rate	0	0.6
rChem1Xpos	0	1
rChem1Ypos	0	0.6
rChem1Str	0	1
rChem1Rate	0	0.6
rChem0Xpos	0	1
rChem0Ypos	0	0.6
rChem0Str	0	1
rChem0Rate	0	0.6
rChem1Xpos	0	1
rChem1Ypos	0	0.6
rChem1Str	0	1
rChem1Rate	0	0.6

Initial Population

Below are shown some of the members of the initial population.

Cantilever A

Fitness	-0.001846
Displacement (mm)	1.846
Parameter	**Value**
propHole	0.6257
propKillWithoutCheck	0.4293
rChemSens	0.0460
rChem0Xpos	0.4414
rChem0Ypos	0.2922
rChem0Str	0.3827
rChem0Rate	0.0962
rChem1Xpos	0.7306
rChem1Ypos	0.0774
rChem1Str	0.2338
rChem1Rate	0.5040
rChem0Xpos	0.8435
rChem0Ypos	0.1380
rChem0Str	0.1916
rChem0Rate	0.3094
rChem1Xpos	0.6002
rChem1Ypos	0.4763
rChem1Str	0.5612
rChem1Rate	0.0846

Figure 8-32 shows snapshots of the growth of cantilever A from the initial population. As can be seen, the growth stage generates reasonable looking shapes even with parameters set at random. This cantilever had a high value for *propHole* and so numerous holes were generated through the development. *propKillWithoutCheck* took a moderate value and so it was reasonably easy for the topology to change through the development. The sensitivity to the chemicals was low and so the shape was not much affected by the chemicals.

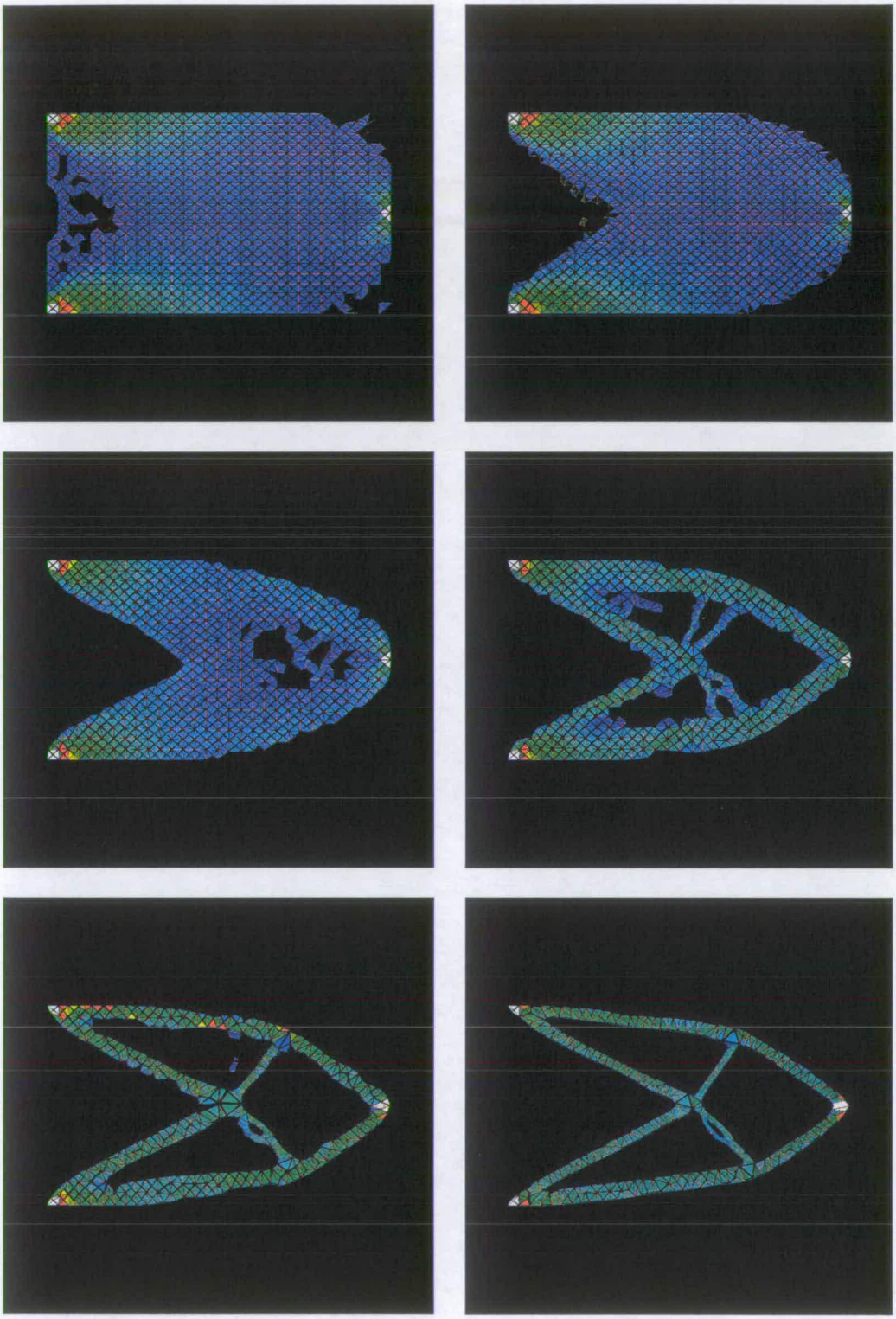


Figure 8-32 **Snapshots of Growth of Cantilever A**

Cantilever B

Fitness	-0.001993
Displacement (mm)	1.993
Parameter	**Value**
propHole	0.1655
propKillWithoutCheck	0.4528
rChemSens	0.0140
rChem0XPos	0.4907
rChem0YPos	0.3598
rChem0Str	0.6170
rChem0Rate	0.4419
rChem1XPos	0.8986
rChem1YPos	0.5297
rChem1Str	0.5102
rChem1Rate	0.5415
rChem2XPos	0.3571
rChem2YPos	0.5716
rChem2Str	0.9401
rChem2Rate	0.5077
rChem3XPos	0.5331
rChem3YPos	0.2335
rChem3Str	0.2661
rChem3Rate	0.2685

Figure 8-33 shows snapshots of the growth of cantilever B from the initial population. The main difference between this cantilever and cantilever A is that *propHole* is much smaller and so fewer holes were generated through the development. Consequently, the development of the shape was changed. More of the material was removed from the left of the design domain before holes appeared in the right hand side. The cross-trusses are therefore attached further to the right for cantilever B than for cantilever A.

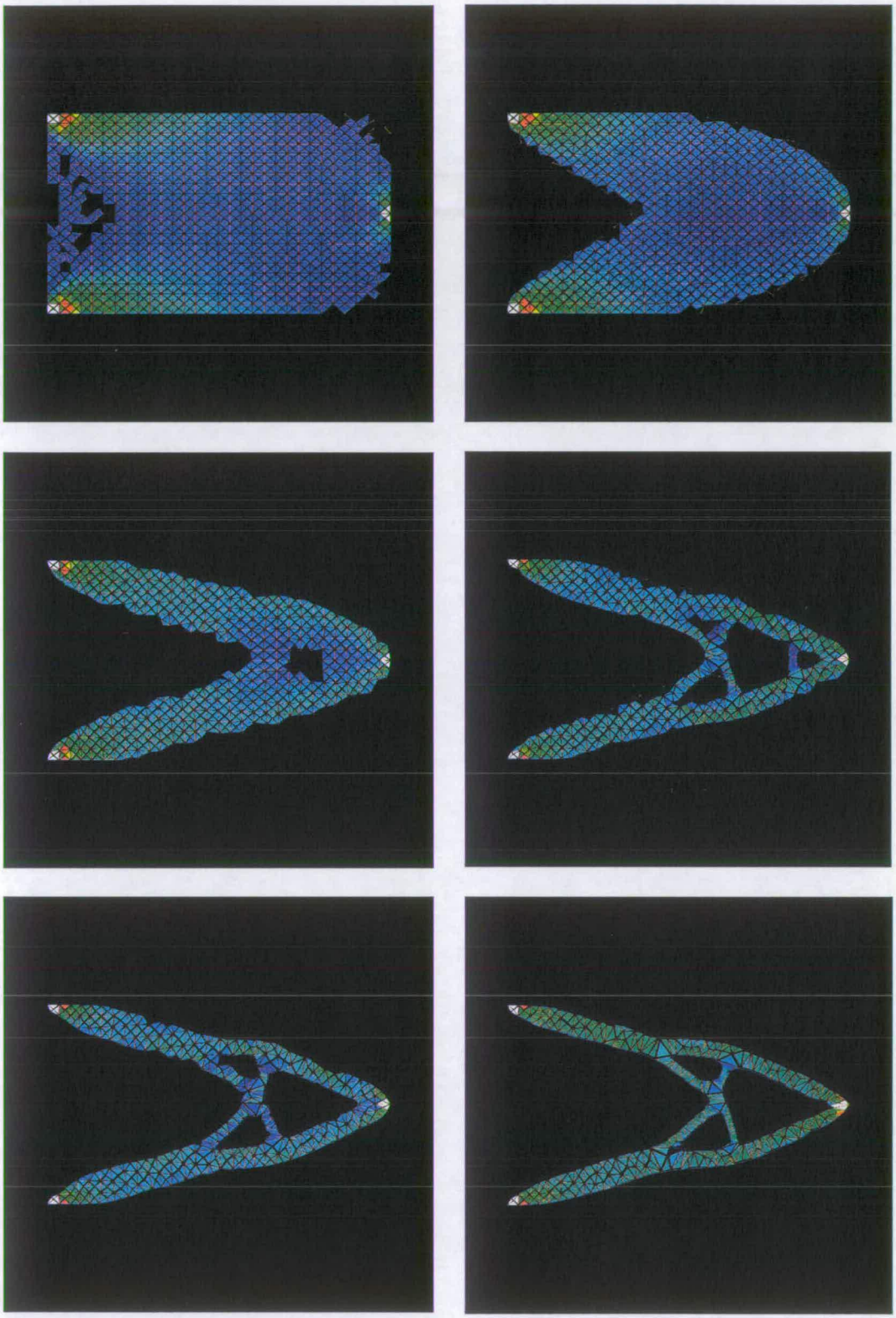


Figure 8-33 **Snapshots of Growth of Cantilever B**

Cantilever C

Fitness	-0.002105
Displacement (mm)	2.105
Parameter	**Value**
propHole	0.2144
propKillWithoutCheck	0.1885
rChemSens	0.0347
rChem0XPos	0.6458
rChem0YPos	0.4066
rChem0Str	0.9179
rChem0Rate	0.4775
rChem1XPos	0.3325
rChem1YPos	0.4028
rChem1Str	0.7692
rChem1Rate	0.1936
rChem2XPos	0.1193
rChem2YPos	0.0983
rChem2Str	0.7568
rChem2Rate	0.0818
rChem3XPos	0.1010
rChem3YPos	0.0213
rChem3Str	0.2083
rChem3Rate	0.2543

Figure 8-34 shows snapshots of the growth of cantilever C from the initial population. For this cantilever very few holes were generated and so material was successively removed from the left, resulting in a two truss structure.

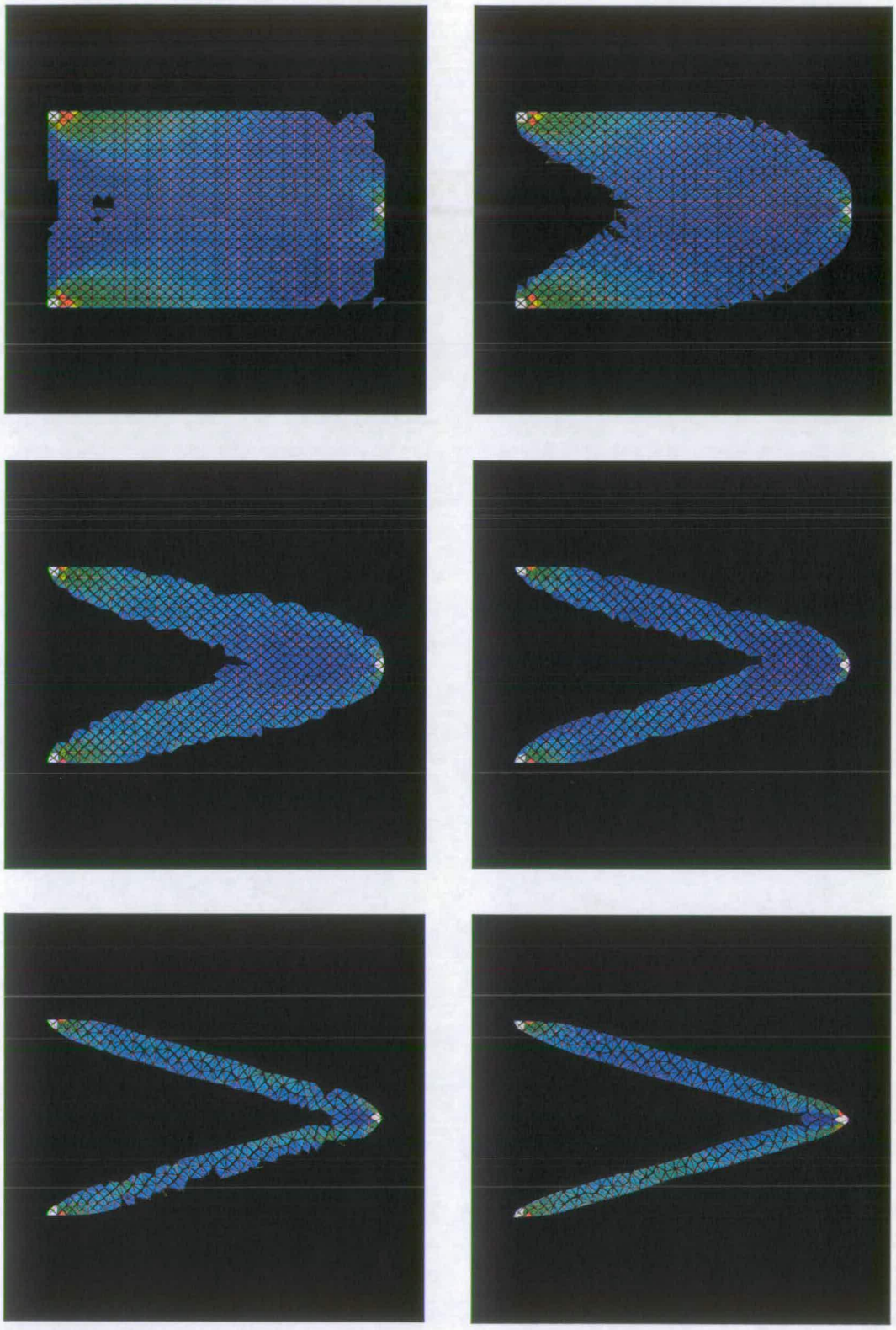


Figure 8-34 Snapshots of Growth of Cantilever C

Cantilever D

Fitness	-0.001828
Displacement (mm)	1.828
Parameter	**Value**
propHole	0.6539
propKillWithoutCheck	0.1903
rChemSens	0.0003
rChem0Xpos	0.1272
rChem0Ypos	0.5361
rChem0Str	0.5338
rChem0Rate	0.2318
rChem1Xpos	0.5543
rChem1Ypos	0.5830
rChem1Str	0.9587
rChem1Rate	0.0209
rChem2Xpos	0.6658
rChem2Ypos	0.3000
rChem2Str	0.7951
rChem2Rate	0.2480
rChem3Xpos	0.5079
rChem3Ypos	0.5254
rChem3Str	0.7991
rChem3Rate	0.3160

Figure 8-35 shows snapshots of the growth of cantilever D from the initial population. This cantilever had a high value for *propHole* and so numerous holes were generated throughout the development. *propKillWithoutCheck* was low and so the sections that formed when holes appeared were only removed when the stress on them was very low. This resulted in a shape with numerous trusses. This shape had the best fitness (i.e. lowest maximum deflection) in the initial population.

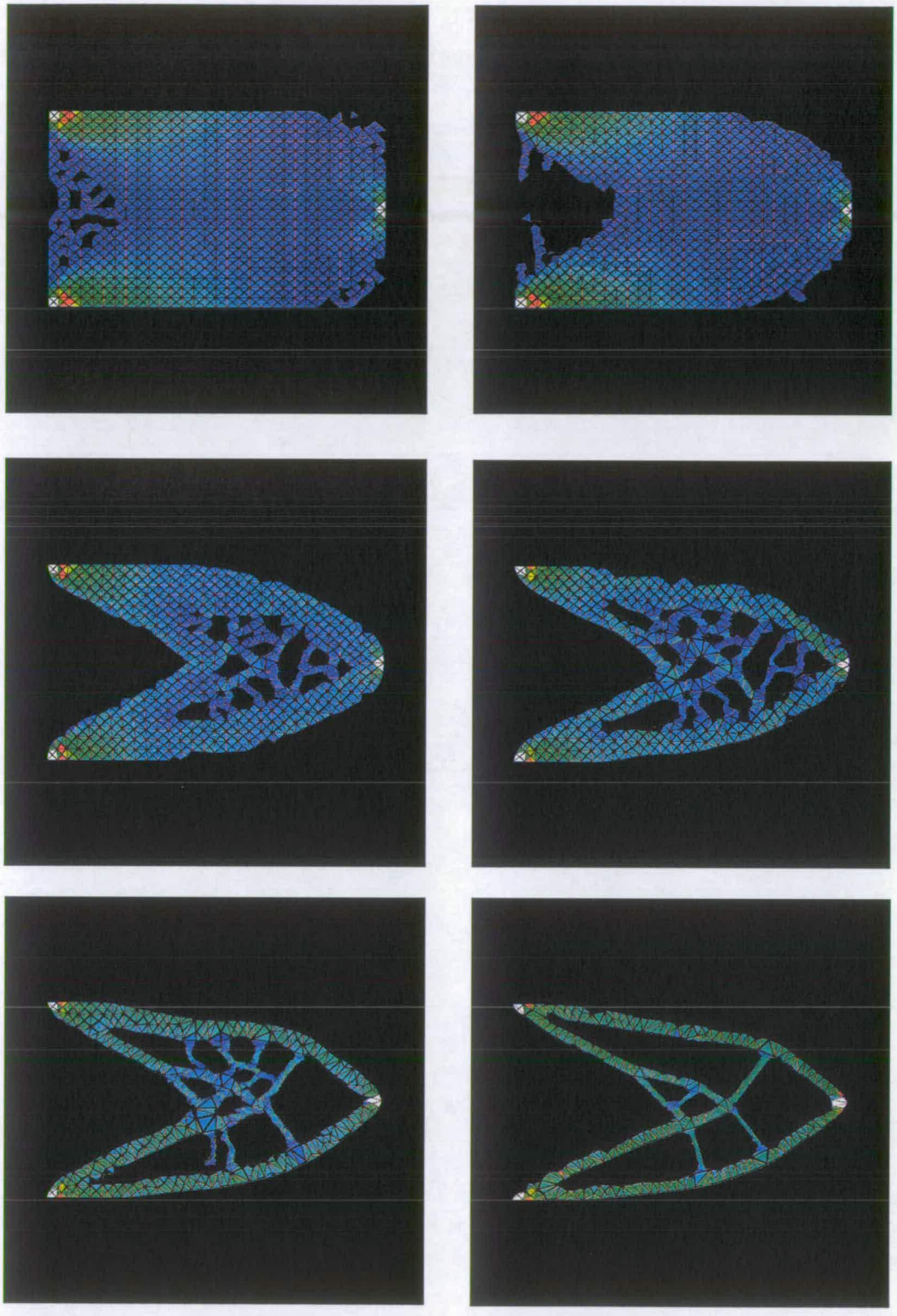


Figure 8-35 **Snapshots of Growth of Cantilever D**

8.5.2.1 Results of GA Optimisation

Below is shown the best cantilever found during the optimisation run.

Fitness	-0.001695
Max Deflection (mm)	1.695
Max Von Mises Stress (x10^6 Nm^2)	384
Mean Von Mises Stress (x10^6 Nm^2)	109
Area (m^2)	0.139
Parameter	**Value**
propHole	0.4868
propKillWithoutCheck	0.5410
rChemSens	0.0058
rChem0Xpos	0.8059
rChem0Ypos	0.2556
rChem0Str	0.9412
rChem0Rate	0.3447
rChem1Xpos	0.6683
rChem1Ypos	0.3138
rChem1Str	0.6261
rChem1Rate	0.3622
rChem2Xpos	0.6254
rChem2Ypos	0.1971
rChem2Str	0.6428
rChem2Rate	0.2337
rChem3Xpos	0.1911
rChem3Ypos	0.0671
rChem3Str	0.8573
rChem3Rate	0.4112

Figure 8-36 shows snapshots of the growth of the optimised cantilever. The maximum deflection was 1.695mm which represented a 9% improvement over the best individual in the initial population (cantilever D). This cantilever had a value for *propHole* around the middle of the range of values for this parameter and so a reasonable number of holes were generated through the development. *propKillWithoutCheck* similarly took a midrange value. The sensitivity to the chemicals was very low and so the chemicals had very little influence over the development of the shape.

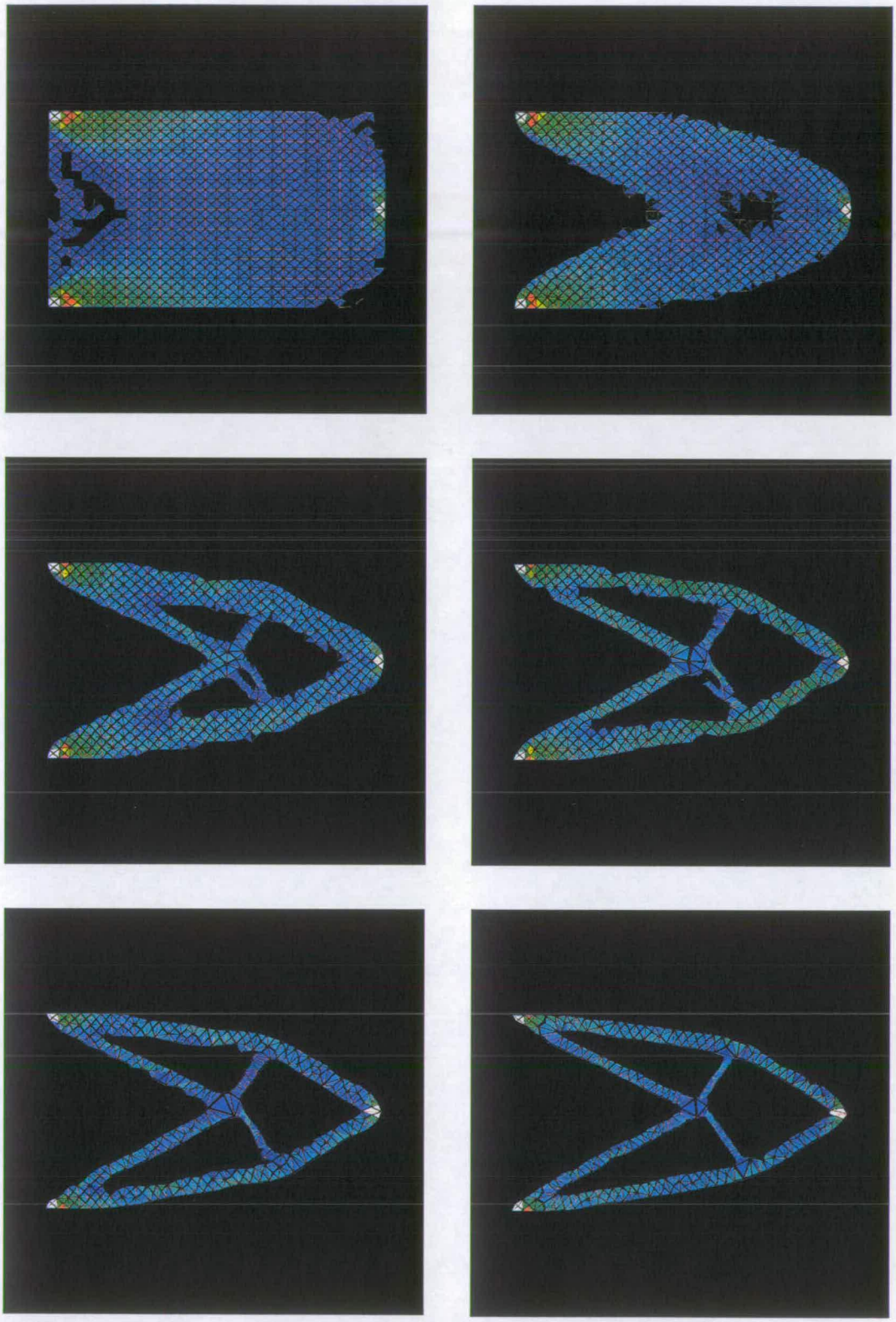


Figure 8-36 **Snapshots of Growth of Optimised Cantilever D**

8.5.3 Arch

The load case for the arch was given in Section 8.4. The genetic algorithm was run with the following settings.

Fitness	-deflection	
Constraint	area < 0.07 m^2	
Population size	15	
Number of Crossovers per generation	3	
Number of Mutations per generation	3	
Number of Generations	20	
Parameter	**Min**	**Max**
propHole	0.05	0.8
propKillWithoutCheck	0.01	0.5
rChemSens	0	0
rChem0Xpos	0	0
rChem0Ypos	0	0
rChem0Str	0	0
rChem0Rate	0	0
rChem1Xpos	0	0
rChem1Ypos	0	0
rChem1Str	0	0
rChem1Rate	0	0
rChem2Xpos	0	0
rChem2Ypos	0	0
rChem2Str	0	0
rChem2Rate	0	0
rChem3Xpos	0	0
rChem3Ypos	0	0
rChem3Str	0	0
rChem3Rate	0	0

As can be seen the chemicals are not used for this problem. This was because the problem was symmetric and it was found in initial tests that any use of chemicals resulted in impaired performance. Consequently, there were only two parameters. Even with two parameters a large variety of shapes could be produced.

8.5.3.1 Initial Population

Below are shown some of the members of the initial population.

Arch A

Fitness	-0.681
Deflection (mm)	0.681
Parameter	**Value**
propHole	0.6002
propKillWithoutCheck	0.1998
rChemSens	0
rChem0Xpos	0
rChem0Ypos	0
rChem0Str	0
rChem0Rate	0
rChem1Xpos	0
rChem1Ypos	0
rChem1Str	0
rChem1Rate	0
rChem0Xpos	0
rChem0Ypos	0
rChem0Str	0
rChem0Rate	0
rChem1Xpos	0
rChem1Ypos	0
rChem1Str	0
rChem1Rate	0

Figure 8-37 shows snapshots of the growth of arch A from the initial population. This arch had a high value for *propHole* and so numerous holes were generated through the development. *propKillWithoutCheck* took a moderate value and so it was reasonably easy for the topology to change through the development. It can be seen that an arch is developed as expected. The load is supported by two trusses up to the arch.

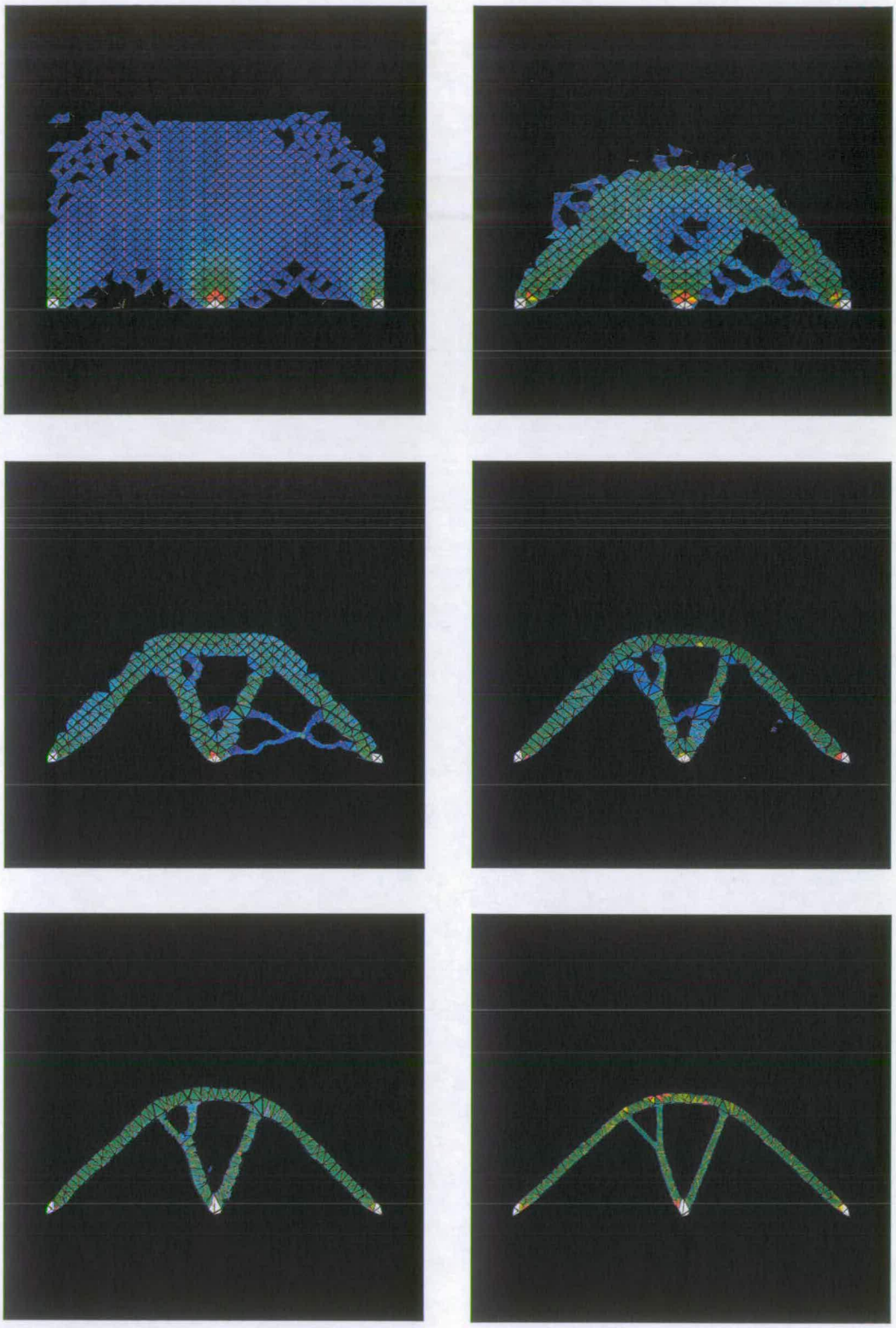


Figure 8-37 **Snapshots of Growth of Arch A**

Arch B

Fitness	-0.000723
Deflection (mm)	0.723
Parameter	**Value**
propHole	0.4781
propKillWithoutCheck	0.3576
rChemSens	0
rChem0Xpos	0
rChem0Ypos	0
rChem0Str	0
rChem0Rate	0
rChem1Xpos	0
rChem1Ypos	0
rChem1Str	0
rChem1Rate	0
rChem0Xpos	0
rChem0Ypos	0
rChem0Str	0
rChem0Rate	0
rChem1Xpos	0
rChem1Ypos	0
rChem1Str	0
rChem1Rate	0

Figure 8-38 shows snapshots of the growth of arch B from the initial population. This arch had a mid-range value for *propHole* and a moderate number of holes were generated through the development. *propKillWithoutCheck* also took a midrange value and so it was reasonably easy for the topology to change through the development. It can be seen that this shape is fairly similar to arch A, although the angle of the trusses is wider.

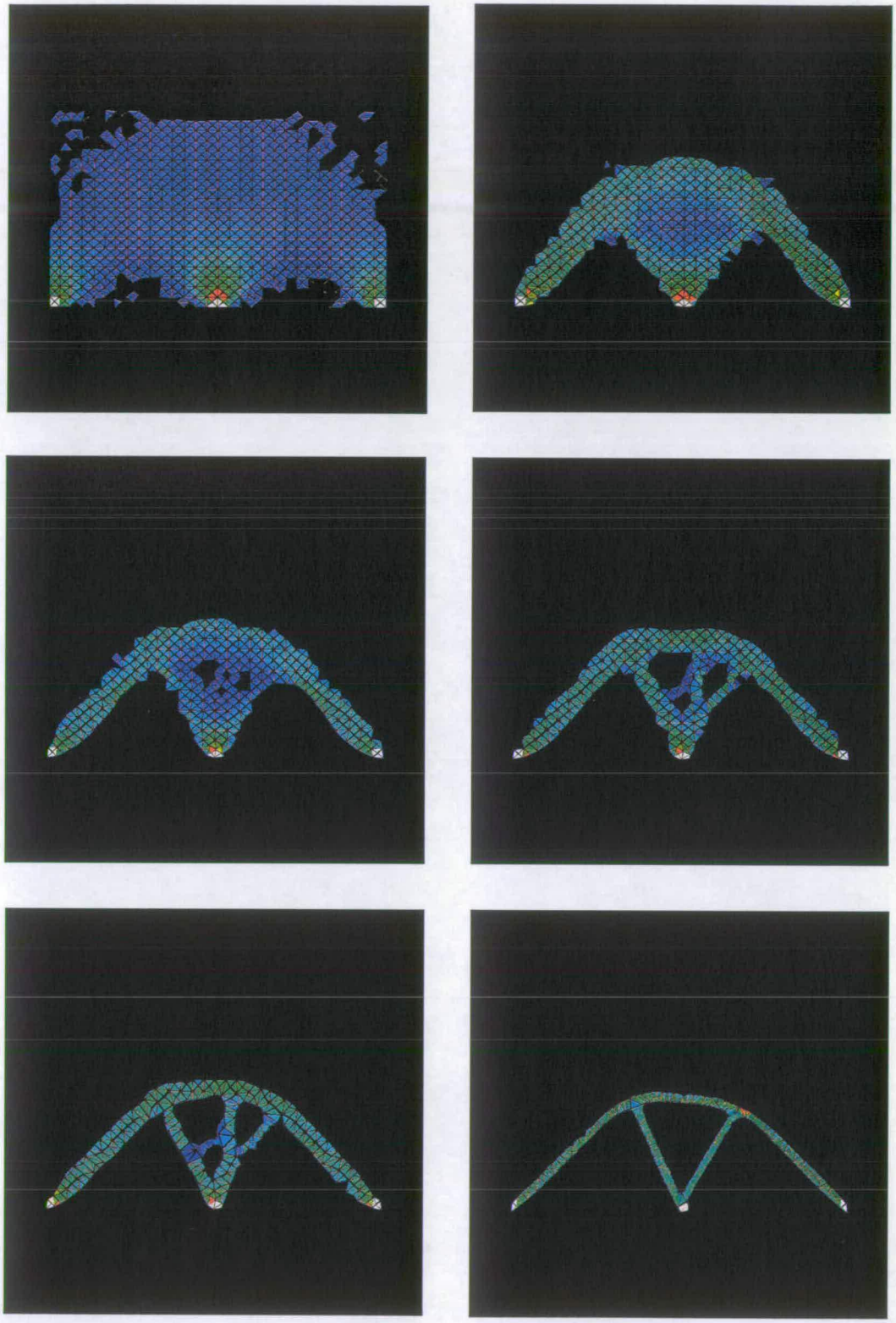


Figure 8-38 **Snapshots of Growth of Arch B**

Arch C

Fitness	-0.000835
Deflection (mm)	0.835
Parameter	**Value**
propHole	0.7210
propKillWithoutCheck	0.3215
rChemSens	0
rChem0Xpos	0
rChem0Ypos	0
rChem0Str	0
rChem0Rate	0
rChem1Xpos	0
rChem1Ypos	0
rChem1Str	0
rChem1Rate	0
rChem0Xpos	0
rChem0Ypos	0
rChem0Str	0
rChem0Rate	0
rChem1Xpos	0
rChem1Ypos	0
rChem1Str	0
rChem1Rate	0

Figure 8-39 shows snapshots of the growth of arch C from the initial population. This arch had a high value for *propHole* and so a large number of holes were generated through the development. *propKillWithoutCheck* took a midrange value and so it was reasonably easy for the topology to change through the development. It can be seen that the height of the arch is similar to those of arches A and B. However the load is supported by three trusses.

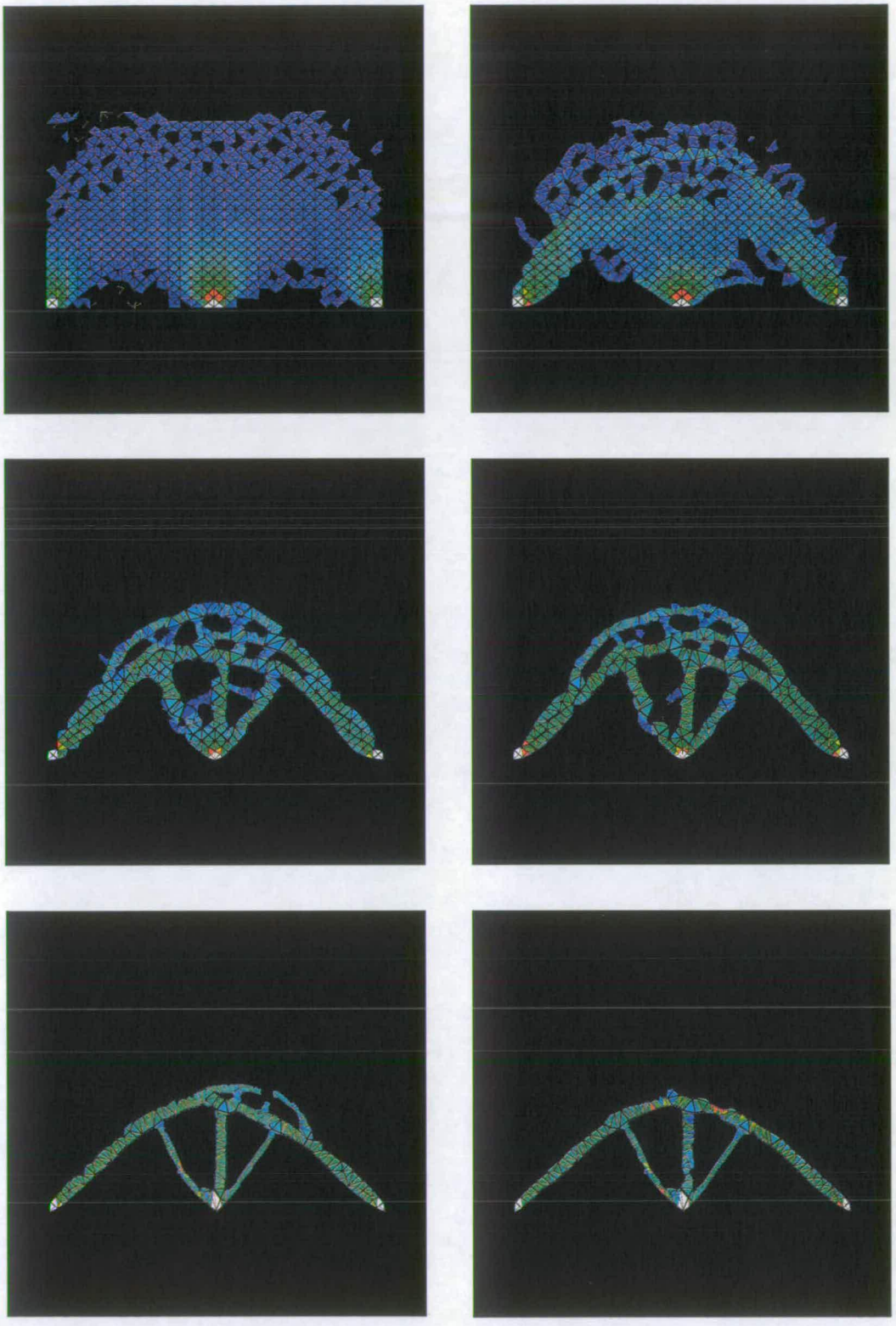


Figure 8-39 **Snapshots of Growth of Arch C**

Arch D

Fitness	-0.00073
Deflection (mm)	0.73
Parameter	**Value**
propHole	0.2764
propKillWithoutCheck	0.2311
rChemSens	0
rChem0XPos	0
rChem0YPos	0
rChem0Str	0
rChem0Rate	0
rChem1XPos	0
rChem1YPos	0
rChem1Str	0
rChem1Rate	0
rChem0XPos	0
rChem0YPos	0
rChem0Str	0
rChem0Rate	0
rChem1XPos	0
rChem1YPos	0
rChem1Str	0
rChem1Rate	0

Figure 8-40 shows snapshots of the growth of arch D from the initial population. This arch had low value for *propHole* and few holes were generated. The shape was primarily formed by moving existing boundaries inwards. This has resulted in a shape with a higher arch and only one truss supporting the load.

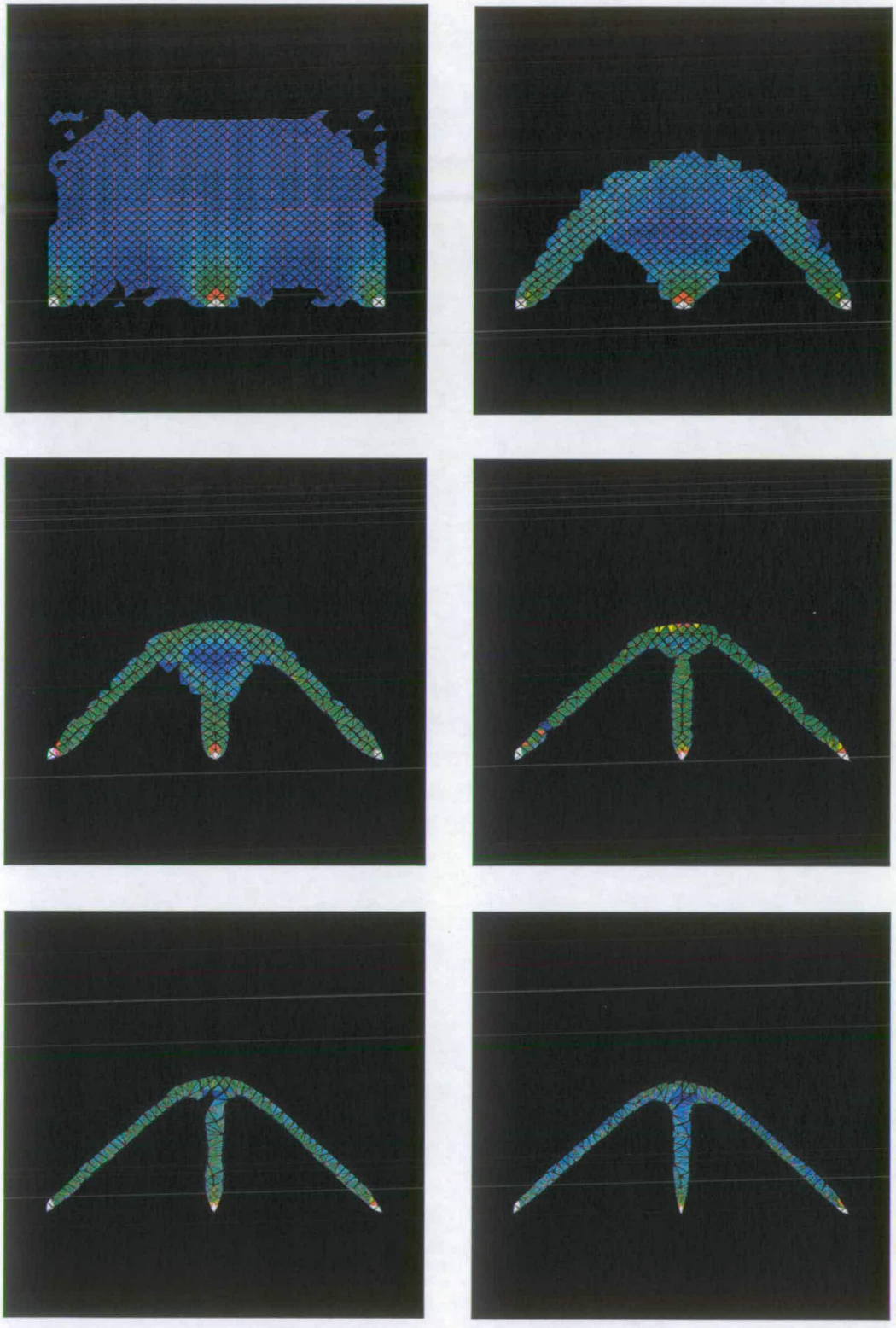


Figure 8-40 **Snapshots of Growth of Arch D**

8.5.3.2 Results of GA Optimisation

Below is shown the best arch found during the optimisation run.

Fitness	-5.5600e-004
Max Deflection (mm)	0.556
Max Von Mises Stress (x10^6 Nm^2)	153
Mean Von Mises Stress (x10^6 Nm^2)	84
Area (m^2)	0.069
Parameter	**Value**
propHole	0.6005
propKillWithoutCheck	0.4546
rChemSens	0
rChem0XPos	0
rChem0YPos	0
rChem0Str	0
rChem0Rate	0
rChem1XPos	0
rChem1YPos	0
rChem1Str	0
rChem1Rate	0
rChem0XPos	0
rChem0YPos	0
rChem0Str	0
rChem0Rate	0
rChem1XPos	0
rChem1YPos	0
rChem1Str	0
rChem1Rate	0

Figure 8-41 shows snapshots of the growth of the optimised arch. This arch had a fairly high value for *propHole* and so a large number of holes were generated through the development. *propKillWithoutCheck* took a high value and so it was easy for the topology to change through the development. It can be seen that this arch is similar to arch C, but with a slightly higher arch and better formed trusses.

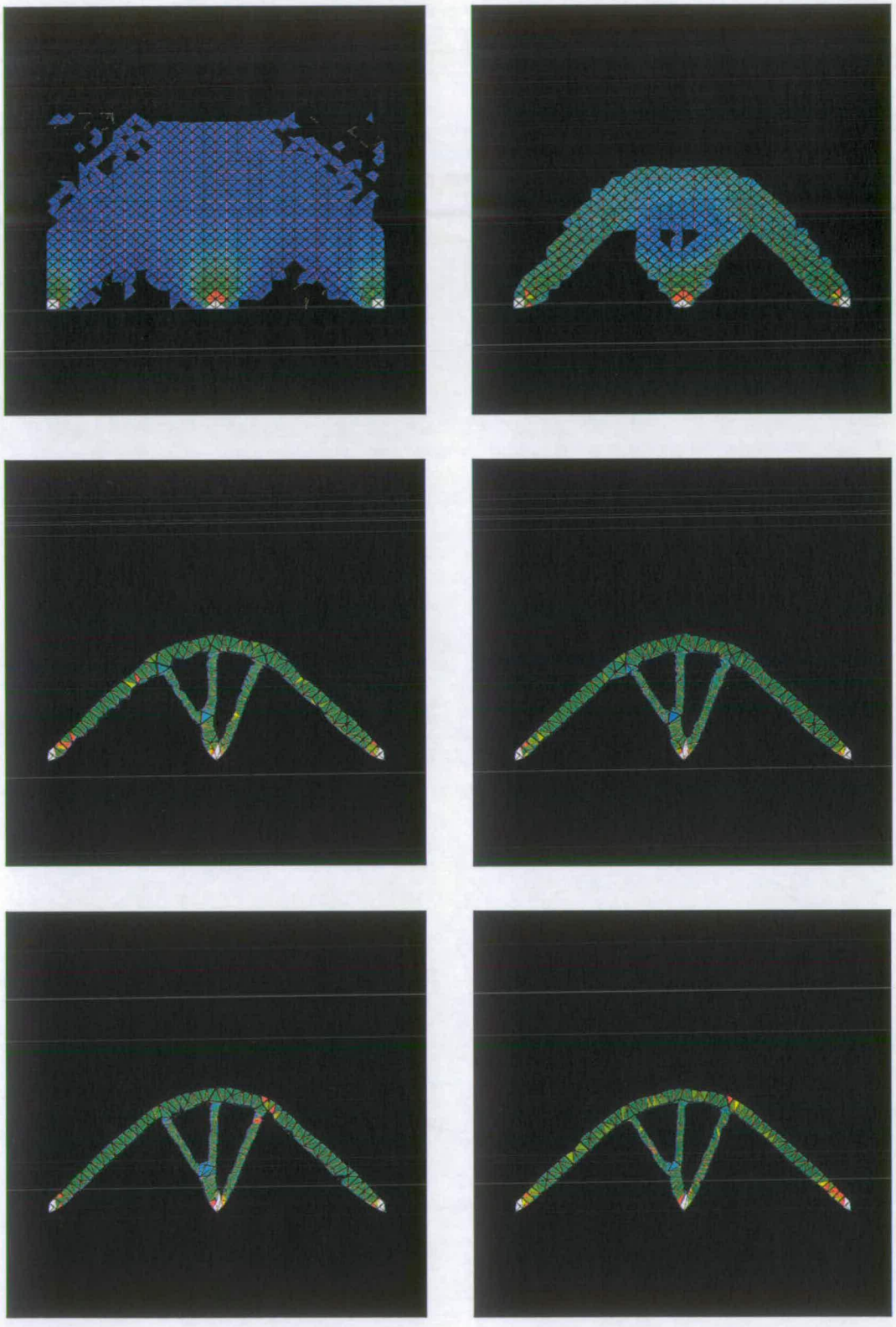


Figure 8-41 **Snapshots of Growth of Optimised Arch**

8.6 Discussion

8.6.1 Discussion of Results

This morphogenic cellular approach to shape and topology optimisation worked well for the three test cases. Shapes were produced which met the objectives (namely minimising deflection whilst remaining within a constraint on the area). The shapes qualitatively matched with what would be expected of optimal shapes for those load cases and were qualitatively similar to those reported by others doing similar optimisations [Chen *et al.* 2002] [Baumgartner & Mattheck 1994] [Mattheck *et al.* 1994] [Chen & Tsai 1993].

It was clear that the shape was primarily determined by the development stage. It was interesting that the choice of the variables represented by the genes often had a considerable effect on the shape early in the run, but later in the run the shapes tended to converge back towards similar shapes. The variables did, though, modulate the final shapes to a reasonable degree, whilst still keeping them 'sensible'. In other words, the shapes were sufficiently different that the variation in parameters produced functionally different shapes.

The method can be seen to be effective at taking a Chain model of the specification of the design required (with loading points, forces, specified displacements at built in points and constraints on area) and producing a cellular model which minimises the required behaviour of displacement, as was discussed in Chapter 7. This was done using a combination of search using the GA and by utilising information generated about the shapes' performance through the development stage.

8.6.2 Issues

The main concern over this approach was in ensuring the quality of the mesh and hence the accuracy of the analysis. The crude adaptive meshing routines that the cells were given provided a reasonable quality of mesh in most areas. There were,

however, circumstances when the mesh looked inadequate. This happened when, for instance, a section was becoming thinner. The cells would recognise that the stress gradient over them was high and divide forming new cells, but, because the stress in that area was low, they would die in the next time step. The cells could only divide once during a time step and so were dying as quickly as they could be generated by the cell division. This could lead to some truss sections only being one element thick during some stages of the development and the results would therefore be somewhat inaccurate. At the end of the development when the area constraint was achieved and the stress threshold stopped increasing, the cell division did then increase the quality of the mesh.

To some extent, these problems were caused by the desire, which was present from the outset, to apply no 'top-down' control to the cells. One of the motivations of this work was to see if shapes could be generated from the 'bottom-up', with as little external interference as possible. It may be possible to periodically extract the boundary and, possibly, fit this with a spline and then remesh the interior. This would, however, go against the bottom-up philosophy of this approach. Another possible solution to the mesh quality problems would be to increase mesh density from the start of the run. This would increase the time taken to produce a shape. Nevertheless, the shapes generated by the development stage did seem to be robust to the inaccuracy of the analysis.

The area constraints were chosen for each of the problems so that feasible shapes could be produced in which the Von Mises stress was around 100 MPa (giving a factor of safety of about three for the steel used). The mean stress was indeed found to be around the stress expected. The maximum stress, however, was often found to be higher at around the yield stress (350 MPa) for the material. This occurred at the loading points and was thought to be because the loads were applied at single nodes and therefore caused locally high stresses. This could have been avoided by applying the loads in a more realistic way.

Another concern was the extent to which the genetic algorithm was able to evolve the shapes. As mentioned earlier, the shapes tended to converge towards the end of the development. In other words, the number of trusses and overall layout of the shapes tended to be similar at the end of a run. However, small events towards the start of the development stage, such as the death of a single cell, could have a large impact later in the run on the shape generated. This was mainly by changing the angle of trusses or where they attached to other sections. This was the main way in which the variety in the shapes was generated. It was thought that this sensitivity to small events might make the search space very noisy and difficult to optimise (although a genetic algorithm would certainly be one of the better ways to optimise in such a search space). Unfortunately, it was not possible to undertake any systematic study of the search space or over genetic operators. This was due to both time constraints on the work and because optimisation runs were very time consuming. Nevertheless, it was possible to conclude that, in the runs given in the results, the GA did improve the shapes generated.

The speed of the development stage was the main limitation of the approach as a single shape development could take three or four minutes. This was because the analysis had to be called frequently. It was therefore not possible to evolve more than a few hundred shapes in a particular run. This was mitigated by the fact that good quality solutions were available almost immediately (typically at least one of the first few shapes generated would be good). Also, no user intervention was required and the space of possible shapes that could be generated was very large and general. Therefore, the aim stated in Chapter 1, that the use of search techniques be extended further into the design process of generating form from function, was to some extent achieved.

One of the ideas postulated in Section 7.5.1 was that the physical elements described could be made autonomous and to some extent intelligent. In the work, some first steps were made to investigate these ideas. By imparting the elements with some simple behaviours it was possible, with little top-down control (the only top down control was the setting of the stress-threshold), to generate efficient shapes.

8.6.3 Further work

This was a preliminary study into whether this morphogenetic approach to shape optimisation shows any promise. This early work does indeed indicate that this may be an effective approach. There is clearly, though, need for further work before this can be concluded with any certainty.

The role of evolution in manipulating the genotype, so that good shapes are grown, needs a great deal more investigation. At present, due to time constraints, only a limited study has been made of the EA. The design of operators and the choice of the rules of development, along with what parameters to allow the EA to manipulate, needs further investigation.

The mesh-based shape representation is very general, and can represent a great many shapes with varying topologies. Despite this generality, the combination of the EA and the development appears to perform well at searching through this space. Further work into the variety of shapes that can be generated, both within a particular run, and under various load cases, would be interesting.

One area where further work could be undertaken is to try other methods of developing the shapes. At present, the development is heavily influenced by the stress. Further work might include other biologically plausible mechanisms for influencing cell behaviour such as cell signalling. This could well help the evolvability of the representation.

At present, new cells can only be created through cell division within the current boundaries of the shape. The primary motor for changing the shape is through cell death. Further work could involve cells dividing out of the existing boundaries so that 'swelling' can take place in high stress areas.

It was also thought that the addition of the 'chemicals' in the environment could be made interactive. This could allow a designer to influence the retention of cells in

certain areas without over-constraining the shapes generated. This would allow the search to be influenced by the designer's intuitions.

Another area of further work would be to investigate how the approach scales when finer meshes are used. The majority of the computation effort takes place in the finite element analysis. The solution time for a linear elastic problem is approximately proportional to nb^2 where n is the number of elements and b is the semi-bandwidth [Desai & Kundu 2001]. Calculating the semi-bandwidth is not necessarily easy, however [Desai & Kundu 2001] give an example, with a two-dimensional problem, where computational cost is proportional to n^2. The computational costs of cell actions are likely to be proportional to the number of cells (although this is not certain). The cell action costs are, though, likely to be dominated by the analysis costs. With some small alterations to the current software, it may be possible to start with a coarse mesh, with low computational costs, and increase the number of elements considerably during the run. This would enable detailed alterations of the shape at the end of the run, without incurring excessive runs for initial parts of the growth.

For all the test cases tried so far, the effect of the removal of material is primarily local. It would be interesting to investigate how the system could be adapted to problems (such as the annulus problem from Chapter 4) where there is a rotational load. In such cases, addition or removal of material at the rim, for example, would have a considerable affect at the hub. For such problems, it may be helpful to mimic cell differentiation, which is exploited by biological development. This would allow cell behaviour to vary according to position. Cells at the hub, for instance, might evolve to behave differently than those at the rim.

8.7 Conclusions

This work used a genetic algorithm to evolve shapes for three load cases, a bicycle frame, a cantilever beam and an arch. The approach was novel because it used a development stage in which the shape was 'grown' so that the genes did not encode the shape explicitly, but rather influenced the behaviour of a development stage of a cellular shape representation. The development stage made use of information generated by a finite element analysis to guide the generation of the shapes with the genes modulating the exact execution of the process.

- This work did not attempt to produce a definitive algorithm for shape or topological optimisation using morphogenesis. It did, however, establish that such an approach is feasible and shows promise for shape optimisation.

- The approach was able to use the information generated in the finite element analysis on a shape's performance, in an effective way, to influence the shapes generated by a morphogenetic evolutionary algorithm approach to shape optimisation.

- The approach was able to make use of 'intelligent cells', where finite elements are endowed with simple behaviours, to produce efficient shapes, with very little 'top-down' control.

9 Conclusions

The primary aim of this thesis was to determine whether shape optimisation could be extended, such that it can be used to increase the automation of the process of shape synthesis for engineering design.

In order to investigate this, there were three objectives:

(a) *To determine whether evolutionary algorithms, along with novel shape representations, which are able to represent a large generality of shapes, would enable more automation of the process of determining form from function, and to identify any obstacles that might be encountered with such an approach.*

Investigations were undertaken into the use of evolutionary algorithms with general shape representations. Firstly, an aerofoil optimisation was undertaken with a parametric aerofoil representation and a Bézier representation. Fluid analysis was done with a vortex panel method and a genetic algorithm optimiser was used. The genetic algorithm worked well with the parametric representation, but failed with the more general Bézier representation because the vortex panel method was unable to accurately model many of the Bézier shapes.

Secondly, a voxel shape representation was used with a genetic algorithm for structural optimisation. This was applied to two problems: the design of a beam cross-section and the optimisation of the axisymmetric cross-section of a jet engine annulus. Specialised genetic operators were developed for this representation. This method proved reasonably successful. Two problems were encountered. There were some difficulties in ensuring that the analysis of the shapes with the finite element method was sufficiently accurate because of the 'stepped' boundary formed by voxels. Problems were also encountered because the problem specification (in terms of stress constraints at particular parts of the design), which was supplied by an industrial collaborator, was not sufficiently rigorous. Because the voxel

representation was very general, it was able to find shapes which met the specification as stated, but which would be undesirable in some other way.

Thirdly, genetic programming was used to generate CSG models. Initial tests were to evolve towards a target three-dimensional body. This proved successful but computationally expensive. Therefore, it was concluded that this method would have not be suitable for the automatic translation of B-Rep solid models to CSG models, an application for which this approach was hoped to be suited.

The conclusions drawn from these studies on general shape representations and various evolutionary algorithms, were that evolutionary algorithms along with some novel shape representations, did show some promise for allowing a designer to use shape optimisation to search for shape designs from a large, general set of possible shapes. However, a number of issues were identified which acted as obstacles to this. Briefly, these were the need to establish an effective shape representation, the need to be able to analyse the all of the shapes in the search space with sufficient accuracy, and the ability to search through these shapes effectively. It was argued that some steps could be made to addressing these issues if a common representation for geometry representation and analysis could be established. This would also offer opportunities to develop algorithms that could make use of the information on the shapes' current physical behaviour in order to modify the shape, and thus search the space of possible shapes more effectively.

(b) *To identify a computational framework which could provide an integrated representation of both component geometry and physical behaviour.*

Chain models, based on chains and cells from algebraic topology, were put forward as a possible common representation for both geometry and physical behaviour. With the use of Chain models, shape optimisation could be reformulated as a systematic transformation of Chain models specifying required function into Chain models representing both geometry and physical behaviour.

(c) *To determine whether a morphogenetic evolutionary algorithm, using the identified integrated representation of geometry and physical behaviour, shows any potential to increase the automation of the process of shape synthesis for engineering design.*

A novel shape optimisation technique was developed using the Chain model framework. This approach used an evolutionary algorithm along with a morphogenetic stage in which a cellular model of the shape was 'grown'. The cells were implemented in the Swarm agent-based modelling language and were free to behave independently based on the stress on them and their genes, which were evolved by the evolutionary algorithm. The generation of the shape could in this way make use of all of the information generated in the finite element analysis to influence the shape produced. Shapes were grown for three test cases: a bicycle frame, a cantilever beam and an arch. The approach proved successful and generated good quality shapes without intervention by the user to set up shape parameterisation. Therefore, it was concluded that is a promising method extending the use of computers to automatically synthesise geometry for engineering design.

In conclusion, the argument presented by this thesis, was that a framework for shape optimisation with a common representation for both geometry and physical behaviour would allow the development of novel and efficient new algorithms better suited to the semi-automatic generation of the geometry for a component given a certain desired behaviour. This was shown by investigations into aerodynamic and structural optimisation, which demonstrated a number of difficulties that could be alleviated by use of such a common representation. Chain models were identified as an appropriate representation. Finally, a morphogenetic approach to shape optimisation was developed, based on this representation, which showed the ability to search effectively through a large set of possible shapes, to produce high quality shapes for a number of structural shape optimisation problems.

References

[Abbot *et al.* 1945] Abbott, I.H.; Von Doenhoff, A.E & Stivers, L.Jr., (1945), ***Summary of Airfoil Data,*** NACA Report 824 NACA-ACR-L5C05 NACA-WR-L-560.

[Acton 1970] Acton, F.S., (1970), ***Numerical Methods That Work***, Mathematical Association of America, Washington, DC, 1990 corrected edition.

[Adeli & Cheng 1993] Adeli, H. & Cheng, N., (1993), '**Integrated Genetic Algorithms for Optimisation of Space Structures**', *Journal of Aerospace Engineering*, vol. 6(4), pp. 315–328.

[Adeli & Cheng 1994a] Adeli, H. & Cheng, N., (1994), '**Augmented Lagrangian Genetic Algorithm for Structural Optimization**', *Journal of Aerospace Engineering*, vol. 7(1), pp.104–118.

[Adeli & Cheng 1994b] Adeli, H. & Cheng, N., (1994), '**Concurrent Genetic Algorithms for Optimization of Large Structures**', *Journal of Aerospace Engineering*, vol. 7(3), pp.276–296.

[Agarwal 1994] Agarwal, P., (1994), '**The Cell Programming Language**', *Artificial Life*, vol. 29(1), pp. 37–77.

[Agarwal 1995] Agarwal, P., (1995), '**Cellular Segregation and Engulfment Simulations using the Cell Programming Language**', *Journal of Theoretical Biology*, vol. 176(1), September 1995, pp. 79–89.

[Agnelli *et al.* 2002] Agnelli, D.; Bollini, A. & Lombardi, L., (2002), '**Image Classification: an Evolutionary Approach**', *Pattern Recognition Letters*, vol. 23(1-3), January 2002, pp. 303–309.

[Alander 1994] Alander, J.T., (1994), ***Indexed Bibliography of Genetic Algorithms in Computer Aided Design***, Report 94-1-CAD, University of Vaasa, Department of Information Technology and Production Economics, ftp://ftp.uwasa.fi/cs/report94-1.

[Anderson & Venkatakrishnan 1999] Anderson, W.K. & Venkatakrishnan, V., (1999), '**Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation**', *Computers & Fluids*, vol. 28, pp. 443–480.

[Angeline 1995] Angeline, P.J. (1995), '**Morphogenic Evolutionary Computations: Introduction, Issues and Examples**', in *Proc. of Evolutionary Programming IV: The Fourth Annual Conference on Evolutionary Programming*, McDonnell, J. R.; Reynolds, R.G. & Fogel, D.G. (eds.), pp. 387–401, MIT Press, ISBN 0-262-13317-2.

[Angeline & Pollack 1992] Angeline, P.J. & Pollack, J.B., (1992), '**Evolutionary Induction of Subroutines**', in *Proc. of the 14th Annual Cognitive Science Conference*, pp. 236–241.

[Annicchiarico & Cerrolaza 1998] Annicchiarico, W. & Cerrolaza, M., (1998), '**Optimization of Finite Element Bidimensional Models: an Approach Based on Genetic Algorithms**', *Finite Elements in Analysis and Design*, vol. 29, pp. 231–257.

[Annicchiarico & Cerrolaza 2001] Annicchiarico, W. & Cerrolaza, M., (2001), '**Structural Shape Optimization 3D Finite-Element Models Based on Genetic Algorithms and Geometric Modeling**', *Finite Elements in Analysis and Design*, vol. 37, pp. 403–415.

[Ariana & Ta'asanb 1999] Ariana, E. & Ta'asanb, S., (1999), '**Analysis of the Hessian for Aerodynamic Optimization: Inviscid Flow**', *Computers & Fluids*, vol. 28, pp. 853–877.

[Bäck 1996] Bäck, T., (1996), ***Evolutionary Algorithms in Theory and Practice***, Oxford University Press, New York.

[Bäck *et al.* 1991] Bäck, T., Hoffmeister, F. & Schwefel, H.-P., (1991), '**A Survey of Evolution Strategies**', in *Proc. of the 4th International Conference on Genetic Algorithms*, Belew, R.K. & Booker, L.B. (eds.), Morgan Kaufmann Publishers Inc., San Francisco.

[Bangtsson *et al.* 2003] Bangtsson, E.; Noreland, D. & Berggren, M., '**Shape Optimization of an Acoustic Horn**', *Computer Methods in Applied Mechanics and Engineering*, vol. 192, pp. 1533–1571.

[Banichuk *et al.* 1995] Banichuk, N.V.; Barthold, F.J.; Falk, A. & Stein, E., (1995), '**Mesh Refinement for Shape Optimization**', *Structural Optimization*, vol. 9, pp. 46–51, Springer-Verlag.

[Barton 2002] Barton, A.C., (2002), ***Integrating Manufacturing Issues into Structural Optimization***, Ph.D. Thesis, University of Sydney, Sydney, Australia.

[Baumgartner & Mattheck 1994] Baumgartner, A. & Mattheck, C., (1994), '**A New Design of a Bicycle Frame: An Example for an Effective Layout Procedure Based on F.E.-Simulation of Biological Growth**', *Advances in Design Automation*, vol. 69(2), ASME.

[Beck & Parmee 1999] Beck, M.A. & Parmee, I.C., (1999), '**Design Exploration: Extending the Bounds of the Search Space**', in *Proc. of IEEE Congress on Evolutionary Computation*, Washington D.C., pp. 519–526.

[Bentley 1996] Bentley, P.J., (1996), ***Generic Evolutionary Design of Solid Objects using a Genetic Algorithm***, Ph.D. Thesis, University of Huddersfield, Huddersfield, UK.

[Bentley 1999] Bentley, P.J., (ed.), (1999), ***Evolutionary Design by Computers***, Morgan Kaufmann Publishers Inc., San Francisco, CA.

[Bentley 2000] Bentley, P.J., (2000), '**Exploring Component-Based Representations - The Secret of Creativity by Evolution?** ', in *Proc. of the Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM 2000)*, April 26th - 28th, 2000, University of Plymouth, UK.

[Bentley & Corne 2001a] Bentley, P.J. & Corne, D.W., (2001), '**An Introduction to Creative Evolutionary Systems**', in *Creative Evolutionary Systems*, Bentley, P.J. & Corne, D.W. (eds.), Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 1–75.

[Bentley & Corne 2001b] Bentley, P.J. & Corne, D.W., (2001), ***Creative Evolutionary Systems***, Morgan Kaufmann Publishers Inc., San Francisco, CA.

[Bentley & Kumar 1999] Bentley, P.J. & Kumar, S., (1999), '**Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem**', in *Genetic and Evolutionary Computation Conference (GECCO '99)*, July 14-17, 1999, Orlando, Florida USA, pp. 35–43, RN/99/2.

[Bentley & Wakefield 1996] Bentley, P.J. & Wakefield, J.P., (1996), '**The Evolution of Solid Object Designs using Genetic Algorithms**', in *Modern Heuristic Search Methods*, Rayward-Smith, V. (ed.), Chapter 12, John Wiley & Sons Inc., pp. 199–215.

[Bhavikatti & Ramakrishnan 1980] Bhavikatti, S.S. & Ramakrishnan, C.V., (1980), '**Optimum Shape Design of Rotating Disks**', *Computational Structures*, vol. 11, pp. 397–401.

[Boden 1991] Boden, M.A., (1991), ***The Creative Mind; Myths And Mechanisms,*** Basic Books, New York.

[Boden 1995] Boden, M.A., (1995), '**Modelling Creativity: Reply to Reviewers**', *Artificial Intelligence*, vol. 79(1), November 1995, pp. 161–182.

[Boden 1998] Boden, M.A., (1998), '**Creativity and Artificial Intelligence**', *Artificial Intelligence*, vol. 103(1-2), August 1998, pp. 347–356.

[Booth 1997] Booth, G. (1997), '**Gecko: A Continuous 2D World for Ecological Modeling**', *Artificial Life*, vol. 3, pp. 147–163.

[Botkin *et al.* 2002] Botkin, M.E; Wang, H-P.; Kim N.H. & Choi, K.K., (2002), '**Shape Optimization of Two-Dimensional Automotive Components using a Meshfree Method**', *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 4-6 September 2002, Atlanta, Georgia, AIAA 2002-5541.

[Bremicker *et al.* 1991] Bremicker M.; Chirehdast, M.; Kikuchi, N. & Papalambros, P.Y., (1991), '**Integrated Topology and Shape Optimization in Structural Design**', *Mechanical Structures and Machines*, vol. 19(4), pp. 551–587.

[Brent 1973] Brent, R., (1973), ***Algorithms for Minimisation without Derivatives***, Prentice-Hall.

[Broughton *et al.* 1999] Broughton, T.; Coates, P. & Jackson, H., (1999), '**Exploring Three-Dimensional Design Worlds using Lindenmeyer Systems and Genetic Programming**', in *Evolutionary Design by Computers*, Bentley, P. (ed.), Kaufmann.

[Bulman *et al.* 2001] Bulman, S.; Sienz, J. & Hinton, E., (2001), '**Comparisons Between Algorithms for Structural Topology Optimization using a Series of Benchmark Studies**', *Computers & Structures*, vol. 79(12), May 2001, pp. 1203–1218.

[Burgreen & Baysal 1994] Burgreen, G.W.& Baysal, O., (1994), '**Aerodynamic Shape Optimization Using Preconditioned Conjugate Gradient Methods**', *AIAA Journal*, vol. 32(11), Nov. 1994, pp. 2145–2152.

[Burgreen *et al.* 1994] Burgreen, G.W.; Baysal, O. & Eleshaky, M.E., (1994), '**Improving the Efficiency of Aerodynamic Shape Optimization**', *AIAA Journal*, vol. 32(1), pp. 69–76.

[Burguburu & le Pape 2003] Burguburu, S. & le Pape, A., (2003), '**Improved Aerodynamic Design of Turbomachinery Bladings by Numerical Optimization**', *Aerospace Science and Technology*, in press.

[Burkhart 1994] Burkhart, R., (1994), '**The Swarm Multi-Agent Simulation System**', *(OOPSLA) 1994 Workshop on 'The Object Engine'*, 7th September 1994.

[Canales *et al.* 1994] Canales, J.; Tárrago; J.A. & Hernández, A., (1994), '**An Adaptive Mesh Refinement Procedure for Shape Optimal Design**', *Advances in Engineering Software*, vol. 18(2), pp. 131–145.

[Cantu-Paz 1997] Cantu-Paz, E., (1997), ***A Survey on Parallel Genetic Algorithms***, ILLIGAL report 97003, University of Illinois at Urbana-Champain.

[Cappello & Mancuso 2003] Cappello, F. & Mancuso, A., (2003), '**A Genetic Algorithm for Combined Topology and Shape Optimisations**', *Computer-Aided Design*, in press.

[Cartwright & Harris 1993] Cartwright, H.M. & Harris, S.P., (1993), '**The Application of the Genetic Algorithm to Two-Dimensional Strings: The Source Apportionment Problem**', in *Proc. of the Fifth International Conference on Genetic Algorithms*, Forrest, S. (ed.), San Mateo, Morgan Kaufmann.

[Cerrolaza *et al.* 2000] Cerrolaza, M.; Annicchiarico, W. & Martinez, M., (2000), **'Optimization of 2D Boundary Element Models Using β-splines and Genetic Algorithms'**, *Engineering Analysis with Boundary Elements*, vol. 24(5), May 2000, pp. 427–440.

[Chang & Tang 2001] Chang, K.-H. & Tang, P.-S., (2001), **'Integration of Design and Manufacturing for Structural Shape Optimization'**, *Advances in Engineering Software*, vol. 32(7), July 2001, pp. 555–567.

[Chapman et al. 1994] Chapman, C.D.; Saitou, K. & Jakiela, M.J., (1994), **'Genetic Algorithms as an Approach to Configuration and Topology Design'**, *Journal of Mechanical Design*, vol. 116, pp. 1005–1012, ASME.

[Chen 2001] Chen, S-Y., (2001), **'An Approach for Impact Structure Optimization using the Robust Genetic Algorithm'**, *Finite Elements in Analysis and Design*, vol. 37, pp. 431–446.

[Chen *et al.* 2002] Chen, Y.-M.; Bhaskar, A. & Keane, A. J., **'A Parallel Nodal-based Evolutionary Structural Optimization Algorithm'**, *Structural & Multidisciplinary Optimization*, vol. 23, pp. 241–251.

[Chen & Tsai 1993] Chen, J.L. & Tsai, W.C., (1993), **'Shape Optimization by Using Simulated Biological Growth Approaches'**, *AIAA Journal*, vol. 31(11), pp. 2143–2147.

[Coello 1999] Coello, C.A.C., (1999), **'Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques'**, *Knowledge and Information Systems*, vol. 1(3), pp. 269–308.

[Coello & Christiansen 2000] Coello, C.A. & Christiansen, A.D., (2000), **'Multiobjective Optimization of Trusses using Genetic Algorithms'**, *Computers & Structures*, vol. 75(6), May 2000, pp. 647–660.

[Corney 1997] Corney, J., (1997), ***3D Modeling with the ACIS Kernel and Toolkit***, Wiley, ISBN 0471965359.

[Corney & Lim 2001] Corney, J. & Lim, T., (2001), ***3D Modeling with ACIS***, Saxe-Coburg Publications, ISBN 1-874672-14-8.

[Cramer 1985] Cramer, N.L., (1985), '**A Representation for the Adaptive Generation of Simple Sequential Programs**', in *Proc. of an International Conference on Genetic Algorithms and the Applications*, Grefenstette, J.J. (ed.), CMU.

[Cvetkovic 2000] Cvetkovic, D., (2000), ***Evolutionary Multi-Objective Decision Support Systems for Conceptual Design***, Ph.D. Thesis, School of Computing, University of Plymouth.

[Cvetkovic & Parmee 1999a] Cvetkovic, D. & Parmee, I.C., (1999), '**Genetic Algorithm-based Multi-objective Optimisation and Conceptual Engineering Design**', in *Proc. Congress on Evolutionary Computation - CEC99*, Washington D.C., USA, 1999, vol. 1, pp. 29–36, IEEE.

[Cvetkovic & Parmee 1999b] Cvetkovic, D. & Parmee, I.C., (1999), '**Genetic Algorithms based Systems for Conceptual Engineering Design**', in *Proc. of the 12th International Conference on Engineering Design ICED'99*, Lindemann, U.; Birkhofer, H.; Meerkamm, H. & Vajna, S. (eds.), München, Germany, August 1999, TU München, vol. 2, pp. 1035–1038.

[Davis 1991] Davis, L. (ed.), (1991), ***Handbook of Genetic Algorithms***, Van Nostrand Reinhold, New York.

[Dawkins 1986] Dawkins, R., (1986), ***The Blind Watchmaker***, Penguin, London.

[Dawkins 1987] Dawkins, R., (1989). **'The Evolution of Evolvability'**, in *Proc. Artificial Life IV*, Langton, C. (ed.), Addison-Wesley.

[Deb 1997] Deb, K., (1997), **'GeneAS: a Robust Optimal Design Technique for Mechanical Component Design'**, in *Evolutionary Algorithms in Engineering Applications*, Springer, Berlin, pp. 497–514.

[Deb & Goyal 1997] Deb, K. & Goyal, M, (1997), **'Optimizing Engineering Designs using a Combined Genetic Search'**, in *Proc. Seventh International Conference on Genetic Algorithms*, Back, T. (ed.), pp. 521–528.

[Deb & Goyal 1998] Deb, K. & Goyal, M, (1998), **'A Robust Optimization Procedure for Mechanical Component Design Based on Genetic Adaptive Search'**, *Journal of Mechanical Design*, vol. 120(2), pp. 162–164, ASME.

[Desai & Kundu 2001] Desai, C. S. & Kundu, T., (2001), ***Introductory Finite Element Method***, CRC Press, Boca Raton, FL, ISBN 0849302439.

[Dibakar & Mruthyunjaya 1999] Dibakar, S. & Mruthyunjaya, T.S., (1999), **'Synthesis of Workspaces of Planar Manipulators with Arbitrary Topology using Shape Representation and Simulated Annealing'**, *Mechanism and Machine Theory*, vol. 34(3), April 1999, pp. 391–420.

[Doorly 1995] Doorly, D. J., (1995), **'Parallel Genetic Algorithms for Optimization in CFD'**, in *Genetic Algorithms in Engineering and Computer Science*, Winter, G.; Périaux, J.; Galan M. & Cuesta, P. (eds.), Wiley.

[Doorly & Peiró 1997] Doorly, D.J. & Peiró, J., (1997), **'Supervised Parallel Genetic Algorithms in Aerodynamic Optimisation'**, in *Proc. 13th AIAA CFD Conference*, June 30-July 2, Snowmass Co., U.S.A., AIAA 97-1852.

[Doorly *et al.* 1996a] Doorly, D.J.; Peiró, J. & Oesterle, J-P., (1996), '**Optimisation of Aerodynamic and Coupled Aerodynamic-Structural Design Using Parallel Genetic Algorithms**', in *Proc. 6th AIAA/NASA/USAF Multidisciplinary Analysis & Optimization Symposium*, September 4-6, Bellevue, Seattle, WA, U.S.A., AIAA 96-4027.

[Doorly *et al.* 1996b] Doorly, D.J.; Peiró, J.; Kuan, T. & Oesterle, J-P., (1996), '**Optimisation of Airfoils using Parallel Genetic Algorithms**', in *Proc. of the 15th AIAA International Conference on Numerical Methods in Fluid Mechanics*, June 24-28, Monterey, CA, U.S.A..

[Duvigneau & Visonneau 2001] Duvigneau, R. & Visonneau M., (2001), ***Shape Optimization of Incompressible and Turbulent Flows using the Simplex Method***, AIAA 2001–2533, Sept. 2001, Anaheim, CA.

[Eby *et al.* 1999a] Eby, D.; Averill, R.C.; Goodman, E.D. & Punch, W.F., (1999), '**The Optimization of Flywheels Using an Injection Island Genetic Algorithm**', in *Evolutionary Design by Computers*, Bentley, P. (ed.), Morgan Kaufmann, San Francisco, 1999, pp.167–190.

[Eby *et al.* 1999b] Eby, D.; Averill, R.C.; Punch, W.F. & Goodman, E.D., (1999), '**Optimal Design of Flywheels using an Injection Island Genetic Algorithm**', *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, vol. 13(5), November 1999, pp. 327–340.

[Eggenberger 1996] Eggenberger, P., (1996), '**Cell Interactions as a Control Tool of Developmental Processes for Evolutionary Robotics**', *From Animals to Animats 4*, Maes, P. *et al.* (eds.), Cambridge, MA, MIT Press.

[Egli & Stewart 2000] Egli, R. & Stewart, N.F., (2000), '**A Framework for System Specification using Chains on Cell Complexes**', *Computer-Aided Design*, vol. 32(7), June 2000, pp. 447–459.

[El-Beltagy & Keane 2001] El-Beltagy, M.A. & Keane, A.J., (2001), '**Evolutionary Optimization for Computationally Expensive Problems using Gaussian Processes**', in *Proc. International Conference on Artificial Intelligence IC-AI'2001*, Arabnia, H., (ed.), pp. 708–714, CSREA Press, Las Vegas.

[Eleshaky & Baysal 1991] Eleshaky, M.E. & Baysal, O., (1991), '**Airfoil Shape Optimization using Sensitivity Analysis on Viscous Flow Equations**', *Multidisciplinary Applications of Computational Fluid Dynamics*, FED-vol. 129, ASME.

[Ellman *et al.* 1993] Ellman, T.; Keane, J. & Schwabacher, M., (1993), '**Intelligent Model Selection for Hillclimbing Search in Computer-Aided Design**', in *Proc. of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C.

[Ferziger & Peric 1996] Ferziger, J.H. & Peric, M., (1996), ***Computational methods for Fluid Dynamics***, Springer.

[Filipiak 1996] Filipiak, M. (1996), ***Technology Watch Report on Mesh Generation***, Edinburgh Parallel Computing Centre, University of Edinburgh, UK.

[Fillipone 1995] Fillipone, A., (1995), '**Airfoil Inverse Design and Optimization by Means of Viscous-Inviscid Techniques**', *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 56, pp. 123–136.

[Fisher 1995] Fisher, K.A., (1995), '**The Application of Genetic Algorithms to Optimising the Design of an Engine Block for Low Noise**', in *Proc. of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 12-14 September 1995, pp. 18–22, Conference Publication No. 414, IEE.

[Fleischer 1995] Fleischer, K., (1995), ***A Multiple-Mechanism Developmental Model for Defining Self-Organizing Geometric Structures***, Ph.D. Thesis, California Institute of Technology.

[Fletcher & Reeves 1964] Fletcher, R. & Reeves, C., (1964), '**Function Minimisation by Conjugate Gradients**', *The Computer Journal*, vol. 7, pp. 163–168.

[Fogel 1995] Fogel D.B., (1995), ***Evolutionary Computation: Toward a New Philosophy of Machine Intelligence***, IEEE Press, Piscataway, NJ.

[Fogel *et al.* 1966] Fogel, L.J., Owens, A.J. & Walsh, M.J., (1966), ***Artificial Intelligence through Simulated Evolution***, Wiley, New York.

[Fugsland & Madsen 1999] Fugsland, P. & Madsen, H.A., (1999), '**Optimization Method for Wind Turbine Rotors**', *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 80, pp. 191–206.

[Funes & Pollack 1999] Funes, P. & Pollack, J., (1999), '**The Evolution of Buildable Objects**', in *Evolutionary Design by Computers*, Bentley, P. (ed.), Morgan Kaufmann, San Francisco.

[Gage *et al.* 1995] Gage, P.; Kroo, I. & Sobieski, I., (1995), '**A Variable-Complexity Genetic Algorithm for Topological Design**', *AIAA Journal*, vol. 33(11), pp. 2212–2217.

[de Garis 1994] de Garis, H., (1994), '**An Artificial Brain**', *New Generation Computing*, vol. 12(2), Springer Verlag.

[Gelsey *et al.* 1998] Gelsey, A., Schwabacher, M. & Smith, D., (1998), '**Using Modeling Knowledge to Guide Design Space Search**', *Artificial Intelligence*, vol. 101 (1-2), pp. 35–62.

[Gen & Cheng 1997] Gen, M. & Cheng, R., (1997), ***Genetic Algorithms and Engineering Design***, Wiley Series in Engineering and Automation, John Wiley & Sons Inc., New York.

[Gere & Timoshenko 1984] Gere, J.M. & Timoshenko, S.P., (1984), ***Mechanics of Materials***, 2nd Edition, Brooks/Cole Engineering Division.

[Gero 1990] Gero, J.S., (1990), '**Design Prototypes: a Knowledge Representation Schema for Design**', *AI Magazine*, vol. 11(4), pp. 26–36.

[Gero 1996] Gero, J.S., (1996), '**Computers and Creative Design**', *The Global Design Studio*, Tan, M. & Teh, R. (eds.), National University of Singapore, pp. 11–19.

[Gero 1998] Gero, J. S., (1998), '**Adaptive Systems in Designing: New Analogies from Genetics and Developmental Biology**', in *Adaptive Computing in Design and Manufacture*, Parmee, I. (ed.), Springer, London, pp. 3–12.

[Gero & Kazakov 1996] Gero, J.S. & Kazakov, V., (1996), '**Evolving Building Blocks for Design using Genetic Engineering: A Formal Approach**', in *Advances in Formal Design Methods for CAD*, Gero, J.S. (ed.), Chapman and Hall, London, pp. 31–50.

[Gero & Kazakov 2000] Gero, J.S. & Kazakov, V., (2000), '**Adaptive Enlargement of State Spaces in Evolutionary Designing**', *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, vol. 14(1), pp. 31–38.

[Giannakoglou 2002] Giannakoglou, K.C., (2002), '**Design of Optimal Aerodynamic Shapes using Stochastic Optimization Methods and Computational Intelligence**', *Progress in Aerospace Sciences*, vol. 38, pp. 43–76.

[Gilbert 1994] Gilbert, S.F., (1994), ***Developmental Biology***, Sineauer Associates Inc., ISBN 0-87893-249-6.

[Goldberg 1989] Goldberg, D.E., (1989), ***Genetic Algorithms in Search, Optimization and Machine Learning***, Addison-Wesley, New York, ISBN 0-201-15767-5.

[Goldberg 1999] Goldberg, D.E., (1999), '**The Race, the Hurdle, and the Sweet Spot: Lessons from Genetic Algorithms For the Automation of Design Innovation and Creativity**', in *Evolutionary Design by Computers*, Bentley, P. (ed.), Morgan Kaufmann, San Francisco.

[Grindeanu *et al.* 2002] Grindeanu, I.; Kim, N.H.; Choi, K.K. & Chen, J.S., (2002), '**CAD-Based Shape Optimization Using a Meshfree Method**', *Concurrent Engineering: Research and Applications*, vol. 10, pp. 55–66.

[Haase 1995] Haase, K.B., (1995), '**Too Many Ideas, Just One Word: a Review of Margaret Boden's the Creative Mind: Myths and Mechanisms: (Basic Books, New York, 1991); 303 pages**', *Artificial Intelligence*, vol. 79(1), November 1995, pp. 69–82.

[Haftka & Grandhi 1986] Haftka, R.T. & Grandhi, R.V., (1986), '**Structural Shape Optimization - A Survey**', *Journal of Computer Methods in Applied Mechanics and Engineering,* vol. 57(1), August 1986, pp. 91–106.

[Hajela 1998] Hajela, P., (1998), ***Implications of Artificial Life Simulations in Structural Analysis and Design***, invited paper at the AIAA/ASME/ASCE/AHS SDM meeting, April 1998, Long Beach, California, AIAA 98-1775.

[Hajela & Kim 2001] Hajela, P. & Kim, B., '**Research Paper: On the Use of Energy Minimization for CA Based Analysis in Elasticity**', *Structural and Multidisciplinary Optimisation*, vol. 23(1), pp. 24–33.

[Hasançebi & Erbatur 2002] Hasançebi, O. & Erbatur, F., (2002), **'Layout Optimisation of Trusses using Simulated Annealing'**, *Advances in Engineering Software*, vol. 33(7-10), July-October 2002, pp. 681–696.

[Hearn & Baker 1994] Hearn, D. & Baker, M.P., (1994), ***Computer Graphics***, Prentice Hall, ISBN 0-13-159690-X.

[Hicks *et al.* 1974] Hicks, R.M.; Murman, E.M. & Vanderplaats, G.N., (1974), ***An Assessment of Airfoil Design by Numerical Optimization***, NASA TM X-3092.

[Hoffmann & Kim 2001] Hoffmann, C. M. & Kim, K.-J., (2001), **'Towards Valid Parametric CAD Models'**, *Computer-Aided Design*, vol. 33(1), January 2001, pp. 81–90.

[Holland 1975] Holland, J.H., (1975), ***Adaptation in Natural and Artificial Systems***, The University of Michigan Press, Ann Arbor.

[Hong & Bhamidimarri 2003] Hong, Y.-S. & Bhamidimarri, R., (2003), **'Evolutionary Self-organising Modelling of a Municipal Wastewater Treatment Plant'**, *Water Research*, vol. 37(6), March 2003, pp. 1199–1212.

[Hooke & Jeeves 1961] Hooke, R. & Jeeves, T., (1961), **'Direct Search Solutions to Numerical Problems'**, *Journal of the Association of Computing Machinery*, vol. 8, pp. 212–229.

[Hornby 2003] Hornby, G.S., (2003), ***Generative Representations for Evolutionary Design Automation***, Ph.D. Dissertation, Brandeis University, Dept. of Computer Science.

[Hornby & Pollack 2001] Hornby, G.S. & Pollack, J.B., (2001), **'The Advantages of Generative Grammatical Encodings for Physical Design'**, in *Proc. 2001 Congress on Evolutionary Computation CEC2001.*

[Hsu 1994] Hsu, Y-H, (1994), '**A Review of Structural Shape Optimization**', *Computers in Industry*, vol. 25(1), pp. 3–13.

[Hsu & Liu 2000] Hsu, W. & Liu, B., (2000), '**Conceptual Design: Issues and Challenges**', *Computer-Aided Design*, vol. 32(14), December 2000, pp. 849–850.

[Hsu & Wonn 1998] Hsu, W. & Woon, I.M.Y., (1998), '**Current Research in the Conceptual Design of Mechanical Products**', *Computer-Aided Design*, vol. 30(5), April 1998, pp. 377–389.

[Hu & Goodman 2002] Hu, J. & Goodman, E.D., (2002), '**Hierarchical Fair Competition Model for Parallel Evolutionary Algorithms**', in *Proc. Congress on Evolutionary Computation CEC 2002*, IEEE World Congress on Computational Intelligence, Honolulu, Hawaii, May 2002.

[Hu *et al.* 2003] Hu, J.; Goodman, E. D.; Seo, K.; Fan, Z. & Rosenberg, R. C., (2003), '**HFC: A Continuing EA Framework for Scalable Evolutionary Synthesis**', in *Proc. of the 2003 AAAI Spring Symposium - Computational Synthesis: From Basic Building Blocks to High Level Functionality*, March 24-26 2003, Stanford, California, pp. 106–113.

[Husbands *et al.* 1996] Husbands, P.; Jeremy, G.; M^{c}Ilhagga, M. & Ives, R., (1996), ***'Two Applications of Genetic Algorithms to Component Design'***, *Selected Papers: AISB Workshop on Evolutionary Computing,* Fogarty, T.C. (ed.), Lecture Notes in Computer Science, No. 1143, pp. 50–61, Springer Verlag.

[Jakobi 1995] Jakobi, N., (1995), ***Harnessing Morphogenesis***, Presented at The International Conference on Information Processing in Cells and Tissues, Liverpool, UK.

[Jakobi 1996] Jakobi, N., (1996), **'Encoding Scheme Issues for Open-Ended Artificial Evolution'**, *Parallel Problem Solving from Nature (PPSN) IV*, Voigt, H-M; Ebeling, W.; Rechenberg, I. & Schwefel, H-P (eds.), pp. 52–61, Springer, Berlin.

[Jameson *et al.* 1998] Jameson, A.; Alonso, J.J.; Reuther, J.J.; Martinelli, L. & Vassberg, J. C., (1998), ***Aerodynamic Shape Optimization Techniques Based On Control Theory***, AIAA 98–2538.

[Jameson 2001] Jameson, A., (2001), **'A Perspective on Computational Algorithms for Aerodynamic Analysis and Design'**, *Progress in Aerospace Sciences*, vol. 37(2), February 2001, pp. 197–243.

[Janikow & Michalewicz 1991] Janikow, C. & Michalewicz, Z., (1991), **'An Experiment Comparison of Binary and Floating Point Representations in Genetic Algorithms'**, in *Proc. 4th International Conference of Genetic Algorithms*, Belew, R.K. & Booker, L.B. (eds.), Morgan Kaufmann Publishers, San Mateo, CA.

[Jenkins 1992] Jenkins, W.M., (1992), **'Plane Frame Optimum Design Environment Based on Genetic Algorithm'**, *Journal of Structural Engineering*, vol. 118(11), pp. 3103–3112, ASCE.

[Jenkins 1997] Jenkins, W.M., (1997), **'On the Application of Natural Algorithms to Structural Design Optimization'**, *Engineering Structures*, vol. 19(4), pp.302–308.

[Johnson 1998] Johnson, P., (1998), ***Adaptive Agents versus Rational Actors: Social Science Implications***, Annual Meeting of the American Political Science Association, 3-6 September 1998, Marriott Copley Place and Sheraton Boston Hotel and TowerBoston.

[Joines & Houck 1994] Joines, J. & Houck, C., (1994), '**On the Use of Non-Stationary Penalty Functions to Solve Constrained Optimization Problems with Genetic Algorithms**', in *Proc. IEEE International Symposium Evolutionary Computation*, Orlando, Fl, pp. 579–584.

[Kasper 1993] Kasper, M., (1993), '**Optimization of FEM Models by Stochastic Methods**', *International Journal of Applied Electromagnetics in Materials*, vol. 4, pp. 107–113.

[Keane 1994] Keane, A.J., (1994), '**Experiences with Optimizers in Structural Design**', in *Proc. of the Conference on Adaptive Computing in Engineering Design and Control 94*, Parmee, I.C. (ed.), Plymouth, U.K., Sept. 1994, pp. 14–27.

[Kirkpatrick *et al.* 1983] Kirkpatrick, S.; Gelatt, C.D. Jr. & Vecchi, M.P., (1983), '**Optimization by Simulated Annealing**', *Science*, vol. 220, pp. 671–680.

[Kita & Toyoda 2000] Kita, E. & Toyoda, T., (2000), '**Structural Design Using Cellular Automata**', *Structural & Multidisciplinary Optimization*, vol. 19(1), pp. 64–73.

[Kodiyalam *et al.* 1992] Kodiyalam, S.; Kumar, V. & Finnigan, P.M., (1992), '**Constructive Solid Geometry Approach to Three-Dimensional Structural Shape Optimization**', *AIAA Journal*, vol. 30(5), May 1992, pp. 1408–1415.

[Kodiyalam & Parthasarathy 1992] Kodiyalam, S. & Parthasarathy, V.N., (1992), '**Optimized/Adapted Finite Elements for Structural Shape Optimization**', *Finite Elements in Analysis and Design*, vol. 12(1), September 1992, pp. 1–11.

[Kodiyalam & Thanedar 1993] Kodiyalam, S. & Thanedar, P.B., (1993), '**Some Practical Aspects of Shape Optimization and its Influence on Intermediate Mesh Refinement**', *Finite Elements in Analysis and Design*, vol. 15(2), December 1993, pp. 125–133.

[Kohli & Carey 1993] Kohli, H.S. & Carey, G.F., (1993), **'Shape Optimization using Adaptive Shape Refinement'**, *Journal for Numerical Methods in Engineering*, vol. 36, pp. 2435–2451.

[Koza 1990] Koza, J., (1990), ***Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems***, Technical Report STAN-CS-90-1314, Department of Computer Science, Stanford University.

[Koza 1992] Koza, J., (1992), ***Genetic Programming: On the Programming of Computers by Means of Natural Selection***, MIT Press.

[Koza 1994] Koza, J., (1994), ***Genetic Programming II: Automatic Discovery of Reusable Programs***, MIT Press.

[Koza *et al.* 1999] Koza, J., Bennett, F.H, Andre, D, & Keane, M.A, (1999), ***Genetic Programming III: Darwinian Invention and Problem Solving***, Morgan Kaufmann.

[Kreft *et al.* 1998] Kreft, J-U.; Booth, G. & Wimpenny, J.W.T., (1998), **'BacSim: a Simulator for Individual-based Modelling of Bacterial Colony Growth'**, *Microbiology*, vol. 144, pp. 3275–3287.

[Kristensen & Madsen 1976] Kristensen, E.S. & Madsen, N.F., (1976), **'On the Optimum Shape of Fillets in Plates Subjected to Multiple In-Plane Loading Cases'**, *International Journal of Numerical Methods in Engineering*, vol. 10, pp. 1007–1009.

[Krothapalli & Deshmukh 1997] Krothapalli, N.K.C. & Deshmukh, A.V., (1997), **'Effects of Negotiation Mechanisms on Performance of Agent Based Manufacturing Systems'**, in *Proc. of the Seventh International Conference on Flexible Automation and Intelligent Manufacturing*, pp. 704–717.

[Kuethe & Chow 1986] Kuethe, A.M. & Chow, C.-Y., (1986), ***Foundations of Aerodynamics,*** Fourth Edition, Wiley, ISBN 0-471-85954-0.

[Kumar & Bentley 2003a] Kumar, S. & Bentley, P.J., (2003), '**Biologically Inspired Evolutionary Development**', in *Proc. International Conference on Evolvable Systems: from biology to hardware (ICES 2003)*, Trondheim, Norway.

[Kumar & Bentley 2003b] Kumar, S. & Bentley, P.J., (2003), '**Computational Embryology: Past, Present and Future**', in *Advances in Evolutionary Computing, Theory and Applications*, Ghosh & Tsutsui (eds.), Springer, pp. 461–478.

[Kumar & Bentley 2003c] Kumar, S. & Bentley, P.J., (2003), '**Mechanisms of Oriented Cell Division in Computational Development**', in *Proc. First Australian Conference on Artificial Life (ACAL 2003)*, Canberra, Australia.

[Langham & Grant 1999] Langham, A.E. & Grant, P.W., (1999), '**Evolving Rules for a Self-organizing Finite Element Mesh Generation Algorithm**', in *Proc. 1999 Congress on Evolutionary Computation*, Washington D.C., USA, July 1999, pp. 161–168, IEEE Computer Science.

[Leite & Topping 1998] Leite, J.P.B. & Topping, B.H.V., (1999), '**Improved Genetic Operators for Structural Engineering Optimisation**', *Advances in Engineering Software*, vol. 29(7-9), pp. 529–562.

[Leite & Topping 1999] Leite, J.P.B. & Topping, B.H.V., (1999), '**Parallel Simulated Annealing for Structural Optimization**', *Computers & Structures*, vol. 73(1-5), October 1999, pp. 545–564.

[LeRiche *et al.* 1995] Le Riche, R.G.; Knopf-Lenoir, C. & Haftka, R.T., (1995), '**A Segregated Genetic Algorithm for Constrained Structural Optimization**', in *Proc. of the Sixth International Conference on Genetic Algorithms*, Eshelman, L. (ed.), pp. 558–565, Morgan Kaufmann, San Francisco.

[Lesieutre *et al.* 1998] Lesieutre, D.; Dillenius, M. & Lesieutre T., (1998), '**Multidisciplinary Design Optimization of Missile Configurations and Fin Planforms for Improved Performance**', in *Proc. 7th Symposium on Multidisciplinary Analysis and Optimization*, September 2–4 1998, St. Louis, AIAA 98-4890.

[Lewis 1997] Lewis, R.M., (1997), ***A Nonlinear Programming Perspective on Sensitivity Calculations for Systems Governed by State Equations***, NASA CR-201659, ICASE Report No. 97-12, February 1997, pp. 37.

[Lewis *et al.* 2000] Lewis, R.M.; Torczon, V. & Trosset, M.W., (2000), '**Direct Search Methods: Then and Now**', Journal of Computational and Applied Mathematics, vol. 124(1-2), pp. 191–207.

[Lin & Chen 2000] Lin, C-Y. & Chen, W-T., (2000), '**Stochastic Multistage Algorithms for Multimodal Structural Optimization**', *Computers & Structures*, vol.74(2), January 2000, pp. 233–241.

[Liu & Batill 2000] Liu, W. & Batill, S.M., (2000), ***Gradient-Enhanced Neural Network Response Surface Approximations***, AIAA Multidisciplinary Analysis and Optimization Conference and Exhibit, Long Beach, California, September 2000, AIAA 2000-4923.

[Lu & Chen 2002] Lu, H. & Chen, J. S., (2002), '**Adaptive Meshfree Particle Method**', in *Lecture Notes in Computational Science and Engineering*, vol. 26, pp. 251–267.

[Luna & Perrone 2001] Luna, F. & Perrone, A., (2001), '**Agent-Based Methods in Economics and Finance: Simulations in Swarm**', in *Advances in Computational Economics*, vol. 17, Amman, H. & Nagurney, A (eds.), Kluwer Academic Publishers, ISBN 0-7923-7419-3.

[Luna & Stefansson 2000] Luna, F. & Stefannson, B., (2000), '**Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming**', in *Advances in Computational Economics*, vol. 14, Amman, H. & Nagurney, A. (eds.), Kluwer Academic Publishers, ISBN 0-7923-8665-5.

[Lustig 1995] Lustig, R., (1995), '**The Creative Mind: Myths and Mechanisms: Margaret Boden, (Basic Books, New York, 1991); 303 pages**', *Artificial Intelligence*, vol. 79(1), November 1995, pp. 83–96.

[M^{c}Ilhagga *et al.* 1996] M^{c}Ilhagga, M.; Husbands, P. & Ives, R., (1996), '**A Comparison of Optimisation Techniques on a Wingbox Optimisation Problem**', in *Parallel Problem Solving from Nature (PPSN) IV*, Voigt, H-M; Ebeling, W.; Rechenberg, I. & Schwefel, H-P (eds.), Springer, Berlin.

[M^{c}Neill *et al.* 1998] McNeill, T.; Gero, J.S. & Warren, J., (1998). '**Understanding Conceptual Electronic Design using Protocol Analysis**', *Research in Engineering Design*, vol. 10, pp. 129–140.

[Mackerle 2003] Mackerle, J., (2003), '**Topology and Shape Optimization of Structures using FEM and BEM: A Bibliography (1999–2001)**', *Finite Elements in Analysis and Design*, vol. 39(3), January 2003, pp. 243–253.

[Mäkinen *et al.* 1999] Mäkinen; R.A.E., Periaux, J. & Toivanen, J., (1999), '**Multidisciplinary Shape Optimization in Aerodynamics and Electromagnetics using Genetic Algorithms**', *International Journal for Numerical Methods in Fluids*, vol. 30, pp. 149–159.

[Mäntylä 1988] Mäntylä, M., (1988), ***An Introduction to Solid Modeling***, Computer Science Press, Maryland, U.S.A..

[Marco & Lanteri 2000] Marco, N. & Lanteri, S., (2000), '**A Two-level Parallelization Strategy for Genetic Algorithms Applied to Optimum Shape Design**', *Parallel Computing*, vol. 26(4), March 2000, pp. 377–397.

[Mattheck *et al.* 1994] Mattheck, C.; Baumgartner, A. & Walther, F., (1994), '**Optimization Procedures by use of the Finite Element Method**', *Engineering Systems Design and Analysis*, vol. 64(4), ASME.

[Meric 1999] Meric, R.A., (1999), '**Boundary Elements and Optimization for Linearized Compressible Flows Around Immersed Airfoils**', *Engineering Analysis with Boundary Elements*, vol. 23, pp. 591–596.

[Metropolis *et al.* 1953] Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H. & Teller, E., (1953), '**Equation of State Calculations by Fast Computing Machines**', *Journal of Chem. Phys.*, vol. 21(6), pp. 1087–1092.

[Michalewicz 1992] Michalewicz, Z., (1992), ***Genetic Algorithms + Data Structures = Evolution Programs***, Springer Verlag.

[Michalewicz & Fogel 2000] Michalewicz, Z. & Fogel, D.B., (2000), ***How to Solve It: Modern Heuristics***, Springer-Verlag, Berlin, ISBN 3-540-66061-5.

[Michalewicz & Schoenauer 1996] Michalewicz, Z. & Schoenauer, M., (1996), '**Evolutionary Algorithms for Constrained Parameter Optimization Problems**', *Evolutionary Computation*, vol. 4(1), pp.1–32.

[Minar *et al.* 1996] Minar, N.; Burkhart, R.; Langton, C. & Askenazi, M., (1996), ***The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations,*** Santa Fe Institute Working Paper 96-06-042.

[Nelder & Mead 1965] Nelder, J. & Mead, R., (1965), '**A Simplex Method for Function Minimisation**', *The Computer Journal*, vol. 7, pp. 308–313.

[Nemec & Zingg 2001] Nemec, M. & Zingg, D.W., (2001), '**Towards Efficient Aerodynamic Shape Optimization Based on the Navier–Stokes Equations**', in *Proc. 15th AIAA Computational Fluid Dynamics Conference*, June 11–14 2001, Anaheim, CA, AIAA 2001–2532.

[Novruzi & Roche 1995] Novruzi, A. & Roche, J.R., (1995), ***Second Order Derivatives, Newton Method, Application to Shape Optimization***, Rapport de Recherche RR2555, Institute National De Recherche en Informatique en Automatique, France, ISSN 0249-6399.

[Nowostawski & Poli 1999] Nowostawski, M. & Poli R., (1999), '**Parallel Genetic Algorithm Taxonomy**', in *Proc. of the Third International Conference on Knowledge-based Intelligent Information Engineering Systems*, KES'99.

[Obayashi & Takanashi 1995] Obayashi, S. & Takanashi, S., (1995), '**Genetic Algorithm for Aerodynamic Inverse Optimization Problems**', in *Proc. of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 12-14 September 1995, pp. 7–12, Conference Publication No. 414, IEE.

[Otto *et al.* 1996] Otto, J.C.; Landman, D. & Paters, A.T., (1996), ***A Surrogate Approach To the Experimental Optimization of Multi-Element Airfoils***, AIAA96-4138-CP.

[Oyamaa *et al.* 2001] Oyamaa, A.; Obayashi, S. & Nakamurac, T., (2001), '**Real-Coded Adaptive Range Genetic Algorithm Applied to Transonic Wing Optimization**', *Applied Soft Computing*, vol. 1, pp. 179–187.

[Palmer 1995] Palmer, R.S., (1995), '**Chain Models and Finite Element Analysis: An Executable Chains Formulation of Plane Stress**', *Computer-Aided Geometric Design*, vol. 12(7), pp. 733–770.

[Palmer & Shapiro 1994] Palmer, R.S. & Shapiro, V., (1994), **'Chain Models of Physical Behavior for Engineering Analysis and Design'**, *Research in Engineering Design*, vol. 5(3).

[Papalambros 2002] Papalambros, P.Y., (2002), **'The Optimization Paradigm in Engineering Design: Promises and Challenges'**, *Computer-Aided Design*, vol. 34, pp. 939–951.

[Parmee 1993] Parmee, I.C., (1993), **'The Concrete Arch Dam - An Evolutionary Model of the Design Process'**, in *Proc. of the International Conference on Neural Nets & Genetic Algorithms*, Innsbruck, Austria, Springer-Verlag Wien.

[Parmee 1996] Parmee, I.C., (1996), **'Towards an Optimal Engineering Design Process using Appropriate Adaptive Search Strategies'**, *Journal of Engineering Design*, vol. 7(4), December, pp 341–362.

[Parmee 2002] Parmee, I.C., (2002), **'Evolutionary Computing Strategies for Preliminary Design Search and Exploration'**, in *Proc. US United Engineering Foundation's 'Optimisation in Industry' Conference*, Tuscany, Italy, 2001; Springer-Verlag, London.

[Parmee *et al.* 1993] Parmee, I.C.; Denham, M.J, & Roberts, A., (1993), **'Evolutionary Engineering Design using the Genetic Algorithm'**, in *Proc. International Conference on Engineering Design '93*, August 17-19 1993, The Hague.

[Parmee *et al.* 2001] Parmee, I.C.; Cvetkovic, D.; Bonham, C. & Packham, I., (2001), **'Introducing Interactive Evolutionary Systems for Ill-defined, Multi-objective Design Environments'**, *Journal of Advances in Engineering Software*, vol. 32(6), pp. 429–441, Elsevier.

[Pedersen & Laursen 1982] Pedersen, P. & Laursen, C.L., (1982), **'Design for Minimum Stress Concentration by Finite Elements and Linear Programming'**, *Journal of Structural Mechanics*, vol. 10, pp. 375–391.

[Pepper & Smuts 1999] Pepper, J.W. & Smuts, B. (1999), **'The Evolution of Co-operation in an Ecological Context: an Agent-based Model'**, in *Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes*, Kohler, T. & Gumerman, G. (eds.), Santa Fe Institute and Oxford University Press.

[Perkins 1995] Perkins, D., (1995), **'An Unfair Review of Margaret Boden's The Creative Mind from the Perspective of Creative Systems: (Basic Books, New York, 1991); 303 pages'**, *Artificial Intelligence*, vol. 79(1), November 1995, pp. 97–109.

[Pourazady & Fu 1996] Pourazady, M. & Fu, Z., (1996), **'An Integrated Approach to Structural Shape Optimization'**, *Computers & Structures*, vol. 60(2), July 1996, pp. 279–289.

[Powell 1964] Powell, M., (1964), **'An Efficient Method for Finding the Minimum of Functions of Several Variables Without Calculating Derivatives'**, *The Computer Journal*, vol. 7, pp. 155–162.

[Press *et al.* 1993] Press, W.H.; Flannery, B.P.; Teukolsky, S.A. & Vetterling, W.T., (1993), ***Numerical Recipes in C The Art of Scientific Computing***, 2nd Edition, February 1993, Cambridge University Press, ISBN: 0521431085.

[Quagliarella & Cioppa 1994] Quagliarella, D. & Cioppa, A.D., (1994), ***Genetic Algorithms Applied to the Aerodynamic Design of Transonic Airfoils***, AIAA Paper 94-1896-CP.

[Quagliarella & Cioppa 1995] Quagliarella, D. & Cioppa, A.D., (1995), '**Genetic Algorithms Applied to the Aerodynamic Design of Transonic Airfoils**', *Journal of Aircraft*, vol. 32(4), pp. 889–890.

[Quagliarella & Vicini 2000] Quagliarella, D. & Vicini, A., (2000), '**GAs for Aerodynamic Shape Design II: Multiobjective Optimization and Multi-criteria Design**', in *Genetic Algorithms for Optimisation in Aeronautics and Turbomachinery*, Von Karman Institute Lecture Series 2000-07, May 2000.

[Quagliarella & Vicini 2001] Quagliarella, D. & Vicini A., (2001), '**Viscous Single and Multicomponent Airfoil design with Genetic Algorithms**', *Finite Elements in Analysis and Design*, vol. 37, pp. 365–380.

[Querin *et al.* 2000] Querin, O. M.; Steven G. P. & Xie, Y. M., '**Evolutionary Structural Optimisation using an Additive Algorithm**', *Finite Elements in Analysis and Design*, vol. 34(3-4), February 2000, pp. 291–308.

[Raich & Ghaboussi 2000] Raich, A.M. & Ghaboussi, J., (2000), '**Evolving Structural Design Solutions using an Implicit Redundant Genetic Algorithm**', *Structural Multidisciplinary Optimization*, vol. 20(3), pp. 222–231.

[Raghothama & Shapiro 2000] Raghothama, S. & Shapiro, V., (2000), '**Consistent Updates in Dual Representation Systems**', *Computer-Aided Design*, vol. 32(8-9), August 2000, pp. 463–477.

[Raghothama & Shapiro 2002] Raghothama, S. & Shapiro, V., (2002), '**Topological Framework for Part Families**', in *Proc. of the Seventh ACM Symposium on Solid Modeling and Applications 2002*, Saarbrücken, Germany, pp. 1–12, ACM Press, New York, ISBN 1-58113-506-8.

[Ram *et al.* 1995] Ram, A.; Wills, L.; Domeshek, E.; Nersessian, N. & Kolodner, J., (1995), **'Understanding the Creative Mind: a Review of Margaret Boden's Creative Mind'**, *Artificial Intelligence*, vol. 79(1), November 1995, pp. 111–128.

[Rasheed & Hirsh 2000] Rasheed, K. & Hirsh, H., (2000), **'Informed Operators: Speeding up Genetic-Algorithm-Based Design Optimization using Reduced Models'**, in *Proc. Genetic and Evolutionary Computation Conference (GECCO'2000).*

[Rasheed *et al.* 1997] Rasheed, K.; Hirsh, H. & Gelsey, A., (1997), **'A Genetic Algorithm for Continuous Design Space Search'**, *Artificial Intelligence in Engineering*, vol. 11(3), pp. 295–305.

[Ratle 2001] Ratle, A., (2001), **'Kriging as a Surrogate Fitness Landscape in Evolutionary Optimization'**, *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, vol. 15, pp. 37–49.

[Rechenberg 1973] Rechenberg, I., (1973), ***Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution***, Fromman-Holzboog, Stuttgart.

[Reddy & Cagan 1995] Reddy, G. & Cagan, J., (1995), **'An Improved Shape Annealing Algorithm for Truss Topology Generation'**, *Journal of Mechanical Design*, vol. 117, pp. 315–321, ASME.

[Renner & Ekárt 2003] Renner, G. & Ekárt, A., (2003), **'Genetic Algorithms in Computer Aided Design'**, *Computer-Aided Design*, vol. 35(8), July 2003, pp. 709–726.

[Reuther *et al.* 1999] Reuther, J.; Alonso, J.J.; Rimlinger, M.J. & Jameson, A., (1999), '**Aerodynamic Shape Optimization of Supersonic Aircraft Configurations via an Adjoint Formulation on Distributed Memory Parallel Computers**', *Computers & Fluids*, vol. 28, pp. 675–700.

[Robinson *et al.* 1999] Robinson, G.; El-Beltagy, M.A. & Keane, A., (1999), '**Optimization in Mechanical Design**', in *Evolutionary Design by Computers*, Bentley, P. (ed.), Morgan Kaufmann, San Francisco, 1999.

[Rogalsky *et al.* 1999a] Rogalsky, T.; Derksen, R.W. & Kocabiyik, S., (1999), '**An Aerodynamic Design Technique for Optimizing Fan Blade Spacing**', in *Proc. of the 7th Annual Conference of the Computational Fluid Dynamics Society of Canada*, May 30 - June 1, pp. 2–34.

[Rogalsky *et al.* 1999b] Rogalsky, T.; Derksen, R.W. & Kocabiyik, S., (1999), '**Differential Evolution in Aerodynamic Optimization**', in *Proc. of the 46th Annual Conference of the Canadian Aeronautics and Space Institute*, May 2-5, pp. 29–36.

[Rong *et al.* 2001] Rong, J.H.; Xie, Y.M. & Yang, X.Y., (2001), '**An Improved Method for Evolutionary Structural Optimisation against Buckling**', *Computers & Structures*, vol. 79(3), January 2001, pp. 253–263.

[Rosenman 1997] Rosenman, M.A., (1997), '**The Generation of Form using an Evolutionary Approach**', in *Evolutionary Algorithms in Engineering Applications*, Dasgupta, D. & Michalewicz, Z. (eds.), Springer-Verlag, Southampton and Berlin, pp.69–85.

[Roy & Bharadwaj 2002] Roy, U. & Bharadwaj, B., (2002), '**Design with Part Behaviors: Behavior Model, Representation and Applications**', *Computer-Aided Design*, vol. 34(9), August 2002, pp. 613–636.

[Roy *et al.* 2001] Roy, U.; Pramanik, N.; Sudarsan, R.; Sriram, R. D. & Lyons, K. W., (2001), '**Function-to-form Mapping: Model, Representation and Applications in Design Synthesis**', *Computer-Aided Design*, vol. 33(10), September 2001, pp. 699–719.

[Ryan *et al.* 2003] Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R. & Costa, E. (eds.), (2003), ***Genetic Programming: 6th European Conference***, EuroGP 2003, Essex, UK, April 14-16, 2003, Lecture Notes in Computer Science, vol. 2610, Springer-Verlag Heidelberg.

[Sadrehaghighi *et al.* 1995] Sadrehaghighi, I.; Smith, R.E. & Tiwari, S.N., (1995), '**Grid Sensitivity and Aerodynamic Optimization of Generic Airfoils**', *Journal of Aircraft*, vol. 32(6), December 1995, pp. 1234.

[Saitou & Jakiela 1994] Saitou, K. & Jakiela, M.J., (1994), '**Meshing of Engineering Domains by Meitotic Cell Division**', in *Artificial Life IV: Proc. of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, July 1994, Cambridge, Massachusetts, The MIT Press.

[Samareh 1999] Samareh, J.A., (1999), '**A Survey of Shape Parameterization Techniques**', in *CEAS/AIAA/ICASE/NASA Langley International Forum on Aeroelasticity and Structural Dynamics*, June 22-25, Williamsburg, VA, NASA/CP-1999-209136, pp. 333–343.

[Sandgren & Wu 1988] Sandgren, E. & Wu, S.J., (1988), '**Shape Optimization using the Boundary Element Method with Substructuring**', *International Journal for Numerical Methods in Engineering*, vol. 26, pp. 1913–1924.

[Schank & Foster 1995] Schank, R.C. & Foster, D.A., (1995), '**The Engineering of Creativity: a Review of Boden's the Creative Mind**', *Artificial Intelligence*, vol. 79(1), November 1995, pp. 129–143.

[Schewchuk 1997] Schewchuk, J.R., (1997), ***Delaunay Refinement Mesh Generation***, Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh.

[Schoenauer 1995] Schoenauer, M., (1995), '**Shape Representations for Evolutionary Optimization and Identification in Structural Mechanics**', in *Genetic Algorithms in Engineering and Computer Science*, Winter, G.; Periaux, J.; Galan M. & Cuesta, P. (eds.), pp. 443–464, Wiley.

[Schoenauer 1996] Schoenauer, M., (1996), '**Shape Representations and Evolution Schemes**', in *Proc. 5th Annual Conference on Evolutionary Programming*, Fogel, L.J.; Angeline, P.J. & Back, T. (eds.), MIT Press, pp. 121–129.

[Schramm & Pilkey 1994] Schramm, U. & Pilkey, W.D., (1994), '**Higher Order Boundary Elements for Shape Optimization using Rational B-splines**', in *Engineering Analysis with Boundary Elements*, vol. 14(3), pp. 255–266.

[Schramm & Pilkey 1995] Schramm, U. & Pilkey, W.D., (1995), '**The Coupling of Geometric Descriptions and Finite Elements using NURBs – A Study in Shape Optimization**', *Finite Elements in Analysis and Design*, vol. 15(1), December 1993, pp.11–34.

[Schwefel 1981] Schwefel, H-P., (1981), ***Numerical Optimization of Computer Models***, Wiley, Chichester.

[Sebald & Fogel 1994] Sebald, A.V. & Fogel, L.J. (eds.), (1994), ***Proceedings of the Third Annual Conference on Evolutionary Programming***, World Scientific Publishers, River Edge, NJ.

[Seller *et al.* 1994] Sellar, R.S.; Batill, S.M. & Renaud, J.E., (1994), '**Optimization of Mixed Discrete/Continuous Design Variable Systems Using Neural Networks**', *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City, Florida, Sept. 1994, AIAA-94-4348.

[Seller *et al.* 1996] Sellar, R.S.; Stelmack, M.; Batill, S.M. & Renaud, J.E., (1996), '**Response Surface Approximations for Discipline Coordination in Multidisciplinary Design Optimization**', *AIAA/ASME/ASCE/AHS/ASC 37th Structures, Structural Dynamics and Materials Conference*, Salt Lake City, Utah, April 1996, AIAA-96-1383.

[Seo *et al.* 2003] Seo, K.; Fan, Z.; Hu, J.; Goodman, E.D. & Rosenberg, R.C., (2003), '**Toward a Unified and Automated Design Methodology for Multi-domain Dynamic Systems using Bond Graphs and Genetic Programming**', *Mechatronics,* In Press.

[Shah 1991] Shah, J.J., (1991), '**Conceptual Development of Form Features and Feature Modelers**', *Research in Engineering Design*, vol. 2, pp. 93–108.

[Shapiro & Voelcker 1989] Shapiro, V. & Voelcker, H., (1989), '**On the Role of Geometry in Mechanical Design**', *Research in Engineering Design*, vol. 1(1), pp. 69–73.

[Shapiro & Vossler 1995] Shapiro, V. & Vossler D.L., (1995), '**What is a Parametric Family of Solids?**', in *Proc. of the 3rd ACM/IEEE Symposium on Solid Modeling and Applications*, Salt Lake City, Utah, May 17-19, 1995.

[Shimada & Gossard 1992] Shimada, K. & Gossard, D., (1992), '**Automated Shape Generation of Components in Mechanical Assemblies**', in *Proc. ASME Design Automation Conference*, Tempe, AZ, September 1992.

[Smith 1995a] Smith, R., (1995), ***A First Investigation into a Voxel Based Shape Representation***, Internal Report No.20, Manufacturing Planning Group, Department of Mechanical Engineering, University of Edinburgh, UK.

[Smith 1995b] Smith, R., (1995), ***A Parameterised Rolls Royce Annulus***, Internal Report No. 21, Manufacturing Planning Group, Department of Mechanical Engineering, University of Edinburgh, UK.

[Sobieszczanski-Sobieski 1986] Sobieszczanski-Sobieski, J, (1986), '**Structural Optimization: Challenges and Opportunities**', *International Journal of Vehicle Design*, vol. 7(3-4), pp. 242–263.

[Soize & Michelucci 2000] Soize, C. & Michelucci, J-C., (2000), '**Structural Shape Parametric Optimization for an Internal Structural-Acoustic Problem**', *Aerospace Science and Technology*, vol. 4(4), June 2000, pp. 263–275.

[Solé *et al.* 1999] Solé, R.V.; Salazar-Cuidad, I. & Garcia-Fernandez, J., (1999), ***Phase Transitions in a Gene Network Model of Morphogenesis***, Santa Fe Institute Working Paper 99-11-075.

[Song *et al.* 2002] Song, W.; Keane, A.; Rees, J.; Bhaskar, A. & Bagnall, S., (2002), '**Turbine Blade Fir-tree Root Design Optimisation using Intelligent CAD and Finite Element Analysis**', *Computers & Structures*, vol. 80(24), September 2002, pp. 1853–1867.

[Spendley *et al.* 1962] Spendley, W.; Hext, G.R. & Himsworth, F.R., (1962), '**Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation**', *Technometrics*, vol. 4(4), pp. 441–461.

[Tanskanen 2002] Tanskanen, P., (2002), '**The Evolutionary Structural Optimization Method: Theoretical Aspects**', *Computer Methods in Applied Mechanics and Engineering*, vol. 191(47-48), November 2002, pp. 5485–5498.

[Taura & Nagasaka 1999] Taura, T. & Nagasaka, I, (1999), '**Adaptive-Growth-Type 3D Representation for Configuration Design**', *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, vol. 13(3), June 1999, pp. 171–184.

[Tavakkoli & Dhande 1991] Tavakkoli, S. & Dhande, S.G, (1991), '**Shape Synthesis and Optimization Using Intrinsic Geometry**', *Journal of Mechanical Design*, Transactions of the ASME, vol. 113(4), pp. 379–386.

[Turner 1995] Turner, S.R., (1995), '**The Creative Mind: Margaret Boden, (Basic Books, New York, 1991); 303 pages**', *Artificial Intelligence*, vol. 79(1), November 1995, pp. 145–159.

[Tuson *et al.* 1997] Tuson, A.; Ross, P. & Duncan, T., (1997), '**On Interactive Neighbourhood Search Schedulers**', *16th Workshop of the UK Planning and Scheduling SIG.*

[Vanderplaats 1993] Vanderplaats, G.N., (1993), '**Thirty Years of Modern Structural Optimization**', *Advances in Engineering Software*, vol. 16, pp. 81–88.

[Vekeria & Parmee 1997] Vekeria, H.D. & Parmee, I.C., (1997), '**Reducing Computational Expense Associated with Evolutionary Detailed Design**', in *Proc. of IEEE International Conference on Evolutionary Computation '97*, Indiana University, Indianapolis, 13-16 April, 1997.

[Versteeg & Malalasekera 1995] Versteeg, H.G. & Malalasekera, W., (1995), ***An Introduction to Computational Fluid Dynamics***, Longman.

[Vicini & Quagliarella 1999] Vicini, A. & Quagliarella, D., (1999), '**Airfoil and Wing Design using Hybrid Optimization Strategies**', *AIAA Journal*, Vol. 37(5), May 1999.

[Vicini & Quagliarella 2000] Vicini, A., Quagliarella, D., (2000), '**A Multiobjective Approach to Transonic Wing Design by Means of Genetic Algorithms**', *NATO RTO AVT Symposium on Aerodynamic Design and Optimization*, Ottawa, Canada, October 1999, RTO-MP-35, June 2000.

[Walsh 1975] Walsh G., (1975), ***Methods of Optimization***, Wiley.

[Wang *et al.* 2002a] Wang, J.F.; Periaux, J. & Sefrioui M., (2002), '**Parallel Evolutionary Algorithms for Optimization Problems in Aerospace Engineering**', *Journal of Computational and Applied Mathematics*, vol. 149, pp. 155–169.

[Wang *et al.* 2002b] Wang, L.; Shen, W.; Xie, H.; Neelamkavil, J. & Pardasani, A., (2002), '**Collaborative Conceptual Design – State of the Art and Future Trends**', *Computer-Aided Design*, vol. 34(13), November 2002, pp. 981–996.

[Watabe & Okino 1993] Watabe, H. & Okino, N., (1993), '**A Study on Genetic Shape Design**', in *Proc. of the Fifth International Conference on Genetic Algorithms,* Forrest, S. (ed.), pp. 445–450. San Mateo, Morgan Kauffman.

[Watson 1981] Watson, D.F., (1981), '**Computing the n-dimensional Delaunay Tessellation with Application to Voronoi Polytopes**', *The Computer Journal*, vol. 24, pp. 167–172.

[Whitley 1989] Whitley, D., (1989), '**The GENITOR Algorithm: Why Rank Based Allocation of Reproductive Trials is Best**', in *Proc. of the Third International Conference on Genetic Algorithms*, pp. 116–121.

[Widmann & Sheppard 1993] Widmann, J.M. & Sheppard, S.D., (1993), '**Shape Optimization using a Variable Number of Conic Patches**', *Advances in Design Automation*, vol. 65–1, ASME.

[Widmann & Sheppard 1994] Widmann, J.M. & Sheppard, S.D., (1994), '**Intrinsic Geometry for Shape Optimal Design with Analysis Model Compatibility**', in *Proc. 1994 ASME Design Technical Conference*, Minneapolis, Sept. 11-14, Advances in Design Automation, vol. 2, pp. 273–281, ASME.

[Winter *et al.* 1995] Winter, G.; Périaux, J.; Galan M. & Cuesta, P. (eds.), (1995), ***Genetic Algorithms in Engineering and Computer Science***, Wiley.

[Xie & Steven 1996] Xie, Y.M. & Steven, G.P., (1996), '**Evolutionary Structural Optimization for Dynamic Problems**', *Computers & Structures*, vol. 58(6), March 1996, pp. 1067–1073.

[Xie & Steven 1997] Xie, Y. M. & Steven, G. P., (1997), ***Evolutionary Structural Optimization***, Springer, ISBN 3540761535.

[Yamazaki *et al.* 1993] Yamazaki, K.; Sakamoto, J. & Kitano, M., (1993), '**An Efficient Shape Optimization Technique of a Two-dimensional Body based on the Boundary Element Method**', *Computers & Structures*, vol. 48(6), September 1993, pp. 1073–1081.

[Yamazaki *et al.* 1994] Yamazaki, K.; Sakamoto, J. & Kitano, M., (1994), '**Three-Dimensional Shape Optimization Using the Boundary Element Method**', *AIAA Journal*, vol. 32(6), pp. 1295–1301.

[Younsi *et al.* 1996] Younsi, R.; Knopf-Lenoir, C. & Selman, A., (1996), '**Multi-mesh and Adaptivity in 3D Shape Optimization**', *Computers & Structures*, vol. 61(6), December 1996, pp. 1125–1133.

[Zhao *et al.* 1998] Zhao, C.; Steven, G. P. & Xie, Y. M., (1998), '**A Generalized Evolutionary Method for Natural Frequency Optimization of Membrane Vibration Problems in Finite Element Analysis**', *Computers & Structures*, vol. 66(2-3), January 1998, pp. 353–364.

[Zienkiewicz & Campbell 1973] Zienkiewicz, O.C. & Campbell, J.S., (1973), **'Shape Optimization and Sequential Linear Programming'**, in *Optimal Structural Design*, Callagher, R.H. & Zienkiewicz, O.C. (eds.), Wiley, New York, pp. 109–126.

A. Appendix A - Results of Aerofoil Optimisation

This Appendix provides a more detailed description of the results of the experiments conducted on aerofoil optimisation from Chapter 3, which were summarised in Section 3.4.

A.1 Aerofoil Parameterisation

Below, the results of a number of experiments undertaken with the aerofoil parameterisation are given, along with a short discussion of each. The genetic algorithm was set up to minimise and so low fitnesses are good. It should also be noted that since the genetic algorithm is a stochastic algorithm, the results of an experiment might not be the same each time it is done. Consequently, each experiment was repeated ten times and the results are given for each run.

Experiments were undertaken with various fitness measures, population sizes, mutation rates and mutation amplitudes. Fitness measures varied dependent on what properties of the aerofoil were to be optimised. For Experiments A, B and C, the population size was set at 50. These experiments were intended to validate the genetic algorithm and vortex panel method, by maximising lift (Experiment A) or minimising drag (Experiments B and C) for which the optimal solution was known to lie at the extreme of the parameter ranges. Preliminary tests showed that a population size of 50 was sufficient to avoid premature convergence for these experiments. Later experiments were more challenging for the genetic algorithm, and therefore population sizes were increased, as described with each set of results below.

The proportion of population to breed per generation was set as 10%, since preliminary tests indicated that this avoided premature convergence of the population. The mutation rate was set so that on average there would be one mutation per new individual created. The mutation amplitude was varied for each experiment; lower amplitudes were desirable for those problems where the optimal solution lay away from the parameter bounds, so that adequate 'fine-tuning' of aerofoil parameters could take place.

A.1.1 Experiment A

Fitness	$-C_l$	
Population size	50	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.5	
Parameter	**Min**	**Max**
Thickness from camber line (% of chord length)	2	8
Position of maximum thickness (% of chord length)	55	80
Camber angle, theta0, at trailing edge (deg)	0	30
Camber angle, theta1, at leading edge (deg)	0	30
Angle of attack (deg)	-8	8

The objective of this experiment was to maximise lift coefficient (i.e. minimise $-C_l$). This experiment was used primarily to validate the analysis and optimisation code. An aerofoil with maximal lift would have the largest camber, thickness and attack angle allowed within the parameter bounds. The experiment was repeated ten times with varying seeds for the random number generator. Every run consistently found exactly the following best solution:

Thickness from camber line (% of chord length)	8.00
Position of maximum thickness (% of chord length)	55.00
Camber angle, theta0, at trailing edge (deg)	30.00
Camber angle, theta1, at leading edge (deg)	30.00
Angle of attack (deg)	8.00
Fitness ($-C_l$)	-2.54
Number of Evaluations	600

As was expected these values were at the extremes of the parameter bounds. In this implementation of the genetic algorithm, mutations could vary the parameters to be outwith the bounds, but these values would subsequently be 'repaired' to lie on the bound. Consequently, it was 'easy' for the genetic algorithm to find extrema which

lie at the boundary of the search-space, and hence each of the ten runs produced exactly the same result. The lift coefficient generated had a reasonable value based on data for similar aerofoils at similar attack angles [Abbot *et al.* 1945]. Figure A-1 and Figure A-2 show the best individual from a number of generations throughout two particular runs, A1 and A2. Figure A-3 shows a plot of individual fitnesses, along with minimum, mean and maximum fitnesses for the run A1.

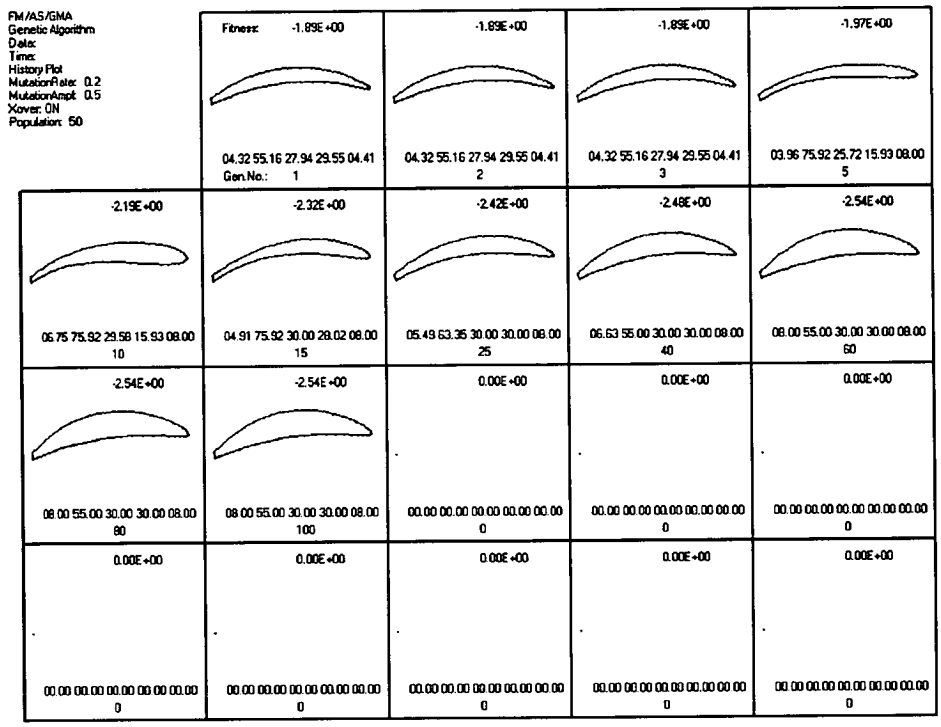


Figure A-1 Best Individual for each Generation for Run A1

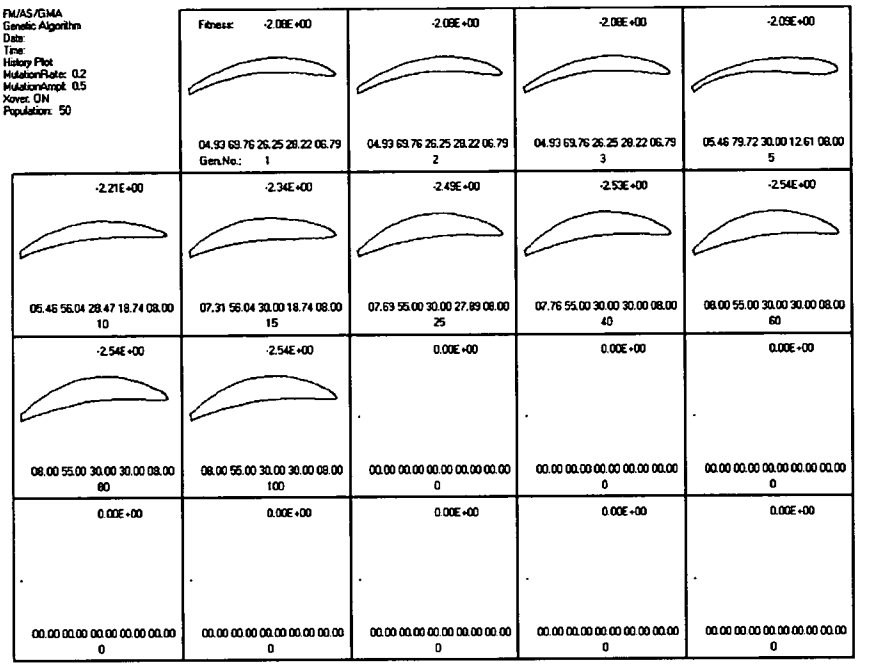


Figure A-2 **Best Individual for each Generation for Run A2**

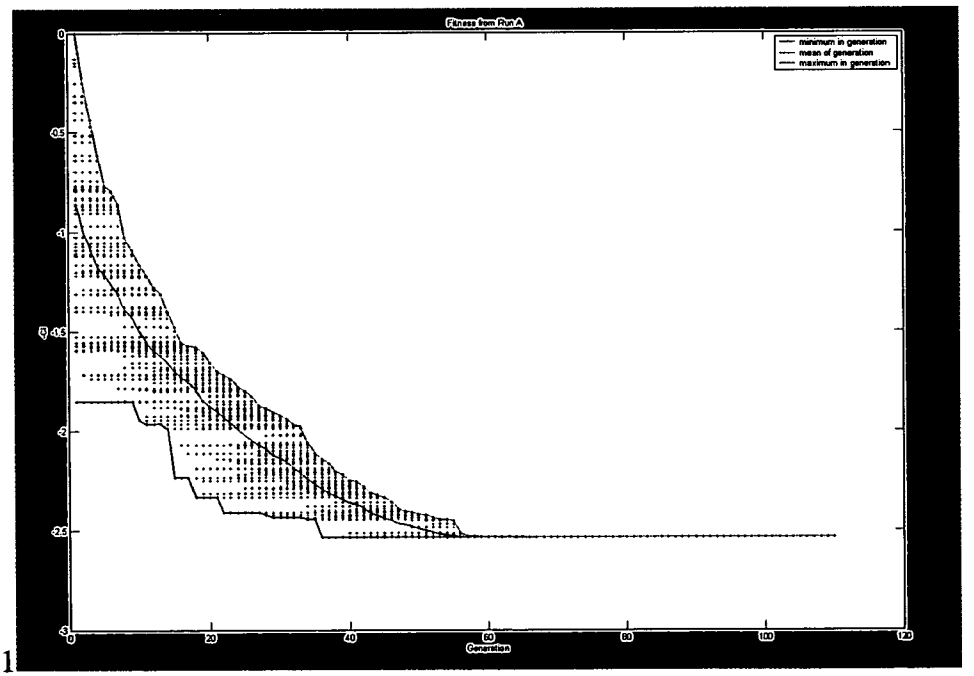


Figure A-3 **Graph of Fitnesses for each Generation for Run A1**

A.1.2 Experiment B

Fitness	C_d	
Population size	50	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.5	
Parameter	**Min**	**Max**
Thickness from camber line (% of chord length)	2	8
Position of maximum thickness (% of chord length)	55	80
Camber angle, theta0, at trailing edge (deg)	0	30
Camber angle, theta1, at leading edge (deg)	0	30
Angle of attack (deg)	-4	4

The objective of this experiment was to minimise the drag coefficient. The experiment was repeated ten times with varying seeds for the random number generator. Every run consistently found exactly the following best solution (for similar reasons as those discussed under Experiment A):

Thickness from camber line (% of chord length)	2.00
Position of maximum thickness (% of chord length)	55.00
Camber angle, theta0, at trailing edge (deg)	30.00
Camber angle, theta1, at leading edge (deg)	30.00
Angle of attack (deg)	-4.00
Fitness	-5.86e-2
Number of Evaluations	1100

This solution was not as expected. It was anticipated that the optimum aerofoil for low drag coefficient would have low or zero camber and would be at an attack angle very close to zero. Figure A-4 and Figure A-5 show the best individual from a number of generations throughout two particular runs, B1 and B2. From these figures, it can be seen that the aerofoil generated had a high camber and negative attack angle. The value of drag coefficient was negative which is clearly unrealistic. There was clearly some problem with the optimisation algorithm or fluid analysis.

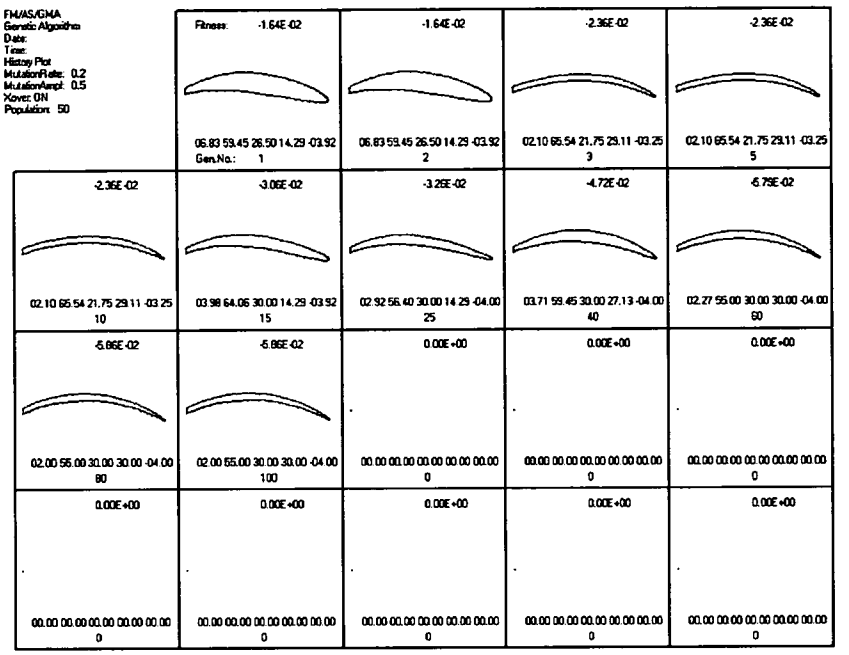


Figure A-4 **Plot of Best Individual for each Generation for Run B1**

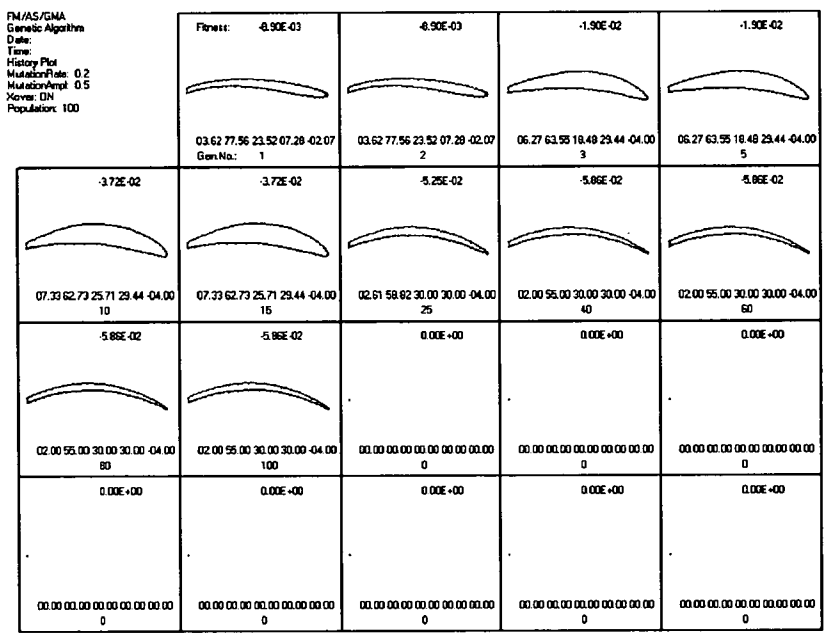


Figure A-5 **Plot of Best Individual for each Generation for Run B2**

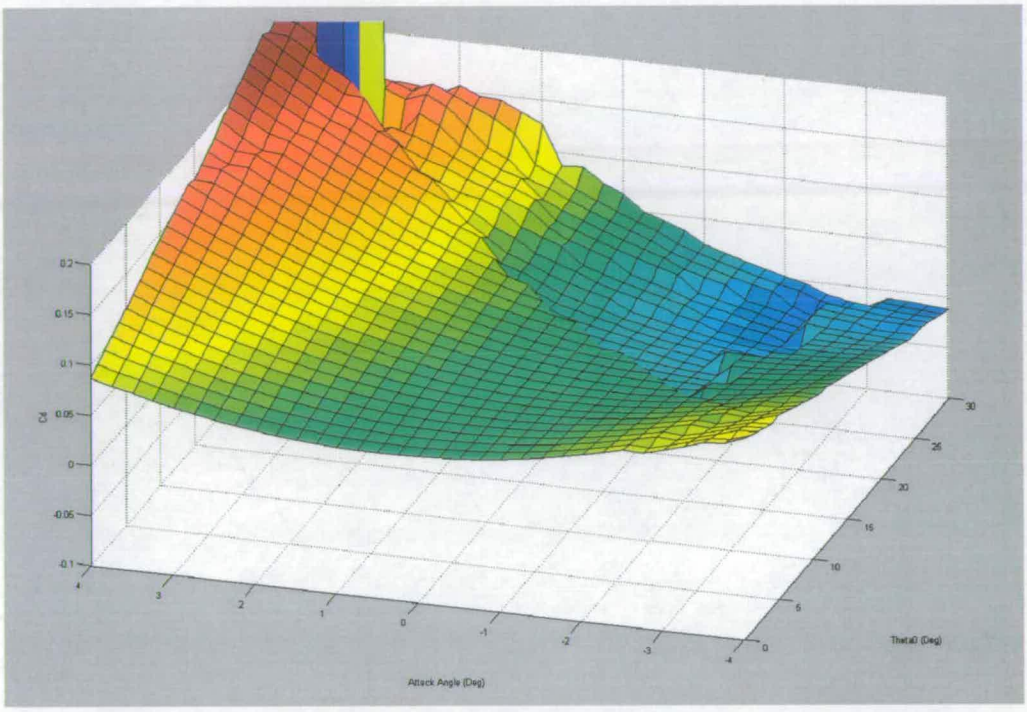


Figure A-6 Surface Plot of C_d against Attack Angle and Theta0

Since the camber angle and attack angle of the fittest seemed to be wrong, it was decided to investigate how the drag coefficient varied with camber angle and the attack angle. All parameters other than theta0 and attack angle were held constant at the values for the best solution. A scan was undertaken calculating the drag coefficient whilst varying theta0 from 0° to 30° and attack angle between –4° to 4°.

Looking at Figure A-6 it can be seen that the drag coefficient varies with the attack angle as expected (it is at minimum at an attack angle of zero) with values of theta0 up to about 15°. However, above theta0 of 15°, the drag coefficient continues decreasing as the attack angle decreases to values below zero, which is clearly incorrect. It can also be seen that the smoothness of the surface in this region is reduced.

This highlighted a problem that was to be frequently encountered with the vortex panel fluid analysis. It worked well for most shapes, but in some areas of the search space, the values for lift and drag it returned were incorrect. It was also difficult to predict in which areas it performed poorly.

A.1.2 Experiment C

This run was a repeat of Experiment B but with reduced upper-bounds on the camber angles in order to avoid the problems encountered with false values for drag coefficient being generated for aerofoils with negative attack angle and high camber angles.

Fitness	C_d	
Population size	50	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.5	
Parameter	**Min**	**Max**
Thickness from camber line (% of chord length)	2	8
Position of maximum thickness (% of chord length)	55	80
Camber angle, theta0, at trailing edge (deg)	0	10
Camber angle, theta1, at leading edge (deg)	0	10
Angle of attack (deg)	-4	4

The experiment was repeated ten times with varying seeds for the random number generator. The best solution for each run was:

Run Title	C1	C2	C3	C4	C5
Thickness from camber line (% of chord length)	8.00	8.00	8.00	8.00	8.00
Position of maximum thickness (% of chord length)	57.85	57.49	57.74	58.03	58.25
Camber angle, theta0, at trailing edge (deg)	0.00	0.00	0.00	0.00	0.00
Camber angle, theta1, at leading edge (deg)	0.00	0.00	0.00	0.00	0.00
Angle of attack (deg)	0.93	1.02	0.92	0.85	0.87
Fitness (10^{-2})	1.92	1.92	1.92	1.92	1.93
Number of Evaluations	2100	2100	2100	2100	2100

Run Title	C6	C7	C8	C9	C10
Thickness from camber line (% of chord length)	8.00	8.00	8.00	8.00	7.93
Position of maximum thickness (% of chord length)	57.92	57.85	57.56	57.36	56.14
Camber angle, theta0, at trailing edge (deg)	0.00	0.00	0.00	0.00	0.00
Camber angle, theta1, at leading edge (deg)	0.00	0.00	0.00	0.00	0.00
Angle of attack (deg)	0.86	0.90	1.02	1.10	0.85
Fitness (10^{-2})	1.92	1.92	1.92	1.92	1.93
Number of Evaluations	2100	2100	2100	2100	2100

The average C_d found was 0.01922 with a standard deviation of 0.0042.

It can be seen that all the runs converged to very similar aerofoils. With the restriction on the camber angles added, the genetic algorithm and analysis produced an aerofoil shape with minimum drag coefficient that looked as expected (low camber and low attack angle) with a realistic drag coefficient. Figure A-7 and Figure A-8 show the best individual from a number of generations for runs C1 and C2.

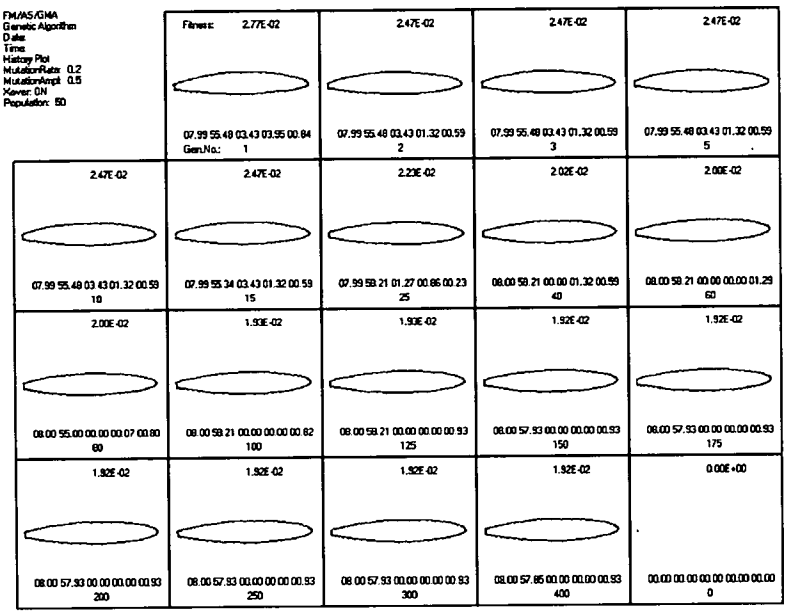


Figure A-7 **Plot of Best Individual for each Generation for Run C1**

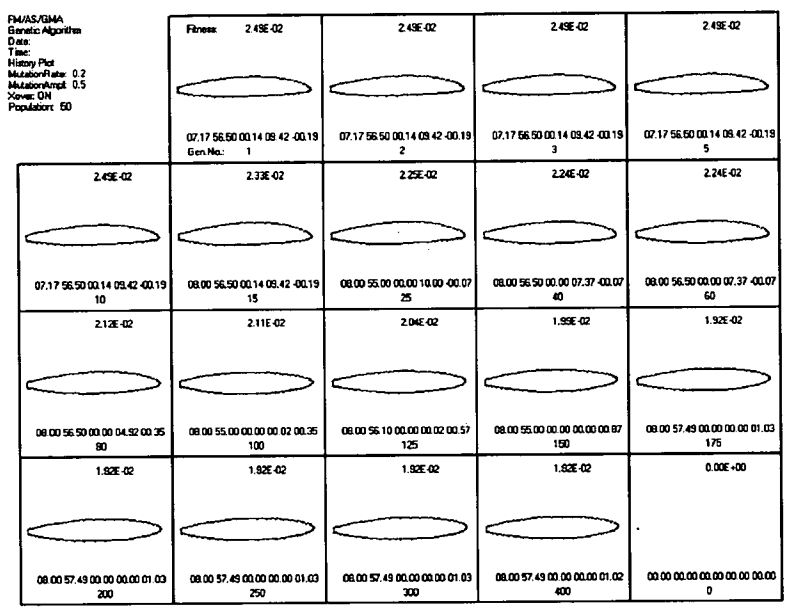


Figure A-8 **Plot of Best Individual for each Generation for Run C2**

A.1.4 Experiment D

This experiment looked to maximise the lift/drag ratio (i.e. minimise C_d / C_l). Candidate aerofoils were penalised heavily for negative values for C_l. As this was the first experiment that attempted to optimise the lift/drag ratio, it was decided to firstly attempt a simplified problem in which the attack angle was constrained to be 0°. It should also be noted that the camber angles were not restricted to be below 10° but were allowed up to 30°. Since the optimal aerofoil was expected not to lie at the extreme of the aerofoil parameter bounds, the mutation amplitude was set lower at 0.05 so that more 'fine-tuning' of aerofoil parameters was possible.

Fitness	C_d / C_l	
Population size	200	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.05	
Parameter	**Min**	**Max**
Thickness from camber line (% of chord length)	2	8
Position of maximum thickness (% of chord length)	55	80
Camber angle, theta0, at trailing edge (deg)	0	30
Camber angle, theta1, at leading edge (deg)	0	30
Angle of attack (deg)	0	0

The experiment was repeated ten times with varying seeds for the random number generator. The best solution for each run was:

Run Title	D1	D2	D3	D4	D5
Thickness from camber line (% of chord length)	3.52	4.53	4.50	4.55	3.43
Position of maximum thickness (% of chord length)	80.00	80.00	80.00	80.00	80.00
Camber angle, theta0, at trailing edge (deg)	20.15	19.78	19.79	19.75	20.17
Camber angle, theta1, at leading edge (deg)	25.00	22.92	23.04	23.03	25.34
Angle of attack (deg)	0	0	0	0	0
Fitness (10^{-2})	3.74	3.74	3.73	3.74	3.71
C_l / C_d	26.74	26.74	26.81	26.74	26.95
Number of Evaluations	8200	8200	8200	8200	8200

Run Title	D6	D7	D8	D9	D10
Thickness from camber line (% of chord length)	3.41	4.55	4.52	3.43	4.55
Position of maximum thickness (% of chord length)	80.00	80.00	80.00	80.00	80.00
Camber angle, theta0, at trailing edge (deg)	20.23	19.76	19.77	20.20	19.74
Camber angle, theta1, at leading edge (deg)	25.27	23.03	23.03	25.28	23.02
Angle of attack (deg)	0	0	0	0	0
Fitness (10^{-2})	3.71	3.74	3.74	3.72	3.74
C_l / C_d	26.95	26.74	26.74	26.88	26.74
Number of Evaluations	8200	8200	8200	8200	8200

The average C_l / C_d found was 26.803 with a standard deviation of 0.09.

It can be seen that all the runs converged to aerofoils with a reasonably low thickness, relatively high camber and maximum thickness as close to the leading edge as the bounds allowed. This matched well with what was expected and the lift-drag ratio was realistic.

The runs converged to two slightly different shapes, both with very similar fitnesses. Runs D1, D5, D6 and D9 produced an aerofoil with thickness between 3.41% and 3.52%, camber angle, theta1, between 25.00° and 25.28° and lift-drag ratio between 26.74 and 26.95. Runs D2, D3, D4, D7, D8 and D10 produced a thicker aerofoil with thickness between 4.50% and 4.55%, camber angle, theta1, between 22.92° and 23.03° and lift-drag ratio between 26.74 and 26.81.

Figure A-9 and Figure A-10 show the best individual from a number of generations for runs D1 and D2. Figure A-11 shows a plot of minimum, mean and maximum fitnesses for run D1. Figure A-12 shows a plot of minimum, mean and maximum lift coefficients for run D1. Figure A-13 shows a plot of minimum, mean and maximum drag coefficients for run D1.

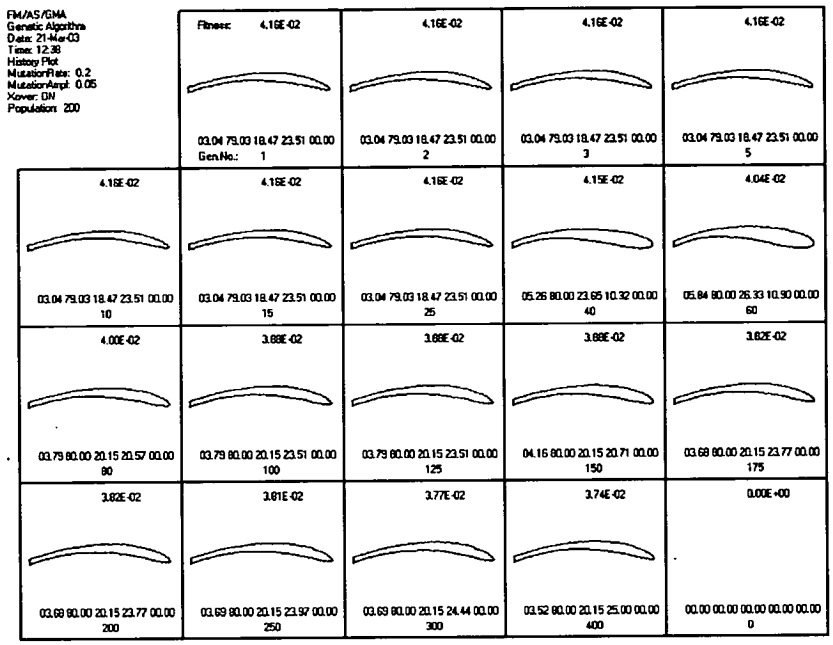


Figure A-9 **Plot of Best Individual for each Generation for Run D1**

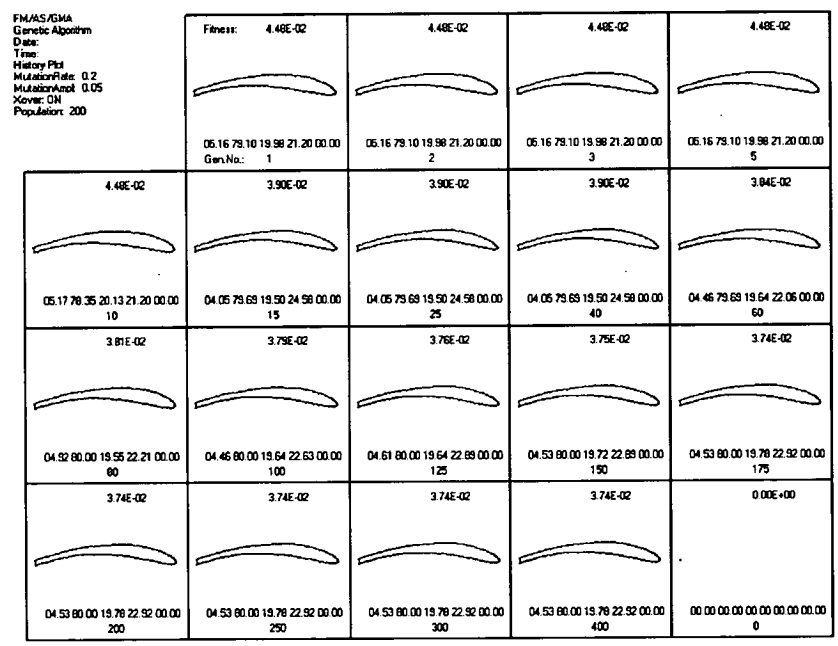


Figure A-10 **Plot of Best Individual for each Generation for Run D2**

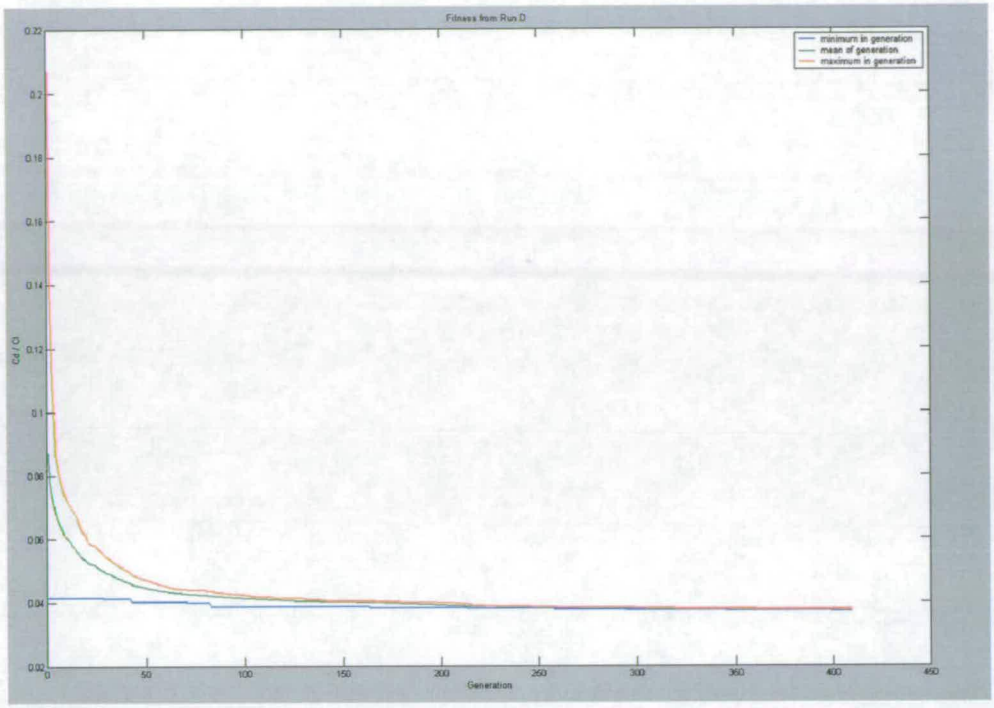


Figure A-11 **Graph of Fitness (C_d / C_l) against Generation for Run D1**

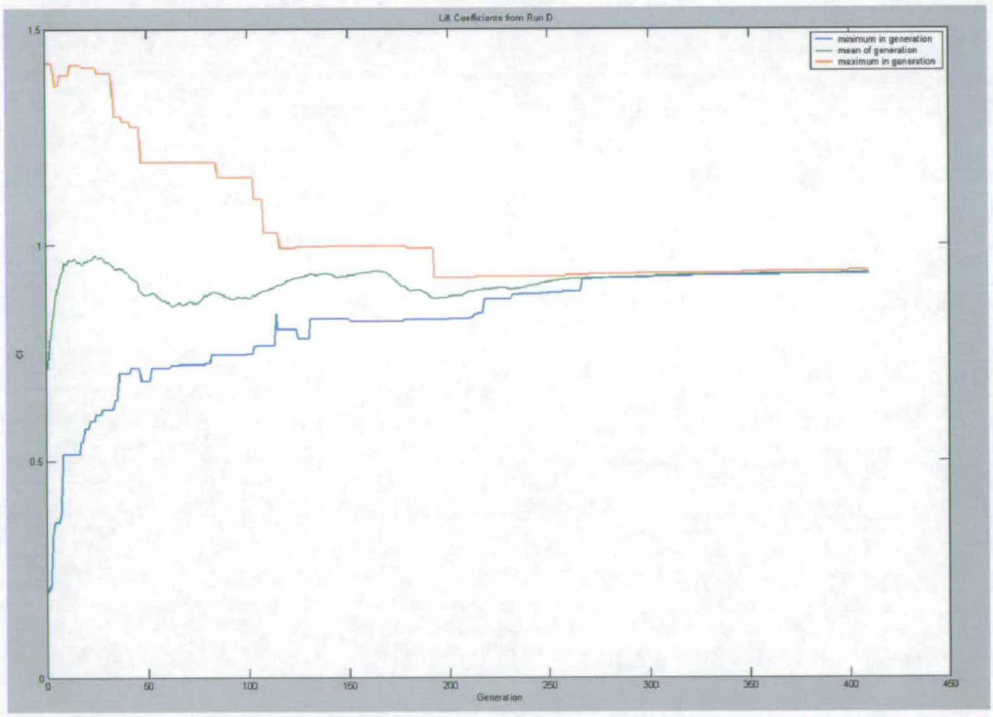


Figure A-12 **Graph of C_l against Generation for Run D1**

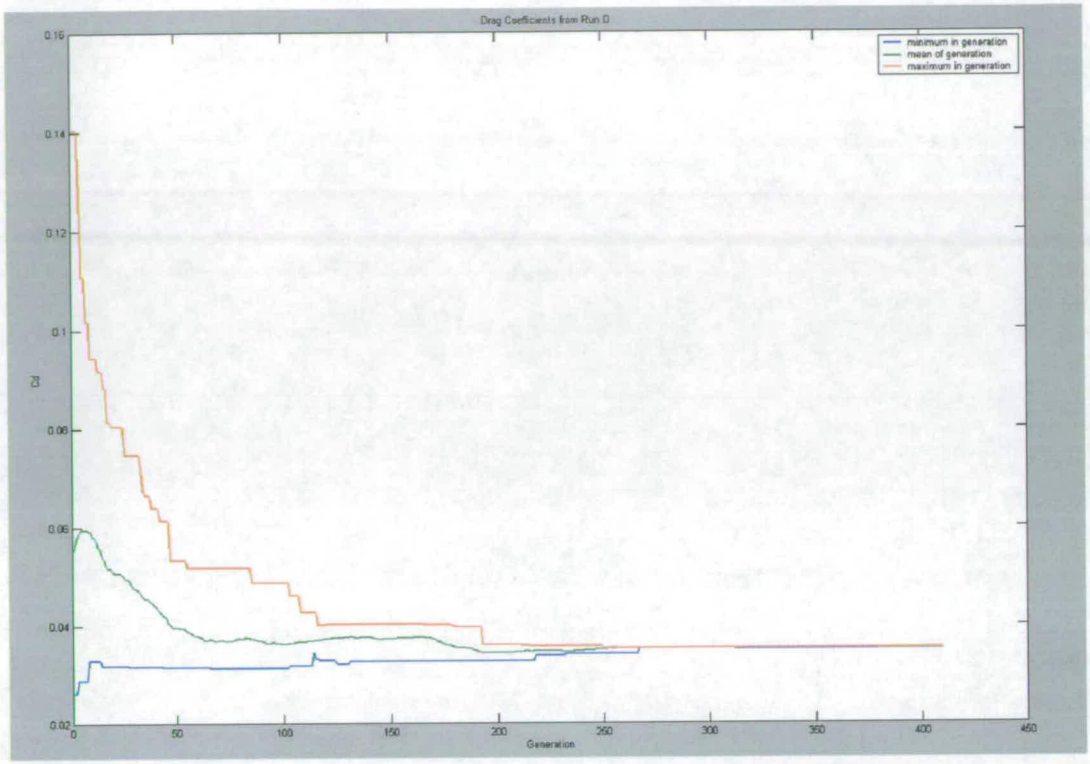


Figure A-13 **Graph of C_d against Generation for Run D1**

A.1.5 Experiment E

This was a repeat of Experiment D with a larger population size. Since the runs in Experiment D had converged to two slightly different aerofoil shapes, it was decided to check that Experiment D had not prematurely converged to a sub-optimal solution. This experiment looked to maximise the lift/drag ratio (i.e. minimise C_d / C_l). Candidate aerofoils were penalised heavily for negative values for C_l .

Fitness	C_d / C_l	
Population size	400	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.05	
Parameter	**Min**	**Max**
Thickness from camber line (% of chord length)	2	8
Position of maximum thickness (% of chord length)	55	80
Camber angle, theta0, at trailing edge (deg)	0	30
Camber angle, theta1, at leading edge (deg)	0	30
Angle of attack (deg)	0	0

The experiment was repeated ten times with varying seeds for the random number generator. The best solution for each run was:

Run Title	E1	E2	E3	E4	E5
Thickness from camber line (% of chord length)	2.00	2.47	3.43	4.49	3.45
Position of maximum thickness (% of chord length)	68.29	63.76	80.00	80.00	80.00
Camber angle, theta0, at trailing edge (deg)	28.55	27.86	20.25	19.79	20.16
Camber angle, theta1, at leading edge (deg)	19.30	21.17	24.89	22.97	25.28
Angle of attack (deg)	0	0	0	0	0
Fitness (10^{-6})	9.42	203	37400	37400	37200
C_l / C_d	1.06e5	4926	26.74	26.74	26.88
Number of Evaluations	8400	8400	8400	8400	8400

Run Title	E6	E7	E8	E9	D10
Thickness from camber line (% of chord length)	2.22	3.45	4.49	3.35	2.00
Position of maximum thickness (% of chord length)	60.68	80.00	80.00	80.00	65.64
Camber angle, theta0, at trailing edge (deg)	28.85	20.14	19.81	20.37	28.59
Camber angle, theta1, at leading edge (deg)	12.60	25.35	22.85	24.63	22.86
Angle of attack (deg)	0	0	0	0	0
Fitness (10^{-6})	113	37200	37500	37400	2310
C_l / C_d	8850	26.88	26.67	26.74	432.9
Number of Evaluations	8400	8400	8400	8400	8400

Figure A-14 and Figure A-15 shows the best individual from a number of generations throughout Run E1 and Run E2. Runs E3, E4, E5, E7, E8 and E9 converged to aerofoil shapes similar to those found in Experiment D. However, Runs E1, E2, E6 and E10 produced an aerofoil with an unrealistic lift-drag ratio. It was suspected that the vortex panel method was producing inaccurate results in some parts of the search space as was found in Experiment B.

Since the thickness of the wrong 'optimal' solution from the run was against the lower bound, it was decided to investigate how the drag and lift coefficient varied with thickness and position of maximum thickness. A scan was undertaken calculating lift and drag with varying thickness from 1 to 20 of chord length and position of maximum thickness from 50 to 95 (both as a percentage of chord length). All other variables were held constant at the values for the best solution. The results of this are shown in Figure A-16 and Figure A-17.

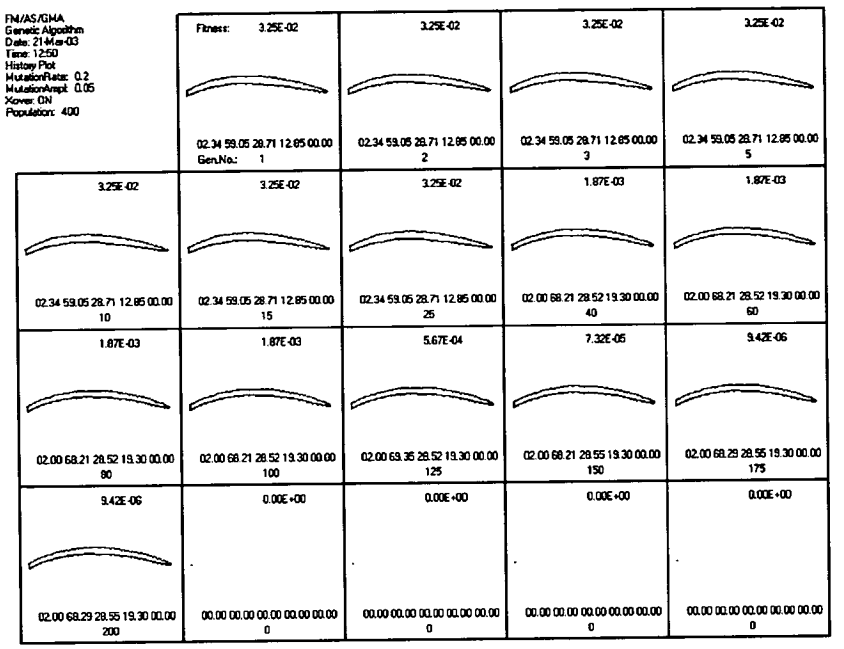


Figure A-14 **Plot of Best Individual for each Generation for Run E1**

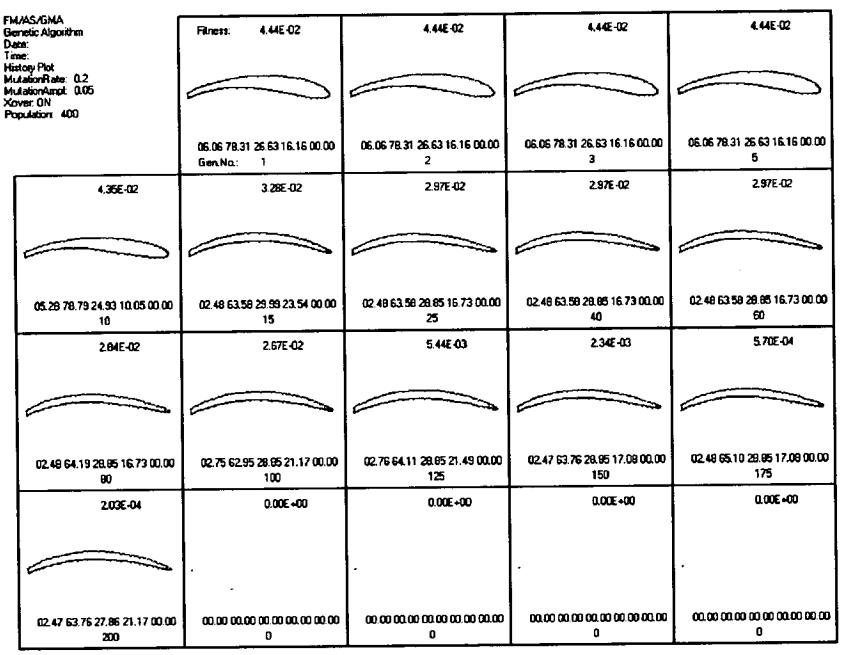


Figure A-15 **Plot of Best Individual for each Generation for Run E2**

By considering Figure A-16 and Figure A-17, it was apparent that the accurate calculation of both lift and drag was not possible for values for the position of maximum thickness above about 85%. More pertinently to the problem encountered on this run, it could be seen than although the lift was calculated accurately at low values of thickness, the calculation of drag was not.

This problem was not encountered when a smaller population size was used because this inaccuracy only occurs in a small part of the search space (it relies on the other parameters, such as camber angles, being in certain ranges). With a large population this area of the search space is more likely to be encountered, either when the initial population was formed, or during the optimisation.

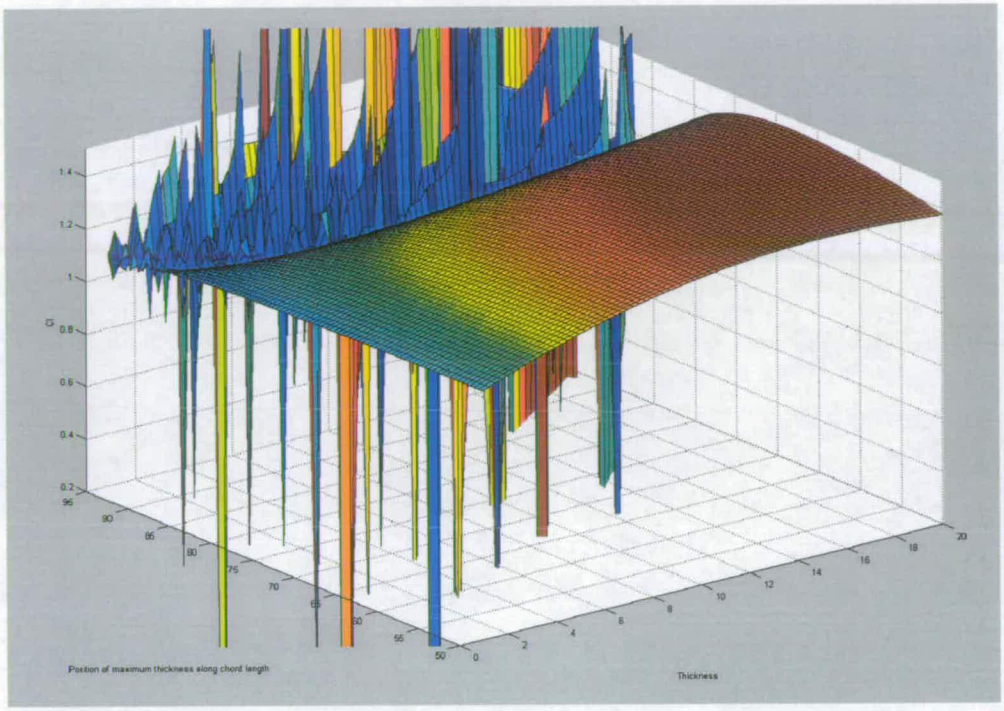


Figure A-16 **Plot of C_l vs. Thickness and Position of Max Thickness**

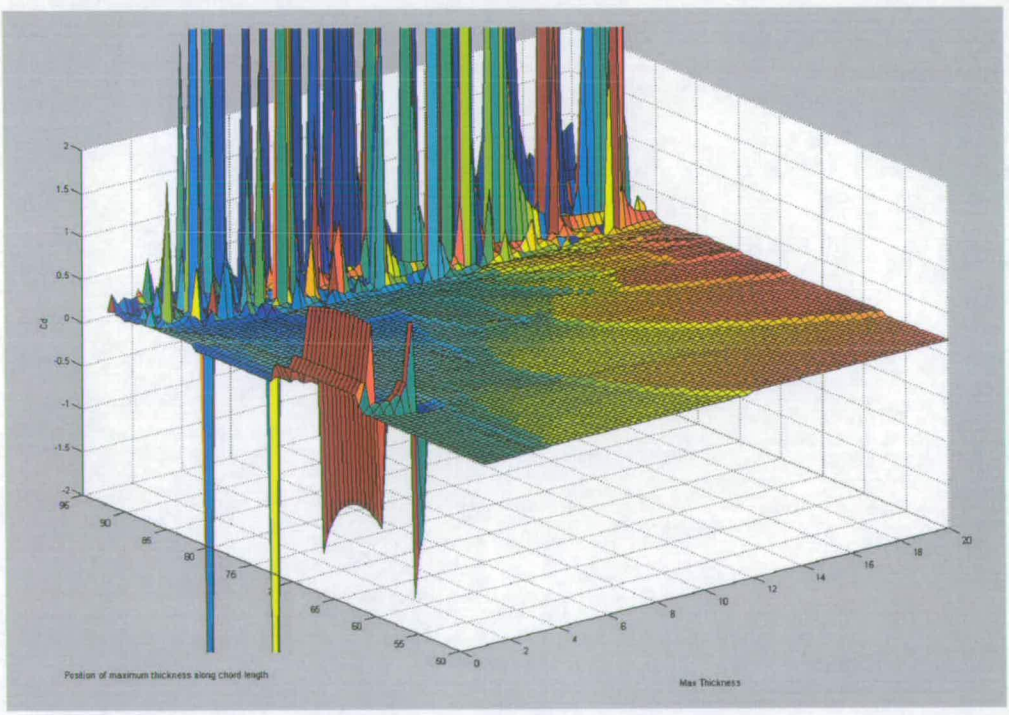


Figure A-17 **Plot of C_d vs. Thickness and Position of Max Thickness**

A.1.6 Experiment F

This was a repeat of Experiment E with the lower bound on aerofoil thickness raised from 2% to 3% in order to avoid the problems encountered with faulty fluid analysis at small thicknesses. The upper bound on the position of maximum thickness was also raised from 80% to 82%.

Fitness	C_d/C_l	
Population size	400	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.05	
Parameter	**Min**	**Max**
Thickness from camber line (% of chord length)	3	8
Position of maximum thickness (% of chord length)	55	82
Camber angle, theta0, at trailing edge (deg)	0	30
Camber angle, theta1, at leading edge (deg)	0	30
Angle of attack (deg)	0	0

The experiment was repeated ten times with varying seeds for the random number generator. The best solution for each run was:

Run Title	F1	F2	F3	F4	F5
Thickness from camber line (% of chord length)	3.11	4.48	3.64	4.47	4.48
Position of maximum thickness (% of chord length)	79.09	79.99	79.98	80.00	79.98
Camber angle, theta0, at trailing edge (deg)	20.69	19.81	20.06	19.82	19.80
Camber angle, theta1, at leading edge (deg)	23.09	22.90	24.98	22.82	22.93
Angle of attack (deg)	0	0	0	0	0
Fitness (10^{-2})	3.77	3.74	3.75	3.75	3.74
C_l/C_d	26.52	26.74	26.67	26.67	26.74
Number of Evaluations	12400	8400	8400	8400	8400

Run Title	F6	F7	F8	F9	F10
Thickness from camber line (% of chord length)	4.50	4.79	3.58	3.35	4.54
Position of maximum thickness (% of chord length)	79.98	79.99	79.80	80.00	79.98
Camber angle, theta0, at trailing edge (deg)	19.77	19.43	19.81	19.87	19.75
Camber angle, theta1, at leading edge (deg)	23.03	23.14	22.85	25.15	23.03
Angle of attack (deg)	0	0	0	0	0
Fitness (10^{-6})	3.74	3.77	3.75	3.77	3.74
C_l / C_d	26.74	26.52	26.67	26.52	26.74
Number of Evaluations	8400	8400	8400	8400	8400

The average C_l / C_d found was 26.653 with a standard deviation of 0.097.

These results matched the solutions found in Experiment D. Figure A-18 shows the best individual from a number of generations throughout the Run F1. Again, as in Run D, the runs converged to two slightly different shapes, both with very similar fitnesses.

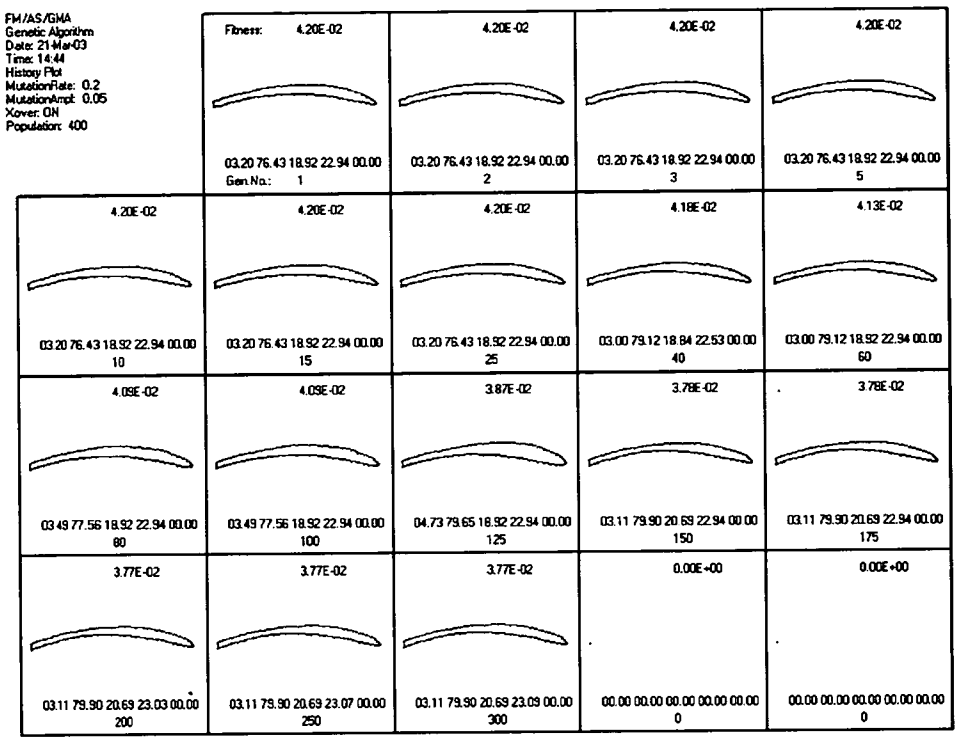


Figure A-18 **Plot of Best Individual for each Generation for Run F1**

A.1.7 Experiment G

This run repeated Experiment F but allowed the attack angle to vary from a lower bound of -4° to an upper bound of 4°. Again, this experiment looked to maximise the lift/drag ratio (i.e. minimise C_d / C_l). Candidate aerofoils were penalised heavily for negative values for C_l .

Fitness	C_d/C_l	
Population size	400	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.05	
Parameter	**Min**	**Max**
Thickness from camber line (% of chord length)	3	8
Position of maximum thickness (% of chord length)	55	82
Camber angle, theta0, at trailing edge (deg)	0	30
Camber angle, theta1, at leading edge (deg)	0	30
Angle of attack (deg)	-4	4

The experiment was repeated ten times with varying seeds for the random number generator. The best solution for each run was:

Run Title	G1	G2	G3	G4	G5
Thickness from camber line (% of chord length)	6.74	4.68	5.08	4.51	4.44
Position of maximum thickness (% of chord length)	64.53	67.05	57.76	76.67	73.00
Camber angle, theta0, at trailing edge (deg)	15.17	18.76	16.17	21.97	26.56
Camber angle, theta1, at leading edge (deg)	22.88	22.77	19.05	16.57	24.98
Angle of attack (deg)	-2.70	-2.54	-2.98	-2.60	-2.24
Fitness (10^{-7})	17.6	5.79	22.9	2.51	30.2
C_l/C_d (10^6)	0.568	1.73	0.437	3.98	0.331
Number of Evaluations	8400	8400	8400	8400	8400

Run Title	G6	G7	G8	G9	G10
Thickness from camber line (% of chord length)	3.00	6.92	5.63	6.02	3.46
Position of maximum thickness (% of chord length)	70.85	63.69	76.62	72.62	78.88
Camber angle, theta0, at trailing edge (deg)	19.17	20.16	28.06	21.85	22.52
Camber angle, theta1, at leading edge (deg)	24.08	23.06	19.97	27.43	28.47
Angle of attack (deg)	-2.56	-2.72	-2.35	-2.48	-2.32
Fitness (10^{-7})	2.66	10.0	12.1	8.32	14.6
C_l / C_d (10^6)	3.76	0.999	0.826	1.20	0.684
Number of Evaluations	8400	8400	8400	8400	8400

Figure A-19 shows the best individual from a number of generations throughout the Run G1. This solution again produced unrealistic results. This was due to the same problem encountered in Experiment B where the vortex panel was unable to calculate drag correctly for aerofoils with large camber angles and negative attack angles. This can be seen by considering Figure A-20 which shows the drag coefficient against generation.

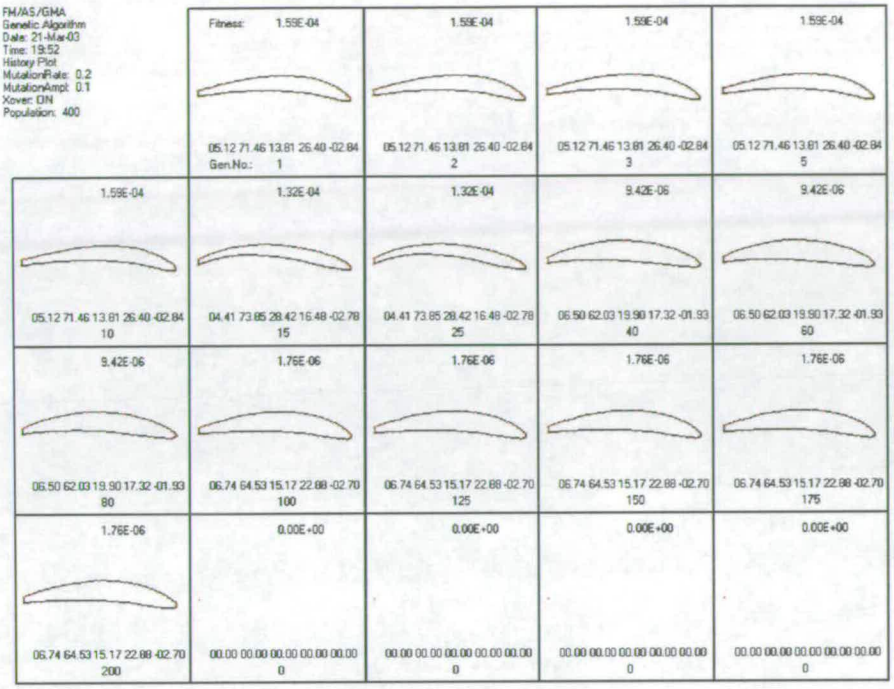


Figure A-19 **Plot of Best Individual for each Generation for Run G1**

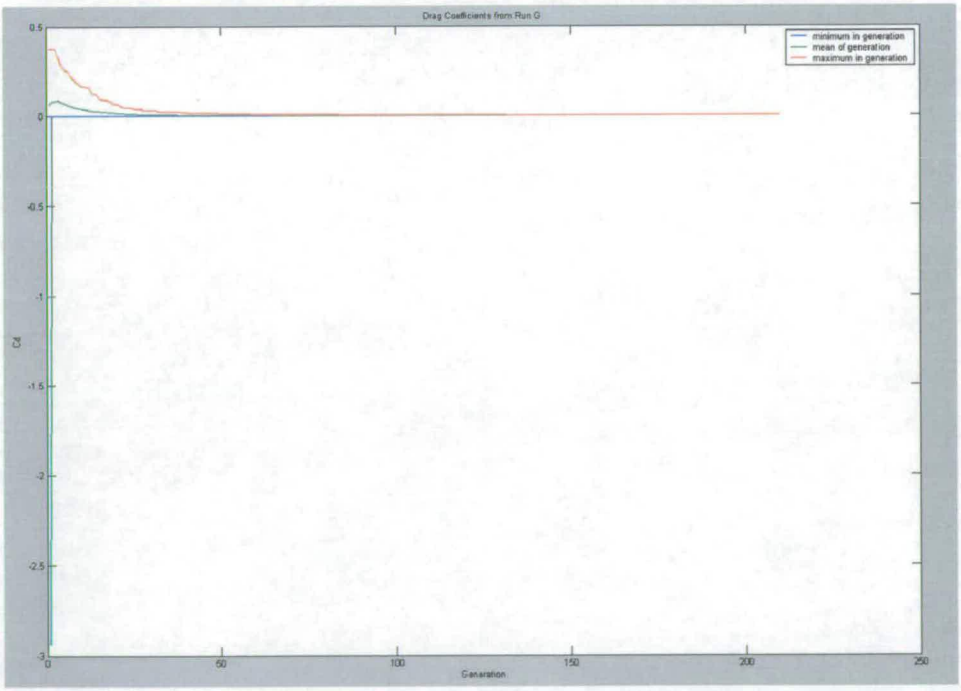


Figure A-20 **Graph of C_d against Generation for Run G1**

A.1.8 Experiment H

This run repeated of Experiment G, but with the maximum camber angles restricted to 10°, in order to avoid the problems with the faulty fluid analysis at high camber angles and negative attack angles.

Fitness	C_d/C_l	
Population size	400	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.05	
Parameter	**Min**	**Max**
Thickness from camber line (% of chord length)	3	8
Position of maximum thickness (% of chord length)	55	82
Camber angle, theta0, at trailing edge (deg)	0	10
Camber angle, theta1, at leading edge (deg)	0	10
Angle of attack (deg)	-4	4

The experiment was repeated ten times with varying seeds for the random number generator. The best solution for each run was:

Run Title	H1	H2	H3	H4	H5
Thickness from camber line (% of chord length)	7.87	8.00	8.00	8.00	8.00
Position of maximum thickness (% of chord length)	73.84	74.11	55.66	65.12	73.91
Camber angle, theta0, At trailing edge (deg)	10.00	10.00	10.00	10.00	10.00
Camber angle, theta1, At leading edge (deg)	10.00	10.00	10.00	10.00	10.00
Angle of attack (deg)	-0.31	-0.14	-0.32	-0.49	-0.23
Fitness (10^{-2})	8.23	8.26	8.97	8.77	8.21
C_l/C_d	12.15	12.11	11.15	11.40	12.18
Number of Evaluations	8400	8400	8400	8400	8400

Run Title	H6	H7	H8	H9	H10
Thickness from camber line (% of chord length)	8.00	8.00	8.00	8.00	8.00
Position of maximum thickness (% of chord length)	55.66	65.35	73.90	73.81	73.93
Camber angle, theta0, At trailing edge (deg)	10.00	10.00	10.00	10.00	10.00
Camber angle, theta1, At leading edge (deg)	10.00	10.00	10.00	10.00	10.00
Angle of attack (deg)	-0.32	-0.45	-0.23	-0.20	-0.23
Fitness (10^{-7})	8.97	8.78	8.21	8.22	8.21
C_l / C_d (10^6)	11.15	11.39	12.18	12.17	12.18
Number of Evaluations	8400	8400	8400	8400	8400

The average C_l / C_d found was 11.806 with a standard deviation of 0.467.

The runs converged to two slightly different shapes. All runs converged to aerofoils with a maximum thickness of 8.00% (except H1 with 7.87%), and camber angles, theta1 and theta2, of 10°. However, runs H1, H2, H5, H8, H9 and H10 produced aerofoils with the position of maximum thickness at between 73.81% to 73.93% of chord length, attack angle of between -0.14° and -0.31°, and lift-drag ratio between 12.11 and 12.18. Runs H3, H4, H6 and H7 produced an aerofoil with the position of maximum thickness at between 55.66% to 65.35% of chord length, attack angle of between -0.32° and -0.49°, and a worse lift-drag ratio between 11.15 and 11.39. This would seem to indicate that runs H3, H4, H6 and H7 had converged prematurely to a sub-optimal solution.

It should be noted that all these solutions were considerably worse than the aerofoil profile found in Experiment F which had a fitness of 3.77e-2 (C_l / C_d = 26.5), despite the fact that this experiment had a considerably larger search space (attack angle was not included in Experiment F). This is discussed further in Section 3.5. Figure A-21 shows the best individual from a number of generations throughout Run H1.

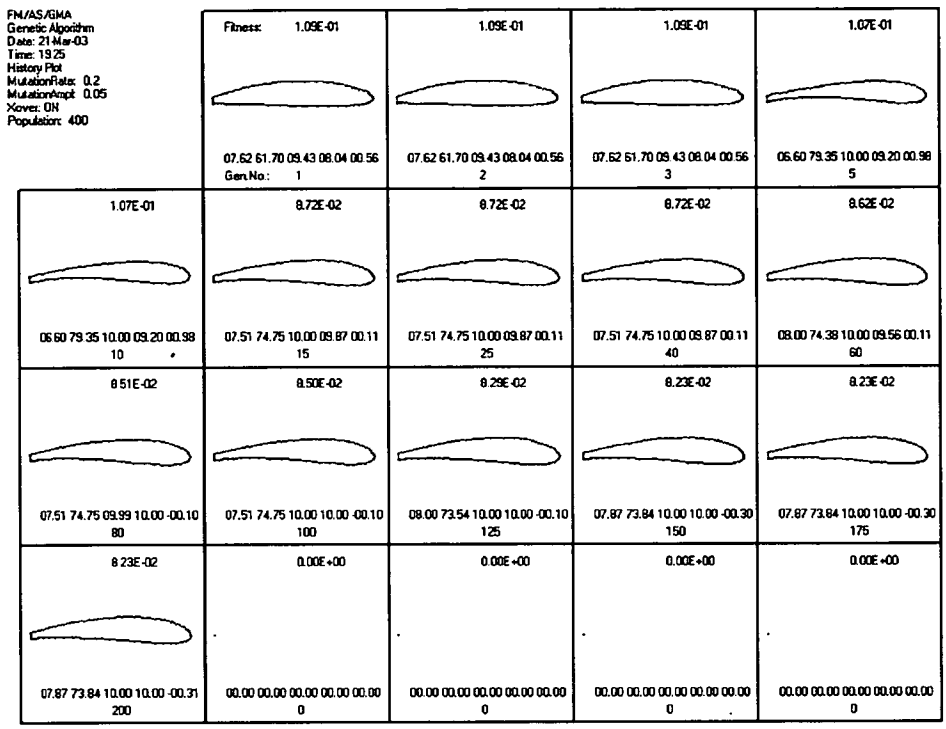


Figure A-21 Plot of Best Individual for each Generation for Run H1

A.2 Unsmoothed Bézier Representation

A.2.1 Experiment I

An experiment was undertaken using the unsmoothed Bézier representation. The population was initialised by perturbing the control points from those of a given aerofoil profile. The size of this perturbation was user-defined as a proportion of the range of the relevant parameter (a value of 0.5 was used for Experiment I). The given aerofoil profile and specified bounds are given below:

Parameter	Lower Bound	Sample Aerofoil	Upper Bound
Control Point 1 Radius (m)	1	1	1
Control Point 1 Angle (rad)	-0.5 π	-0.5 π	-0.5 π
Control Point 2 Radius (m)	0.2	0.25	0.5
Control Point 2 Angle (rad)	0.4 π	0.42 π	0.44 π
Control Point 3 Radius (m)	0.2	0.25	0.5
Control Point 3 Angle (rad)	0.31 π	0.32 π	0.35 π
Control Point 4 Radius (m)	0.2	0.25	0.5
Control Point 4 Angle (rad)	0.23 π	0.25 π	0.27 π
Control Point 5 Radius (m)	0.2	0.25	0.5
Control Point 5 Angle (rad)	0.14 π	0.16 π	0.18 π
Control Point 6 Radius (m)	0.2	0.25	0.5
Control Point 6 Angle (rad)	0.06 π	0.08 π	0.10 π
Control Point 7 Radius (m)	0.2	0.25	0.5
Control Point 7 Angle (rad)	-0.02 π	0.00 π	0.02 π
Control Point 8 Radius (m)	0.1	0.2	0.3
Control Point 8 Angle (rad)	-0.1 π	-0.08 π	-0.06 π
Control Point 9 Radius (m)	0.1	0.2	0.3
Control Point 9 Angle (rad)	-0.18 π	-0.16 π	-0.14 π
Control Point 10 Radius (m)	0.1	0.2	0.3
Control Point 10 Angle (rad)	-0.27π	-0.25 π	-0.23 π
Control Point 11 Radius (m)	0.1	0.2	0.3
Control Point 11 Angle (rad)	-0.35 π	-0.33 π	-0.31 π
Control Point 12 Radius (m)	0.1	0.2	0.3
Control Point 12 Angle (rad)	-0.44 π	-0.42 π	-0.40 π

Experiment I used the following genetic algorithm parameters:

Fitness	C_d / C_l	
Population size	400	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	1.0	

Figure A-22 shows the best individuals generated through the run. The best individual had an unrealistic fitness of 8.57e-5 (C_l/C_d of 11700). Numerous runs were undertaken with various initial base aerofoil profiles, each produced similarly unrealistic results to these.

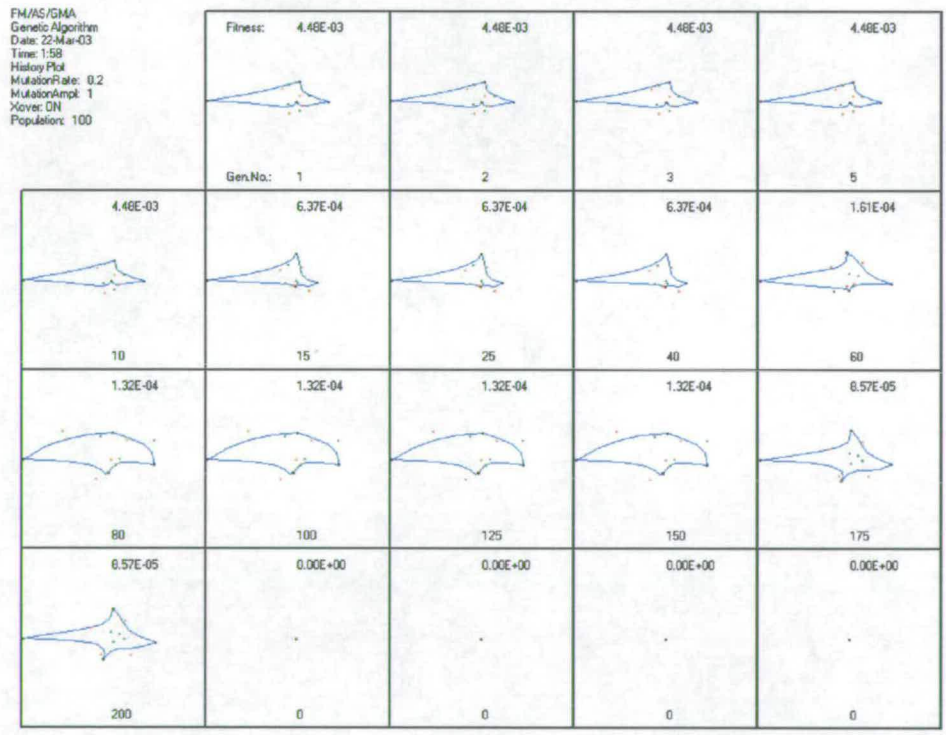


Figure A-22 Plot of Best Individual for each Generation for Run I1

A.3 Smooth Bézier Representation

A.3.1 Experiment J

It was thought that one possible cause of the analysis problems encountered in Experiment I was the presence of 'kinks' in the aerofoil shapes. Therefore, a second Bézier representation was tried in which C_1 continuity was imposed between the Bézier curves except at the trailing edge, as described in Section 3.3.1.3.

The population was initialised by perturbing the control points from those of a given aerofoil profile within a set of specified bounds, in the same was as described in Section A.2.1 for Experiment I. It should be noted that, with this representation, the position of control points 4,8 and 12 are not set independently of the other control points, but instead are 'repaired' to lie halfway between adjacent control points.

The following run was undertaken using the smooth Bézier representation.

Fitness	C_d/C_l	
Population size	400	
Proportion of population to breed per generation	0.1	
Mutation rate	0.2	
Mutation amplitude	0.01	

Figure A-23 shows the best individuals generated through the run. The best individual had an unrealistic fitness of 5.55e-8 (C_l/C_d of 1.8e6). Again, numerous runs were undertaken with various initial base aerofoil profiles, each produced similarly unrealistic results to these.

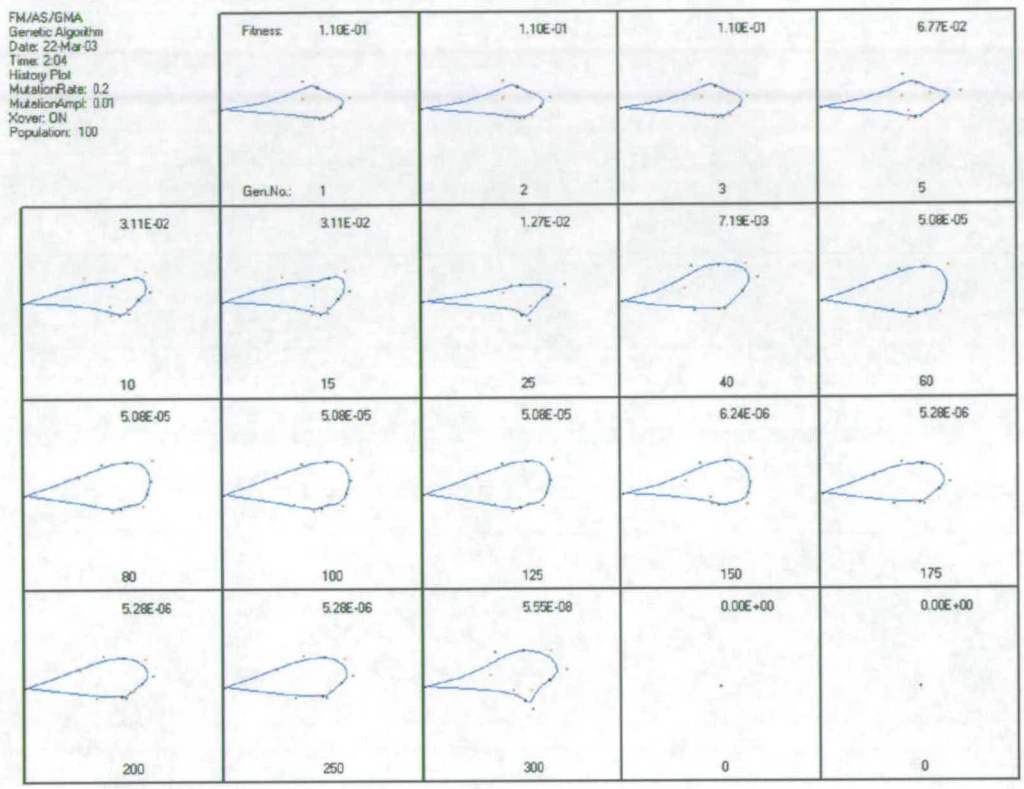


Figure A-23 Plot of Best Individual for each Generation for Run J1