UNIVERSITY OF SOUTHAMPTON

An automated Shape Grammar approach to structural design description and optimisation

by
Alkin Nasuf

A thesis submitted in partial fulfilment for the degree of Doctor of Philosophy

in the
Computational Engineering and Design Group
Faculty of Engineering and the Environment

October 8, 2013

Contents

Declaration of Authorship

I, Alkin Nasuf, declare that the thesis entitled, "*An automated Shape Grammar approach to structural design description and optimisation*" and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

Signed: _____

Date: _____

Acknowledgements

I would like to express my gratitude to Dr. Atul Bhaskar, for his sound advice, warm support and thoughtful guidance. I appreciate his patience and assistance throughout my thesis writing period. He was always available to pick me up whenever I stumbled during the course of my doctoral research. This thesis would not have been materialised without him.

I am grateful to Prof. Andy J. Keane for sharing his wisdom and great expertise in the area of computational engineering. His professional thinking and intelligent comments have helped me to work in the right direction.

A special thanks goes out to Dr. Ivan I. Voutchkov, without whose motivation and encouragement I would not have considered a graduate career in computational engineering.

I am in great debt to my wife, Maya, for her endless patience and understanding in my most difficult times.

Last but not the least I wish to thank my colleague and best friend Sanjay Pant for all the fruitful discussions and discourses.

I would also like to deeply acknowledge the generosity of Engineering and Physics Science Research Council (EPSRC), UK for the grant EP/E004547/1: *The role of topology and shape in structural design.* I recognize that this research would not have been possible without such a generous financial assistance.

UNIVERSITY OF SOUTHAMPTON

Abstract

COMPUTATIONAL ENGINEERING AND DESIGN GROUP
FACULTY OF ENGINEERING AND THE ENVIRONMENT

Doctor of Philosophy

by Alkin Nasuf

The main objective of this research is to develop and evaluate an automated design description and optimisation framework based on the Shape Grammar (SG) syntax. In particular, an algorithmic methodology to automate the SG syntax using evolutionary intelligence is proposed. The proposed automation of SG is achieved by mapping the genetic information provided by a Genetic Algorithm (GA) to context-free grammar rules by a means of Backus-Naur Form (BNF) syntax known as Grammatical Evolution (GE). GE is an efficient optimisation tool, which can be used in a variety of optimisation problems. First, its use in single and multi-objective optimisation of mathematical functions is demonstrated. Several techniques for synthesis of variables using specific BNF syntaxes are proposed. The results obtained from numerical experiments are compared with those obtained using a standard GA. Interestingly, the results show a notable improvement in the convergence speed over the standard GA for the functions tested. This observation surpassed expectations, since the GE is based on the GA.

The use of GE is then extended to automate the SG syntax by deriving a grammar based design shape description and optimisation framework. To evaluate its efficacy and to demonstrate the concept, this framework is applied to two distinctive classes of problems frequently encountered in engineering practice. The first class of problems is related to shape descriptions and optimisation aspects of structural design. A specific BNF syntax is developed for planar shapes which makes use of four SG rules with arc primitives of variable size given by a radius and an angle of rotation. These SG rules are then used in the synthesis of piecewise parametric curves for the shape description and optimisation of a planar crane hook. The experimental results show superiority in convergence speed when compared to shape optimisation of the same problem based on Non-Uniform Rational B-Spline (NURBS) combined with a GA search strategy.

The second class of problems considered here relates to the topology description and optimisation of planar trusses. Several BNF syntaxes are developed to achieve simultaneous topology, size and configuration optimisation. The experimental results thus obtained show good agreement with the results reported in the literature using an alternative truss optimisation method based on GA. Furthermore, by using the proposed truss description and optimisation method the computational expense is significantly reduced.

The proposed design description and optimisation framework based on the SG syntax is a fast and efficient design exploration tool. The successful application of this proposed framework combining SG with design exploration in a range of structural problems validates the proposed idea.

Publications

The following research publications have arisen out of this work:

(i) Nasuf, A, Bhaskar, A and Keane, A (2009) Shape exploration for structural efficiency. Proceedings of the 17th UK Conference on Computational Mechanics (ACME-UK) Nottingham, UK, Association of Computational Mechanics, 315-318.

(ii) Nasuf A, Bhaskar A, Keane A (2011) Multi-objective optimisation using grammatical evolution. Proceedings of the Anniversary Scientific Conference: 40 Years Department of Industrial Automation, UCTM, Sofia, Bulgaria, pp 20–25

(iii) Nasuf A, Bhaskar A, Keane A (2013) Grammatical evolution of shape and its application to structural shape optimisation. *Structural and Multidisciplinary Optimization*, DOI:10.1007/s00158-013-0890-0

List of Figures

List of Tables

Acronyms

BEM Boundary Element Method

BNF Backus-Naur Form

CAD Computer Aided Design

EAP Evolutionary Automatic Programming

EBNF Extended Backus-Naur Form

FE Finite Element

FEA Finite Element Analysis

FEM Finite Element Method

FFD Free-Form Deformation

TPS Thin-Plate Spline

ESO Evolutionary Structural Optimisation

GA Genetic Algorithm

GE Grammatical Evolution

GP Genetic Programming

SG Shape Grammar

NSGAII Non-dominated Sorting Genetic Algorithm II

NURBS Non-Uniform Rational B-Spline

Symbols

β	Codon length	–
γ	Shear strain	–
Γ	Rotation matrix	–
δ	Displacement	mm
ϵ	Strain	–
ζ	Eigenvalue	–
η	Search step of normalised variables	–
θ	Angle of rotation	$^\circ$
κ	Search step of optimisation variables	–
λ	Number of codons	–
μ	Binary string length	–
ν	Poisson's ratio	–
ξ	Number of grammar rules in a BNF expression	–
ρ	Density	$\mathrm{kg} \times \mathrm{mm}^{-3}$
σ	Stress	Pa
σ^{vm}	von Mises stress	Pa
τ	Shear stress	Pa
υ	Eigenvector	–
χ^T	Translation vector	–
ψ	Scale parameter	–
$\langle \mathrm{A} \rangle, \langle \mathrm{B} \rangle, \langle \mathrm{C} \rangle, \cdots, \langle \mathrm{Z} \rangle$	Non-terminal set of grammar primitives	–
$\mathrm{A}, \mathrm{B}, \mathrm{C}, \cdots, \mathrm{Z}$	Terminal set of grammar primitives	–
A	Area	mm^2
a	Cross-section area	mm^2
$\bar{a}$	Normalised cross-section area	–
$B(u)$	Bernstein polynomial	–
b	Beam cross-section area	mm^2
$C(u)$	Parametric Bézier curve	–
C_n	Parametric geometry continuity	–
E	Young's modulus	Pa
F	Force	N
$\mathbf{f}$	Applied loads	–
G	Share modulus of elasticity	Pa
H	Breadth	mm

$J(\Phi)$	Thin-Plate Spline energy bending function	–
$\mathbf{K}$	Stiffness matrix	–
k	Curve knot weight	–
L	Linguistic grammar sentence	–
l	Length	mm
M	Total structure mass	–
q	GA population member	–
O	Optimisation objective	–
o	Circular arc centre	–
P	Surface pressure	$\mathrm{N} \times \mathrm{mm}^{-2}$
p	Control point	–
p_{c}	Crossover probability	–
p_{m}	Mutation probability	–
$R(u)$	Rational basis function	–
R	Radius	mm
S	Grammar sentence	–
$S(u, v)$	Parametric surface	–
t	Thickness	mm
U	Parametric piecewise curve by arcs	–
$\mathbf{u}$	Displacements at nodes	–
$\mathrm{V_P}$	Vocabulary set	–
$\mathrm{V_N}$	Non-terminal vocabulary set	–
$\mathrm{V_T}$	Terminal vocabulary set	–
W	Width	mm
$\mathbf{w}$	Optimisation objective weights	–
X_{ffd}	Free-Form Deformation function	–

To my wife Maya...

Chapter 1

Introduction

Design optimisation covers a range of multidisciplinary problems in engineering. This is a process where design parameters are systematically altered so as to improve the performance of designs characterised by some given measures of merit subject to prescribed constraints. Structural design optimisation find extensive application in various industrial sectors such as Aerospace, Automotive, Biomedical, Aeronautics, Marine, Power, Civil engineering, etc. A common objective of structural optimisation is to reduce the weight of a given structure by optimising its shape and topology, while maintaining or improving its structural properties. In the Aerospace & Aeronautics industries, for example, strong and light structures can make a significant difference in the overall performance of the final product, where the overall weight of the vehicle strongly correlates with its fuel consumption. Usually, weight reduction is achieved by removing unnecessary material from low stress areas of the structure – thus changing the initial design. This thesis is about the design description and optimisation aspects of structural problems. The complexity of structures determines the choice of design description and optimisation methods. The existing methods mainly differ in the adopted design manipulation strategy based on various parameterisation techniques. The current trend is to reduce the computational resources needed for solving particularly complex structural design problems. This can be done by development of new innovative state-of-art techniques which will follow the trend or even provide new avenues.

The rapid development of computational machines, in recent years has led to significant advances in the computational engineering field. Previously, designs were analysed using approximations, idealisations and analytical formulations, when possible. Analytical methods are only applicable to simple engineering problems

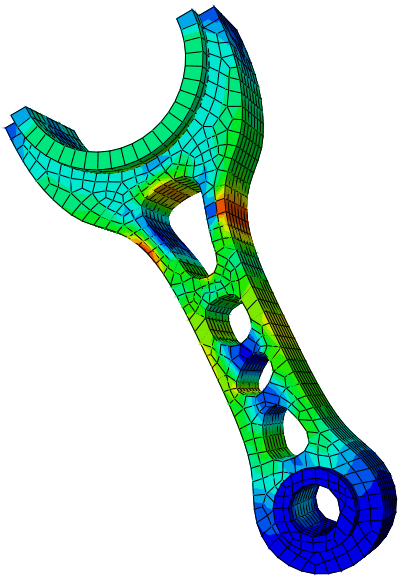


FIGURE 1.1: Structural finite element 3D CAD model – a connecting rod

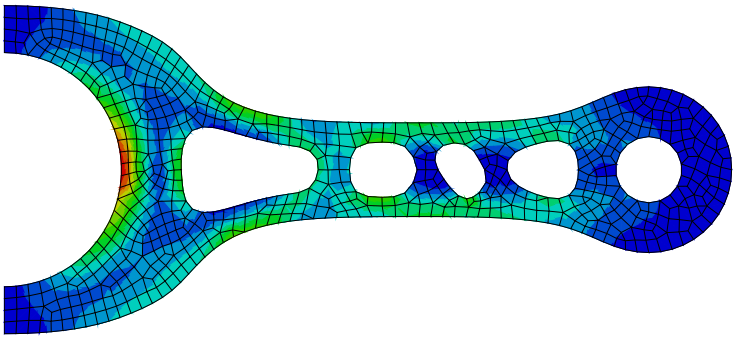
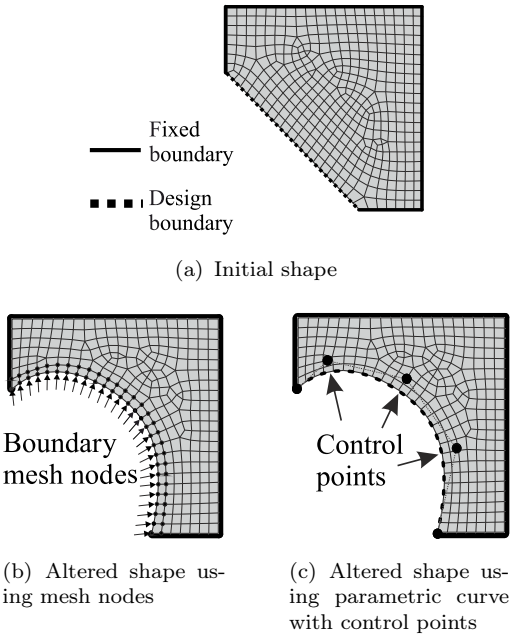


FIGURE 1.2: An example of shape and topology in structural design

only which is very restrictive for practical situations. The use of analytical methods in complex engineering problems often requires unrealistic simplifications about the problem. The extensive use of analytical methods further led to the development of a host of computational methods. Following the technological advancement in computer hardware, this played a fundamental role in the development of modern computational methods such as the Finite Element Method (FEM) (Zienkiewicz et al., 2005). Advances in such methods have been especially noticeable in the area of computational mechanics, where the FEM has had a key role. In general, the FEM is a method for calculating the elastic response of solids by dividing the geometry into finite elements. An example of a structural finite element 3D Computer Aided Design (CAD) model is shown in Fig. 1.1. The geometry is divided into finite elements using a mesh. During a Finite Element Analysis (FEA) (Cook et al., 1989) the elements interact with each other based on physical laws of equilibrium and compatibility. In computational mechanics,



(a) Initial shape

(b) Altered shape using mesh nodes

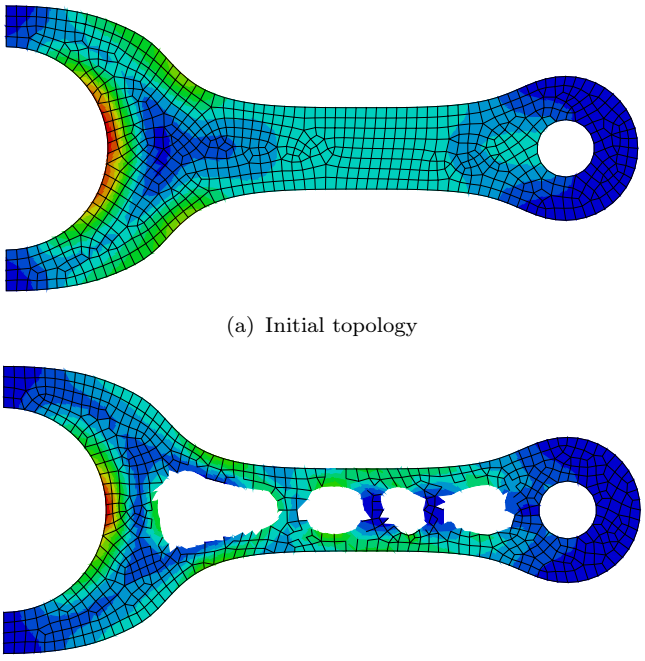
(c) Altered shape using parametric curve with control points

FIGURE 1.3: An example of two distinctive geometry based shape manipulation techniques

the FEM is used to calculate the mechanical response of real structures. The improvement of FEM theory in the last four decades has led to remarkable advances in the field of design optimisation.

Design optimisation of structures has two important aspects. These are the overall shape of the structure and its topology. Shape optimisation is concerned with the overall boundary geometry of structures, while topology optimisation is concerned with topological features of structures that are related to the connectivity of material, see Fig. 1.2.

Shape optimisation. Kendall provided the following definition of shape – "Shape is all geometrical information that remains when location, scale and rotational effects are filtered out from any geometrical form" (Kendall, 1989). In structural design and many other shape related problems, size is a matter of scaling and is trivial. Location and rotation, on the other hand, are rather spatial than shape descriptive parameters. The initial shape in most shape optimisation studies is usually a rough geometry template, which is far from being structurally optimal. The initial design domain of a structure is often divided into designable and non-designable boundaries as illustrated in Fig. 1.3(a). The geometry of non-designable boundaries remains fixed – the designable boundary can be manipulated during an optimisation process. Often, systematic shape alteration of the designable boundary is achieved by first parametrising its shape, and then manipulating the parameters.



(a) Initial topology

(b) Altered topology

FIGURE 1.4: Topology optimisation driven by the local stress level

This is illustrated in Fig. 1.3(b) and Fig. 1.3(c), where two distinctive geometry based shape manipulation techniques are presented:

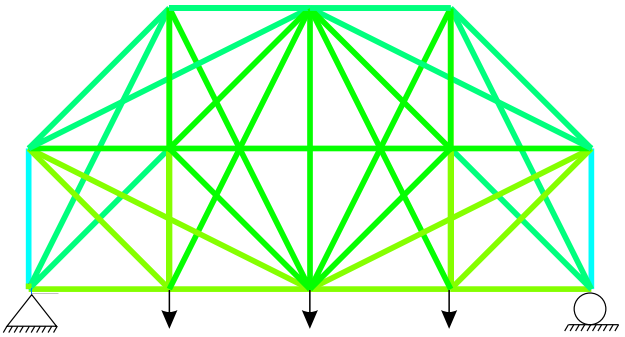
1. The first technique uses the boundary mesh nodes of the Finite Element (FE) mesh (Fig. 1.3(b)).

2. The second technique relies on parametric curves or surfaces (Fig. 1.3(c)). Such curves or surfaces are commonly used in CAD modelling for sketching purpose and are usually described mathematically by means of B-spline basis functions.

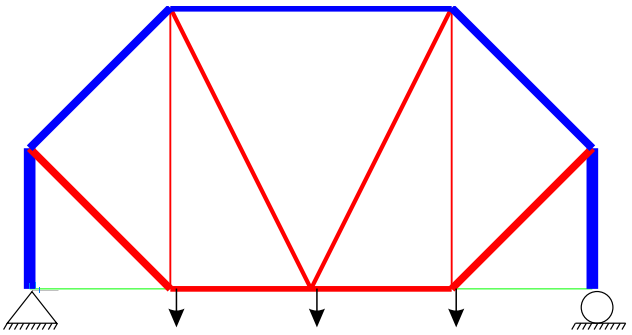
Topology optimisation. Topology optimisation is mainly concerned with optimising the internal material connectivity of structures so as to reduce their weight while keeping or improving the mechanical performance. From a structural design perspective, there are two distinctive classes of topology optimisation problems:

1. The class of problems relates to optimal topological connectivity of solid parts in an object. For example, the meshed structural CAD model shown in Fig. 1.4(a) represents a solid object – a rod. The initial design of the rod is continuous. Of interest is the desire to reduce its weight by removing unnecessary material, thus making new topological features in the structure.

An example of an optimised rod design with topological features due to removed elements is shown in Fig. 1.4(b). The approach here is to first analyse the structure given external load and identify regions of low stress. The elements below a specified threshold of stress are then removed from the mesh (and hence from the design). This process can be carried out iteratively in several steps of stress analysis followed by element removal. Occasionally, the process of element removal is supplemented by element addition. There are a range of algorithms that implement the spirit of this idea – referred generally as Evolutionary Structural Optimisation (ESO). Algorithms that allow element removal as well as element addition are called Bi-directional Evolutionary Structural Optimisation (BESO) (Querin, 1997; Chen et al., 2002; Young et al., 1999; Huang and Xie, 2008). Sometimes (particularly for 2D structures), the thickness of the elements is used as a design variable. In such situations, one uses a threshold thickness for removing/retaining an element in the design. The drawback of this topology optimisation strategy is that the number of free variables is as many as the number of elements in the mesh. This increases the computational effort during optimisation significantly.



(a) Initial truss topology



(b) Optimal truss topology

FIGURE 1.5: An example of initial and optimised truss design. The free variables of optimisation are the cross-section areas of structural member. The line thickness represent the size of cross-section

2. The same strategy can be applied to another class of topology optimisation problems such as the one shown in Fig. 1.5(a). Here, the free variables of optimisation are given as cross-sectional areas of the structural members. The optimisation starts similarly with an initial structural design containing all possible members. There are usually two ways of driving the ESO strategy in such cases. Most commonly, a prescribed minimum stress level is used to either accept or reject a structural member in the design. Sometimes this rejection criterion is rendered more graduated by altering the cross-sectional area of each member – the areas being reduced for members with lower stress. A criterion based on a threshold minimum area is then used to eliminate (or retain) a member within the design. An example of optimal connectivity derived from the initial network of structural members is shown in Fig. 1.5(b). Note that in this figure the cross-sectional area of each member is represented as line thickness. This strategy shares many features with the previously known topology optimisation strategies i.e. the high number of optimisation variables. The strength of these strategies, however, is the simplicity of the algorithm. On the other hand, the design evolution often tends to "get trapped" into local optima, thus missing out high quality designs – obtainable by more expensive and often stochastic methods (Bendsøe and Kikuchi, 1988; Deb and Gulati, 2001).

The present thesis focuses on design description by means of SG and its further use in structural optimisation problems. The structure of the thesis is arranged in the following order. In Chapter 2, existing methods in the computational design description and optimisation field are reviewed. In Chapter 3, a design description and optimisation framework based on GE is proposed. The proposed framework benefits from recent developments in Genetic Programming (GP) and particularly the Grammatical Evolution (GE). In Chapter 4, an example of a bi-objective design optimisation of a planar crane hook shape using the proposed design synthesis and optimisation framework is presented. The results obtained for crane hook designs are compared with those using an alternative and more conventional shape description and parameterisation method based on NURBS and a GA search strategy. The proposed framework is further applied to a new class of structural problems concerned with the topology optimisation of trusses in Chapter 5. In this chapter, a performance comparison is carried out, where the efficacy of the proposed method is compared to that of existing alternative truss optimisation methods based on GA. The experimental results thus obtained indicate that the

proposed truss optimisation method based on GE converges faster than the available GA based truss optimisation methods. Another application of the proposed framework is presented in Chapter 6, where structural design problems related to networks of plate stiffeners and their optimal topology are considered. Finally conclusions are drawn in Chapter 7 where promising lines for future work are also indicated.

Chapter 2

Literature review

The purpose of this chapter is to introduce relevant tools frequently used in the areas of computational linguistics, artificial intelligence, computer science, and logic and to explore their application to problems in structural design. Also other tools commonly used for shape manipulation are presented with illustrations that are often used manually. An automated framework to achieve shape alteration required within an optimisation loop is developed with the aim of reducing the lead time in design search studies. In particular, Grammatical Evolution (GE), Shape Grammar (SG), Genetic Programming (GP) and Evolutionary Automatic Programming (EAP) approaches have been proposed. The framework is further developed in the following chapters.

This chapter conducts a survey on some existing engineering methods and tools for design description and optimisation in the following areas:

- Stress analysis tools – Finite Element Method (FEM).
- Parametric curves and surfaces – Shape Grammar (SG), B-splines, Non-Uniform Rational B-Spline (NURBS).
- Design synthesis tools – Phrase-structure grammar, Backus-Naur Form (BNF) syntax, Shape Grammar (SG).
- Evolutionary computation tools – Genetic Algorithm (GA), Non-dominated Sorting Genetic Algorithm II (NSGAII), Genetic Programming (GP) and Grammatical Evolution (GE).
- Shape optimisation methods.

- Truss optimisation methods.

2.1 The Finite Element Method

The FEM is a numerical procedure for simulating mechanical, thermal, electrical and many other aspects of physics of the problem in various engineering contexts (Cook et al., 1989). The method was initially developed for stress analysis problems in solid mechanics. The fundamental procedure of FEM is to divide a solid body into finite elements and then apply physical laws of equilibrium and compatibility to each of the elements which result in a set of algebraic equations and calculate its response.

During FEA procedure the stiffness matrix is used to compute physical responses of simulated objects at finite elements nodes. The physical response of an object under loading conditions is given by $\mathbf{Ku} = \mathbf{f}$, where $\mathbf{f}$ is a vector of applied loads and $\mathbf{u}$ is the response of simulated object. Given the material properties of each element, the displacements can be then used to calculate the stresses and strains.

The overall behaviour of the structure is not the same as the actual one and many assumptions are made. However, the method could be used quite satisfactorily for many applications. An important part of FEA is the initial set-up procedure – it has a significant impact on authenticity of simulation results. Some of the important FEA set-up parameters are:

- Mesh size. In general, the accuracy of FEA simulation increases with the discretisation resolution (number of elements used), however the required computational expense is also strongly related to the number of elements. This is because the size of the stiffness matrix $\mathbf{K}$ becomes larger due to increased number of nodes. Usually, the mesh size is defined arbitrarily based on engineering judgement about the problem and convergence, therefore it is problem specific. Nevertheless, a preliminary study can be also conducted to determine the optimal mesh size. Mesh nodes can be seeded uniformly across the object or non-uniformly. A frequently used technique is to use a non-uniform mesh seed in order to speed-up the FEA simulation by controlling the number of elements. The advantage of the non-uniform mesh is that it allocates the computation resources to simulation of critical regions such as boundaries of high curvatures e.g. sharp edges which are areas of high stress. This is achieved by refining the mesh in those regions at the expense of less

important regions such as internal regions or linear boundaries of low stress where a coarse mesh could provide acceptable results. There are a range of non-uniform mesh seed algorithms such as the adaptive mesh technique which are out of the scope of this brief FEA introduction.

- Material properties. Proper definition of material properties is crucial for obtaining realistic results. The Young's modulus of elasticity E and Poisson's ratio ν are two of the most commonly required material properties during modelling of linear elastic solids. FEA of non-linear elastic or elasto-plastic solids requires definition of additional non-linear stress-strain relationships. There are numerous other physical properties assigned to elements based on the type of FEA e.g. heat transfer coefficient, conductivity, electrical conductivity, coefficient of viscosity, etc.

- Element types. There are families of elements types for simulation of the most frequently encountered engineering problems. Element types are selected based on information about the simulation problem being faced by the user. Commercially available software packages such ABAQUS and ANSYS maintain rich element database libraries, but also provide tools for creating custom element types. Element types mainly differ in the number of construction nodes, degrees-of-freedom at nodes, integration formulations and the underlying physics of the problem.

- Boundary conditions. They are used to define the interaction of the simulated object at its boundaries within a surrounding virtual world. The boundary conditions are usually applied to individual nodes or groups of boundary nodes. In solid mechanics, the boundary conditions are divided into constraints and loads. The constraints impose displacement and rotation at the nodes, while loads are used to impose an external forces.

2.2 Parametric curves and surfaces

An accurate definition of parametric curves or surfaces, frequently used to represent free-form shapes (free of form asymmetric geometry), is given in (Piegl and Tiller, 1997). The parametric curve or surface possesses a natural direction of traversal in some complete space (for a surface $S(u,v) = [x(u,v), y(u,v), z(u,v)]$ and $(a < u < b)$; $(c < v < d)$, see Fig 2.1), while the implicit curve or surface does not. In the parametric shape representation, $S(u,v)$ can be seen as parametric

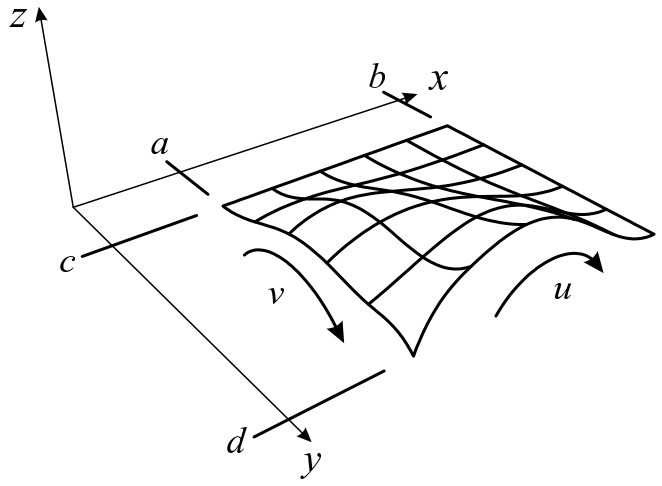


FIGURE 2.1: An example of parametric surface in complete space $[a, b]$ and $[c, d]$ with natural direction of traversal u and v

basis function with unknown coefficients u, v in intervals $[a, b]$ and $[c, d]$. These unknown coefficients can be characterised as being the location, scale and rotation invariant shape representative parameters. Note that in a 2D space, the shape is represented by a parametric basis function $C(u)$ with natural direction of traversal for the interval $a < u < b$. The parametric basis functions are superior tools compared to traditional parameterisation techniques which fail to parameterise free-form geometries with reasonable number of variables. They are commonly used to represent and manipulate complex free-form shapes. The research on basis functions for engineering applications was pioneered by Pierre Bézier and Paul de Casteljau who developed the well known Bézier curve (Piegl and Tiller, 1997). Each Bézier curve is characterised by a degree of curvature n and the number of

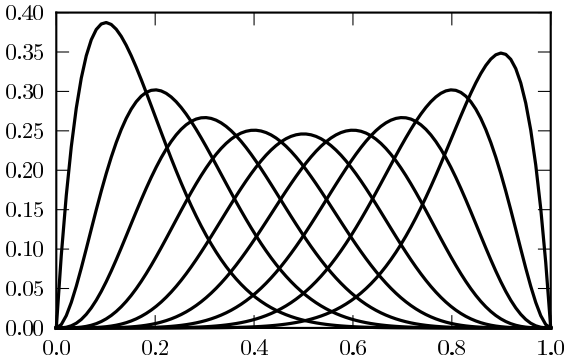


FIGURE 2.2: An illustration of Bernstein polynomial of degree $n = 10$ given by Eq. 2.2

control points p_i (basis function weights) as follows

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) p_i;\ 0 \leq u \leq 1, \tag{2.1}$$

where $B_{i,n}(u)$ is Bernstein basis polynomial

$$B_{i,n}(u) = \left(\frac{n!}{i!(n-i)!} \right) u^i (1-u)^{n-1}, \tag{2.2}$$

of degree n evaluated for the interval $0 \leq u \leq 1$. A sketch of the Bernstein polynomial of degree 10 is shown in Fig. 2.2.

Although the Bézier curve is a powerful tool for representing highly complex free-form shapes, it has two important limitations:

1. It is unable to represent some simple but important curves and surfaces such as circles, ellipses, cylinders, etc. that are frequently used in engineering practice. This limitation is overcome by employing rational basis functions of the type

$$C(u) = \sum_{i=0}^{n} R_{i,n}(u) p_i, \tag{2.3}$$

where

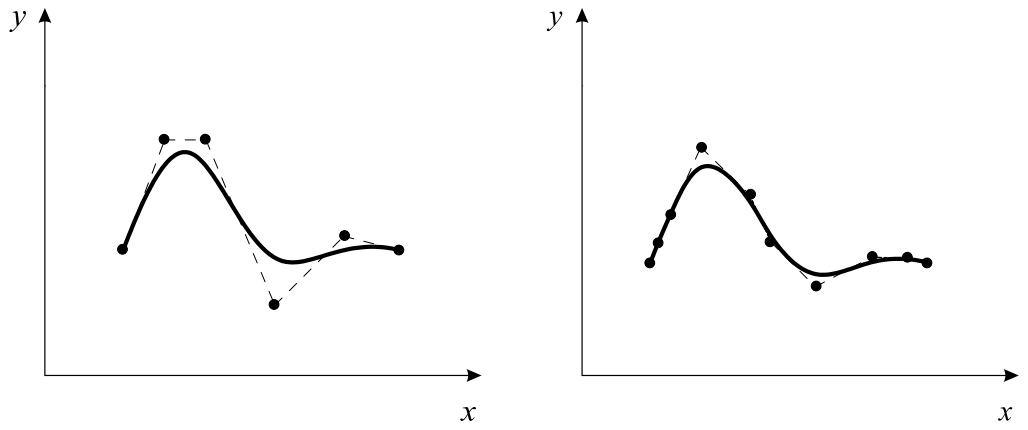
$$R_{i,n}(u) = \frac{B_{i,n}(u) w_i}{\sum_{j=0}^{n} B_{j,n}(u) w_j} \tag{2.4}$$

and k_i are the knot weights for each of the control points in the rational Bézier curve.

2. Representation of some complex shapes require a high number of control points p_i, which inevitably leads to construction of high degree Bézier curves. In (Piegl and Tiller, 1997) the high degree Bézier curves are characterised as "inefficient to process and numerically unstable". One of the elegant ways to overcome this limitation is to employ B-spline basis functions.

B-Splines are curves created by piecewise union of several low degree polynomial segments such as cubic or quadratic Bézier curves. The construction of B-splines reduces the number of control points to reasonable levels and consequently the degree of curvature at the expense of an additional parameter known as the knot vector. The knot vector is important for each B-spline, as its role is to maintain a desired level of continuity between curve segments.

One of the most frequently used parametric curves, in engineering design, is the NURBS (Piegl and Tiller, 1997). Each NURBS is a piecewise curve derived by



(a) 6 control points (b) 10 control points

FIGURE 2.3: An example of two NURBS parameterised by 6 and 10 control points respectively. Note that the two curves have equal shape

uniting several Beziér curves which maintain curvature continuity. A NURBS curve is capable of representing simple shapes (e.g. circles, ellipses, cylinders, etc.) as well as complex shapes with a reasonable number of parameters (co-ordinates of control points). A comparative discussion over the advantages and disadvantages of employing NURBS rather than Bézier curves is given in (Piegl and Tiller, 1997). Each NURBS curve is parameterised by a finite set of control points. In Fig. 2.3, a NURBS curve is parameterised by 6 (Fig. 2.3(a)) and 10 (Fig. 2.3(b)) control points, respectively.

2.3 Shape deformation tools

Shape manipulation is fundamental operation in every design process. Often existing shapes are deformed and transformed in order to improve the performance of design. NURBS is convenient and versatile shape description and manipulation tool frequently used in engineering design. Some of the factors that contribute to popularity of NURBS are:

(i) Intuitive shape manipulation using control points positioned in space.

(ii) Possibility of altering the complexity of shape by increasing or decreasing the number of control points.

(iii) Possibility of parametrising almost any shape (excluding sharp corners) with sufficient number of control points within specified tolerance.

(iv) An ability to maintain accurate, smooth and natural continuity of represented shape.

Design optimisation requires an automated manipulation of shape parameters driven by an optimisation algorithm. The advantages listed above are mainly valid during interactive design search where the shape is altered by an experienced modeller manually.

Some of the drawbacks of using NURBS in automated design optimisation fashion are as follows:

1. The number of control points (optimisation variables) increases proportionally to complexity of design. This on the other hand increases convergence time of optimisation.
2. Difficulties in maintaining natural continuity between control points – automated manipulation of several control points without taking into account any movement relationships is likely to result in wavy and unrealistic designs.
3. Definition of boundaries for the optimisation variables is not straightforward procedure.

Thin-Plate Spline (TPS) deformation and Free-Form Deformation (FFD) are similar but not identical shape manipulation methods which aim to overcome the disadvantages of NURBS. The common feature between two methods is the reduced number of shape manipulation parameters based on mathematical formulations of deformations which maintain accurate representation of complex shapes.

The idea of deforming existing shape to obtain new unique shape is not new. In fact deformation has been in use since the discovery of plasticity of materials. Mathematical formulation of deformations dates to work of Thompson (1917) and his grid transformations proposal. In the proposed technique a transformation grid is placed over an object of interest. The deformation of the object is then achieved by manipulating the surrounding transformation grid, see Fig. 2.4.

Influenced by the idea Bookstein (2002) develops the Thin-Plate Spline (TPS) technique and later Sederberg and Parry (1986) develops Free-Form Deformation (FFD). The main purpose of their work is to describe mathematically the physical deformations.

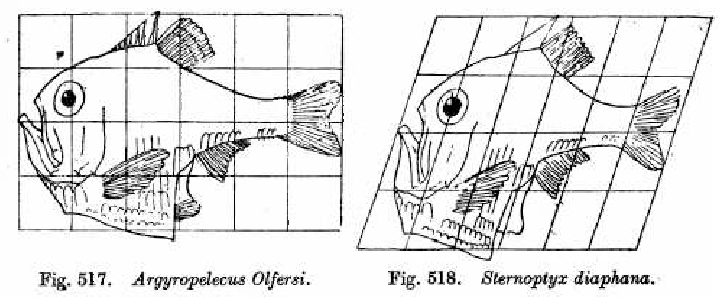


FIGURE 2.4: Formulation of fish geometry deformation by D'Arcy Thompson. The image is courtesy of Cambridge University Press

2.3.1 Thin-plate spline deformation

Thin-Plate Spline (TPS) deformation is an algebraic method to transform geometrical properties of virtual objects. The representation of physical deformation is achieved by iteratively evaluating a natural interpolating function known as thin-plate spline. This function represents an infinitely thin stiff plate under certain constraints. For better understanding reader may consider a very thin plate with embedded shape on it. The essential geometry regions of the embedded shape are mapped by points known as landmark points. These points represent the control points of deformation. The deformation of the thin-plate is achieved by moving the landmark points. The landmark points are tied to the thin plate through embedded geometry. The objective is to morph one shape into another unique shape considering the physical properties of thin-plate deformation.

The Thin-Plate Spline (TPS) function interpolates landmark points by minimizing the amount of bending required for the deformation. To interpolate a point in 2D a pair of TPS functions are used for x and y direction.

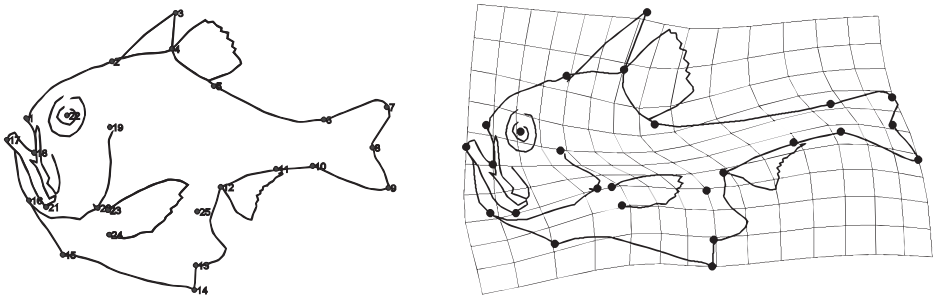
$$J(\Phi) = \sum_{j=1}^{2} \int\int_{R^2} \left(\left(\frac{\partial^2 z_j}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 z_j}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 z_j}{\partial y^2} \right)^2 \right) dxdy, \tag{2.5}$$

The total bending energy of all possible interpolating functions can be found by Eq. 2.5 where,

$$z(t) = (z_1(t), z_2(t))^T \tag{2.6}$$

is a pair of TPS (Dryden and Mardia, 1998).

Fig. 2.5 shows an example of deformed fish geometry using TPS deformation. The TPS deformation in this example is performed by TPSSPLIN Rohlf (2008). The deformed shape is found as 14-*th* principal warp of initial shape with an eigenvalue



(a) Initial (b) Deformed

FIGURE 2.5: Initial and deformed shape of D'Arcy's fish using TPS

of bending energy matrix equal to 1.192×10^{-5}. The bending energy matrix is $k \times k$ matrix $B_e = \Gamma^{11}$ (Dryden and Mardia, 1998).

TPS deformation is mostly used in area known as geometric mophometrics (Adams et al., 2004) to represent shape changes of biometric analysis. This deformation method can be also used in structural design optimisation, but has some disadvantages when dealing with 3D deformations. A 3D deformation using TPS is proposed in (Bookstein, 2002), where 2D layers of thin-plates are superimposed to represent the overall 3D geometry deformation. Further information about superimposition of TPS can be found in (Gunz et al., 2005). An advantage of using TPS deformation is that the deformation is applied directly on geometry points (landmark points) rather than on some pseudo parametric control points providing intuitive deformation control.

2.3.2 Free-Form Deformation

Free-Form Deformation (FFD) is an alternative technique for representing physical deformations of solid geometric objects. Free-Form Deformation (FFD) uses grid of points known as lattice points surrounding the geometric object. The parameters of FFD are the lattice vertices. The deformation is achieved by iteratively evaluating the Bernstein polynomial blending function.

$$X_{ffd} = \sum_{i=0}^{l} \binom{l}{i} (1-s)^{l-i} s^i \left[\sum_{j=0}^{m} \binom{m}{j} (1-t)^{m-j} t^j \left[\sum_{k=0}^{n} \binom{n}{k} (1-u)^{n-k} u^i P_{i,j,k} \right] \right], \tag{2.7}$$

The Eq. 2.7 is used to perform 3D deformations employing a composition of three Bernstein polynomial blending functions. These functions are parametrised

by lattice points $p_{i,j,k}$, where i, j and k denote the lattice points in x, y and z direction respectively.

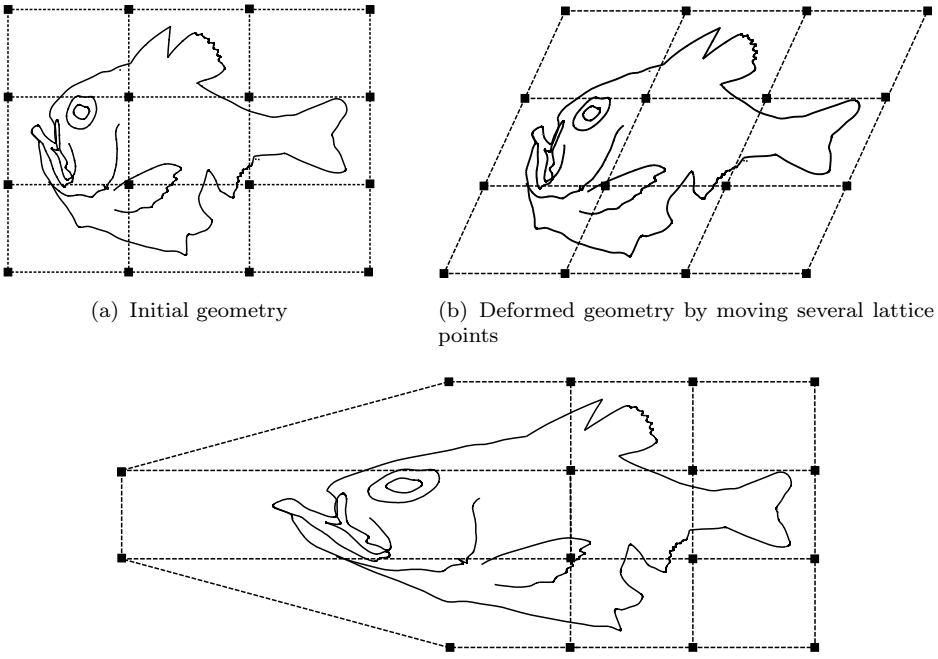
Among main advantages of FFD are:

(i) Better control on local and global deformations.

(ii) Usability and implementation in computer vision.

(iii) Sensible and natural representation in 3D.

(iv) The use of Bernstein polynomial or NURBS.



(a) Initial geometry

(b) Deformed geometry by moving several lattice points

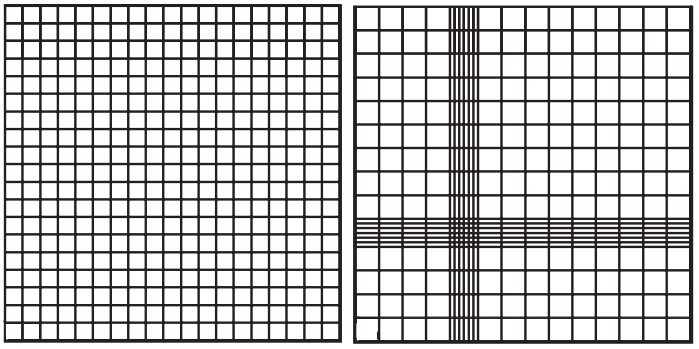
(c) Deformed geometry by moving two lattice points in x direction

FIGURE 2.6: FFD of D'Arcy's fish geometry

An example FFD of D'Arcy's fish geometry is shown in Fig. 2.6, where the initial and deformed fish geometry is presented. The deformation example presented in Fig. 2.6(c) covers approximately 50% of the initial geometry.

Local FFD based on Bernstein polynomial is performed by increasing the density of the grid – thus increasing the number of uniformly distributed lattice points as shown in Fig. 2.7(a). This solution has two disadvantages:

1. The user must take into account the continuity.



(a) Uniform grid (b) Non-uniform grid

FIGURE 2.7: Two types of grids, uniform and non-uniform. The Bernstein polynomial cannot be applied to the second grid

2. It is inefficient to perform local and global deformation simultaneously because the number of parameters is increased in regions where global deformation is required. This on the other hand requires a group of points to be moved together so that the continuity of the overall geometry is maintained.

To overcome these limitations Lamousin and W.N. Jr (2002) propose the NURBS based FFD, where Bernstein polynomial in replaced by NURBS. The NURBS based FFD enables the user to increase the density of lattice points only around the area of local deformation leaving the rest of the grid coarse. Thus a local grid is generated emending only the area of interest. The NURBS based FFD is a preferable method for local deformation when computational time is an issue.

A significant disadvantage of FFD compared to TPS is the intuitive deformation control. In TPS landmark points co-ordinates are the parameters of deformation which means that the geometry is deformed directly. The parameters of deformation in FFD are in fact the grid points and the deformation is indirect or pseudo controlled. The solutions is presented in (Hsu et al., 1992) as direct manipulation FFD. The direct manipulation is a modification of FFD which gives the user an intuitive deformation control. Using direct manipulation FFD, one can easily perform very complex deformations by manipulating directly several points on virtual geometry to desired place such as in TPS. Hu et al. (2001) propose an elegant technique for direct manipulation of FFD. This approach is based on constrained optimisation problem given by Eq. 2.8

$$f(x)_{\min} = \sum_{i,j,k=0}^{l,m,n} \|\delta_{i,j,k}\|^2. \tag{2.8}$$

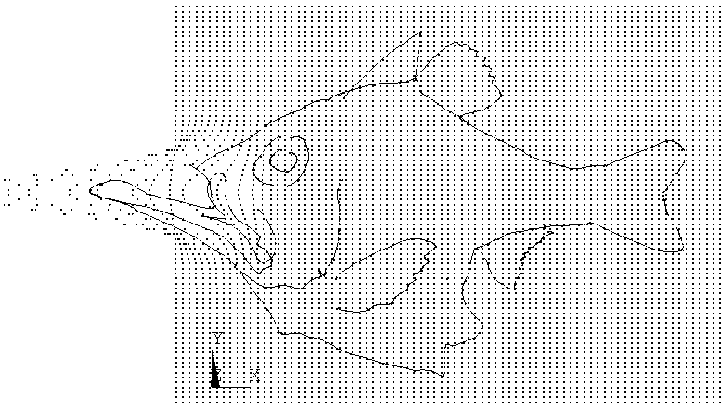


FIGURE 2.8: Local direct manipulation FFD

In (Hu et al., 2001) it is proven that the solution of minimization problem is:

$$\delta_{i,j,k} = \frac{R_{i,j,k}(s_s, t_s, u_{,s})}{\sum_{i,j,k=0}^{l,m,n} R^2_{i,j,k}(s_s, t_s, u_s)}(T - S), \tag{2.9}$$

where $R_{i,j,k} = \frac{W_{i,j,k}}{\sum_{i,j,k=0}^{l,m,n} W_{i,j,k}}$ for NURBS and $R_{i,j,k} = W_{i,j,k}$ for Bernstein polynomial, S are the co-ordinates of grid point before deformation and T are the desired co-ordinates of the grid point after deformation. The weights are given by

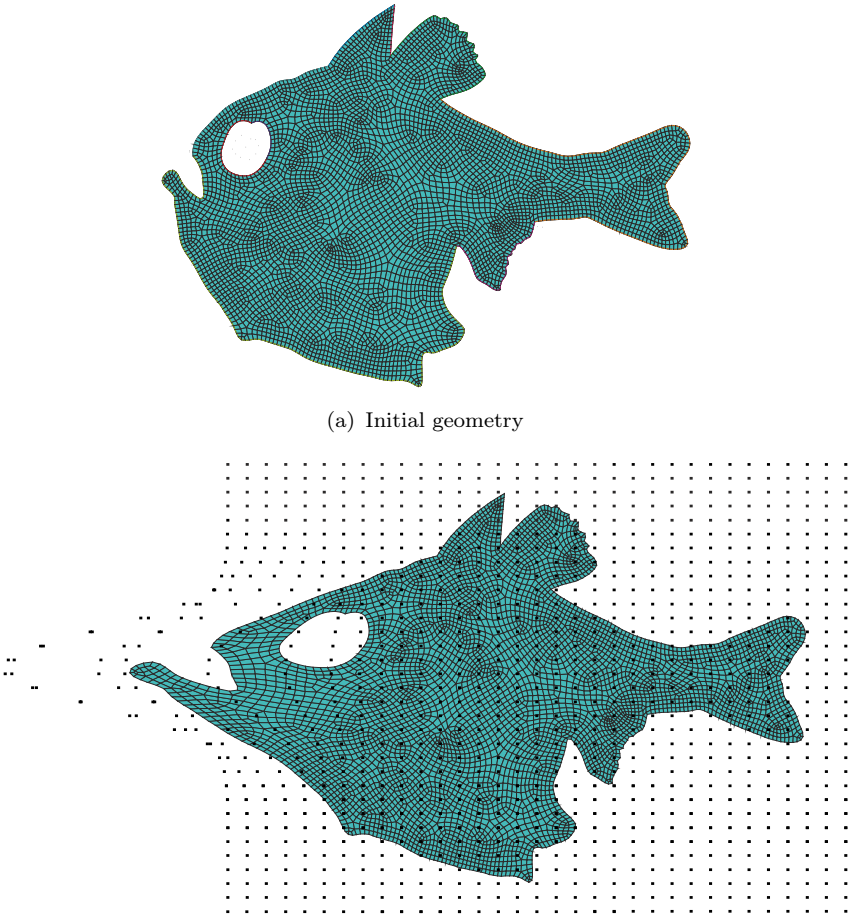
$$W_{i,j,k} = \binom{l}{i}(1-s)^{l-i}s^i\left[\binom{m}{j}(1-t)^{m-j}t^j\left[\binom{n}{k}(1-u)^{n-k}u^i\right]\right], \tag{2.10}$$

where $i = 1, 2, \cdots, l$; $j = 1, 2, \cdots, m$; $k = 1, 2, \cdots, n$ and l, m, n are the number of nodes in each direction. The new co-ordinates of the lattice points are calculated as:

$$\Phi_{i,j,k} = P_{i,j,k} + \delta_{i,j,k}. \tag{2.11}$$

Using Eq. 2.11 one can obtain the new coordinates of lattice points which will deformed geometry in desired way. An example of direct manipulation FFD of D'Arcy's planar fish geometry with dense grid is shown in Fig. 2.8. Similarly the direct manipulation FFD is performed with 32×32 grid size, but on a meshed FE model of fish geometry shown in Fig. 2.9.

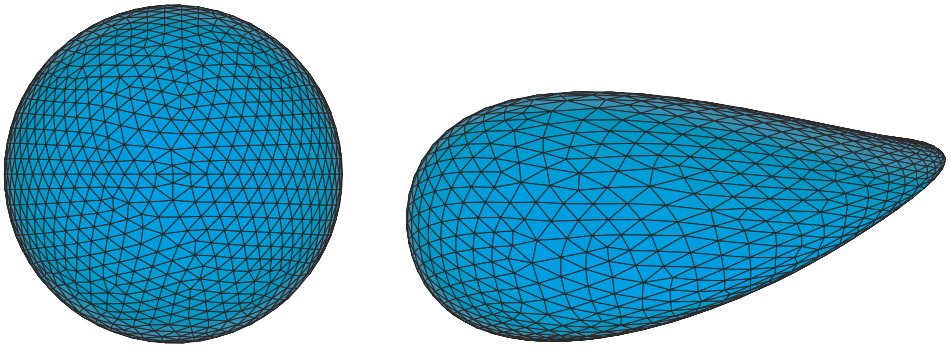
Initial and deformed 3D meshed sphere by direct manipulation FFD are shown on Fig. 2.10 respectively. All elements and nodes of the 3D model are deformed in the intuitive way, which saves the computation time needed to re-mesh the resulting model.



(a) Initial geometry

(b) Deformed geometry

FIGURE 2.9: Direct manipulation FFD with grid size 32×32



(a) Initial geometry

(b) Deformed geometry

FIGURE 2.10: Direct manipulation FFD of meshed sphere with elements

2.4 Design synthesis tools

2.4.1 Phrase-structure grammar

Grammars are effective means to synthesise a variety of complex linguistic structures. Every grammar contains a set of rules. The rules are applied to a predefined set of primitives in order to select some – and arrange those selected – in a logical order. In general, each stand-alone primitive is meaningless, but when combined with other primitives, following certain grammar rules, it plays a central role in the final complex linguistic structure. Logical linguistic structures have a great potential in non-linguistic contexts, including shape synthesis.

The use of grammar to describe complex linguistic structures can be traced back to the work of Panini who created 3,959 linguistic rules to formalise Sanskrit grammar around the 4-th century BC (Katre et al., 1989). Seminal contributions towards the development of automated grammar syntaxes were made by Chomsky (1956) who credits this ancient source and further develops three natural language grammar syntaxes. One of the syntaxes is named as the "phrase-structure grammar". This syntax can construct grammatically correct sentences as combinations of 'terminal' and 'non-terminal' linguistic primitives following iterative application of finite grammar rules in sequential order. The rules are introduced as expressions with two sides of the form

$$\mathrm{X} \rightarrow \mathrm{Y}, \tag{2.12}$$

where X and Y are linguistic primitives (strings) from a finite vocabulary set $\mathrm{V_P} = \{\mathrm{X}, \mathrm{Y}\}$. The two primitives can be further classified to be members of a non-terminal $\mathrm{V_N} = \{\mathrm{X}\}$ and terminal $\mathrm{V_T} = \{\mathrm{Y}\}$ vocabulary sets respectively. The arrow here, is used to divide the rule into the left-side and right-side. Rules, in phrase-structure grammar, are interpreted as instructions to rewrite string X as string Y.

A finite set of n grammar rules is given as

$$\begin{aligned} \mathrm{X}_1 &\rightarrow \mathrm{Y}_1 \\ \mathrm{X}_2 &\rightarrow \mathrm{Y}_2 \\ &\vdots \\ \mathrm{X}_n &\rightarrow \mathrm{Y}_n. \end{aligned} \tag{2.13}$$

Here, each rule instructs string X_i to be rewritten as string Y_i, where $V_N = \{X_1, X_2, \cdots, X_n\}$ and $V_T = \{Y_1, Y_2, \cdots, Y_n\}$.

A distinctive characteristics of phrase-structure grammar is that rewriting instructions can be grouped in expressions i.e. a non-terminal string X_i, on the left-side, can be used to rewrite a range of strings and thus provide a choice of rules. This means that the left-side of such expression is always a single non-terminal, while the right side can be a mixed group of terminals and/or non-terminals of the form

$$X_i \rightarrow Y_1, Y_2, \cdots, Y_n, Z_1, Z_2, \cdots, Z_n, \tag{2.14}$$

where $V_T = \{Y_1, Y_2, \cdots, Y_n\}$ is an array of terminal primitives separated by comma while $V_N = \{X, Z\}$ are different sets of non-terminal primitives. The role of terminal primitives is to terminate addition of rewriting instructions, while non-terminal primitives are used to assist the rule selection process by adding new instructions to an existing sequence of strings (initial sentence S) deriving final linguistic sentence L. One can consider non-terminal primitives (in this case X_i and Z_i) as 'wild cards' corresponding to finite set of terminal and non-terminal primitives where the instruction $X_i \rightarrow Z_i$ appends additional instructions to already existing sequence of strings S, while instruction $X_i \rightarrow Y_i$ terminates addition of new primitives.

A comprehensive theoretical background on iterative synthesis of linguistic sentences using phrase-structure grammar is given in (Chomsky, 1956).

For a practical example consider the initial English language sentence

THE EIFFEL TOWER IS 320 [*UNITS*] TALL.

One can define a terminal and non-terminal grammar vocabularies as follows:

$$\begin{aligned} V_N &= \{\text{UNITS}\} \\ V_T &= \{\text{m}, \text{cm}, \text{mm}, \text{nm}\}. \end{aligned} \tag{2.15}$$

This leads to definition of four phrase-structure grammar rules grouped in a single expression of the form

$$\text{UNITS} \rightarrow \text{m}, \text{cm}, \text{mm}, \text{nm} \tag{2.16}$$

The non-terminal UNITS, from the left-side of this expression may correspond to any of the four terminals – m, cm, mm, and nm. In this way, an overall structure of four grammar rules are grouped in a single expression. If the rule UNITS $\rightarrow$

nm is applied, it results in instruction to rewrite the non-terminal string UNITS with the terminal string nm synthesising the final sentence to

THE EIFFEL TOWER IS 320 [nm] TALL,

which is a grammatically correct sentence.

2.4.2 Backus-Naur Form

Chomsky's proposal of phrase-structure grammar plays a crucial role in the development of Algorithmic programming Languages (ALGOL) (Backus et al., 1963). In particular, the work of Backus (1959) which transforms the phrase-structure grammar syntax to provide a syntax for programming languages initially known as Backus Normal Form (BNF). The main advantage of the BNF over the phrase-structure grammar is the improved notation of grammar rules as follows:

(i) Non-terminal primitives in BNF are enclosed in angular brackets ($\langle\ \rangle$) e.g. $\langle$non-terminal$\rangle$ instead of a plain string for recognition purposes.

(ii) The arrow symbol ($\rightarrow$) dividing expression to left-side and right-side is replaced with a symbol $:\equiv$.

(iii) The comma symbol (,) separating alternative primitives on the right-side is replaced by ($\bar{\text{or}}$) symbol.

All these changes are introduced mainly to ease the implementation of the phrase-structure grammar as a computer code. As a result the generic form of phrase-structure grammar expressed in BNF syntax becomes

$$\langle \mathrm{X}_i \rangle :\equiv \mathrm{Y}_1\ \bar{\text{or}}\ \mathrm{Y}_2\ \bar{\text{or}}\ \cdots\ \bar{\text{or}}\ \mathrm{Y}_n\ \bar{\text{or}}\ \langle \mathrm{Z}_1 \rangle\ \bar{\text{or}}\ \langle \mathrm{Z}_2 \rangle\ \bar{\text{or}}\ \cdots\ \bar{\text{or}}\ \langle \mathrm{Z}_n \rangle, \tag{2.17}$$

where $\mathrm{V_T} = \{\mathrm{Y}_1, \mathrm{Y}_2, \cdots, \mathrm{Y}_n\}$ is an array of terminal primitives while $\mathrm{V_N} = \{\langle \mathrm{X} \rangle, \langle \mathrm{Z} \rangle\}$ are different arrays of non-terminal primitives.

More importantly, in his work, Backus first highlights the connection between grammar of natural languages and syntax of programming languages by giving

examples of vocabulary where linguistic primitives are replaced by integers, algebraic operands, arithmetic operand, logical operands, Boolean operands, etc. Backus uses the following vocabulary

$$\begin{aligned} \mathrm{V_T} &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\} \\ \mathrm{V_N} &= \{\langle \text{number} \rangle, \langle \text{integer} \rangle, \langle \text{digit} \rangle, \langle \text{dn} \rangle\}. \end{aligned} \tag{2.18}$$

to define grammar rules similar to the following four expressions

$$\begin{aligned} \langle \text{digit} \rangle &:\equiv 0 \ \bar{\text{or}}\ 1 \ \bar{\text{or}}\ 2 \ \bar{\text{or}}\ 3 \ \bar{\text{or}}\ 4 \ \bar{\text{or}}\ 5 \ \bar{\text{or}}\ 6 \ \bar{\text{or}}\ 7 \ \bar{\text{or}}\ 8 \ \bar{\text{or}}\ 9 \\ \langle \text{integer} \rangle &:\equiv \langle \text{digit} \rangle \ \bar{\text{or}}\ \langle \text{integer} \rangle \langle \text{digit} \rangle \\ \langle \text{dn} \rangle &:\equiv \langle \text{integer} \rangle . \ \bar{\text{or}}\ . \langle \text{integer} \rangle \ \bar{\text{or}}\ \langle \text{integer} \rangle \ \bar{\text{or}}\ \langle \text{dn} \rangle \langle \text{integer} \rangle \\ \langle \text{number} \rangle &:\equiv \langle \text{integer} \rangle \ \bar{\text{or}}\ \langle \text{dn} \rangle, \end{aligned} \tag{2.19}$$

where the initial sentence is $\mathrm{S} = \langle \text{number} \rangle$. The final sentence L is variable and depends on iterative selection of finite rules from the four grammar expressions above. Thus an integer will be synthesised if the instruction $\langle \text{number} \rangle :\equiv \langle \text{integer} \rangle$ is applied and decimal number will be synthesised if $\langle \text{number} \rangle :\equiv \langle \text{dn} \rangle$ is applied instead of the previous instruction. This simple example illustrates how one can use grammar rules to synthesise a variable – an integer or decimal number.

A little later, Knuth (1964) argued that the Backus Normal Form should be renamed to Backus-Naur Form because:

(i) It is not a 'Normal Form'.

(ii) Naur revised the notation proposed by Backus replacing symbol ($:\equiv$) with ($:=$) and ($\bar{\text{or}}$) with ($|$).

(iii) Naur recognised the potential of Backus's ideas and popularised them with the ALGOL committee.

These argument are widely accepted and as a result the BNF is better known as Backus-Naur Form, which has the following syntax

$$\langle \mathrm{X}_i \rangle := \mathrm{Y}_1 \mid \mathrm{Y}_2 \mid \cdots \mid \mathrm{Y}_n \mid \langle \mathrm{Z}_1 \rangle \mid \langle \mathrm{Z}_2 \rangle \mid \cdots \mid \langle \mathrm{Z}_n \rangle. \tag{2.20}$$

To visualise the main characteristics of the phrase-structure grammar Chomsky uses tree-like diagrams, where the grammar primitives are structured in verb phrases, noun phrases, etc. as shown in Fig. 2.11.

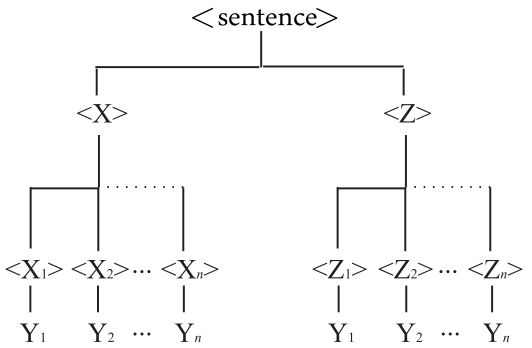


FIGURE 2.11: Tree like logical structure of phrase-structure grammar

For a practical example let us assume that there exists a complete finite vocabulary $V_P = \{V_T; V_N\}$, where

$$\begin{aligned} V_T = \{&0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \\ &\text{THE, A, SOME,} \\ &\text{BEAM, BEAMS, PLATE, PLATES,} \\ &\text{IS, ARE, IS NOT, ARE NOT,} \\ &\text{LONG, SHORT, THICK, WIDE} \\ &\text{m, cm, mm, nm}\} \end{aligned} \tag{2.21}$$

and

$$\begin{aligned} V_N = \{&\langle\text{sentence}\rangle, \langle\text{subject}\rangle, \langle\text{predicate}\rangle\langle\text{article}\rangle, \langle\text{noun}\rangle, \langle\text{verb}\rangle, \langle\text{num}\rangle, \langle\text{unit}\rangle, \\ &\langle\text{adjective}\rangle\} \end{aligned} \tag{2.22}$$

One can use this vocabulary to construct the BNF syntax shown in Table 2.1,

TABLE 2.1: An example of a BNF to representing English language grammar rules.

⟨subject⟩	::=	⟨article⟩⟨noun⟩
⟨predicate⟩	::=	⟨verb⟩⟨num⟩⟨num⟩⟨unit⟩⟨adjective⟩ | ⟨verb⟩⟨adjective⟩⟨num⟩⟨num⟩⟨unit⟩
⟨article⟩	::=	THE | A | SOME
⟨noun⟩	::=	BEAM | PLATE | STRING | BEAMS | PLATES | STRINGS
⟨verb⟩	::=	IS | ARE | IS NOT | ARE NOT
⟨adjective⟩	::=	LONG | SHORT | THICK | WIDE
⟨num⟩	::=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
⟨unit⟩	::=	m | cm | mm | nm
⟨sentence⟩	::=	⟨subject⟩⟨predicate⟩

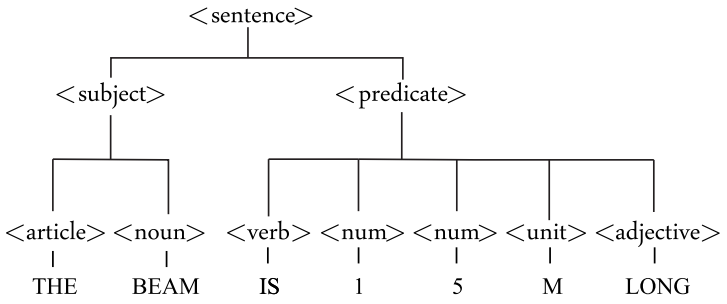


FIGURE 2.12: Tree like logical structure of an English language sentence 'THE BEAM IS 15 M LONG' synthesised using phrase-structure grammar

where the initial sentence is S = ⟨sentence⟩. This BNF syntax can be visualised as a tree structure of phrases shown in Fig. 2.12 that illustrates the sentence

THE BEAM IS 15 M LONG

derived from 9 unique non-terminal (⟨sentence⟩ ⟨subject⟩, ⟨predicate⟩, ⟨article⟩, ⟨noun⟩, ⟨verb⟩, ⟨num⟩, ⟨unit⟩, and ⟨adjective⟩) and 7 unique terminal (THE, BEAM, IS, 1, 5, M, and LONG) English language words and numbers. Each expression represents a particular phrase of the tree structure, e.g. ⟨sentence⟩ ::= ⟨subject⟩⟨predicate⟩ represents the initial branch of the tree.

The BNF syntax presented in Table 2.1 can, therefore, be used as part of a computer program for synthesis of linguistic sentences by selecting rules from each expression. The difference in this example is that some of the rules provide instructions to rewrite a single non-terminal to a phrase containing several terminal and non-terminal primitives.

2.4.3 Extended Backus-Naur Form

Consider the non-terminal ⟨num⟩ from the BNF listed in Table 2.1. This non-terminal can be repeated twice in a sequence to allow the synthesis of a double digit variable number i.e. the length of the beam in the final sentence can be between 00 and 99 and in this example it is prescribed as 15. The synthesis of a number with more digits say n would require to repeat the primitive ⟨num⟩ n times. To avoid this inconvenience in syntax notation e.g. 20 digit number would require 20 repeating ⟨num⟩ strings, which makes the phrase too long. An Extended Backus-Naur Form (EBNF) notation was hence proposed (Wirth, 1977).

In Extended Backus-Naur Form (EBNF), sequentially repeating primitives are enclosed in braces $\{\langle X\rangle\}$. This indicates that the primitive is repeated zero or more times in the sequence e.g. $\langle X\rangle\langle X\rangle\cdots\langle X\rangle$.

Another useful advantage of EBNF over regular BNF is the notation of optional primitives in square brackets $[\langle X\rangle]$. As the name optional suggests these are primitives which may or may not result in rewriting rule. For example, the synthesis of a two digit number requires two non-terminals ($\langle num\rangle\langle num\rangle$), however numbers from 00 to 09 will carry a padding 0. To avoid this one can use the EBNF notation $[\langle opnum\rangle]\langle num\rangle$, where the rewriting rule produced by the non-terminal enclosed in square brackets is optional. This is equivalent to regular BNF expression

$$\begin{aligned} \langle opnum\rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid \text{None} \\ \langle num\rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned} \tag{2.23}$$

where the rule ($\langle opnum\rangle$ ::= None) results in the removal of the non-terminal form the sentence without instructing it to be rewritten as a terminal integer. Various other notations have been suggested since the proposal of EBNF. Most of these are included in the EBNF standard issued by International Organization of Standardization (ISO) (Scowen, 1998; ISO, 1996). In this thesis, it is demonstrated how the BNF syntax can be used to represent by Shape Grammar (SG) rules by replacing linguistic grammar primitives by geometric shapes.

2.4.4 Shape Grammar (SG)

Just like the grammars in natural languages, Shape Grammar (SG) is a syntax, where certain rules are applied to a set of primitive shapes for the synthesis of complex shapes. The SG syntax was introduced by Stiny and Gips (1971) as a geometry synthesis framework for manipulating a finite number of primitive shapes. In fact, SG is special case of phrase-structure grammar, where strings are replaced by primitive shapes such as lines, splines, circles, rectangles, etc. As in phrase-structure grammar, SG requires a vocabulary set of terminal and non-terminal primitives V_P. Each SG rule, as in the phrase-structure grammar, has a left-side and right-side separated by an arrow. The rewriting rules are of the

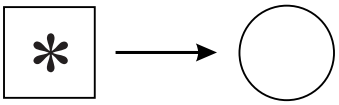
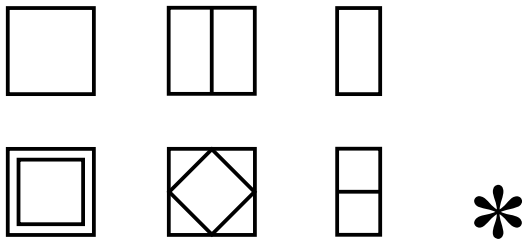


FIGURE 2.13: SG rule



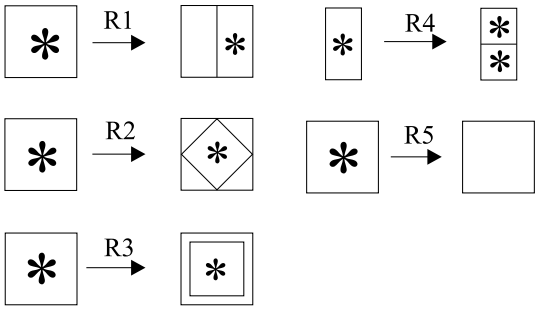
(a) A terminal vocabulary set of SG primitives – polygons

(b) Non-terminal SG primitive

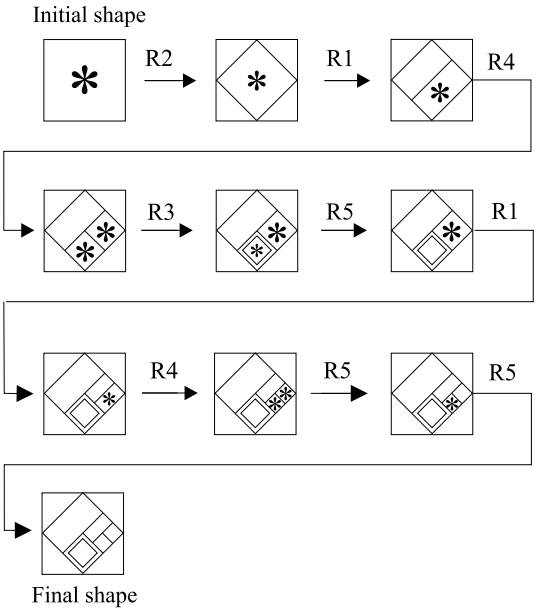
FIGURE 2.14: An example of SG vocabulary set of primitives

form shown in Fig. 2.13. The SG rule in this figure instructs the shape from the left-side to be replaced with the shape from the right-side. Ideally, the shape on the left-side should be distinguishable as it is of non-terminal type, therefore it is given as a combination of two shapes. Here, the rectangular shape is the actual primitive, while the asterisk shape inside the rectangle plays role of a marker for non-terminal types. The marker shape is an equivalent of angled enclosing brackets in phrase-structure grammar.

Consider a finite vocabulary set $V_P = \{V_N; V_T\}$, where members of V_N and V_T are shown in Fig. 2.14(a) and 2.14(b), respectively. In this example, the terminal primitive shapes are sketched as various polygons with topological features. There is only one marker shape and it is represented by an asterisk shown in Fig. 2.14(a). This vocabulary is used to define five unique SG rules denoted by R1, R2, R3, R4 and R5 in Fig. 2.15(a). The initial shape shown in Fig. 2.15(b) is given as a combination of two shapes – a polygon and a marker. This shape is the same as the shape on the left hand side of each of the rules R1, R2, R3 and R5 shown in Fig. 2.15(a). This means that any of these rules is applicable. In this case, rule R2 is selected arbitrarily and applied first. The resulting shape also contains the combination of a polygon and a marker. This triggers the arbitrary application of rule R1, which results in another shape and so on. In this way the five unique SG rules are applied in the following order: R2, R1, R4, R3, R5, R1, R4, R5 and R5. Note that the application of rule R4 introduces parallelism to the grammar sentence by adding another marker. Such a rule creates ambiguity by breaking the sequential order of the rule application procedure. Parallelism may lead to non-unique outputs of the final SG sentence. This is because in such a situation, the rule application order is not strictly defined. In this example of SG, after the application of the rule R4, first the lower pair of polygon and marker has been



(a) Five unique SG rules



(b) A sequence of applied SG rules

FIGURE 2.15: Illustration of SG syntax using five unique SG rules. The rules are applied by the user in particular order

picked, this leads to the application of rule R3. The final sentence would have been different if one had first picked the upper pair and then applied the same rule (R3). The ambiguity can be avoided by applying SG rules in parallel or by defining a strict order in which the rules will be applied in a parallelism situation. The opposite functionality of rule R4 is provided by rule R5. R5 provides a stopping criteria for the SG by removing the marker shape (non-terminal shape). In SG, the rule application is carried out until all markers are removed i.e. all shapes are of terminal type. The complexity of the shapes thus generated may be increased by adding more types of unique terminal primitives such as triangles, polygons, circles, ellipses, arcs, B-splines, patches, surfaces, etc.

The SG syntax was developed to enhance visual computations (Stiny and Gips, 1971). The realisation of SG in a computer code requires an algorithm for automatic recognition of shapes – it is required during the SG rule selection and

application. While the recognition of simple shapes such as lines, polygons and circles is a straightforward procedure, the recognition of complex shapes constructed from several curves is difficult and ambiguous. A possible solution is presented in (McCormack and Cagan, 2002b, 2008) as an SG interpreter. This is an algorithm which automates the shape recognition process by dividing the initial (complex) shape into sub-shapes with a hierarchy. Each level contains a group of primitive sub-shapes from the initial complex shape. The hierarchy of the levels are created recursively from the most complicated shapes to the simplest ones. Each sub-shape is then searched for a match passing through all levels.

An idea for the possible application of SG in the design of an aerofoil shape is presented by Keane and Nair (2005). In this proposal the initial aerofoil geometry – a circle – evolves by applying a set of SG rules, where the non-terminal shape is a triangle surrounding the aerofoil shape. McCormack et al. (2004) use SG syntax in a design study to preserve the brand design line identity of vehicles by introducing specific SG rules. These rules enable the synthesis of alternative designs preserving the brand identity of vehicles. Prats et al. (2009) conduct a survey on sketching habits of experienced designers. The observation leads to a unique set of SG rules. These rules are then identified as fundamental (general) rules which are followed by most designers in the sketching practice when exploring a range of designs.

Unfortunately in all these studies, the shape recognition and rule application operations are conducted manually. This increases the design search time significantly, due to its iterative nature and also requires human intervention. The Design Synthesis and Shape Generation (DSSG, 2011) project implements the SG syntax as computer code and provides at least an automated shape recognition. However, after recognition, the program offers a range of possible alternative SG rules to the designer. Such implementation of SG syntax still requires significant humans decision making. One of the ideas suggested to automate the SG design process is an eye tracking system (Garner et al., 2011). Such a system could follow the designer's attention and offer them alternative shape rules.

2.5 Evolutionary computation tools

2.5.1 The Genetic Algorithm

The GA is inspired by natural selection of living organisms (Goldberg, 1989) – it mimics the evolution cycle of living organisms, where the search for the fittest

individuals is continuous. In the GA, the genotype of two parents is paired in order to produce an 'offspring'. In natural selection, species strive to select their partner based on an estimate of their survival probability. In this way the chances of survival of their genes carried by the offspring in the following generations are increased significantly.

Due to its search pattern, the GA is mainly used in optimisation of multi-modal (highly non-linear) functions. In GA implementation, each member from a 'population' is represented by a genetic code often in binary string form with certain length. Each binary string is usually decoded to a decimal base values in order to calculate the survival probability of the members.

In general, evolutionary search algorithms are based on evolutionary cycles of individuals through generations. In GA, evolution cycles are carried out by applying three simple operations to the members of a population. The three operations are selection, crossover and mutation, see (Goldberg, 1989).

Selection is an operation where the best performing members from a given generation are paired for reproduction based on their fitness value. Two of the most popular selection techniques are roulette wheel selection and tournament selection. In roulette wheel selection, the member with the best fitness value has the highest probability of being selected in a simulated roulette wheel spin. In tournament selection, two or more randomly selected members from the same population compete and the one with the best fitness value gets selected to produce an offspring.

Crossover is an operation where two selected members from the same population are paired in order to produce an offspring. This is carried out by dividing the binary strings of each selected member at a particular place or several places (multipoint crossover) and then swapping the divided parts. For example, suppose that two members are selected using tournament and are represented by the binary strings 11101101 and 01101100. A crossover at position 3 in the binary string of these members will produce the following offsprings: 111|01100 $\equiv$ 236, 011|01101 $\equiv$ 109.

Mutation is an operation where a randomly selected bit from a binary string is altered with the probability p_m. Each string can undergo single or multiple mutations. During a single mutation process, the bit to be exposed to mutation is selected randomly, while during multiple mutations some or all of the bits in the string are exposed to mutation with the probability p_m. An example of single bit mutation is the change of the last bit in the binary string 1110110*0* to 1110110*1*.

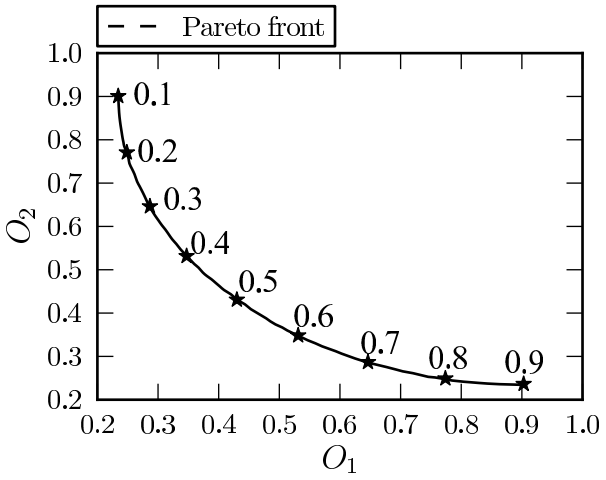


FIGURE 2.16: An example of Pareto-front by weighted sum of objective. Each weight w is used to calculate a point of the Pareto-front by using Eq. 2.24

In a GA, the old generation is commonly replaced by a new generation once all members of the offspring population emerge. Usually the offspring population is of the same size as the generation of the parents, but this is not a requirement. Recently, substantial amount of work has been carried out towards improving the efficacy of GA reproduction operations (Jadaan et al., 2008; Arabas et al., 1994; Zhou and Han, 2005; Mühlenbein et al., 1991; Perry et al., 2006). As a result, several distinctive reproduction strategies such as elitism have been developed.

2.5.2 NSGAII

Often structural design optimisation problems have multiple objectives such as to minimise the total weight of the structure and stress within the structure, to minimise the total weight and to maximise the stiffness of the structure, etc. One of the solutions is to merge multiple objectives into a single objective by taking the weighted sum of the objectives. For example, suppose the minimisation of two objectives O_1 and O_2 is required. The bi-objective problem is turned into a single objective problem by using a weighted sum of the fitnesses that linearly combines the original two objectives

$$O = wO_1 + (1 - w)O_2, \tag{2.24}$$

where $0 \leq w \leq 1$ are the weights. Different weighting values w can be readily used to explore a Pareto-front such as the one shown in Fig. 2.16. In this example

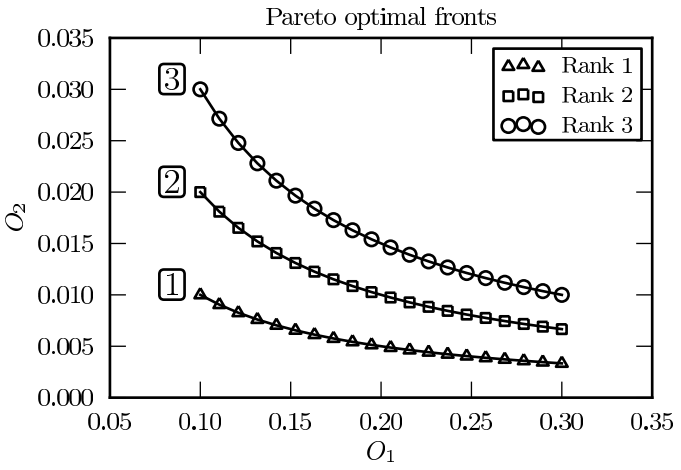


FIGURE 2.17: Ranked Pareto-fronts example – Pareto-front of rank 1 dominates those of rank 2 and 3, Pareto-front of rank 2 dominates that of rank 3, but it is dominated by that of rank 1. All points represent members in a population pool

each weight is used to explore a particular region of the Pareto-front. The weights can take values from a vector $\mathbf{w} = 0.1, 0.2, \cdots, 0.9$ are selected arbitrarily. If the fitness values of the two objectives O_1 and O_2 have different scales they are normalised so that they are in a similar range when calculating their sum O. This can be difficult to achieve in practice.

An elegant approach for multi-objective optimisation is the modification of GA known as NSGAII (Deb et al., 2000). In NSGAII, each member from the mating population pool formed by merging the parent and the offspring population is ranked taking into account the values of multiple objectives. The rank values of all members are obtained by facing each population member – in terms of its fitness taking into account multiple objectives – with other members from the mating population pool. In this way the non-dominated members are assigned rank 1 – all these member are of equal rank 1 and represent a Pareto-front. The members dominated only by those of rank 1, but non-dominated in between receive rank 2 and represent another Pareto-front which is dominated by that of rank 1. The members dominated only by those of rank 1 and 2 receive rank 3 and represent a third Pareto-front and so on, see Fig 2.17. The main difference between GA and NSGAII is that the selection of members for reproduction is based on their rank values instead of their fitness values of multiple objectives. Hence, the newly formed population consist of members with the lowest possible rank.

Consider five members from a population pool listed in Table 2.2. These members are to be ranked for minimisation of the objectives O_1 and O_2. Considering the

values of the two objectives O_1 and O_2, the population members q_1, q_2 have rank 1, q_3, q_4 have rank 2 and q_5 has rank 3. NSGAII is known to use an 'elitist' reproduction strategy, since it keeps 'alive' only the best performing members. A detailed description of technical implementation of NSGAII is given in (Deb et al., 2000).

An interesting multi-objective design optimisation of surrogate models[1] using NSGAII is presented in (Prats and Garner, 2006). In (Voutchkov et al., 2006), a structural design optimisation of a simplified jet engine FE model using NSGAII is carried out in order to find a robust optimal design. Another interesting multi-objective study (Malyna et al., 2006) investigates the possibility of synthesising power electronic converters using NSGAII.

NSGAII finds its roots in the development of Multi-Objective Genetic Algorithms (MOGA) (Fonseca and Fleming, 1993) and Niched Pareto GA (NPGA) (Horn et al., 1994). A performance comparison between NSGAII and a rival multi-objective algorithm named as Strength Pareto Evolutionary Algorithm II (SPEAII) is presented in (Zitzler et al., 2002).

Because the NSGAII is a heuristic search method, one could expect other gradient based search methods to perform better in locating the global optima with high accuracy if the whereabouts of the global optimum were roughly known. This means that the search is already constrained to uni-modal region and the gradient towards the global optima can be calculated. Unfortunately this is often a difficult task, especially when the global optimum is extremely sharp. This is a reason for using the NSGAII with relatively large population size. A disadvantage of using a large population size is the slow convergence due to the increased number of objective function evaluations. In this thesis an efficient truss optimisation using GE is proposed, which overcomes the limitations and outperforms some currently available truss optimisation algorithms.

[1]Surrogate models are computationally inexpensive approximations of computationally expensive complex models usually given in mathematical form.

TABLE 2.2: Ranking of 5 members in a population based on fitness values of two objectives O_1 and O_2

member	q_1	q_2	q_3	q_4	q_5
rank	1	1	2	2	3
O_1	3	1	3	2	3
O_2	1	3	2	3	3

2.5.3 Grammatical Evolution

GE provides an automated way of selecting grammar rules using evolutionary intelligence. This is achieved by a combination of a GA with the use of Backus-Naur Form (BNF) syntax. As previously mentioned the GA replicates natural selection based on fitness measure to breed population of trial solutions that improve over time. The ability of GAs to efficiently search large conceptual spaces makes them suitable for the discovery and induction in generalisations such as grammars. According to Mckay et al. (2010) early attempts of using GAs to explore grammar generalisations can be traced back to the work of Cramer (1985) and Hicklin (1986). A firm definition, of this newly emerging field of evolutionary computation, is given in (Koza, 1994; Koza and Poli, 2005) as GP. The first formulation of GP is based upon tree like topological diagrams representing logical structures derived from generalised formulation of a conceptual search space. GA specific crossover and mutation operations are applied directly at tree branches to explore alternative solutions. Further development of GP is the use of phrase-structure grammar (Whigham et al., 1995), which plays a key role in development of BNF syntax.

GE (O'Neill and Ryan, 2003) is one of the most widely applied GP method and as such is classified as an evolutionary automatic programming method. The automated programming in GE is achieved by applying grammar rules to building blocks of the potential program – the program primitives. These primitives can be operands, variables, constants, functions, classes, objects, modules, etc. Initially GE was developed as an automatic programming method (Ryan et al., 1998; O'Neill and Ryan, 1998, 2002, 2003), but since then it has been successfully applied to solve a variety of complex problems (O'Neill, 2011). Some interesting applications of GE include automatic composition of music (de la Puente et al., 2002), evolving financial prediction models and market index trading rules (Brabazon and O'Neill, 2006; O'Neill et al., 2001a), automatic programming of robots (O'Neill et al., 2000), evolutionary design (O'Neill et al., 2009, 2010), and numerical parametric optimisation (Murphyand et al., 2009; Nasuf et al., 2011). In the following section, the automated grammar rule selection mechanism provided by the GE is described.

2.5.4 Intelligent selection of grammar rules with GE

The human intelligence behind rule selection in grammars is common. Recently, in pattern learning and machine intelligence, it has been of interest to study how such intelligence can be transferred to computer programs. The advantage of phrase-structure grammar is that it provides a simple mechanism to represent grammar rules in a computer code (BNF syntax). If a machine can decide which rules to select from a grammar it will be able to construct logical sentences just as humans do. One of the proposals to plant intelligence into a machine is to develop an evolutionary intelligent algorithm, where through evolution based on individuals, the best sequence of selected grammar rules survives.

The fundamental part of GE is based on the GA, however its innovative feature is the ability to separate search and solution space using BNF syntax mapping procedure illustrated in Fig. 2.18. The BNF syntax used in this mapping is listed in Table 2.3, where the vocabulary sets are

$$\begin{aligned} \mathrm{V_T} &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .\} \\ \mathrm{V_N} &= \{\langle \mathrm{int} \rangle\}. \end{aligned} \tag{2.25}$$

In GE, the genotype (a binary string of length μ) is generated by the GA. This binary string is then divided into smaller fragments of binary code known as codons. The codons are then mapped on to phenotype (grammar rules). The grammar rule selection with GE is done in the following order:

1. The genotype is generated by the GA as a binary string of length μ e.g. $00110111 \cdots 00001011$.
2. This string is then split into smaller binary strings (codons) of length β. The number of codons in a string is $\lambda = \mu/\beta$, where μ is the total binary string length and β is the codon binary string length.
3. Each codon x_i, $i = 1, 2, \cdots, \lambda$, is then decoded into an integer e.g. for $\beta = 8$ bit codon the integer is $0 \leq x_i \leq 255$.

TABLE 2.3: A example of BNF syntax used to synthesise a real number

⟨int⟩	::= ⟨int⟩⟨int⟩ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
S	= ⟨int⟩.⟨int⟩

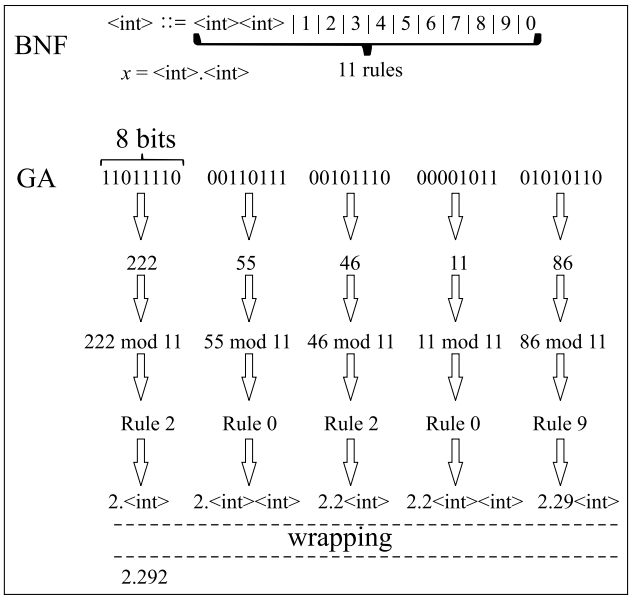


FIGURE 2.18: An illustration of grammar rules selection process in GE. The rules are represented by a BNF syntax and selected using a binary string with length μ. The wrapping operation has been used once, where the first codon with length β from the binary string is overused once to select another grammar rule

4. A binary string of length β caries enough information to select 2^β unique rules. If the number of possible rules in a BNF expression is ξ the rule number $(y_j; j = 1, 2, \cdots, \xi)$ to be selected from that BNF expression is determined as the modulus of the sequential decoded integer x_i by ξ

$$y_i = x_i \bmod \xi. \tag{2.26}$$

5. If, at the end of the decoded integer sequence (x_λ), the grammar sentence is still incomplete (due to the presence of non-terminal primitives in the sentence), the selection continues from the beginning of codon set (x_1) in sequential order. The information carried by each codons is reused to select additional rules until the sentence is complete or a certain number of loops is reached. This may happen when the binary string of length μ is relatively short compared to complexity of the sentence to be synthesised. This is known as the 'wrapping' process. Suppose that there is a sequence of λ codons, but the complexity of the grammar requires $\lambda+n$ rules to be selected. In such a situation, wrapping operation is applied to reuse n number of codons.

The rule selection process in GE is illustrated by an example to generate a real number in Fig. 2.18. Suppose a genotype with length $\mu = 5 \times 8$ is produced by

TABLE 2.4: Grammar rules to synthesise a real number

#	modulus	applied rule
1.	222 mod 11 = Rule 2	$\langle int \rangle.\langle \text{int} \rangle \rightarrow 2.\langle \text{int} \rangle$
2.	55 mod 11 = Rule 0	$2.\langle int \rangle \rightarrow 2.\langle int \rangle\langle int \rangle$
3.	46 mod 11 = Rule 2	$2.\langle int \rangle\langle \text{int} \rangle \rightarrow 2.2\langle \text{int} \rangle$
4.	11 mod 11 = Rule 0	$2.2\langle int \rangle \rightarrow 2.2\langle int \rangle\langle int \rangle$
5.	86 mod 11 = Rule 9	$2.2\langle int \rangle\langle \text{int} \rangle \rightarrow 2.29\langle \text{int} \rangle$
	wrapping	
6.	222 mod 11 = Rule 2	$2.29\langle int \rangle \rightarrow 2.292$

the GA as a binary string

$$11011110001101110010111000001101101010110.$$

The decoded integers for each 8 bit codon ($\beta = 8$) from this string are: 11011110, 00110111, 00101110, 00001011, 01010110 = 222, 55, 46, 11, 86 i.e. $\lambda = 5$. In order to generate a real number with grammar rules, this sequence of integers is applied to BNF syntax form Table 2.3. The number of rules in the BNF expression

$$\langle \text{int} \rangle ::= \langle \text{int} \rangle\langle \text{int} \rangle \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0 \tag{2.27}$$

is $\xi = 11$ – each rule is separated by the '|' symbol. The initial sentence in BNF syntax from Table 2.3 is S= $\langle \text{int} \rangle.\langle \text{int} \rangle$. The rule selection is triggered by the first non-terminal $\langle \text{int} \rangle$ in this sentence. The modulus of the first value in the decoded integer sequence (222, 55, 46, 11, 86) by the number of available rules in BNF expression is

$$y_1 = 222 \bmod 11 = 2. \tag{2.28}$$

This corresponds to *Rule* 2 ($\langle \text{int} \rangle \rightarrow 2$) counted from left to right because the rule count in GE starts from 0. Similarly a mapped value of $y_2 = 55 \bmod 11 = 0$ selects *Rule* 0 ($\langle \text{int} \rangle \rightarrow \langle \text{int} \rangle\langle \text{int} \rangle$) triggered by the remaining, second non-terminal ($\langle \text{int} \rangle$). The third rule is $\langle \text{int} \rangle \rightarrow 2$ ($y_3 = 46 \bmod 11 = 2$) and so on. The rule selection continues until all primitives are of terminal type. The applied integer sequence decoded from the binary string selects 6 out of 11 unique rules synthesising the final sentence 2.292, shown in Table 2.4 and illustrated in Fig. 2.18. In this example, the wrapping has occurred once to select the last digit of the sentence (2), where the value of the first codon ($x_1 = 222$) was reused because the number of rules to be selected was 6 while the available codons were $\lambda = 5$. If more wrappings than the allowed had occurred the selection process would have been terminated and a penalty fitness value for the genotype assigned.

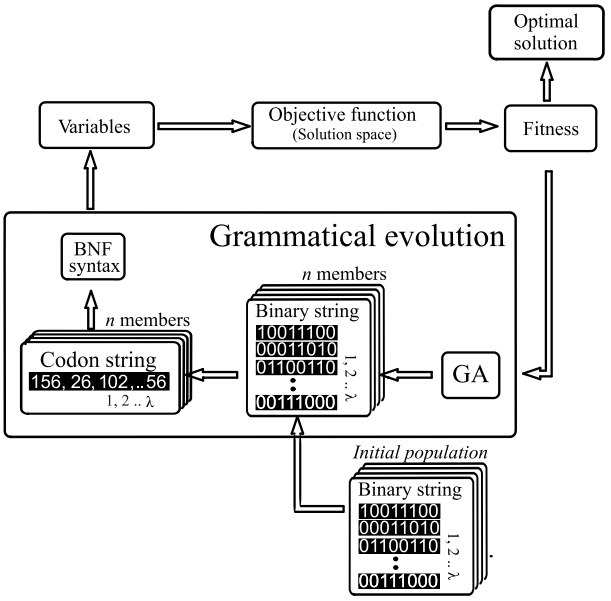


FIGURE 2.19: An optimisation flowchart of GE which shows the connection between GA and BNF syntax

This example shows how the BNF syntax may be used to synthesise a real number by selecting grammar rules. It also demonstrates how grammar rule selection in GE is guided by a binary string of length μ generated by the GA. One of the application of synthesised real numbers based on grammar rules is numerical optimisation and this is covered next

2.5.5 Optimisation using GE

The GE can be used for optimisation purposes, since it incorporates the GA. The difference is that the procedure of mapping the binary string to real variables is carried out using the BNF syntax. This, on the other hand, separates the search space (binary string) and the solution space (selected sequence of grammar rules).

A flow chart outlining the GE optimisation concept is shown in Fig. 2.19. The optimisation loop, in this flowchart, is initiated by generating an initial population of n members. Each member in this population is generated randomly as a binary sting of length μ. Note that the binary strings can be replaced with decimal numbers in case decimal base GA is used. The next block, in this flowchart, illustrates that each binary string is split into smaller strings of length β called codons. The total number of codons is $\lambda = \mu/\beta$. Each codon is then decoded into an integer. The decoded integers are then mapped to a set of grammar rules using already defined BNF syntax synthesising one or more optimisation variables. The

variables are then used as unknown parameters of optimisation to evaluate the objective function in order to obtain fitness values.

Second iteration over the flowchart is continued by feeding back the fitness values to the GA, where reproduction operations such as selection, crossover and mutation are applied to binary level in order to produce an offspring population. The offspring population is then used to select new sets of grammar rules using the BNF syntax defined. The newly synthesised optimisation variables are used to evaluate the objective function once again and so on. The optimisation iteration continues until certain number of iterations is reached or values satisfying the defined optimisation objective are discovered.

2.5.5.1 The role of crossover in numerical optimisation with GE

A distinctive feature of optimisation with GE compared to a regular GA is the separation of search and solution spaces. This is carried out by a BNF syntax which is used to map binary strings to grammar rules. In GE, the crossover operation is applied to the binary strings, however the resulting binary string, after crossover, is mapped to grammar rules (solution space) using problem specific BNF syntax. This, on the other hand, may lead to unpredictable behaviour of optimisation convergence which is thought to depend upon the selected crossover position as follows:

(i) When the crossover position is near the end of the binary string the grammar sentence thus synthesised could remain unchanged although the binary string has been altered.

(ii) When the crossover position is near the beginning of the binary string the grammar sentence thus synthesised could be altered significantly so that it becomes invalid although the performance of crossover pair used could be positively superior in the previous population.

Both situations listed above are valid only when the binary string is interpreted in sequential order through a decoded codon sequence.

For example, suppose a binary string

$$11011110001101110010111000001011 \cdots 01010110,$$

TABLE 2.5: A example of BNF syntax used to synthesise a real number

⟨int⟩	::= ⟨int⟩⟨int⟩ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
x	= ⟨int⟩.⟨int⟩

is decoded into decimal base codons in sequential order i.e. 222, 55, 46, 11, $\cdots$, 86. The situation described in (i) is observed when the crossover position is near the end of the binary string as follows:

$$11011110001101110010111000001011 \cdots 010 \mid 10110,$$

where the symbol '|' indicates crossover position on the binary string. As a result the binary string gets altered, however the synthesised sentence remains the same because the number of rules to be selected is less that the available codons.

In GE the number of rules to be selected is not constant but depends on the sequence of selected rules, where already selected rules influence the decisions made for selecting the rules after. For instance, consider the BNF syntax in Table 2.5. The initial sentence here is $x = \langle \text{int} \rangle.\langle \text{int} \rangle$, where $0 \leq x < \infty$ is the variable to be synthesised. The variable is theoretically unconstrained and goes to infinity. This is because the number of digits before and after the decimal point is theoretically unlimited and depends on how many times the rule ($\langle \text{int} \rangle \rightarrow \langle \text{int} \rangle \langle \text{int} \rangle$) is selected e.g.

$$x = \langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle . \langle \text{int} \rangle \langle \text{int} \rangle = 958745.36. \tag{2.29}$$

The cost of increasing the number of non-terminals in the initial sentence is a longer binary string length. This is because each non-terminal ⟨int⟩ requires an additional codon to select a grammar rule provided that the wrapping operation is disabled. Because the number of grammar rules to be selected is not fixed, the length of the binary string has to be chosen large enough to accommodate all possible scenarios. Nevertheless, if the rule ($\langle \text{int} \rangle \rightarrow \langle \text{int} \rangle \langle \text{int} \rangle$) is selected only a few times, the synthesised number is much smaller e.g. $x = \langle \text{int} \rangle.\langle \text{int} \rangle = 5.2$. Therefore, the remaining binary string remains unused. As a result, crossover applied near the end of the binary string will not make any difference because the information carried by the binary string in that region becomes irrelevant. This is considered as drawback of GE because it reduced the convergence performance of GE due to meaningless iterations.

Opposite to behaviour of (i) is the situation described in (ii), where the crossover happens near the beginning of the binary string e.g.

$$11011 \mid 1100011011100101110000001011 \cdots 01010110.$$

In the literature (Rothlauf and Goldberg, 1999), this situation is described as the locality problem. Locality is a distance measure between binary string neighbours in the search space and corresponding neighbouring solutions in the solution space i.e. how well neighbours in the search space correspond to grammar rule sequence neighbours in the solution space.

For example, a crossover applied at the beginning of two similar binary strings (neighbours) may lead to a new binary string which is close to its parents in the search space, however it selects completely different grammar rules in the solution space. This is because, in most cases, the influence of the first grammar rule is greater than those selected later.

Consider the BNF syntax in Table 2.6. Here an expression with two grammar rules

$$\langle \text{sign} \rangle ::= - \mid + \tag{2.30}$$

is used to determine the sign of the synthesised real number. Taking into account the initial sentence, a binary string which contains three codons would be sufficient to select a rule for each non-terminal. The first codon from this binary string would be used to select the sign of synthesised real number e.g. $\langle \text{sign} \rangle \langle \text{int} \rangle . \langle \text{int} \rangle = +1.5$. Therefore, its role is more important than the second and third codons which select a digit before and after the decimal point. This is because the synthesis of variable -1.5 would be significantly different compared to synthesis of $+1.6$ in the solution space. Hence a crossover applied at the beginning of the binary string would change the sign, while a crossover applied near the end would only change the digit after the decimal point.

The role of high-locality in evolutionary algorithms is controversial and is subject to extensive research (Rothlauf and Goldberg, 1999; Gottlieb et al., 2000a,b, 2001; Rothlauf and Goldberg, 2000). According to Rothlauf and Oetzel (2006) the convergence performance of GE suffers due non-locality because many neighbouring genotypes do not correspond to neighbouring phenotypes. Another study

TABLE 2.6: A example of BNF syntax used to synthesise positive or negative real number

⟨sign⟩	::= - | +
⟨int⟩	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
x	= ⟨sign⟩⟨int⟩.⟨int⟩

conducted by Castle and Johnson (2010) suggests that crossover in GE can be equally productive and destructive. The results from experiments with several well known benchmark problems are presented in (Hugosson et al., 2010). The results obtained are compared statistically to investigate the role of crossover, mutation and non-locality in GE.

Studies mentioned above do not take into account the role of BNF syntax, which is discussed next.

2.5.5.2 The role of BNF syntax in numerical optimisation with GE

The definition of BNF syntax, the initial grammar sentence in particular, plays an important role in improving the convergence speed of optimisation. For the illustration of BNF mapping between search and solution spaces, consider a binary string of length μ e.g. $0001110010000110 \cdots 0100$. This binary string represents a $\lambda = \mu/\beta$ dimensional search space given by λ codons. Each codon is a sub-binary string of length 4 bits e.g. $(0001, 1100, 1000, 0110, \cdots, 0100)$ and represents one dimension in the search space where the decoded codon values are between 0 and 15. This is because 4 bits codon provides $2^4 = 16$ positions in the decoded search space. Therefore, each sub-binary string would be decoded to $(1, 12, 8, 6, \cdots, 4)$.

Suppose this codon sequence is applied to the BNF syntax listed in Table 2.7 to select grammar rules. The initial sentence $\mathrm{S} = (\langle \mathrm{num} \rangle, \langle \mathrm{num} \rangle, \cdots, \langle \mathrm{num} \rangle)$ in this syntax triggers the synthesis of n variables by the non-terminal $\langle \mathrm{num} \rangle$. The first codon of value 1 in the codon sequence selects a rule for the first non-terminal $\langle \mathrm{num} \rangle$.

Taking this into account, the mapping of the variable from the first dimension of the search space to available grammar rules is represented by a discrete periodic function with two unique positions ($0.\langle \mathrm{int} \rangle \langle \mathrm{int} \rangle$ and 0) as shown in Fig. 2.20(a). This is because the first codon is used to select one of the two grammar rules for the non-terminal $\langle \mathrm{num} \rangle$. Recall that in GE the rule selection is carried out

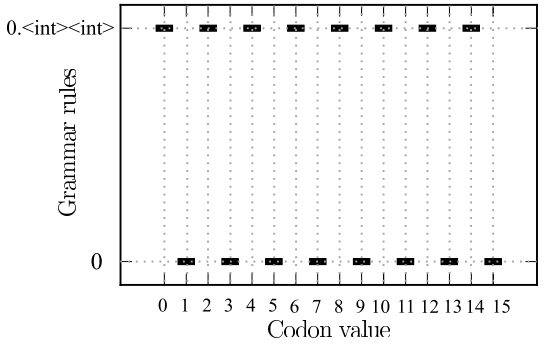
TABLE 2.7: An example of BNF syntax to visualise search and solution space mapping

$\langle \mathrm{int} \rangle$	::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
$\langle \mathrm{num} \rangle$	::= $0.\langle \mathrm{int} \rangle \langle \mathrm{int} \rangle$ | 0
S	$= (\langle \mathrm{num} \rangle, \langle \mathrm{num} \rangle, \cdots, \langle \mathrm{num} \rangle)$

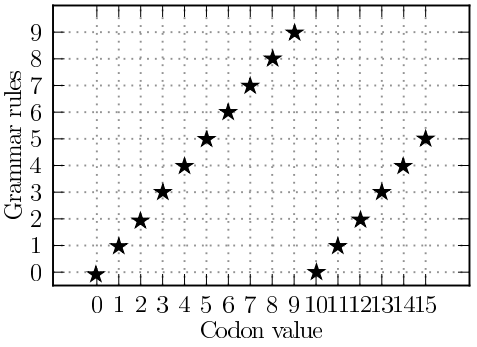
by taking the modulus of codon value by the number of available rules in a BNF expression. In this case the grammar rule ($\langle num \rangle \rightarrow 0$) is selected ($1 \bmod 2 = 1$), since the value 0 corresponds to $\langle num \rangle \rightarrow 0.\langle int \rangle \langle int \rangle$ and the value 1 to $\langle num \rangle \rightarrow 0$.

The search space mapping for the second codon is also visualised as discrete periodic function shown in Fig. 2.20(a) with two levels. The second codon value of 12 selects the grammar rule ($\langle num \rangle \rightarrow 0.\langle int \rangle \langle int \rangle$), since $12 \bmod 2 = 0$.

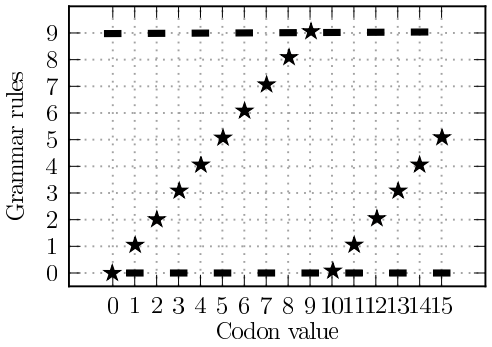
The search space mapping of the third search space dimension is a discrete linear function shown in Fig. 2.20(b). It represents 10 unique grammar rules provided by the non-terminal $\langle int \rangle$. In this case, the codon of value 8 selects the grammar rule ($\langle int \rangle \rightarrow 8$), since $8 \bmod 10 = 8$. Note that some of the grammar rules are repeated for codon values greater than 10, see Fig. 2.20(b). The search space mapping for the fourth codon is the same as the previous, where the grammar rule ($\langle int \rangle \rightarrow 6$) is selected by the sequential codon value of 6 and so on.



(a) Search space mapping for grammar rules ($\langle num \rangle ::= 0.\langle int \rangle \langle int \rangle \mid$ reject)



(b) Search space mapping for grammar rules ($\langle int \rangle ::= 0 \mid 1 \mid 2 \mid \cdots \mid 9$)



(c) Superimposed search spaces mapping when a codon value is used twice for the two sets of rules

FIGURE 2.20: A visualisation of codon search space mapping for grammar rules ($\langle int \rangle ::= 0 \mid 1 \mid 2 \mid \cdots \mid 9$). Each point represents a grammar rule corresponding to a codon value

TABLE 2.8: An example of modified BNF syntax to visualise search and solution space mapping

$\langle \texttt{int} \rangle$	$::= \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \texttt{5} \mid \texttt{6} \mid \texttt{7} \mid \texttt{8} \mid \texttt{9}$
$\langle \texttt{num} \rangle$	$::= \texttt{0}.\langle \texttt{int} \rangle \langle \texttt{int} \rangle \mid \texttt{0}.\langle \texttt{int} \rangle \langle \texttt{int} \rangle \times 0$
S	$= (\langle \texttt{num} \rangle, \langle \texttt{num} \rangle, \cdots, \langle \texttt{num} \rangle)$

Now taking into account all the selected rules using λ codons, the final sentence would be $\mathrm{S} = (0, 0.86, \cdots, 0.56)$. It indicates that the first variable is 0, since the rule $\langle \mathrm{num} \rangle \rightarrow 0$ is selected for the first non-terminal $\langle \mathrm{num} \rangle$. In this scenario, only $(\lambda - 2)$ out of λ codons would be used, i.e. the information provided by the last two codons is unused. This is because the synthesis of the first variable used only 1 codon instead of 3.

To avoid unused binary information in the search space the BNF expression

$$\langle \mathrm{num} \rangle ::= 0.\langle \mathrm{int} \rangle \langle \mathrm{int} \rangle \mid 0 \tag{2.31}$$

is modified to

$$\langle \mathrm{num} \rangle ::= 0.\langle \mathrm{int} \rangle \langle \mathrm{int} \rangle \mid 0.\langle \mathrm{int} \rangle \langle \mathrm{int} \rangle \times 0. \tag{2.32}$$

The second rule in the modified BNF expression in Table 2.8 leads to synthesis of a real number which is then multiplied by 0 ($\langle \mathrm{num} \rangle \rightarrow 0.\langle \mathrm{int} \rangle \langle \mathrm{int} \rangle \times 0$). As a results, the synthesised variable is always 0. This does not change the possible outcomes in the solution space because the value will always be 0; However, there are two immediate benefits of using the modified BNF syntax:

1. It requires a fixed number of codons because of equal number of non-terminal in the sentence even though the variable is selected to be 0.

2. More importantly the sequence of selecting grammar rules is preserved. This, increases the locality of GE because neighbouring binary strings in the search space correspond to neighbouring sentences in the solution space.

Note that the value of 0.86 was previously synthesised for the second variable while the first variable was 0.

Now consider the case where $\lambda + 1$ codons are required i.e. the number of non-terminal to select grammar rules are more than available codons. In such a scenario the wrapping procedure can be used successfully or an additional codon is required. The wrapping will reuse the value of the first codon.

The visual illustration of wrapping over the value of the first codon (search space dimension) is shown in Fig. 2.20(c), where $\lambda + 1$ search space mapping is superimposed over the first search space mapping for the corresponding two sets of grammar rules.

Overusing codons has a limitation – the same information provided by the binary string is used to select two grammar rules. When the codon value 1 is used for the first time it selects the grammar rule ($\langle \text{num} \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \times 0$), see Fig. 2.20(c). If the wrapping occurs, the first codon is used again but for a different set of rules ($\langle \text{int} \rangle ::= 0 \mid 1 \mid 2 \mid \cdots \mid 9$), where the rule ($\langle \text{int} \rangle \rightarrow 1$) is selected. Using the same codon for the two sets of grammar rules means that they are dependent i.e. the codon value selecting the rule ($\langle \text{num} \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \times 0$) will select only rules representing odd integer values on the right side from the second set of rules and vice-versa.

The importance of BNF syntax was pointed out in this section. The original BNF syntax was modified intentionally so that it results in the selection of a fixed number of grammar rules, although not all of them contribute to diversity of the solution space. Experimental results on how this affects the performance on GE are presented in Chapter 5 later in this work.

2.6 Shape analysis tools

2.6.1 Geometric morphometrics

Initial work in the area of morphometrics coincides with foundation of the journal Biometrika by Pearson (1926). A combination of shape and size is known as form. The study of the relation between shape and size is known as allometry. Multivariate morphometrics (Blackith and Reyment, 1971; Adams et al., 2004) is a statistical analysis method for comparing forms. It uses multivariate statistics to compare forms within and among groups of forms. To represent a form using parameters, geometric properties of the form are transferred to a group of finite number of points on the object by measuring distances and sometimes angles. Once the form is parametrised by distances and angles, its variance and covariance with other variables can be studied. Developed in the first quarter of $19th$ century, this method for form analysis has a major drawback – it works with distances and angles between points instead of coordinates. This results in failure to remove size from the shape description.

Geometric morphometrics (Adams et al., 2004) overcomes the size problem in the shape analysis. This new improved method for statistical shape analysis initially was used in zoology to investigate morphological variations within and among samples of organisms. Since then it has been used in archaeology (Dryden and Mardia, 1998; Dryden, 2000), biology (Bookstein, 1984; Monteiro et al., 2005; Rohlf and Marcus, 1993; Adams et al., 2004), artificial intelligence (Jin and Mokhtarian, 2006) and many other research areas. Unlike its predecessor, geometric morphometrics operates with co-ordinates of the points known also as *landmarks* instead of distances and angles between them.

According to Dryden and Mardia (1998) there are three type of landmarks:

(i) Anatomical

(ii) Mathematical

(iii) Pseudo

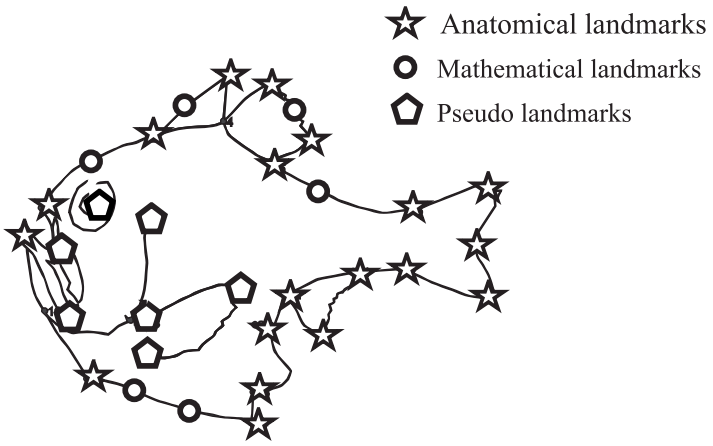


FIGURE 2.21: Three types of landmarks in Geometric Morphometrics

It can be seen in Fig. 2.21 that anatomical landmarks are points assigned by an expert such as a zoologist or a biologist and they represent some biologically meaningful information about the form. Mathematical landmarks are points assigned to location with mathematically or geometrically meaningful information about form. Pseudo-landmarks are points in between anatomical and mathematical landmarks or around the outline.

Usually landmarks are mapped with labels. The landmark mapping is very important procedure since a pair of landmarks with the same label correspond to

the same geometrical point on two forms regardless of their orientation and size in space.

Geometric morphometrics is mainly used for statistical shape analysis of a finite group of geometrical forms in the following order:

1. Landmark points are distributed at shape descriptive geometrical locations on each form within the group.
2. The position of each landmark point is then captured in terms of its co-ordinates in space and labelled.
3. The captured data are then used to refine the shape of each form by removing translation, rotation and scale factors.
4. The refined data are finally used to draw probabilistic conclusions about shape deformation and possible factors influencing these deformations.

The simplest way of refining a 2D shape over a finite group of geometrical forms is to transfer the initial shape descriptive landmark co-ordinates of each form to Bookstein's shape space co-ordinates (Dryden and Mardia, 1998). The transfer is carried out by arbitrarily selecting two landmark points on one of the forms. The distance between the selected points is used as constant base line distance.

The Bookstein's shape space co-ordinates s^B, t^B of initial x, y landmark co-ordinates are calculated relative to the baseline distance using Eq. 2.33.

$$\begin{aligned} s_i^B &= \{(x_2 - x_1)(x_i - x_1) + (y_2 - y_1)(y_i - y_1)\}/D_{12}^2 \\ t_i^B &= \{(x_2 - x_1)(y_i - y_1) - (y_2 - y_1)(x_i - x_1)\}/D_{12}^2, \end{aligned} \tag{2.33}$$

where

$$D_{12} = (x_2 - x_1)^2 + (y_2 - y_1)^2 \tag{2.34}$$

is baseline size between landmark point 1 and 2. This is the squared distance between landmark 1 and 2 in each direction. The equation can be applied only to 2D shapes, where $i = 3, 4, \cdots, k$ and k is the number of landmark points on each geometric form. Note that in this technique the co-ordinates of the first two landmark points ($i = 1, 2$) remain unchanged when transferred to Bookstein's shape space i.e.

$$(s_1^B, t_1^B) = (x_1, y_1)(s_2^B, t_2^B) = (x_2, y_2). \tag{2.35}$$

An alternative to baseline size is the centroid size. In this technique all available landmark co-ordinates are transferred relative to a centroid point calculated using Eq. 2.36

$$S(x) = \sqrt{\sum_{i=1}^{k}\sum_{j=1}^{m}(x_{ij} - \bar{x_j})^2}, \tag{2.36}$$

where m is the dimensionality of analysed geometry.

The centroid size technique is the most commonly used size measure however, according to Dryden and Mardia (1998) sometimes "*the variability in the points away from origin is larger than the points near the origin*" in either of shape spaces.

Better and advanced way of refining the shape of 2D and higher dimensional geometrical objects is by generalised Procrustes method (Dryden and Mardia, 1998), where several geometries are fitted using Procrustes superimposition. The Procrustes method compares shapes by superimposing a group of shape to obtain information as distances and residuals which can be then used for further analysis. The concept of Procrustes method is easily representable in 2D using complex numbers. Suppose that there are n configurations of geometrical forms $X_1, X_2, \cdots, X_n$ mapped with k number of landmark co-ordinates given as complex numbers.

One can use the following equation (Dryden and Mardia, 1998)

$$G(X_1, X_2, \cdots, X_n) = \frac{1}{n}\sum_{i=1}^{n}\sum_{l=i+1}^{n}\left\|(\psi_i X_i \Gamma_i + 1_k \chi_i^T) - (\psi_l X_l \Gamma_l + 1_k \chi_l^T)\right\|^2, \tag{2.37}$$

to minimize the distances between configurations, where

- χ_i^T is a translation vector,
- $\hat{\Gamma}_i$ is $m \times m$ rotation matrix,
- $\hat{\psi}_i > 0$ is scale parameter

and $l = i + 1$.

This is carried out by:

1. Translating each pair of shapes by χ_i^T.
2. Rotating each pair by matrix $\hat{\Gamma}_i$ with angle $0 < \theta < 2\pi$.

3. Scaling each pair by $\hat{\psi}_i > 0$.

The full Procrustes fit for each configuration X_i can be found by Eq. 2.38

$$X_i^P = \hat{\psi}_i X_i \hat{\Gamma}_i + 1_k \hat{\chi}_i^T, \tag{2.38}$$

where the arithmetic mean of the Procrustes fits is

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i^P. \tag{2.39}$$

In Eq. 2.37 and 2.38 the scale parameter is calculated as

$$\hat{\psi}_i = \sqrt{\frac{(X_i^T X_l X_l^T X_i)}{(X_i^T X_i)}}, \tag{2.40}$$

rotation angle as

$$\hat{\theta} = arg(X_i^T X_l) = -arg(X_l^T X_i) \tag{2.41}$$

and

$$\hat{\chi} = 0, \tag{2.42}$$

because it is assumed that configurations has been centred.

A useful quantity for shape comparison derived from Procrustes method is Procrustes residuals calculated using Eq. 2.43

$$r_i = X_i^P - \left(\frac{1}{n} \sum_{i=1}^{n} X_i^P \right). \tag{2.43}$$

For 3D and higher dimensional geometries the following iterative procedure is used

1. Remove translation. Translate each geometric object to a new centroid position

   $$t_{ij} = (x_{ij} - \bar{x_j}), \tag{2.44}$$

   where t_{ij} is denoted as translated shape space co-ordinate and x_{ij} as old shape space co-ordinate of landmark i at j-th direction and $\bar{x_j}$ is average calculated for original co-ordinates of all landmarks in each direction j.

2. Remove scale. Rescale shape to new coordinate space by finding the centroid size $S(x)$ and divide each translated landmark t_{ij} to shape centroid.

3. Remove rotation. Rotate shape around x, y or z axis by multiplying shape coordinates with rotation matrix

$$\mathbf{A} = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}, \tag{2.45}$$

where $0 < \Theta < 2\pi$.

A good method for further analysis is to investigate Procrustes residuals (shape differences) by principal component analysis.

Procrustes residual gives the difference between two shapes as quantity $r \in [0, 1]$, where 0 means that the two shapes are completely different and 1 means that the shapes match.

2.6.2 Principal Component Analysis

According to Shlens (2009) one applications of principal component analysis is "to reduce a complex data set to a lower dimension by extracting relevant information from confusing data revealing hidden, simplified structure that often underlie it". Relevant information extracted from data set in terms of principal components best explains variance in the data. More interestingly principal components can be tested for correlation with other underlying attributes/variables (Dryden and Mardia, 1998).

Eigenvectors and eigenvalues extracted from covariance matrix of data set are the most important components. The most commonly used algorithms to extract eigenvectors and eigenvalues from datasets are singular value decomposition (Shlens, 2009) and non-linear iterative partial least squares (Henning, 2008).

To illustrate the principal component analysis suppose $\mathbf{D}$ is a $k \times m$ data matrix, where k variables are tested m times. If not all variables in the data set have the same scale the variables should be firs normalised. This can be done by dividing each variable by its standard deviation. The principal components of $\mathbf{D}$ can be found as eigenvectors and eigenvalues of corresponding covariance matrix

$$\mathbf{D} = \begin{vmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{1,m} \\ \vdots & \vdots & \ddots & \cdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{vmatrix}. \tag{2.46}$$

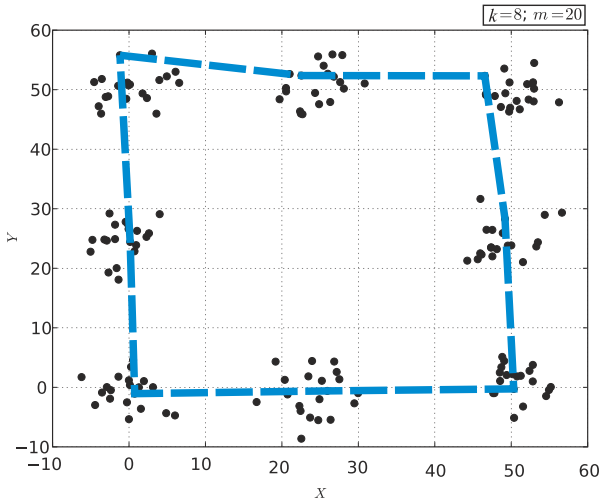


FIGURE 2.22: 20 superimposed 2D rectangles parametrized by eight landmarks.

To enhance the understanding the reader may refer to example shown on Fig. 2.22. This example consist of $m = 20$ polygons parametrised by $k = 8$ geometric landmarks in plane – the point clouds in the figure represent the landmarks. The number of variables in this case is $n = k \times 2 = 16$, since each landmark is defined by two variables. The size of data set is 16×20 i.e.

$$\mathbf{D} = \begin{vmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,20} \\ y_{1,1} & y_{1,2} & \cdots & y_{1,20} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k,1} & x_{k,2} & \cdots & x_{k,m} \\ y_{k,1} & y_{k,2} & \cdots & y_{k,m} \end{vmatrix}. \tag{2.47}$$

All variables have the same dimensionality which means that normalization of data set is not necessary. First the mean shape is calculated by

$$\bar{d}_j = \frac{\sum_{i=1}^{m} D_{ij}}{(m-1)}, \tag{2.48}$$

where $i = 1, 2, \cdots, m$ and $j = 1, 2, \cdots, n$. Then each shape in $\mathbf{D}$ is subtracted from mean shape

$$\bar{\mathbf{D}} = \mathbf{D} - \bar{\mathbf{d}}. \tag{2.49}$$

Eigenvectors υ_j and eigenvalues ζ_j of the covariance matrix

$$\mathbf{C} = \frac{\bar{\mathbf{D}} \times \bar{\mathbf{D}}^T}{(m-1)} \tag{2.50}$$

are the principal components for the data set. The percentage of significance for each PC is given as

$$\mathrm{PC_j} = \frac{100\zeta_j}{\sum_{j=1}^{p}\zeta_j}, \tag{2.51}$$

where p is the number of PC with percentage of significance greater than 0.

Extracted relevant information can be used to describe significant shape changes with fewer variables as well as to search for correlation of shape changes with other underlying variables. To examine the effect of each significant principal component to shape changes in arbitrary range of $-3 \leq c \leq 3$ (three times the standard deviation) the following equation is used

$$\mathbf{S} = \bar{\mathbf{D}} + c\sqrt{\zeta_j}\upsilon_j; \qquad j = 1, 2, \cdots, p, \tag{2.52}$$

where $j = 1, 2, \cdots, p$ and $c \approx N(0, 1)$ is random number with mean 0 and standard deviation 1.

2.6.3 Example application of geometric morphometrics in a design study.

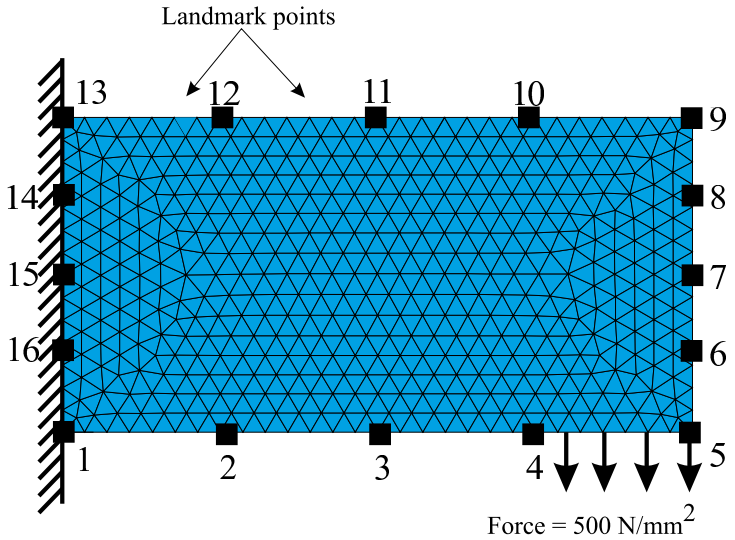


FIGURE 2.23: Constrained plate loaded with surface pleasure of $F = 500$ [N/mm^2]

Possible implementation of statistical shape analysis in structural design optimisation is investigated. For the purpose, first an initial design domain is constructed as a thin plate with dimensions 100×50 [mm] and thickness of 13 [mm]. The model is further parametrised with $k = 16$ geometric landmark points and four B-splines

i	PC	$\sqrt{\text{PC}}$	PC(%)	i	PC	$\sqrt{\text{PC}}$	PC(%)
1	366.34	9.14	85.4	6	7.49	2.74	1.8
2	22.95	4.79	5.4	7	0	0	0
3	13.37	3.66	3.1	8	0	0	0
4	10.39	3.23	2.4	.	.	.	.
5	8.52	2.92	1.9	32	0	0	0

TABLE 2.9: The percentage of shape variation explained by principal components.

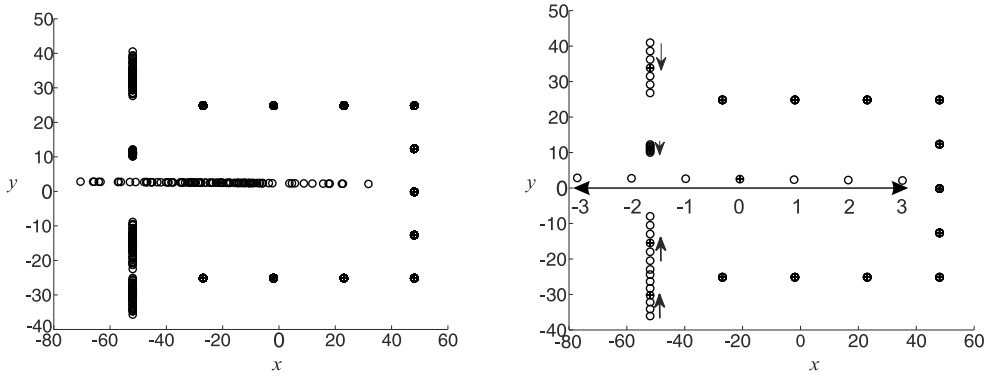
passing through the landmarks. Note that other types of parametric curve such as NURBS can be also successfully used to represent the design boundaries. This model is constrained and meshed with plane stress elements with thickness. The loading is given as $F = 500$ [N/mm^2] of surface pressure as shown in Fig. 2.23. The material properties are defined as steel with Young's modulus $E = 200$ GPa and Poisson's ration $\nu = 0.3$.

The objective of this example case study is to reveal possible connections between the significant shape variations and the two objective parameters:

- Area – A
- Maximum von Mises stress – σ^{vm}

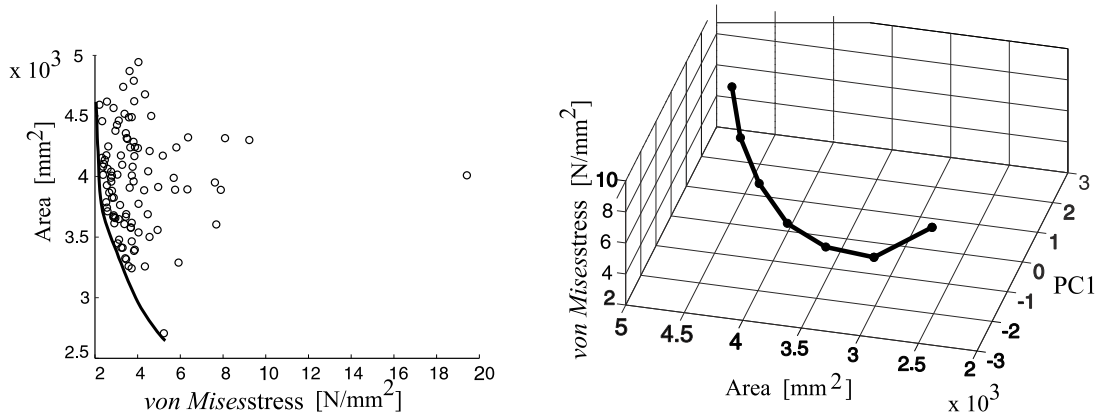
The minimisation of the two objectives is desirable, therefore a simple random search optimisation algorithm (Zhigljavsky, 1991), is used. For the purpose the co-ordinates of landmark points 1, 13, 14, 15 and 16 are used as free variables based on Gaussian distribution with arbitrary mean and standard deviation. The landmark points 1, 13, 14 and 16 are moved only along y axis while landmark 15 is moved along x and y. By moving these points 100 alternative designs of the structure are generated. All 100 new landmark co-ordinates are then used to extract the principal components explaining shape variations. The possibility of obtaining residuals using generalised Procrustes analysis is not explored since all analysed geometries are by default superimposed without any rotation, scale and translation components.

For convenience the notation of each principle component is PC_i, where $i = 1, 2, \cdots, 32$. The study reveals that the first and second principal component (PC_1,PC_2) explain 90.8% of shape variations, while the remaining principal components showed insignificant, explaining only 9.2% of shape variations listed in Table 2.9. The shape variations explained by PC_1, is illustrated in Fig. 2.24(a).



(a) PC_1 (b) PC_1 in the interval (-3,3)

FIGURE 2.24: Shape variations explained by principal components

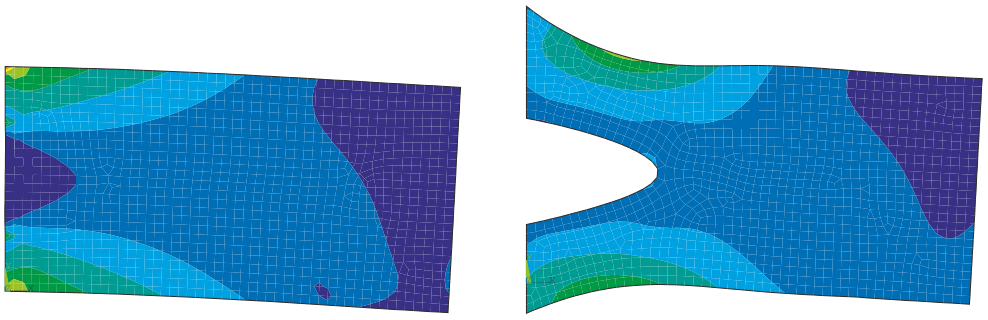


(a) Pareto optimal front (b) Scatter plot of A, σ^{vm} and PC_1

FIGURE 2.25: Relationship between shape variations and design optimisation objectives

Since only PC_1 describes significant shape variations it was decided to investigate its relation with A and σ^{vm}. The effect of PC_1 over the shape variation, was calculated by Eq. 2.52 for the range $c \in [-3, 3]$, where c is random number with mean 0 and standard deviation 1. The results are shown in Fig. 2.24(b). The Pareto-front between von Mises stress and the area is shown in Fig. 2.25(a). Scatter plot in Fig. 2.25(b) shows relationship between PC_1, A and σ^{vm}. It can be seen that minimal value for A and σ^{vm} is possible when $c = 0$.

A structurally improved design of the plate is shown on Fig. 2.26(b), where $\sigma^{\text{vm}} = 2.80[\text{N/mm}^2]$ and $A = 4.092 \times 10^3$ [mm^2]. Compared to the initial shape on Fig. 2.26(a) it can be concluded that σ^{vm} decreased by 50% and the A decreased by 12%.



(a) Initial – $\sigma^{vm} = 4.64$ N/mm^2; $A = 5 \times 10^3$ mm^2

(b) Optimised – $\sigma^{vm} = 2.80$ N/mm^2; $A = 4.092 \times 10^3$ mm^2

FIGURE 2.26: The von Mises stress contours in FEA of structural design

2.7 Shape optimisation methods

Meske et al. (2005) categorise shape optimisation methods into parametric and non-parametric approaches. The division made thus is not very clear because it depends on the choice of the geometry manipulation variables. In general, there are two distinctive shape optimisation techniques based on the choice of geometry manipulation variables – those based on manipulation of boundary mesh nodes of the FE model and those based on manipulation of parametric design boundary curves usually given as mathematical functions.

2.7.1 Shape optimisation based on movement of boundary mesh nodes

In the shape optimisation method presented in (Meske et al., 2005; Clausen and Pedersen, 2006; Meske et al., 2006) the geometry manipulation variables are identified as the co-ordinates of the mesh nodes. The nodes are moved inwards or outwards in the required search direction depending on the choice of a shape exploration algorithm. A commonly used strategy is to force the magnitude of the movement for each node to be proportional to the difference between the stress at that node σ and the reference stress σ_{ref}. Since the movement of the boundary nodes may lead to distortion of the inner mesh structure, the boundary nodal modifications are propagated to the interior region at each step of iteration. Due to non-uniform movement of boundary mesh nodes, usually the outlining design boundary becomes 'jagged'. A final smooth shape through the resulting 'jagged' boundary is achieved by interpolating curves or surfaces through the displaced boundary mesh nodes. An interesting algorithm for parallel topology and shape

optimisation based on movement of boundary mesh nodes is proposed in (Chen et al., 2002). Two important drawbacks of moving boundary mesh nodes for shape optimisation are:

1. The number of optimisation variables is quite large. This issue is addressed by Meske et al. (2005), where a method to overcome this is proposed.
2. The optimisation of shape is achieved only on design boundary areas selected by the user. A consequence of this is the transfer of stress concentration to other unselected boundary areas during the optimisation due to local geometry deformations at selected design boundary areas. In this way, the areas which appear to be of less interest to the designer may become a region of high stress concentration. This, on the other hand, may lead to a highly iterative shape optimisation procedure.

2.7.2 Shape optimisation based on parametric design boundary curves

The design variables during shape optimisation based on parametric design boundaries are the parameters of the curves. The altered CAD model defined by parametric curves or surfaces is re-meshed during each iteration. This is required since the existing FE mesh of the whole design domain is no longer valid, after morphing the design boundary. Some of the main advantages of this technique are:

- The ability to get closer to a global optima in most shape optimisation problems.
- Smaller number of optimisation variables and, therefore, faster convergence speed.
- Flexibility to increase or decrease the number of design variables.
- Intuitive integration with existing CAD modelling packages.

In (Chen and Torterelli, 1997), NURBS curves are used to optimise the shape of 3D connecting rod model. Other methods similar to this using NURBS parameterisation can be found in (Tang and Chang, 2001; Cervera and Trevelyan, 2005a). The former methodology depends on existing conceptual designs obtained through preliminary topology optimisation of rough initial design template using things such

as the ESO or homogenisation method. Then rather than manipulating boundary mesh nodes or elements, the shape optimisation algorithm manipulates the control points of the NURBS curve or surface. NURBS also have been used to optimise the cross sectional shape of 3D geometries involving prismatic parts.

The work presented in (Cervera and Trevelyan, 2005a) uses a similar approach to manipulate the design boundaries, but it avoids the use of preliminary topology optimisation. Also in this method the Boundary Element Method (BEM) (Banerjee and Butterfield, 1981) is preferred over the FEM for stress analysis – this is to reduce the meshing times, since in BEM the CAD model is meshed only at boundaries. The information obtained about the stress concentration on the surface mesh nodes is used to calculate the direction of movement and the magnitude of NURBS control points. This strategy is very similar to the boundary mesh nodes manipulation. The difference is that the optimisation variables here are the position of the control points of parametric NURBS rather than the surface mesh nodes. Another distinctive feature is that the number of control points (optimisation variables) can be increased and decreased during the search process.

The drawback of shape optimisation methods based on NURBS parameterisation is that the complexity of design to be generated determines the number of control points needed. For this reason, successful design optimisation using NURBS parameterisation is often design specific. The use of a small number of control points may shrink the design search space (simple shapes with less degrees of freedom), while that of more control points may expand it (allowing unrealistic 'bumpy' shapes with high curvatures). An appropriate selection of the number of control points is one of the conditions for finding the global optima within a reasonable number of search iterations. This is an unwanted artefact of NURBS parameterisation.

A method to avoid unrealistic shapes, in design exploration with NURBS, is presented by Chen et al. (2007). In this methodology, Boolean operations (union, intersection, set difference, etc.) are applied to NURBS and implicit parametric shapes (square, circle, polygon, etc.), deriving a common shape representational framework. This framework possesses the ability to maintain realistic geometries where they are necessary, while exploring larger solution spaces.

2.7.3 Parametric control points versus boundary mesh nodes

Shape optimisation using boundary mesh nodes can provide fast and reliable results if several conditions are fulfilled. Most of these conditions coincide with the conditions for successful shape optimisation using parametric control points. The two most important conditions are an existing conceptual design obtained through preliminary operations and sufficient a priori knowledge about the structural problem.

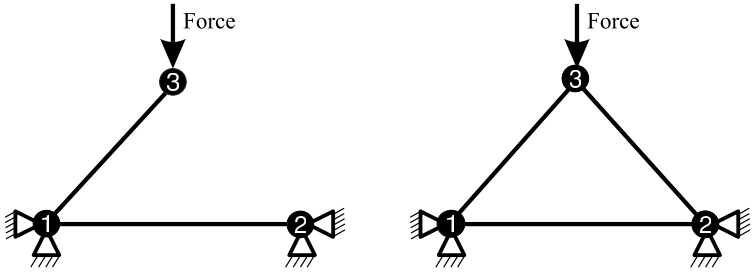
Despite the high number of optimisation variables, shape optimisation using boundary mesh nodes is often fast and efficient compared to parametric control points shape optimisation methods (Meske et al., 2005). Some of the arguments are that:

1. Shape optimisation using boundary nodes benefits from a priori knowledge about the stress concentration on the outer surface.
2. Numerical instabilities of FEA due to removed boundary elements are eliminated by applying filtering techniques to achieve smooth boundary shapes after each optimisation loop.
3. Re-meshing the geometry at each iteration during optimisation is avoided by simultaneous movement of all the affected nodes. This is probably the most significant benefit of using boundary mesh nodes as free variables.

The advantages listed above are not exclusive to boundary mesh nodes shape optimisation only. For example, in (Cervera and Trevelyan, 2005a,b) the parametric control points shape optimisation also benefits from a priori knowledge about stress concentration at the nodes. In fact, the control point of the parametric curve are moved according to the stress level at closest surface node.

While shape optimisation using boundary mesh nodes may be fast and efficient it is most effective in linear optimisation problems. Using this shape optimisation method, one can obtain relatively fast and reliable results if the shape is already in the vicinity of a global optima. On the other hand, shape optimisation using parametric control points is capable of discovering a global optima, since it operates with complex free-form curvatures to explore a larger design space.

Another drawback of using boundary mesh nodes as optimisation variables is the smoothing process required at the boundaries. Smoothing of the shape can be regarded as fitting free-form parametric B-splines through boundary nodes which



(a) Kinematically unstable truss – a mechanism

(b) Kinematically rigid truss

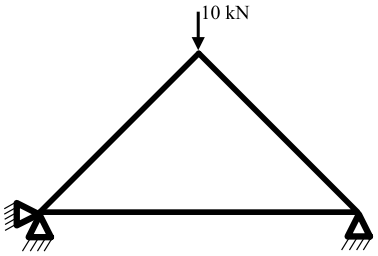
FIGURE 2.27: An illustration of kinematically stable and unstable truss structures. The Chebyshev–Grübler–Kutzbach's criterion (Sharma and Purohit, 2006) $3(n-1)-2j \geq 0$ can be used to determine if a truss is mechanism or not

are already an approximation of the desired shape. This procedure influences the optimisation process if the optimal solution is not robust enough to small shape deviations. The accuracy of fitting splines through surface nodes is also strongly dependent on the mesh size.

2.8 Optimisation of planar trusses

A truss is a network of discrete structural members (struts) connected at joints. The joints are hinges that allow rotation. A valid truss structure must be kinematically rigid not a mechanism. To determine if a structure is truss or mechanism one can use the Chebyshev–Grübler–Kutzbach's criterion (Sharma and Purohit, 2006). This criterion asserts that a planar (2D) structure is a mechanism if $3(n-1)-2j \geq 0$, where n is the number of structural members and j is the total number of degrees-of-freedom lost at joints. In a planar truss each joint has 3 degrees-of-freedom – translation in x and y-direction and rotation. A joint loses 1 degree-of-freedom when constructed by two members (links) and all 3 degrees-of-freedom when constructed by three or more members. In all other situations (none or only one member) the joint retains all three degrees-of-freedom. Consider the two structural members connected by flexible joint shown in Fig. 2.27(a). This is a mechanism because $3(2-1)-2\times 1=1$, since $j=1$ lost at joint 1 and $n=2$. To avoid generation of kinematically unstable truss structures, one can connect members in triangular formation as shown in Fig. 2.27(a). This structure is kinematically stable, because the total number of degrees-of-freedom lost at joints is $j=3$ (one for each node), therefore $3(3-1)-2\times 3=0$.

(a) A member



(b) A truss structure

FIGURE 2.28: A illustration a structural member and a structure. The structure consist of three structural members connected at nodes in triangular formation. The triangular formation of members is used to reduce the bending stress at members by transferring it to axial stress (tension or compression)

It is well accepted fact that triangular arrangement of structural members has another advantage. Such formation is used to build lightweight and strong structures. Consider the structural beam member shown in Fig.2.28(a). This member is constrained in plane at both ends. If a transverse force is applied in the middle of the member, this member will experience a high bending stress in the middle. Now consider a similar situation, but this time the structure is constructed by three members in triangular formation (see Fig. 2.28(a)). In this situation the structure will experience significantly lower bending stress compared to the previous case. In structural mechanics, a triangular formation of members is used to reduce the bending stress at members by transferring it to axial stress (tension or compression). Thus the magnitude of the bending stress becomes inconsiderably small compared to the magnitude of any axial stress. Most of the stress reduction is due to the geometrical arrangements of members, but there is also stress reduction due to introduction of additional structural members. The trade off in adding more members – thus reducing the overall stress – is adding weight to the structure. The principal objective in truss optimisation is to find the optimal number of truss members, their connectivity and cross-sectional area for a given structural problem.

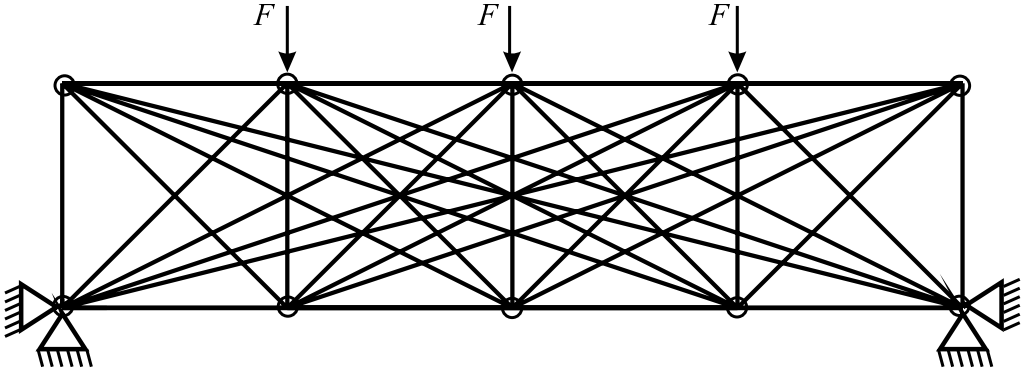


FIGURE 2.29: A illustration of 10-node, 45-member truss structure with boundary condition

2.8.1 Truss optimisation based on GA

Most truss problems consist of three optimisation stages (Deb and Gulati, 2001). These stages are: topology, sizing, and shape optimisation. In topology optimisation, the co-ordinates of all connection nodes are previously known and fixed. The problem to be solved is to find the optimal connectivity (topology) of members. In sizing optimisation, the cross-sectional area of each structural member is given as a free variable. In shape optimisation, the co-ordinates of connection nodes is of interest too.

Until recently, the three stages of truss optimisation have been considered as separate optimisation problems (Zhou and Rozvany, 1991; Kirsch, 1989; Topping, 1983). Such consideration is computationally inefficient, since it requires a separate optimisation run for each aspect of truss optimisation. On the other hand, simultaneous topology, sizing and shape optimisation of truss structures is considered to be a highly multi-modal optimisation problem. Therefore, various truss optimisation methodologies based on GA have been developed (Goldberg and Samtani, 1986; Ohsaki, 1995; Rajeev and Krishnamoorthy, 1992). An interesting method for simultaneous topology and sizing optimisation is presented in (Hajela and Lee, 1995). It consists of a bi-layered GA, where the first layer is responsible for finding a set of near optimal truss topologies. The optimised set of topologies are then used as initial seeds for sizing optimisation in the second layer of the GA. The drawback is that the methodology is still fragmented into optimisation stages. An alternative representation scheme for simultaneous topology, sizing and shape optimisation based on GA is proposed by Deb and Gulati (2001). In this procedure, initially all possible connections between the nodes are present, see Fig. 2.29. The unnecessary structural members are to be removed during the optimisation process. Here, the free variables are defined as cross-sectional areas of

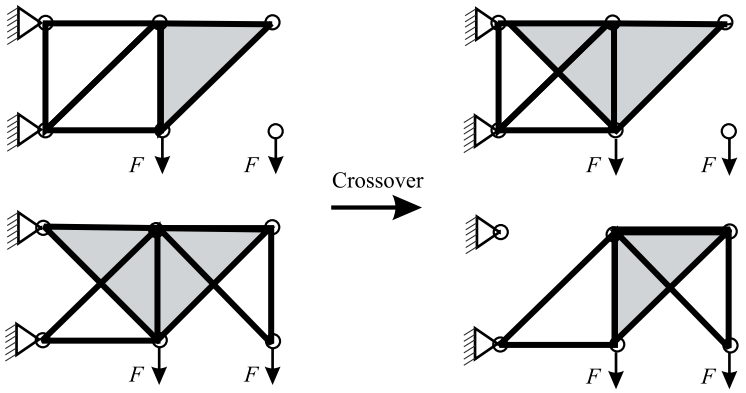


FIGURE 2.30: Crossover operation on two truss structures by swapping a pair of truss members in triangular formation

truss members. The 'removal' of each member depends on size of each members's cross-sectional area. A threshold minimum cross-section area is defined as $a_{\min}$. If the cross-sectional area of any member is below the threshold minimum $a_i \leq a_{\min}$ the corresponding members is removed. Sometimes this turns the structure into a mechanism. Such structures are accepted as invalid and penalised.

Another interesting methodology for simultaneous truss optimisation based on GA is described in (Kawamura et al., 2002). Contrary to the previously discussed method, here initially all members are missing – only the nodes are present. The necessary members are added during the optimisation process. Two of the most interesting features of this method are:

- The new members are added in sets of 3 in triangular formation.
- Unlike a standard GA, the crossover and mutation operations are applied directly to the phenotype, i.e. to the connectivity of two parent truss structures. The crossover is carried out by swapping some of the triangular formations of two truss structures, see Fig. 2.30.

The triangular formation of members is used to decrease the search time by avoiding generation of mechanisms (the constrained area in the solution space).

Chapter 3

A design description and optimisation framework based on Grammatical Evolution

In this chapter, a novel methodology for design synthesis and optimisation using Grammatical Evolution (GE) is presented. First several techniques for synthesis of variables in optimisation problems employing specific grammar rules presented in the BNF syntax are proposed. Single and multi-objective optimisation using GE are studied by considering several mathematical test functions. During the optimisation experiments, a modification to improve the convergence speed of the GE is proposed. The multi-objective grammar rule selection concept of GE is demonstrated by a proposal to extend its functionality, replacing the GA engine with NSGAII. The results from numerical optimisation experiments with GE are presented. The results obtained are then compared to those of optimisation with regular GA.

A design synthesis concept using GE (the automation concept of the SG in particular) is proposed. This includes the development of two alternative techniques to represent SG rules in BNF syntax. One of the proposed techniques is then further developed in Chapter 4 in a shape optimisation study of planar structures. This is then applied to a specific example of the optimisation of a crane hook.

3.1 BNF syntax for synthesis of constrained optimisation variables

TABLE 3.1: A BNF syntax to synthesise n number of variables in interval $0.00 \leq x_i \leq 0.99$.

⟨int⟩	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
$\bar{x}_i$	= 0.⟨int⟩⟨int⟩

In practice, the free variables of optimisation are usually constrained to an interval of values. A BNF syntax listed in Table 3.1 can be used to synthesise n number of constrained optimisation variables in the interval $0 \leq \bar{x}_i < 1$, where $i = 1, 2, \cdots, n$ is the number of variables.

This syntax requires a significantly shorter binary string compared to more general BNF syntax presented in Section 2.5.5, since the overall number of rules to be selected is limited to $2 \times n$ – two rules for each variable ($\bar{x}_i = 0.\langle\text{int}\rangle\langle\text{int}\rangle$). As a result, the optimisation would converge much faster. The synthesised optimisation variables $\bar{x}_i$ are taken as normalised values of the real variables x_i. The normalisation is carried out for the interval $x_\Delta = x_{\max} - x_{\min}$, where $x_{\max}$ and $x_{\min}$ are the upper and lower constraints of x_i. The value of x_i is calculated using the normalised value as $x_i = x_{\min} + \bar{x}_i \times x_\Delta$. This means that when $\bar{x}_i = 0$, $x_i = x_{\min}$ and when $\bar{x}_i = 1$, $x_i = x_{\max}$

The search step η of the synthesised variables $0 \leq \bar{x}_i < 1$ is related to required precision. The precision, on the other hand, is defined by a number of non-terminals ⟨int⟩ after the decimal point in the initial sentence e.g. the search step for double digit precision is $\eta = 0.01$. The search step for the real variables κ is calculated as

$$\kappa = \eta \times x_\Delta \tag{3.1}$$

The search step κ can be decreased by appending more non-terminals ⟨int⟩ after the decimal point. For example, if $x_\Delta = 5$ and $\eta = 0.001$ i.e. the search resolution for normalised value is given as $\bar{x} = 0.\langle\text{int}\rangle\langle\text{int}\rangle\langle\text{int}\rangle$, $\kappa = 0.005$.

TABLE 3.2: A example of BNF syntax used to synthesise a real number

⟨int⟩	::= ⟨int⟩⟨int⟩ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
x	= ⟨int⟩.⟨int⟩

The convergence speed of optimisation depends on the required precision of the synthesised values (number of non-terminals ⟨int⟩ after the decimal point) and the number of variables. If the required accuracy and boundaries of the variable to be synthesised are previously unknown, the BNF in Table 3.2 would converge slowly but with better solution than BNF in Table 3.1. Therefore, each BNF is problem specific and is selected based on preliminary information about the optimisation problem.

During numerical optimisation with GE, the binary string length μ is calculated by taking into account:

- The number of optimisation variables n.
- The desired search resolution (number of digits after the decimal point) of synthesised variables.
- The codon length β.
- The amount of allowed overuse of genetic information – number of wrappings over the codon string.

For example, suppose that the number of optimisation variables $n = 2$ and that search resolution of 2 digits after the decimal point is required ($\bar{x}_i = 0.\langle\text{int}\rangle\langle\text{int}\rangle$). The binary string length is calculated as $\mu = \beta \times n \times 2 = 16$ or $\lambda = \mu/\beta = 4$ codons, where $\beta = 4$ bits. Each codon is used to select a grammar rule, therefore an overall of 4 grammar rules are selected (one for each non-terminal ⟨int⟩).

A codon of length β bits provides selection of up to 2^β unique grammar rules. For efficiency, it is recommended to define β as small as possible. The minimum valid value of β is calculated by taking into account the number of rules in the BNF expression with the largest set.

The minimum possible codon length for BNF syntax from Table 3.2 would be $\beta = 4$ bit, since the number of unique rules in the largest BNF expressions (⟨int⟩ ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0) is 10. A 4 bit codon provides selection of up to $2^4 = 16$ unique grammar rules. In comparison, a 3 bit codon would be insufficient for the purpose, since it provides selection of up to $2^3 = 8$ unique grammar rules. This means that if 3 bit codons are used two of the grammar rules in the BNF expression (⟨int⟩ ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0) will be disregarded.

Another version of grammar rules used for the synthesis of optimisation variables is presented in Table 3.3. This BNF syntax is suitable for the synthesis of constrained

TABLE 3.3: A BNF syntax to synthesise n number of constrained optimisation variables in the interval $15.0 \leq x_i < 35.0$ without normalisation.

⟨int⟩	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
⟨int1⟩	::= 1⟨int3⟩ | 2⟨int⟩ | 3⟨int2⟩
⟨int2⟩	::= 1 | 2 | 3 | 4 | 0
⟨int3⟩	::= 5 | 6 | 7 | 8 | 9
x_i	= ⟨int1⟩.⟨int⟩

TABLE 3.4: An BNF syntax to synthesise n number of variables in the interval $0.0 \leq x_i \leq 9.9$

⟨int⟩	::= ⟨int⟩⟨int⟩ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
x_i	= 0.⟨int⟩×10

variables in the interval $15.0 \leq x_i < 35.0$ without normalisation. For example, an initial sentence $x_i = 1\langle\text{int3}\rangle.\langle\text{int}\rangle$ will generate values between 15.0 and 19.9. Again, the number of non-terminals ⟨int⟩ after the decimal point defines the search resolution of the synthesised values.

3.1.1 Single-objective optimisation

To illustrate single-objective optimisation using GE consider the 'bump function'

$$f(x)_{\max} = \frac{|\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)|}{\sqrt{\sum_{i=1}^{n} i x_i^2}}. \tag{3.2}$$

The optimisation of this function is subject to following constraints:
$\prod_{i=1}^{n} x_i > 0.75$ and $\sum_{i=1}^{n} x_i < 15n/2$, where $0 \leq x_i \leq 10$, $i = 1, 2, \cdots, n$ and n is the number of variables (Keane and Nair, 2005). The optimisation problem is stated as the maximisation of this function.

First, the number of optimisation variables is set to $n = 2$. Therefore, a synthesis of two optimisation variables for the interval $0 \leq x_i < 10$ by a BNF syntax is required. Suppose that the required search resolution of variables to be synthesised is unknown and the BNF syntax listed in Table 3.4 is used. The search completion criterion, for this run, is set to 20,000 evaluations of the function. The binary string length is arbitrarily selected as $\mu = 160$, since the search resolution of the synthesised optimisation variables is unknown. All other GE specific optimisation parameters are listed Table 3.5.

TABLE 3.5: Optimisation parameters used with BNF syntax listed in Table 3.4

Generations	500
Population size	50
Crossover probability(p_c)	0.8
Crossover type	multiple
Crossover points	1
Mutation type	multiple
Mutation probability (p_m)	0.1
Codon length (β)	8 bit
Genotype length (μ)	160 bit
Codons (λ)	20

The use of multiple mutations with high probability appears to be beneficial in GE optimisation, when the binary string length is relatively short. It introduces rather random search behaviour.

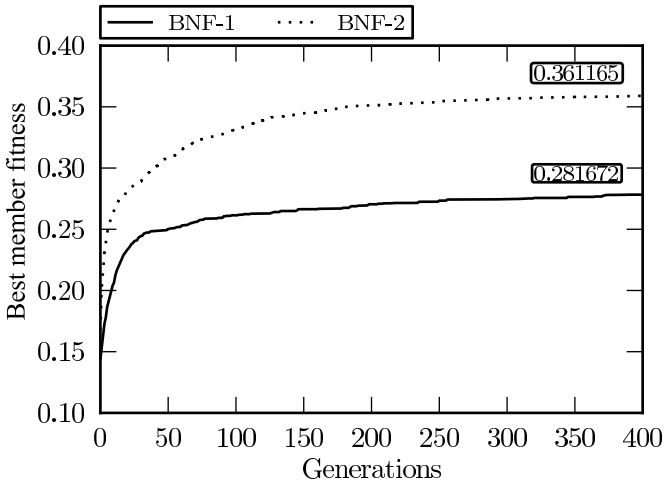


FIGURE 3.1: Averaged convergence comparison between BNF syntaxes presented in Table 3.4 and Table 3.6 for the 'bump' function with 2 variables.

The averaged convergence history by best member in a generation obtained from 100 optimisation runs of 'bump' problem with 2 variables is shown in Fig. 3.1 marked as BNF1. It shows that the optimal solution is away from the global optima of 0.36485 located at $x_1 = 1.5903$, $x_2 = 0.4716$, see (Keane and Nair, 2005). Now, consider that the required search resolution for generated variable is 5 digits after the decimal point. The more promising BNF syntax in this case will be the one presented in Table 3.6.

TABLE 3.6: An BNF syntax to synthesise n number of variables in the interval $0 \leq x_i \leq 9.9$.

$\langle$int$\rangle$	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
x_i	= 0.$\langle$int$\rangle\langle$int$\rangle\langle$int$\rangle\langle$int$\rangle\langle$int$\rangle \times 10$

TABLE 3.7: Optimisation specific parameters used in GE and GA

Parameter	GE 8 bit codons	GE 4 bit codons	GA
Generations	316	316	316
Population size	100	100	100
Crossover probability(p_c)	0.5	0.5	0.8
Crossover type	multiple	multiple	single
Crossover points	5	5	1
Mutation type	multiple	multiple	single
Mutation probability (p_m)	0.003	0.003	0.005
Codon length (β)	8 bit	4 bit	12 bit
Genotype length (μ)	800 bit	400 bit	–
Codons (λ)	100	100	–

In the BNF presented in Table 3.6 the number of rules to be selected is fixed to 10 i.e. the rule ($\langle$int$\rangle \rightarrow \langle$int$\rangle\langle$int$\rangle$) responsible for increasing the number of non-terminals $\langle$int$\rangle$ is removed. The initial sentence $x_i = 0.\langle$int$\rangle\langle$int$\rangle\langle$int$\rangle\langle$int$\rangle\langle$int$\rangle \times 10$ is used to select 5 grammar rules for each variable. The overall binary string length is calculated considering a disabled wrapping operation as follows $\mu = \beta \times 2 \times 5$, where 2×5 represents the number of grammar rules to be selected (5 for each variable) and $\beta = 4$.

A convergence comparison between GE optimisation using BNF syntax from Table 3.4 and Table 3.6 is shown in Fig. 3.1. They are marked as BNF1 and BNF2 respectively. The results are averaged for 100 runs by the best member in generation. The figure shows that the BNF2 run outperforms the BNF1 run in terms of convergence speed and the quality of the solution 0.3611 at $x_1 = 1.5813$, $x_2 = 0.4745$. This demonstrates that BNF2 is superior to BNF1 when used in numerical optimisation.

In the next numerical experiment, a convergence comparison between the GE optimisation and a standard GA is considered. The number of variables in the 'bump' problem is now increased to $n = 20$. In this comparison, BNF syntax from Table 3.6 is preferred for the GE runs. To evaluate the role of the codon length β, two cases are considered for the GE optimisation part. In the first case,

the length of codons is set to $\beta = 8$ bits, compared to $\beta = 4$ bits in the second case. The GA run is conducted using the OPTIONSMATLAB software package (Keane et al., 2008). All optimisation specific parameters in this comparison are listed in Table 3.7. The completion criteria is 60,000 evaluations. The averaged convergence comparison from 100 optimisation runs between GE $\beta = 8$ bit, GE $\beta = 4$ bit and GA by best member in generation is shown in Fig. 3.2. This figure shows that the reduction of codon length in GE from 8 bit to 4 bit has a positive effect to convergence speed. Note that compared to the GE runs the GA converges slowly on average, but in the end finds a better solutions. On the other hand, the GE optimisation with 4 bit codons converges much faster and outperforms the GA significantly in terms of convergence speed. Also, there is a sudden improvement of the optimisation convergence with GA starting from the 300-th generation. This might be due to a better constraint penalty handling mechanism provided by the OPTIONSMATLAB software package, since the global optima lies exactly on the constraint border. In comparison, the GE optimisation is not able to handle constraint penalties. The optimal solution obtained by the 4 bit GE run is 0.787 located at (3.1725, 5.2794, 3.2033, 3.1364, 3.0033, 3.0183, 3.2015, 0.5234, 3.0004, 2.4642, 0.5482, 0.4646, 0.5036, 0.5810, 0.4701, 0.4526, 0.374, 0.4490, 0.4403, 0.374).

3.1.2 Multi-objective optimisation

Multi-objective optimisation using GE is slightly more complicated in the sense that the selection procedure of the GA is extended. In this implementation of a multi-objective GE, the multi-objectiveness is achieved using NSGAII. The GA

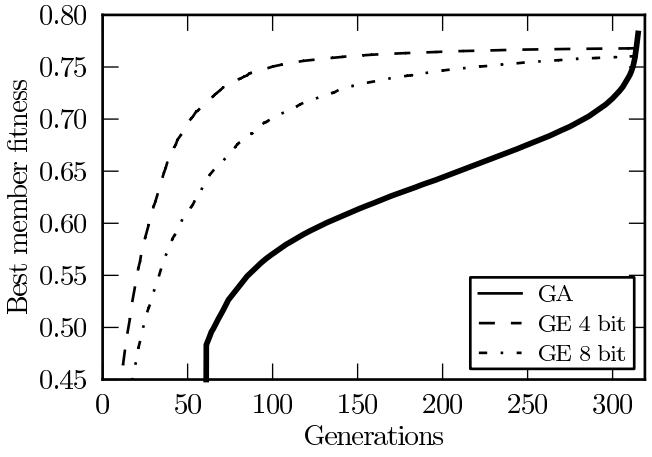


FIGURE 3.2: Averaged convergence comparison between GA, GE (4 bit) and GE (8 bit) from 100 repetitive runs of the 'bump' function with $n = 20$ variables

engine, which is basically the core of GE, is now replaced by NSGAII. As previously discussed in Chapter 2, the main difference between the GA and the NSGAII is that the latter uses a ranking selection strategy. The first multi-objective optimisation experiment is conducted using the mathematical test functions

$$\begin{aligned} O_1(x) &= 1 - \exp(-4x_1)\sin(6\pi x_1)^6 \\ O_2(x) &= g(x)\left(1 - (O_1(x)/g(x))^2\right), \end{aligned} \tag{3.3}$$

where

$$g(x) = 1 + 9\left(\sum_{i=2}^{10}(x_i/9)\right)^{0.25} \tag{3.4}$$

and $0 \leq x_i \leq 1$; $n = 10$. In the literature, this function is frequently referred to as EC6 function (Deb et al., 2000), where minimisation of $O_1(x)$ and $O_2(x)$ is required. The BNF syntax used to tackle this problem is listed in Table 3.8.

TABLE 3.8: A BNF syntax used in optimisation of the EC6 function from Eq. 3.3

⟨int⟩	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
x_i	= 0.⟨int⟩⟨int⟩⟨int⟩

The averaged Pareto-fronts obtained from 100 repetitive optimisation runs with GE and NSGAII for the EC6 function are presented in Fig. 3.3.

The effectiveness of multi-objective optimisation algorithms is measured by taking into account:

- The number of objective function evaluations required to reach an optimal (non-dominated) Pareto-front – usually a good measure for this is when all members in a population are of rank 1 for several sequential generations.
- The accuracy of the Pareto-front – how close the discovered front is to the real Pareto-front. It is often determined qualitatively by using a visual inspection.
- A quantitative measure of the discovered near optimal Pareto-front – here, the discovered Pareto-fronts are compared quantitatively based on Hypervolume indicator, distance, diversity, spread or cardinality, see (Fonseca et al., 2005). Most commonly used measure is the Hypervolume indicator (Auger

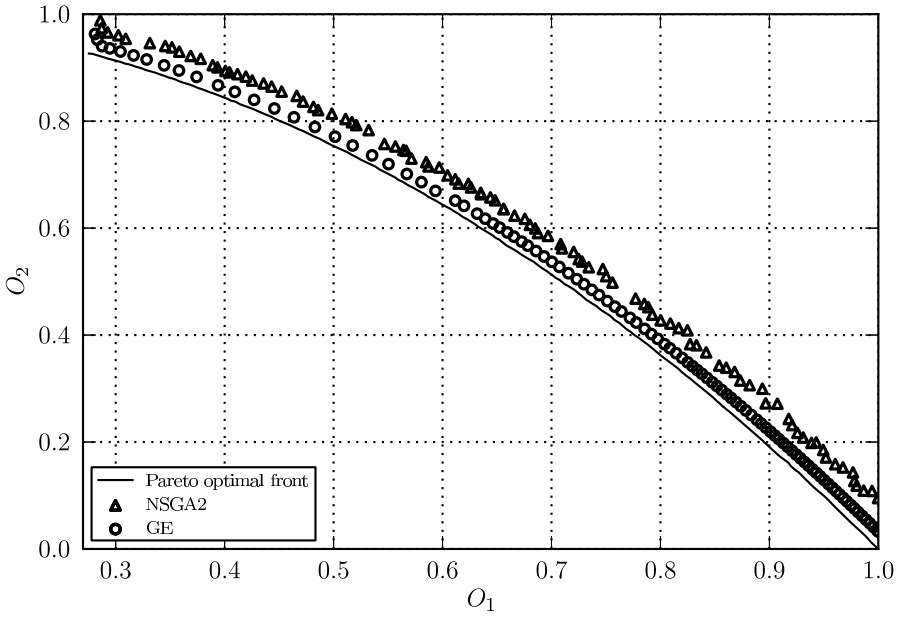


FIGURE 3.3: An averaged comparison between GE and NSGA2 (C++ implementation – Deb) from 100 repetitive runs using the bi-objective test function EC6 (Deb et al., 2000). The optimisation variables are $n = 10$ given in the interval $0 \leq x_i \leq 1$

TABLE 3.9: Optimisation specific parameters used in GE and NSGAII

	GE		NSGAII	
Parameter	EC6	KUR	EC6	KUR
Generations	300	50	300	50
Population size	100	50	100	50
Crossover type	multiple	multiple	single	single
Crossover probability(p_c)	0.8	0.8	0.8	0.8
Crossover points	10	5	1	1
Mutation type	multiple	multiple	single	single
Mutation probability (p_m)	0.025	0.025	0.05	0.005
Codon length (β)	4 bit	4 bit	12 bit	12 bit
Genotype length (μ)	120 bit	60 bit	–	–
Codons (λ)	30	15	–	–

et al., 2009), where a reference point on the solution space is used to calculate the Hypervolume of each Pareto-front to be compared.

The C++ implementation of NSGAII by Deb et al. (2000) provides better solution with approximately similar optimisation parameters for the same problem. This is shown in Fig. 3.3, where the continuous line represents the real Pareto-front.

The second multi-objective optimisation experiment is conducted using the test functions

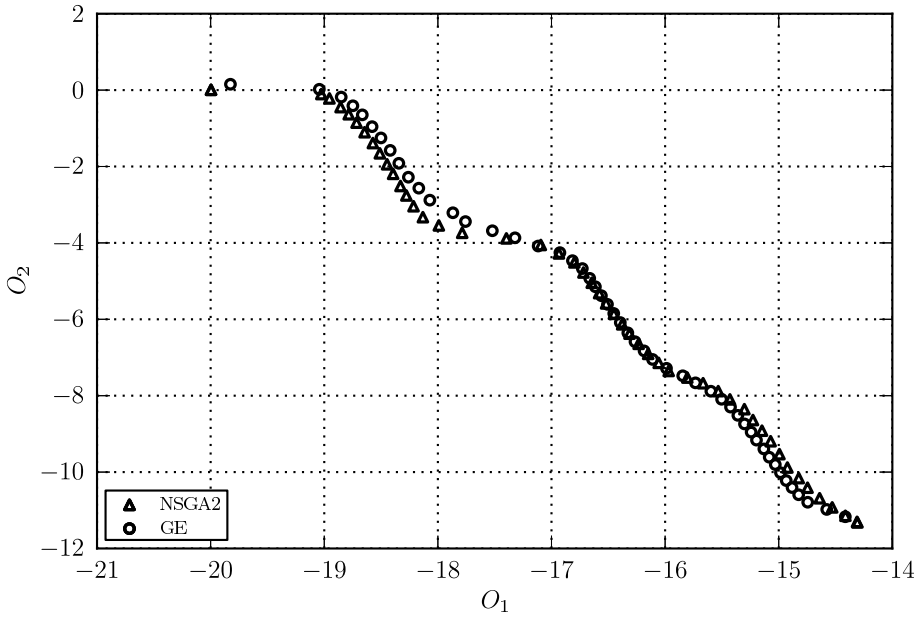
$$\begin{aligned} O_1(x) &= \sum_{i=1}^{n-1} \left(-10 \exp \left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right) \\ O_2(x) &= \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5 \sin (x_i)^3 \right), \end{aligned} \tag{3.5}$$

where $-5 \leq x_i \leq -5$ and $i = 1, 2, 3$. In the literature, this function is frequently referred to as KUR function (Deb et al., 2000), where minimisation of $O_1(x)$ and $O_2(x)$ is required. The GE and NSGAII specific optimisation parameters used in this problem are also listed in Table 3.9. The preferred BNF syntax in this case is listed in Table 3.10.

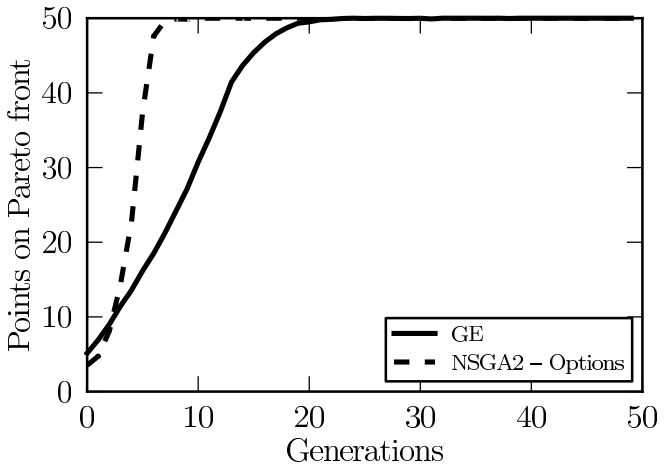
TABLE 3.10: A BNF syntax used in optimisation of the KUR function from Eq. 3.5

⟨int⟩	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
$\bar{x}_i$	= 0.⟨int⟩⟨int⟩⟨int⟩⟨int⟩⟨int⟩

The averaged Pareto-fronts obtained from 100 optimisation runs with GE and GA for the KUR problem are presented in Fig. 3.4(a). In this comparison the performance of OptionsNSGA2 is on par with the multi-objective GE. Fig. 3.4(b) show the average number of rank 1 members by generations. A quantitative comparison of the Pareto-fronts is not required because a visual comparison is sufficient to reveal that the fronts are incomparable i.e. obtained by GE fronts are almost identical to those obtained by NSGAII for the test functions considered.

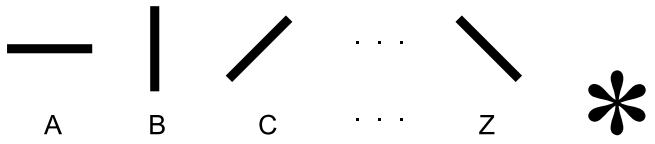


(a) Near optimal Pareto-fronts



(b) Members of rank 1 by generations

FIGURE 3.4: An averaged convergence comparison between GE and NSGA2 (OptionsNSGA2 implementation) from 100 repetitive runs using the bi-objective test fiction KUR (Deb et al., 2000). The optimisation variable are $n = 3$ given in the interval $-5 \leq x_i \leq -5$



(a) A terminal SG vocabulary set V_T of straight lines (b) Non-terminal primitive set V_N

FIGURE 3.5: SG vocabulary V_T with straight lines and V_N with non-terminal primitive

3.2 A generic framework for the proposed automation of SG using GE

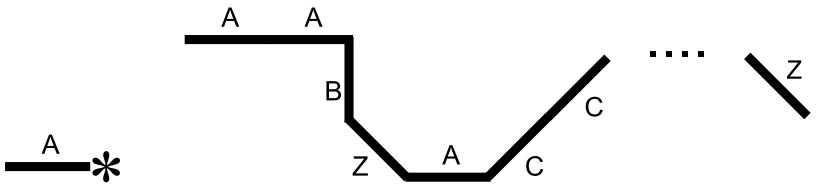
The proposed automation of SG for shape synthesis are described here in general terms first. *An attempt is made to allow the shape of the SG sentence to be kept variable.* The development of SG and its automation requires an SG vocabulary and a non-terminal primitive. The shapes belonging to the terminal vocabulary can be parametrised in a number of ways – one such set that uses straight lines is shown in Fig. 3.5(a). The vocabulary set containing these primitives becomes $V_T = \{A, B, C, \cdots, Z\}$. The non-terminal asterisk is a member of the non-terminal vocabulary set $V_N = \{\langle X \rangle\}$ (Fig.3.5). With this vocabulary and with the possibility of choosing a range of shapes belonging to the two sets that can be parametrised in a variety of ways, a strategy of automating SG is presented next.

The proposed design synthesis and optimisation framework based on automated SG is a special case of GE, where SG rules are introduced as BNF syntax. In the proposed design synthesis and optimisation framework, the SG rules are given in the following form

$$\langle X \rangle ::= A \mid B \mid C \mid \cdots \mid Z \mid A\langle X \rangle \mid B\langle X \rangle \mid C\langle X \rangle \mid \cdots \mid Z\langle X \rangle. \tag{3.6}$$

These rules are selected by GA, where A, B, C, $\cdots$, Z are primitive shapes and $\langle X \rangle$ is called marker non-terminal primitive shape.

If the initial shape grammar sentence is given as S = A $\langle X \rangle$ shown in Fig. 3.6(a), the final sentence, after application of SG, is not limited to a particular number and sequential arrangement of the primitives. It depends on decisions made of selecting SG rules using binary string provided by the GA – similar to GE. This is because the non-terminal $\langle X \rangle$ adds additional shape primitives to the initial



(a) An initial SG sentence

(b) Final SG sentence after application of SG rules

FIGURE 3.6: SG sentences that build on given vocabulary sets $\mathrm{V_T}$ and $\mathrm{V_N}$

sentence. Therefore, the length and shape of synthesised curve is variable. In this way, an infinite number of lines can be connected in sequential order synthesising a range of complex piecewise 2D curves such as

$$\mathrm{S} = \mathrm{A\ A\ B\ Z\ A\ C\ C\ \cdots\ Z}, \tag{3.7}$$

shown in Fig 3.6(b).

Curves with the flexibility regarding the number of primitives and their parametrisation are particularly useful in shape optimisation because they are capable of representing complex geometric boundaries with good accuracy. Each of the synthesised curves represents an unique design of the boundary geometry, where the fitness of the curve is returned to the GA as feedback. The numerical representation of primitive shapes can be given as parametric curves or mathematical functions.

One of the ways of parametrising a planar straight line is to use a pair of points with x and y co-ordinates i.e.

$$l = p^{\mathrm{b}} \_\ p^{\mathrm{e}}, \tag{3.8}$$

where $p^{\mathrm{b}} = (x, y)$ and $p^{\mathrm{e}} = (x, y)$ are two points representing the beginning and the end of the line. A set of n lines parametrised in this way can be used as a terminal vocabulary of shape grammar primitives i.e. $\mathrm{V_T} = \{l_1, l_2, \cdots, l_n\}$.

3.2.1 A BNF syntax to achieve positional continuity

The piecewise positional continuity between two terminal line primitives can be achieved by the following rule,

$$p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}} \rightarrow p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+1}^{\mathrm{e}}, \tag{3.9}$$

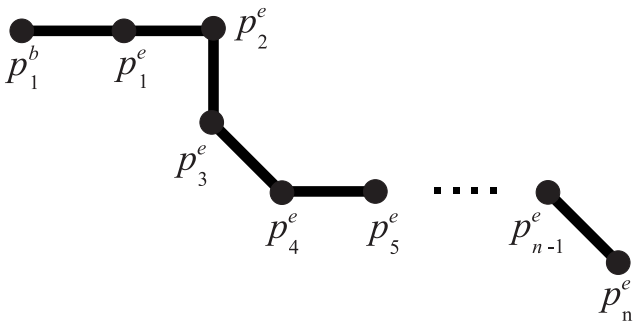


FIGURE 3.7: Piecewise curve constructed by n number of line primitives

TABLE 3.11: A BNF syntax representing two SG rules with line primitives.

⟨asterisk⟩	::= $p_i^{\mathrm{e}}\_p_{i+1}^{\mathrm{e}}$; ⟨asterisk⟩ | $p_i^{\mathrm{e}}\_p_{i+1}^{\mathrm{e}}$
S	= $p_1^{\mathrm{b}}\_p_1^{\mathrm{e}}$; ⟨asterisk⟩

where $i = 1, 2, \cdots, n$. Note that the subscript to the variables in Eq. 3.9 is now introduced which refers to the terminal primitive number. This rule appends a line l_{i+1} to an already existing line l_i where the end point (p_i^{e}) of l_i is the beginning point ($p_{i+1}^{\mathrm{b}} = p_i^{\mathrm{e}}$) of l_{i+1}. Defined in this way, the rule allows positional continuity of synthesised complex curve. Successive application of this rule would lead to the synthesis of a piecewise curve constructed by n number of line primitives which maintains positional continuity as follows

$$p_1^{\mathrm{b}}\_\ p_1^{\mathrm{e}};\ p_1^{\mathrm{e}}\_\ p_2^{\mathrm{e}};\ p_2^{\mathrm{e}}\_\ p_3^{\mathrm{e}};\ \cdots;\ p_{n-1}^{\mathrm{e}}\_\ p_n^{\mathrm{e}} \tag{3.10}$$

which is shown in Fig. 3.7.

If the non-terminal vocabulary is given as $\mathrm{V_N} = \{\langle\text{asterisk}\rangle\}$, the rule from Eq. 3.9 can be formalised in the BNF syntax presented in Table 3.11. This BNF syntax contains two grammar rules similar to the one shown in Eq. 3.9. One of the rules terminates the addition of new primitives to the sentence by removing the non-terminal ⟨asterisk⟩, while the second rules appends a primitive to the existing initial sentence and keeps the non-terminal primitive ⟨asterisk⟩ in the sentence.

Here, the analogy between SG and BNF syntax is clear. Each terminal primitive $l_i = p_i^{\mathrm{b}}\_\ p_i^{\mathrm{e}}$ represents a line with orientation and location given by point co-ordinates p_i^{b} and p_i^{e}. The non-terminal ⟨asterisk⟩ represents the marker primitive shape, which in this case is an asterisk.

3.2.2 A BNF syntax to allow branching of lines

A rule which allows split of the initial line into two branches can be defined as follows

$$p_i^{\mathrm{b}}\_\ p_i^{\mathrm{e}} \rightarrow p_i^{\mathrm{b}}\_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}}\_\ p_{i+1}^{\mathrm{e}};\ p_i^{\mathrm{e}}\_\ p_{i+2}^{\mathrm{e}}, \tag{3.11}$$

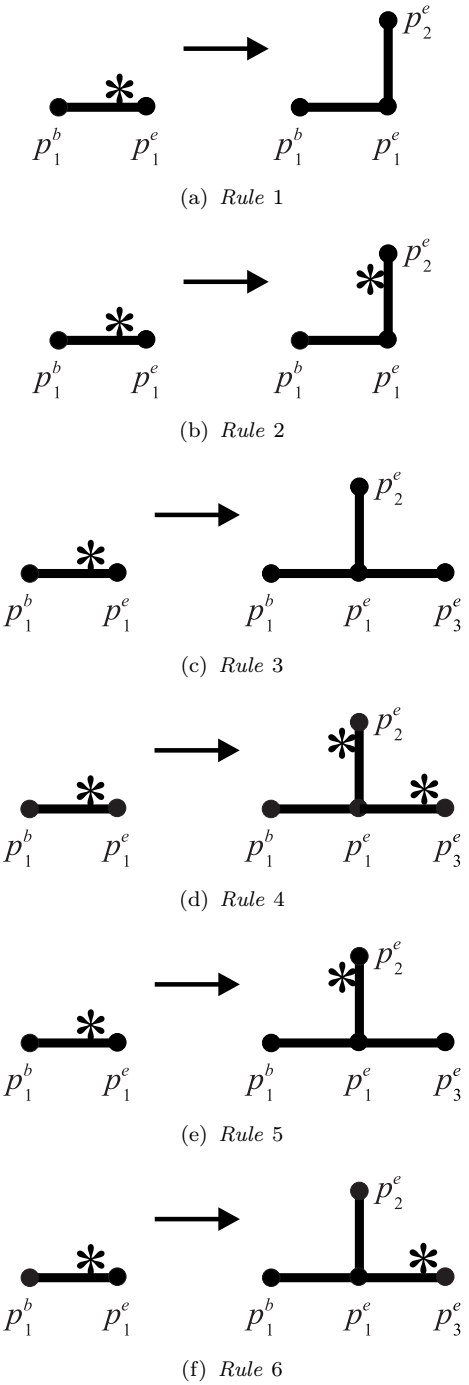
where $i = 1, 2, \cdots, n$.

This rule instructs lines l_{i+1} and l_{i+2} to be appended to already existing line l_i, where l_{i+1} and l_{i+2} share a common construction point which is the end point of l_i i.e. $p_{i+1}^{\mathrm{b}} = p_{i+2}^{\mathrm{b}} = p_i^{\mathrm{e}}$.

TABLE 3.12: A BNF syntax representing six SG rules with line primitives.

⟨asterisk⟩	::=	p_i^{e}_p_{i+1}^{e}; ⟨asterisk⟩ | p_i^{e}_p_{i+1}^{e} | p_i^{e}_p_{i+1}^{e}; p_i^{e}_p_{i+2}^{e} | p_i^{e}_p_{i+1}^{e}; ⟨asterisk⟩; p_i^{e}_p_{i+2}^{e}; ⟨asterisk⟩ | p_i^{e}_p_{i+1}^{e}; ⟨asterisk⟩; p_i^{e}_p_{i+2}^{e} | p_i^{e}_p_{i+1}^{e}; p_i^{e}_p_{i+2}^{e}; ⟨asterisk⟩
S	=	p_1^{b}_p_1^{e}; ⟨asterisk⟩

The rules shown in Eq. 3.9 and Eq. 3.11 can be formalised via the BNF syntax presented in Table 3.12. The six grammar rules from the BNF syntax shown in Table 3.12 can be be used to synthesise piecewise curves by line primitives which are capable to split into branches. The non-terminal primitive ⟨asterisk⟩ is used to append new primitives to the initial sentence – a single line represented by two points S = $p_1^{\mathrm{b}}\_\ p_1^{\mathrm{e}}$.



(a) *Rule* 1

(b) *Rule* 2

(c) *Rule* 3

(d) *Rule* 4

(e) *Rule* 5

(f) *Rule* 6

FIGURE 3.8: Six SG rules with line primitives

These six rules are visualised in Fig. 3.8 as follows:

(i) $\{p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}}; \langle \text{asterisk} \rangle \rightarrow p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+1}^{\mathrm{e}}\}$ – This rule appends a line to already existing line and terminates the addition of new lines by removing the non-terminal $\langle \text{asterisk} \rangle$ as shown in Fig. 3.8(a).

(ii) $\{p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}}; \langle \text{asterisk} \rangle \rightarrow p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+1}^{\mathrm{e}};\ \langle \text{asterisk} \rangle\}$ – This rule appends a line to already existing line, however it does not terminate the addition of new lines by keeping the non-terminal $\langle \text{asterisk} \rangle$ in the sentence as shown in Fig. 3.8(b).

(iii) $\{p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ \langle \text{asterisk} \rangle \rightarrow p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+1}^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+2}^{\mathrm{e}}\}$ – This rule splits the initial line into two by appending two lines at the end point of the initial line as shown in Fig 3.8(c). The addition of new lines is terminated by removing the non-terminal $\langle \text{asterisk} \rangle$ at each of the branches as shown in the same figure.

(iv) $\{p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}}; \langle \text{asterisk} \rangle \rightarrow p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+1}^{\mathrm{e}};\ \langle \text{asterisk} \rangle;\ p_i^{\mathrm{e}} \_\ p_{i+2}^{\mathrm{e}};\ \langle \text{asterisk} \rangle\}$ – This rule is the same as the rule (iii), however the non-terminal $\langle \text{asterisk} \rangle$ is preserved in both branches as shown in Fig. 3.8(d). This means that additional primitives can be appended to each of the branches.

(v) $\{p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}}; \langle \text{asterisk} \rangle \rightarrow p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+1}^{\mathrm{e}};\ \langle \text{asterisk} \rangle;\ p_i^{\mathrm{e}} \_\ p_{i+2}^{\mathrm{e}}\}$ – This rule is the same as the rule (iii), however the non-terminal $\langle \text{asterisk} \rangle$ is preserved only at the first branch as shown in Fig. 3.8(e). This means that the second branch terminates while the first branch can be further developed.

(vi) $\{p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ \langle \text{asterisk} \rangle \rightarrow p_i^{\mathrm{b}} \_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+1}^{\mathrm{e}};\ p_i^{\mathrm{e}} \_\ p_{i+2}^{\mathrm{e}};\ \langle \text{asterisk} \rangle\}$ – This rule is the same as the rule (v). The only difference is that the first branch is terminated while the second branch can be further developed as shown in Fig. 3.8(f).

3.2.3 A BNF syntax to allow variable length of the primitives

The proposed BNFs in Table 3.11 and Table 3.12 do not take into account the variability in the primitive lines length, because the length of each line in the terminal vocabulary set $\mathrm{V_T}$ is already fixed. This leads to a large terminal vocabulary set in order to accommodate all possible length variations of line primitives,

TABLE 3.13: A BNF syntax representing two SG rules with line primitives of variable size.

⟨asterisk⟩	::= $p_i^{\mathrm{e}}\_p_{i+1}^{\mathrm{e}}$; ⟨asterisk⟩ | $p_i^{\mathrm{e}}\_p_{i+1}^{\mathrm{e}}$
⟨int⟩	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
p_i^{e}	= (⟨int⟩, ⟨int⟩, ⟨int⟩)
S	= $p_1^{\mathrm{b}}\_p_1^{\mathrm{e}}$; ⟨asterisk⟩

which is a drawback. An extended BNF syntax is presented in Table 3.13. This syntax overcomes the drawback by introduction of additional grammar rules and primitives to synthesise end point co-ordinates p_i^{e}. In this way, a real value for x and y co-ordinates of the end point p_i^{e} can be generated by the GA in the same fashion as in the earlier optimisation examples, where the non-terminal ⟨int⟩ is used to synthesise the x and y co-ordinates of the ending point p_i^{e}. It should be noted that the BNF syntax can be further modified to allow fractional lengths using the strategy presented earlier combining decimal with digits.

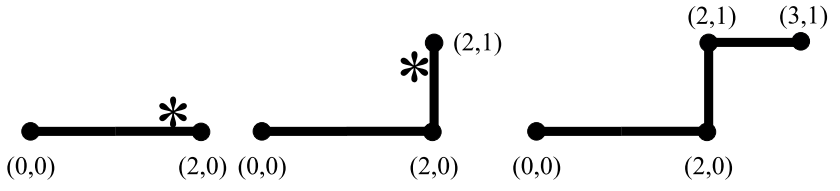
3.3 Illustrative examples

(i) Example 1: Synthesis of a step shape using line primitives.

Consider a planar line $l_1 = p_1^{\mathrm{b}}\_\ p_1^{\mathrm{e}}$ given by two construction points i.e., where $p_1^{\mathrm{b}} = (0, 0)$, and $p_1^{\mathrm{e}} = (2, 0)$ shown in Fig. 3.9(a). If this line is given as initial sentence the application of the grammar rule in Table 3.11

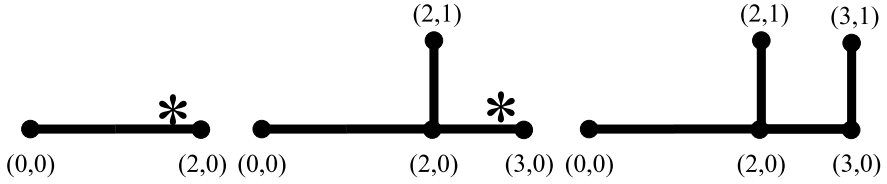
$$p_1^{\mathrm{b}}\_\ p_1^{\mathrm{e}};\ \langle\text{asterisk}\rangle \rightarrow p_1^{\mathrm{b}}\_\ p_1^{\mathrm{e}};\ p_1^{\mathrm{b}}\_\ p_2^{\mathrm{e}};\ \langle\text{asterisk}\rangle \tag{3.12}$$

transforms the straight line into a corner shape as shown in Fig. 3.9(b), where $p_2^{\mathrm{e}} = (2,\ 1)$. The presence of the non-terminal ⟨asterisk⟩ on the left hand side suggests that the application of this rule will be repeated. The



(a) Initial SG sentence (b) Resulting intermediate SG sentence (c) Final SG sentence

FIGURE 3.9: An illustration of grammar rules in Table 3.11 applied to line primitives



(a) Initial SG sentence (b) Resulting intermediate SG sentence (c) Final SG sentence

FIGURE 3.10: An example of a line bifurcated into an F-shape using grammar rules in Table 3.12

application of the second rule transforms the final sentence to

$$p_1^{\mathrm{b}}\_\ p_1^{\mathrm{e}};\ p_1^{\mathrm{e}}\_\ p_2^{\mathrm{e}};\ p_2^{\mathrm{e}}\_\ p_3^{\mathrm{e}} = (0,0)\_\ (2,0);\ (2,0)\_\ (2,1);\ (2,1)\_\ (3,1) \quad (3.13)$$

as shown in Fig 3.9(c).

(ii) Example 2: Synthesis of branched F-shape using line primitives.

The use of BNF syntax presented in Table 3.12 would follow the same sequence of grammar rule application events. The initial sentence as in Example 1 is a straight line, however the application of the grammar rule in Table 3.12

$$p_i^{\mathrm{b}}\_\ p_i^{\mathrm{e}};\ \langle\text{asterisk}\rangle \rightarrow p_i^{\mathrm{b}}\_\ p_i^{\mathrm{e}};\ p_i^{\mathrm{e}}\_\ p_{i+1}^{\mathrm{e}};\ p_i^{\mathrm{e}}\_\ p_{i+2}^{\mathrm{e}};\ \langle\text{asterisk}\rangle \quad (3.14)$$

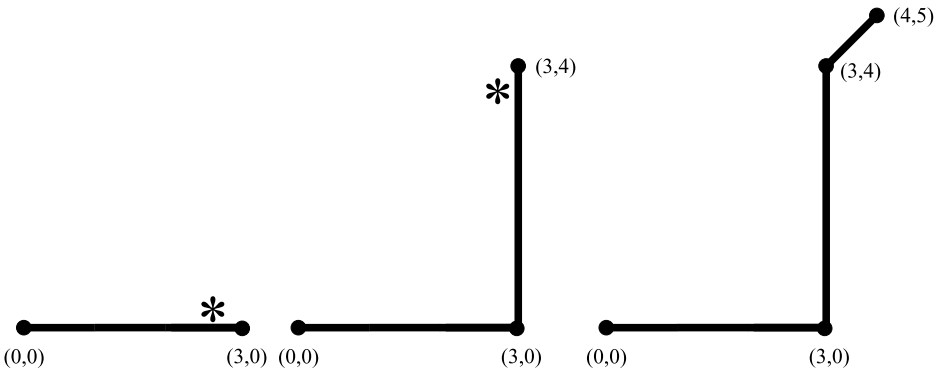
splits the synthesised curve as shown in Fig. 3.10(b). This rule preserves the non-terminal ⟨asterisk⟩ in one of the branches, therefore a second rule is applied which leads to synthesis on the F-shape curve shown in Fig. 3.10(c). The final sentence of the curve shown in Fig. 3.10(c) is

$$\begin{aligned} &p_1^{\mathrm{b}}\_\ p_1^{\mathrm{e}};\ p_1^{\mathrm{e}}\_\ p_2^{\mathrm{e}};\ p_1^{\mathrm{e}}\_\ p_3^{\mathrm{e}};\ p_1^{\mathrm{e}}\_\ p_4^{\mathrm{e}} = \\ &(0,0)\_\ (2,0);\ (2,0)\_\ (2,1);\ (2,0)\_\ (3,0);\ (3,0)\_\ (3,1) \end{aligned} \quad (3.15)$$

(iii) Example 3: Shapes combining primitives of *variable* length.

In case of variable length line, the non-terminal ⟨int⟩ from BNF syntax in Table 3.13 could synthesise the x and y co-ordinates for $p_1^{\mathrm{e}}, p_2^{\mathrm{e}}$ and p_3^{e} as follows

$$\begin{aligned} p_1^{\mathrm{e}} &= (\langle\text{int}\rangle, \langle\text{int}\rangle) = (3,0) \\ p_2^{\mathrm{e}} &= (\langle\text{int}\rangle, \langle\text{int}\rangle) = (3,4) \\ p_3^{\mathrm{e}} &= (\langle\text{int}\rangle, \langle\text{int}\rangle) = (4,5) \end{aligned} \quad (3.16)$$



(a) Initial SG sentence (b) Resulting intermediate SG sentence (c) Final SG sentence

FIGURE 3.11: An illustration of the application of the BNF syntax in Table 3.13 implemented on a 3-segment line

Here, the rule selection process is similar to Example 1, however additional rules are selected in order to synthesise point coordinates. An example of curve synthesis using the BNF syntax is presented in Table 3.13 with lines of variable length realised in Fig. 3.11.

3.4 Discussions

This chapter demonstrated how GE can be used as a numerical optimisation tool. During numerical optimisation experiments, it was noticed that the convergence speed of optimisation with GE strongly depends on the selected BNF syntax, the crossover type, and the rate. Other important parameters are the length of the binary string μ and the multiple mutation probability p_m. The binary string calculation of efficient length μ depends on the number of optimisation variables and the desired search resolution of synthesised variables. Increasing the binary string length μ has a negative effect of increasing the search time.

Numerical optimisation with GE can be seen as an enhanced version of GA or NSGAII, where the genetic material (a binary string representing the search space) is mapped onto the phenotype (the objective function variables representing the solution space) using grammar rules. The benefit of using GE is an improved convergence speed due to the advanced mapping procedure between the binary string and the real variables using phrase-structure grammar rules.

There has been much discussion about the efficacy of single point crossover operation in GE. One study introduces a new improved crossover operation (Francone

et al., 1999), while another study remarks on the importance of the traditional single point crossover to GE (O'Neill et al., 2001b). In some of the optimisation experiments a 20 points crossover operation with probability $p_c = 0.8$ was used successfully. The use of multiple point crossovers appears to be beneficial when the binary string length is particularly long ($\mu > 50$). In contrast, a GE with relatively short binary sting length ($\mu < 50$) may perform best with single point crossover and high probability of multiple mutations ($p_m = 0.025$).

The role of grammar in the synthesis of complex systems. The specific techniques (BNF syntaxes) proposed for the synthesis of optimisation variables and geometrical designs (automated SG) are the building blocks of the design description and optimisation framework proposed here. Using phrase-structure grammar one can develop alternative BNF syntaxes efficient in the synthesis of various complex systems. This is the most innovative feature of the proposed framework i.e. the proposed framework is generic and yet efficient for multidisciplinary design description and optimisation. The development of this approach to shape and topology description is provided in the following chapters.

3.5 Conclusions

The numerical optimisation experiments using various BNF syntaxes – for synthesis of constrained optimisation variables – confirmed that the GE can be successfully employed as a single and multi objective optimisation tool. The convergence speed of GE was further improved by a modification in the grammar rule mapping procedure. The modification was described as dynamic adjustment of codon bit string length β based on defined BNF syntax. This led to efficient calculation of genotype length μ, which influenced the convergence speed of numerical optimisation in a positive way. Following this modification the GE showed significant improvement, where the standard codon length was reduced from 8 bit to 4 bit. On the average, the GE outperformed the standard GA in terms of convergence speed in the test function considered. This was a surprising outcome, since the core engine of the GE was a GA. The multi-objective concept of GE was realised by replacing the GA engine with NSGAII. Experimental results with several multi-objective test function matched with available results in the literature.

During optimisation experiments with GE, it was noticed that its convergence speed depends on:

- The BNF syntax selected.
- Selected binary crossover type and the rate p_c i.e. single point crossover vs. multiple point crossover.
- Selected binary mutation type and rate p_m i.e. single point mutation vs. multiple point mutation.
- Efficient calculation of genetic code (binary sting) length μ.

The last three factors listed are related to standard GA, however their influence is different in GE because of the BNF mapping procedure of optimisation variables.

An experimental finding was that correctly setting the crossover is an essential part of GE. In GE, the selection of crossover type and rate strongly depend on binary string length and the BNF syntax used, i.e. it is problem specific.

The algorithmic methodology to automate the SG presented here fills the gap between the original proposal of SG by Stiny and Gips (1971) and its actual realisation in automated design search studies using CAD modelling tools. The implementation in the following chapter fully automates the shape identification and grammar rule selection processes. The automation is carried out using evolutionary intelligence, in particular GE. Despite its illustration with one specific example, this approach is generic. The automation of the SG presented has the promise to be used as an design search tool involving grammatical operation with primitive shapes in design, architecture and engineering. The next chapter explores the ideas developed in the present chapter to design optimisation where the function evaluation is replaced by a finite element analysis and the calculation of the overall weight for a bi-objective problem. The latter is a relatively cheap calculation whereas stress analysis may in practice be a very demanding computational task.

Chapter 4

Synthesis and optimisation of planar shapes using an automated SG: application to crane hook design

In this chapter, the design description and optimisation framework proposed in the previous chapter is applied to shape description and optimisation of planar structures. The technique presented here is general and can be applied to any 2D shape description followed by optimisation. The example of a crane hook is taken as a case study since it typically represents frequently encountered situations in engineering design. A BNF syntax used for the synthesis of planar parametric curves based on GE is proposed. The syntax provides a generic yet efficient shape synthesis scheme using SG rules with arc primitives of variable size. To evaluate the effectiveness of the proposed methodology, the results obtained are compared with these from design optimisation using NURBS control points manipulation based on a GA search strategy.

4.1 Planar shape synthesis using SG

In shape optimisation, the initial design domain is usually represented by domain boundaries. The boundaries are classified into designable and non-designable regions. The designable boundaries are of shape optimisation interest. In the planar design domain shown in Fig. 4.1 the non-designable boundaries are the fixed part of the design. They are required for applying boundary conditions and to ensure

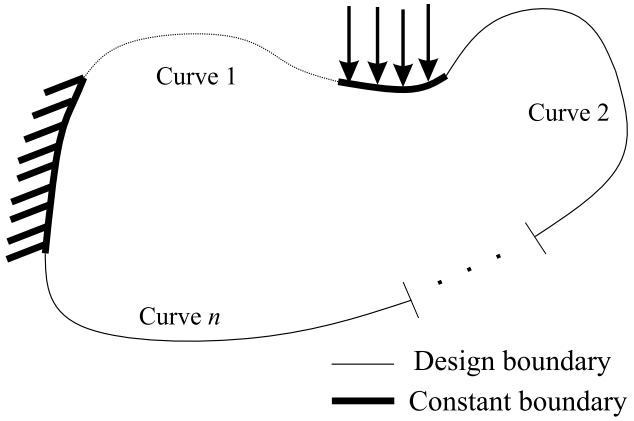


FIGURE 4.1: Structural design domain represented with design and constant boundaries. The boundary conditions are given as a displacement constraint and load applied to highlighted constant regions.

some essential design features. One set of the constant boundaries, in this domain, is used to constrain the movement of the domain in the x and y directions and to apply the loads as shown. These regions are highlighted in Fig. 4.1. The design boundaries of interest are represented by n number of planar curves connecting the constant boundaries. Their shape is of optimisation interest.

The design synthesis and optimisation framework derived in this chapter can be used to synthesise n number of 2D curves representing the design boundaries. Furthermore, a specific BNF syntax for optimising the shape of the synthesised curves is proposed in the next chapter.

The proposed shape synthesis technique based on SG is generic and is applicable to a class of boundary shape optimisation problems. In this thesis it is used in structural optimisation context. If the shape of a 2D profile shown in Fig. 4.1 is taken up as structural design optimisation problem, the objective of optimisation is the minimisation of maximum von Mises stress and the weight of the structure. The point where the stress is maximum within the structure changes in the design when the shape is altered and this quantity needs to be minimised. In the next section the proposed methodology for synthesis of planar piecewise curves using SG rules with arc primitive shapes is described.

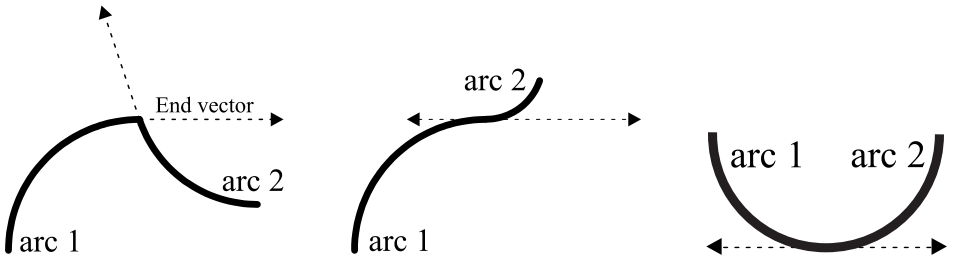
4.2 A technique for the synthesis of planar curves using arcs

The synthesis of planar shapes using SG is further developed in this section using arcs as the primitive geometrical shape. These arcs are parametrised using a radius and a signed angle. The general approach accounting for the desired continuity condition is presented. The methodology is generic and can be applied to a host of planar shapes restricted only by the choice of circular arcs. This is justified given that a large number of engineering components are described using circular features. Further, the proposed methodology is applied to a simple example of a 3-piece planar curve (in section 4.3) and to more complex design problem combined with stress analysis in the subsequent section 4.4.

The implementation of the proposed grammar based design description and optimisation framework for 2D shapes includes the following three stages:

- Stage 1: Selection of the SG primitives. One of the common requirements in structural shape optimisation is to avoid generation of sharp corners and jagged design boundaries unless they are explicitly required to represent a particular design boundary feature. This is because such boundaries pose a risk of high stress concentration, which may result in early material failure and increased risk of crack propagation. Therefore, an important feature of any piecewise parametric curve is its continuity level C_n, where n provides a measure of curve smoothness. In functional parametric representation of curves, C_n continuity means that at least n derivatives of two parametric curves are tangential.

  Let us assume that in an automated SG, all primitive shapes are given as open curves (each curve has a beginning and end which does not coincide in space). The first level of geometric continuity between open curves arranged in a piecewise manner is C_0 (positional) continuity. This continuity is achieved when primitive shapes are just connected. An example of C_0 continuity between two arcs is shown in Fig. 4.2(a). The next two levels of geometric continuities are the C_1 (tangential) and C_2 (curvature) continuity. C_1 continuity requires that the end vectors of connecting open curves are tangential, while C_2 continuity requires that the end vectors of open curves are tangential and of equal length. An example of C_1 and C_2 continuity by two arcs is shown in Fig. 4.2(b) and Fig. 4.2(c), respectively. Usually C_1



(a) C_0 continuity (b) C_1 continuity (c) C_2 continuity

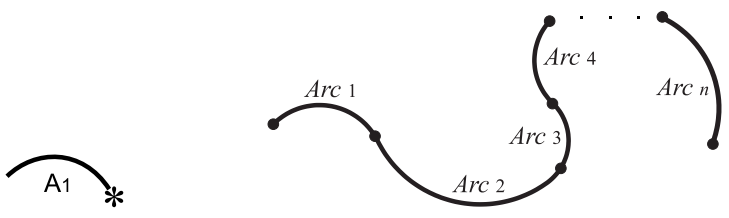
FIGURE 4.2: Three types of parametric continuities between two planar arcs known as C_0, C_1 and C_2 continuities

continuity (Fig. 4.2(b)) between open primitive curves is sufficient for synthesis of a smooth and natural looking piecewise curve for most engineering requirements.

In the design problem undertaken here, the SG primitives are selected as arcs of variable radius. The idea of selecting arcs as primitive shapes is driven by the fact that a C_1 continuous curve can be synthesised as a SG sentence by arranging m number of arc primitives in a particular order. An arc can be defined by two parameters:

1. A radius R.
2. A rotation θ, given its centre.

The radius R_i determines the size, while θ_i prescribes the extent of the i-th arc. Also, the sign of the angle θ_i determines the direction of rotation(i.e. clockwise or counter clockwise). Consider a terminal vocabulary $\mathrm{V_T} = \{Arc_1, Arc_2, \cdots, Arc_m\}$, where A_i is an arc of variable size. The non-terminal vocabulary is $\mathrm{V_N} = \{\langle \mathrm{X} \rangle\}$, where $\langle \mathrm{X} \rangle$ is single non-terminal primitive shape.



(a) An initial sentence with arcs

(b) A piecewise C_1 continuous curve synthesised by SG with arcs

FIGURE 4.3: SG with arc primitives

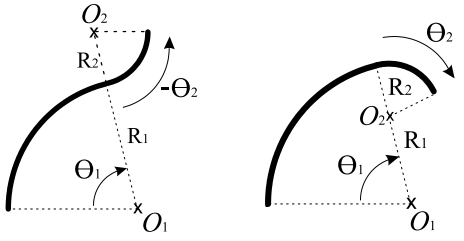
The two vocabulary sets are used to define the following BNF expression

$$\langle \mathrm{X} \rangle ::= \mathrm{A}_1 \mid \mathrm{A}_2 \mid \cdots \mid \mathrm{A}_m \mid \mathrm{A}_1\langle \mathrm{X} \rangle \mid \mathrm{A}_2\langle \mathrm{X} \rangle \mid \cdots \mid \mathrm{A}_m\langle \mathrm{X} \rangle \tag{4.1}$$

This expression can be used to synthesise a piecewise C_1 continuous curve of m arcs connected in sequential order. If an initial SG sentence is given as $\mathrm{S} = \mathrm{A}_1\langle \mathrm{X} \rangle$ (Fig. 4.3(a)), the final sentence could be

$$\mathrm{S} = \mathrm{A}_1, \mathrm{A}_5, \mathrm{A}_2, \mathrm{A}_4, \cdots, \mathrm{A}_m \tag{4.2}$$

as shown in Fig. 4.3(b).



(a) Outer C_1 continuity between two arcs

(b) Inner C_1 continuity between two arcs

FIGURE 4.4: C_1 continuity between two arcs

An important part of arranging arcs in piecewise manner is to maintain C_1 continuity between them. The two arc primitives in Fig. 4.4(a) share a common construction point and maintain C_1 continuity. The end vector of the smaller (second) arc defined by radius R_2, and rotation angle -θ_2 is tangential to the end vector of the larger (first) arc. In the same figure, the direction of the rotation of the second arc is inverted. The location of the centre of the second circular arc O_2 is calculated as a distance $D = |R_i + R_{i+1}|$ perpendicular to the end vector of the first arc starting from its centre O_1. Another tangency between the two arcs is shown in Fig. 4.4(b). Here, the location of O_2 is calculated using $D = |R_i - R_{i+1}|$. Note that in this case the sign of the rotation angle is preserved. To maintain C_1 continuity between two arcs, it is necessary to invert the sign of the rotation angle θ_2 every time $D = |R_i + R_{i+1}|$ and preserve it, when $D = |R_i - R_{i+1}|$. The C_1 continuity between two arcs presented in this way depends on the sign of the second radius R_{i+1}.

- Stage 2: Definition of the BNF syntax.

  The arc primitive shapes of variable size defined by radius R_i and angle θ_i are involved in the definition of four SG rules given as BNF syntax presented

TABLE 4.1: SG rules with arc primitives used to synthesise C_1 continuous piecewise curves.

$\langle \text{tri} \rangle$	::= $(R_i,\ \theta_i)$ | $(-R_i,\ \theta_i)$ | $(R_i,\ \theta_i), \langle \text{tri} \rangle$ | $(-R_i,\ \theta_i), \langle \text{tri} \rangle$
$\langle \text{int1} \rangle$	::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
$\langle \text{int2} \rangle$	::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
$\langle \text{int3} \rangle$	::= 3 | 4 | 5 | 6 | 7 | 8 | 9
U_j	$= p, \langle \text{tri} \rangle$

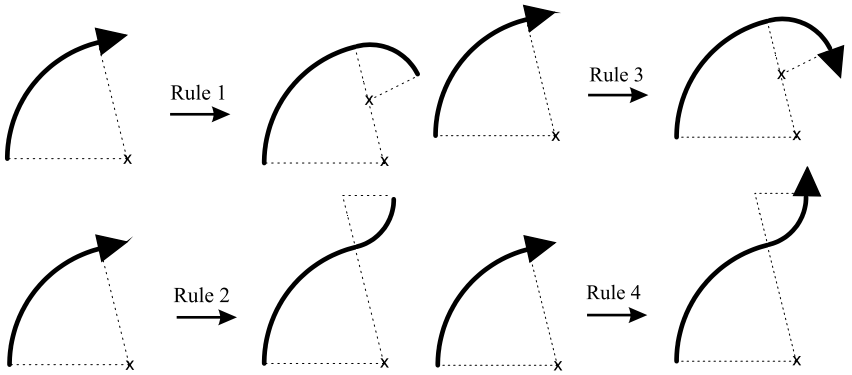
in Table 4.1, where $R_i = 0.\langle \text{int2} \rangle \langle \text{int1} \rangle$ and $\theta_i = 0.\langle \text{int2} \rangle \langle \text{int1} \rangle \times 90$. The initial sentence

$$U_j = p, \langle \text{tri} \rangle \tag{4.3}$$

in this BNF syntax triggers the synthesis of n piecewise parametric planar curves U_j, where $j = 1, 2, \cdots, n$ and p is the initial construction point of each curve. Point p is also the initial construction point of the first arc, its role is discussed later in this section.

The synthesis of parametric piecewise curves U_j is carried out by applying one of the four SG rules:

- *Rule* 1 – $\{\langle \text{tri} \rangle \rightarrow (R_i, \theta_i)\}$
- *Rule* 2 – $\{\langle \text{tri} \rangle \rightarrow (-R_i, \theta_i)\}$
- *Rule* 3 – $\{\langle \text{tri} \rangle \rightarrow (R_i, \theta_i), \langle \text{tri} \rangle\}$
- *Rule* 4 – $\{\langle \text{tri} \rangle \rightarrow (-R_i, \theta_i), \langle \text{tri} \rangle\}$



(a) Terminating SG rules (b) Non-terminating SG rules

FIGURE 4.5: Four unique SG rules with planar arcs used in synthesis of piecewise parametric curves

for each non-terminal ⟨tri⟩.

These SG rules are illustrated in Fig. 4.5. They can be divided into two groups: terminating rules *Rule* 1 and *Rule* 2 (Fig. 4.5(a)) and non-terminating rules *Rule* 3 and *Rule* 4 (Fig. 4.5(b)). The non-terminal ⟨tri⟩ represents the non-terminal shape primitive shown in Fig. 4.5(a) and Fig. 4.5(b) as a filled triangle sketched at the end of the arcs. This marker is used to extend the piecewise curve by appending arcs by the non-terminating rules. The terminating rules are used to stop the addition of arcs by removing this marker.

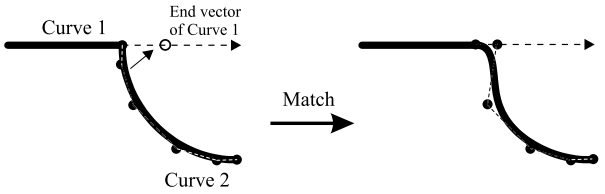
The maximum number of arcs is not limited to a fixed value but it depends on how many times the non-terminating SG rules are selected. However, the minimum number of arcs is one – in case the first selected rule is terminal e.g. *Rule* 2 or *Rule* 1, which remove the non-terminal ⟨tri⟩ from the sentence. In the proposed BNF syntax, the domain of each arc is of variable size. It is determined by the unit arc radius R_i and angle of rotation θ_i in degrees. The smoothness of the synthesised piecewise curves generated from arcs can be controlled by selecting only arcs with large radius. The local shape features of the synthesised curve can be controlled by mixing large and small arcs.

- Stage 3: Fitting synthesised piecewise curves using rotation, translation and scale.

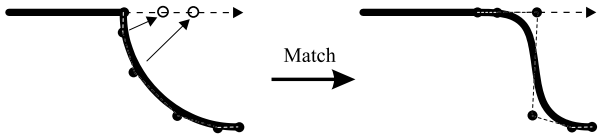
  In the described earlier shape optimisation problem the design boundaries were given as n planar curves. The BNF syntax presented in Stage 2 can be used for the synthesis of n piecewise parametric curves U_j, where $j = 1, 2, \cdots, n$.

  The shape of the synthesised curves U_j is invariant under rotation, translation and scale (Kendall et al., 1999), these three geometric transformation operations are used to fit the curve between the constant design boundaries, see Fig. 4.2. Unfortunately, after the transformation the fitted curves maintain only C_0 continuity with the constant design boundaries. This is because the end vectors of the fitted piecewise curves do not match to the end vectors of the constant curves.

  Here C_1 continuity of the overall 2D geometry (constant boundaries) is achieved by employing the *Match* function provided in the commercial CAD package RHINOCEROS3D. During this operation the two end vectors of the transformed piecewise curves (U_j) are modified to match with the end vectors of the constant (matched) boundaries.



(a) C_1 continuity



(b) C_2 continuity

FIGURE 4.6: The *Match* function provided by RHINOCEROS3D. The two C_0 continuous curves are matched to achieve C_1 and C_2 continuity, respectively

The *Match* transformation is illustrated in Fig. 4.6(a), where one of the control points of the matching curve is moved. The movement of this control point modifies the end of the curve in such a way that its end vector becomes tangential to the end vector of the constant (matched) curve. Fig. 4.6(b) shows *Match* operation to maintain C_2 continuity, where 2 of the control points from the matching curve are moved. Note that the RHINOCEROS3D package utilises NURBS to represent all types of curves including arc primitives.

4.3 An example of the synthesis of a continuously parametrised 3-piece curve

Considering the BNF syntax presented in Table 4.1 the terminal vocabulary set for the proposed shape synthesis technique is

$$\mathrm{V_T} = \{(R_i, \theta_i), (-R_i, \theta_i), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, \tag{4.4}$$

while the non-terminal vocabulary is

$$\mathrm{V_N} = \{\langle \mathrm{tri} \rangle, \langle \mathrm{int1} \rangle, \langle \mathrm{int2} \rangle, \langle \mathrm{int3} \rangle\}. \tag{4.5}$$

The rule selection is initiated by the non-terminal $\langle \mathrm{tri} \rangle$. Lets assume that the first applied rule is *Rule* 3 which transforms the initial sentence to

$$U = p, (R_1, \theta_1), \langle \mathrm{tri} \rangle. \tag{4.6}$$

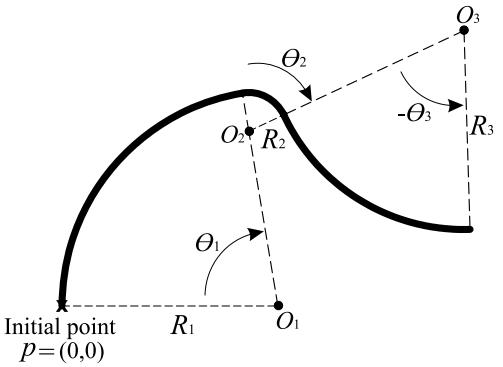


FIGURE 4.7: A piecewise C_1 continuous parametric curve synthesised by 3 planar arcs of variable size

This sentence also contains the non-terminal ⟨tri⟩, therefore *Rule* 3 is applied one more time transforming the sentence to

$$U = p, (R_1, \theta_1), (R_2, \theta_2), \langle \text{tri} \rangle. \tag{4.7}$$

The last applied rule is *Rule* 2, this rule terminates the rule application by removing the non-terminal primitive

$$U = p, (R_1, \theta_1), (R_2, \theta_2), (-R_3, \theta_3). \tag{4.8}$$

After the selection of the SG rules, the value for each arc parameter $R_i = 0.\langle \text{int2} \rangle \langle \text{int1} \rangle$ and $\theta_i = 0.\langle \text{int2} \rangle \langle \text{int1} \rangle \times 90$ is synthesised in the same fashion by selecting a digit for each non-terminal ⟨int2⟩ and ⟨int1⟩. Considering the proposed BNF syntax in Table 4.1, in this example, the first arc is synthesised with radius $R_1 = 0.\langle \text{int2} \rangle \langle \text{int1} \rangle = 0.79$ and rotation angle $\theta_1 = 0.\langle \text{int2} \rangle \langle \text{int1} \rangle \times 90 = 83$ and so on. This leads to the final parametric curve sentence as

$$U = p, (0.79, 83), (0.12, 79), (-0.70, 65). \tag{4.9}$$

Point p provides the co-ordinates of the initial construction point of the first arc – in this example $p = (0, 0)$. The centre of the first circular arc O_1 is calculated as a distance given by R_1 starting from the initial point p in the x direction, see Fig. 4.7.

The piecewise curve shown in this figure is synthesised from the initial sentence

$$U = p, \langle \text{tri} \rangle, \tag{4.10}$$

by applying SG shown in Fig. 4.5 in following order:

1. *Rule* 3
2. *Rule* 3
3. *Rule* 2

The second arc in Fig. 4.5 is defined by relatively small radius $R_2 = 0.12$. It represents a local curve feature compared to $R_1 = 0.79$ and $R_3 = -0.70$. The proposed shape synthesis approach using SG will be further used in conjunction with FE-based stress analysis in order to optimise the shape of a crane hook for the bi-objective optimisation problem simultaneously minimising the weight and the maximum stress

4.4 An illustrative design problem – 2D shape of a crane hook

The shape of the 2D profile of a crane hook is now taken up to apply and demonstrate the proposed methodology for synthesis and optimisation of planar curves using SG. The 2D geometry is divided in to designable and non-designable boundaries shown in Fig. 4.8. The non-designable boundaries are the fixed part of the hook design. One set of the constant boundaries is used to constrain the structure in the x and y directions and to apply the loads. One of the constant boundaries is also used to represent an important safety feature of the hook design – emphasising its possible engineering application. This boundary is used to preserve the lip of the hook (extending from the uniform loading region to the tip of the hook, Fig. 4.8), which plays an important role as a safety feature in any hook design as it prevents a slip-off of the attached load during practical operation.

An additional design parameter shown in this figure is the distance d, since it is an important parameter of a hook design. This parameter provides the co-ordinates of

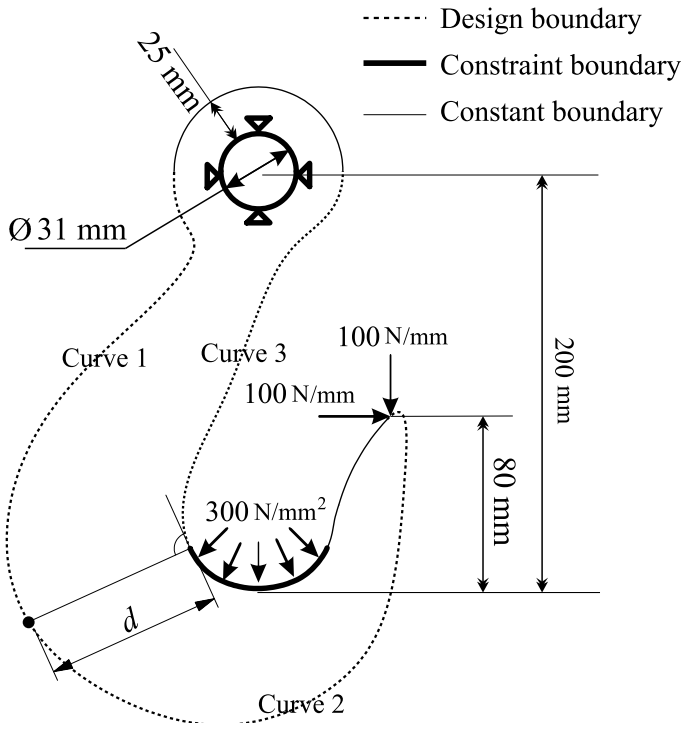


FIGURE 4.8: Structural design problem – a crane hook. The boundary conditions are given as a displacement constraint and two forces applied to highlighted regions. The two forces are given as surface pressure $F_1 = 300$ N/mm^2 and a point load $F_2 = 100$ N/mm

TABLE 4.2: Material and geometric properties of the structural design problem – a crane hook

Hook thickness (t)	=	20 mm
Primary pressure load (F_1)	=	300 N×mm^{-2}
Secondary line load (F_2)	=	100 N×mm^{-1}
Young's modulus (E)	=	200 GPa
Poisson's ratio (ν)	=	0.3

the connection point between *Curve* 1 and *Curve* 2 calculated as the perpendicular distance to the end vector of the constraint boundary curve, see Fig. 4.8.

Stress calculations are carried out using FEA. During FEA runs, the hook geometry is regarded as a plane stress problem having thickness ($t = 20$ mm). Forces are applied to the hook structure as shown. They are designated as *primary* and *secondary* loading. The primary load is applied as a surface pressure and it represents the load that needs to be carried by the hook. The secondary load is given as point load (having two components) applied to the hook tip as shown. The secondary load is introduced in order to artificially preserve some material at the tip of the hook. If this is not kept as a part of the loading, the optimisation process will eliminate the tip of the hook from the design altogether because it is nearly stress-free. This is unrealistic because hook tips are required for functions related to handling safety and this is achieved by a fixed arc with a small load at the end. The magnitude of the secondary loading is chosen small enough so as not to affect the final optimal design of the hook for the prescribed primary static load very significantly. The material and geometric properties are given in Table 4.2.

The objective of the design optimisation is formalised as: for the given loading condition, find the continuous 3D geometry which minimises the maximum von Mises stress (σ^{vm}) within the structure and the overall planar hook geometry area (A) simultaneously. The area is taken as an objective instead of the total weight of the crane hook, since the thickness in the depth direction is constant.

This is a bi-objective optimisation problem, however a GE that handles only single objective optimisation tasks (GA) is used. To get around this problem the, bi-objective problem is turned into one using a weighted sum of the fitness and thus working with a single objective that linearly combines the original two objectives:

$$\chi = w \times \sigma^{\text{vm}} + (1 - w)A, \tag{4.11}$$

where w is the weighting. Nine values of the weight (0.1, 0.2, $\cdots$, 0.9) were used in this search. Different weighting for the two objectives were chosen to explore its effect on the optimal design and to explore the Pareto-front. To calculate the weighted sum, the two design performance variables are normalised so that they are in a similar range.

In this case study, the crane hook model is simple enough that the FEA simulation run time is small. If the run time were an issue, one could consider implementing the NSGAII (Deb et al., 2000) in GE instead of GA. This algorithm is known to handle multi-objective optimisation tasks without additional runs for each weighted sum w. A numerical multi-objective optimisation using GE where the GA is replaced by NSGAII is presented in (Nasuf et al., 2011).

Considering the defined crane hook problem, the BNF syntax presented in Table 4.1 is used for synthesis of three piecewise planar curve representing the hook design boundaries. The initial sentence used in combination with this BNF syntax is

$$\begin{aligned} U_j &= p, \langle \text{tri} \rangle, \langle \text{tri} \rangle, \langle \text{tri} \rangle \\ d &= \langle \text{int3} \rangle \langle \text{int1} \rangle .0, \end{aligned} \tag{4.12}$$

where $j = 1, 2, 3$ and p is the initial construction point of each curve. The part of the initial sentence $d = \langle \text{int3} \rangle \langle \text{int1} \rangle .0$ is used to synthesise a value for the distance variable in the interval $30.0 \leq d \leq 99.0$ mm.

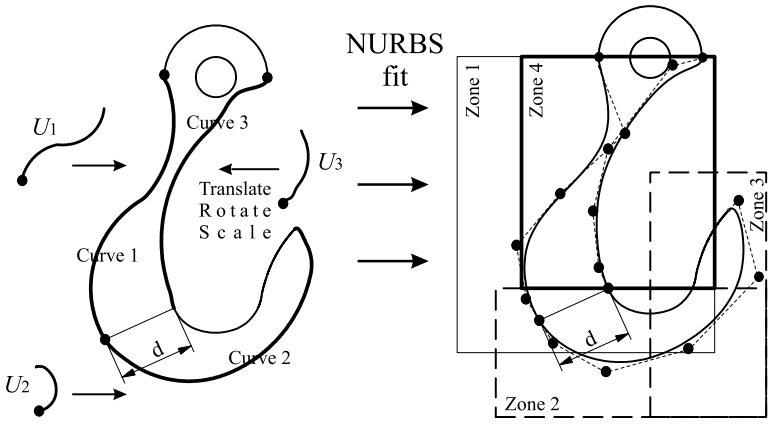
Taking into account the defined SG rules listed in Table 4.1 and the initial SG sentence, the minimum number of arcs in each piecewise curve is constrained to three – in case only the terminating SG rules are selected. Using these SG rules, three separate piecewise curves are synthesised describing the design boundaries of a planar crane hook shape. Note that the initially defined sentence introduces parallelism to the SG sentence (three non-terminals in parallel). Therefore, a sequential order for applying SG for each marker is adopted.

The three piecewise curves of primitive arcs of variable size derived in this way are then fitted in between the constant boundaries of the hook geometry. The *Match* operation pointed out at Stage 3 is used to maintain an overall C_1 continuity of hook geometry (transition regions of constant boundaries to design boundaries). The overall C_1 continuous hook geometry constructed as described is then included in an optimisation loop, where during iterations, a significant range of hook deigns (design space) is explored and evaluated. Following this procedure the hook design

is optimised from an initially unoptimised structure by synthesising planar curves using automated SG with arc primitives.

4.5 Design space exploration using SG and NURBS for shape generation and manipulation

In Chapter 2, the Non-Uniform Rational B-Spline (NURBS) was introduced as a parametric curve frequently used in engineering design. Each NURBS provides a finite set of control points and their co-ordinates can be used as design variables during optimisation. The effectiveness of using automated SG with arc primitive shapes for design exploration is compared here with conventional parameterisation technique based on NURBS in the context of structural optimisation. A GA search strategy for manipulating the NURBS control points is adopted to keep the comparison fair, so that both search strategies rely on GA guidance. Unlike the automated SG, the NURBS parameterisation requires an existing initial design. The first design synthesised by the SG, for each $w = 0.1, 0.2, \cdots, 0.9$, is taken as the initial design in the NURBS shape parameterisation. An example of initial



(a) Initial design using SG with arc primitives

(b) NURBS parameterisation with boundary limits

FIGURE 4.9: An example of a valid planar crane hook design derived from an initial random selection of SG rules and its NURBS representation

design synthesised by the SG is shown in Fig. 4.9a. It is constructed from an initial (random) application of SG rules. The three piecewise curves (U_1, U_2 and U_3), after their transformation, are rebuilt and parameterised by three cubic NURBS with an arbitrary number of 6 control points each, see Figure 4.9(b). This is achieved using the closest NURBS fit of 6 control points – a function in the commercial

TABLE 4.3: Movement boundary limits of the NURBS control points

Zone	Corner 1	Corner 2
1	$(50, 0)$	$(-150, -230)$
2	$(90, -180)$	$(-120, -280)$
3	$(90, -90)$	$(0, -280)$
4	$(50, 0)$	$(-100, -180)$

package RHINOCEROS3D. The co-ordinates of the control points for each NURBS excluding the first and the last point are then defined as design variables in the GA optimiser provided by OPTIONSMATLAB.

The total number of design variables in the NURBS parameterisation is

$$n = \sum_{i=1}^{3} 2(n_i - 2) + 1 = 25 \tag{4.13}$$

where $n_i = 6$ is the number of control points for each NURBS. Here, subtraction by 2 is necessary to exclude the first and last control points that are fixed and multiplication by 2 is necessary since each control point has two degrees of freedom. The last (25th) design variable is the distance 30.0 mm $\leq$ d $\leq$ 99.0 mm. It provides the connection point between *Curve 1* and *Curve 2*, see Fig 4.9(a).

The limits of the boundary movement of the NURBS control points are given as four rectangular zones shown in Fig. 4.9(b). These Zones are identified arbitrarily based on engineering judgement as corner points of a rectangular area listed in Table 4.3.

Zones 1 and 4 represent the boundary limits for the four control points of curves 1 and 3 respectively. Zone 2 represents the boundary limits for the first two movable control points of curve 2, see Fig 4.9(b), while zone 3 represents the boundary limits for the last two movable control points of curve 2.

Since the most expensive part of the crane hook structural optimisation is the FEA run time, the overall computational time in this comparison is limited to 10,000 successful FEA iterations for the compared techniques. A successful FEA iteration is possible only if the geometry of the CAD model is uniform and continuous i.e. without any line intersections of boundaries. In the case of any line intersections in the resulting geometry a penalty value is assigned to the member responsible for the unacceptable design. The GE and GA specific parameters used in this comparative study are listed in Table 4.4. A random seed to generate the initial binary string was used in both runs.

TABLE 4.4: optimisation specific parameters used in GE and GA

Parameter	GE	GA
Budget	10,000 FEA runs	10,000 FEA runs
Population size	100	100
Crossover probability(p_c)	0.5	0.8
Crossover type	multiple	single
Crossover points	3	1
Mutation type	multiple	single
Mutation probability (p_m)	0.005	0.005
Codon length (β)	4 bit	12 bit
Genotype length (μ)	400 bit	–
Codons (λ)	50	–

4.6 Results and discussions

Given the fixed computational budget, the design search was carried out in conjunction with the two different shape description strategies – using SG and using NURBS, respectively. The results are discussed below.

4.6.1 Results

The averaged results of 10 repetitive runs from design optimisation are presented in Fig. 4.10 in the form of two superimposed Pareto-fronts. The star shaped symbols represent the set of designs discovered using the NURBS parameterisation, while the circles represent the set obtained using automated SG with arc primitives. These points indicate the trade-off between the two objectives σ^{vm} and A.

They are averaged from 10 repetitive runs of the optimiser with weights being $w = 0.1, 0.2, \cdots, 0.9$, respectively. A design optimisation run with $w = 0.1$ tolerates the reduction of hook design area at the expense of increased maximum von Mises stress, while $w = 0.9$ favours stress reduction at the expense of increased area, for example.

The Pareto-fronts thus obtained by the two search methods are very close to each other. The main difference can be noticed at the Pareto-front ends – around hook designs obtained by the runs $w = 0.1$ and $w = 0.9$. Taking this difference into account, the NURBS parameterisation gives designs with the lowest von Mises stress at the expense of slightly increased area. On the other hand, the proposed method for shape synthesis using SG with arcs discovers hook designs with the

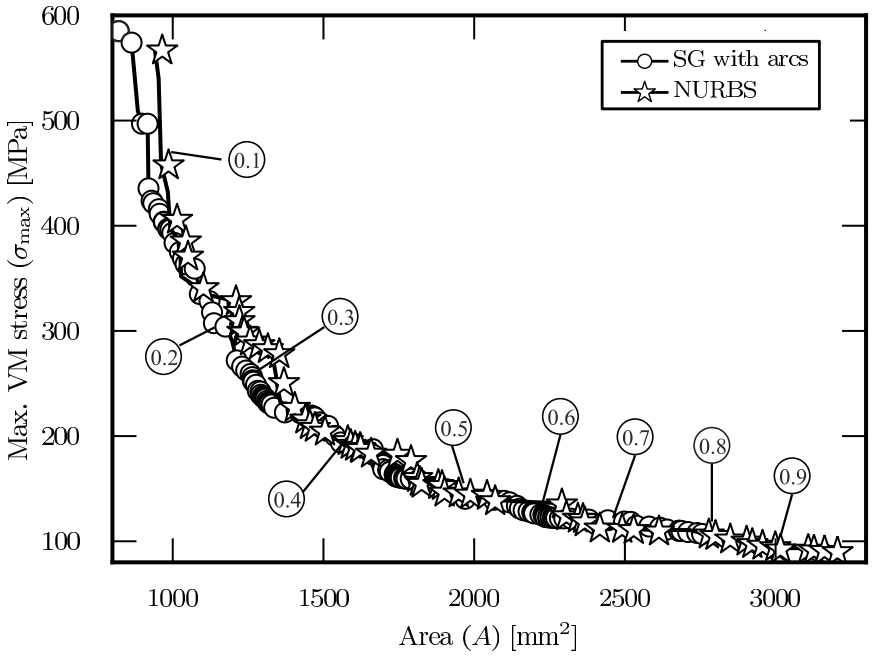
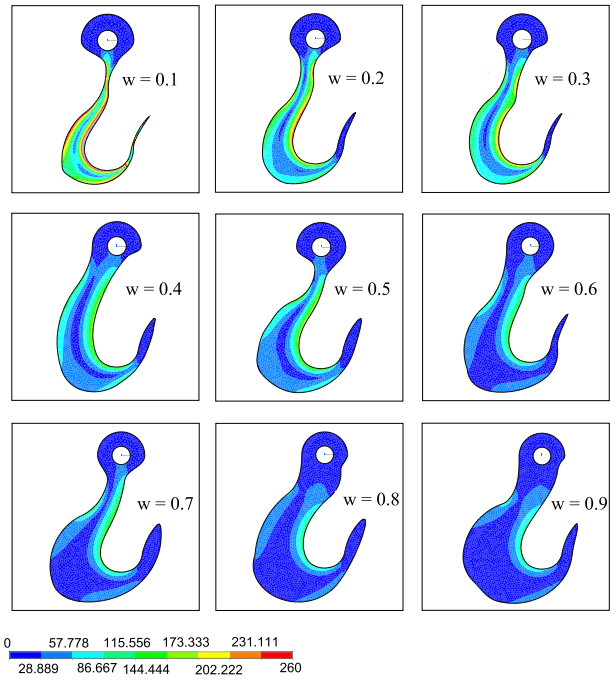
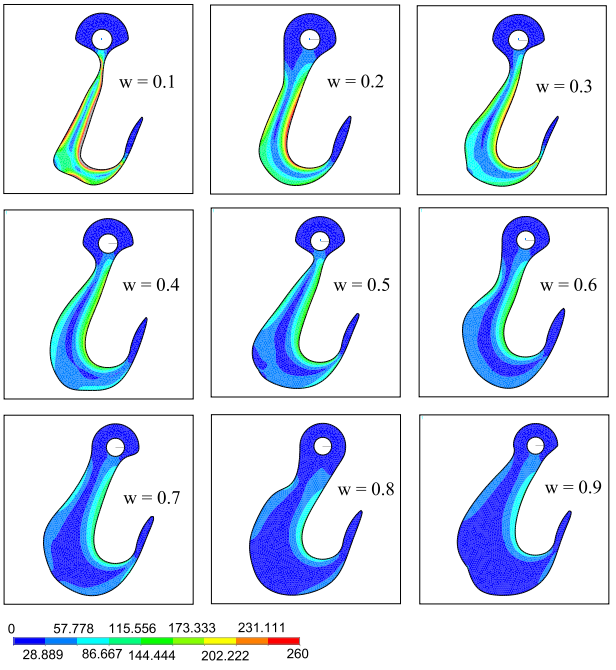


FIGURE 4.10: Superimposed Pareto-fronts obtained during design optimisation of the crane hook problem with SG using arcs and NURBS representation. The Pareto-fronts are obtained by calculating the weighted sum of fitnesses for objectives σ^{vm} and A with different weights w

smallest area. Also, there are a small discrepancies between values for run $w = 0.3$. The similarity in the two sets of designs on the bi-objective plane does not necessarily mean that the two shape parameterisation perform equally efficiently. It may be that the design search convergence speed was different. This aspect will be further explored.



(a) Hook designs obtained by SG with arcs using GE



(b) Hook designs obtained by NURBS representation and GA search strategy

FIGURE 4.11: A range of optimal planar crane hook designs for each w picked from both Pareto-fronts. Hook designs with small area have higher stress compared to those with low stress and larger area. The hook designs clearly show the trade-off between the two objectives

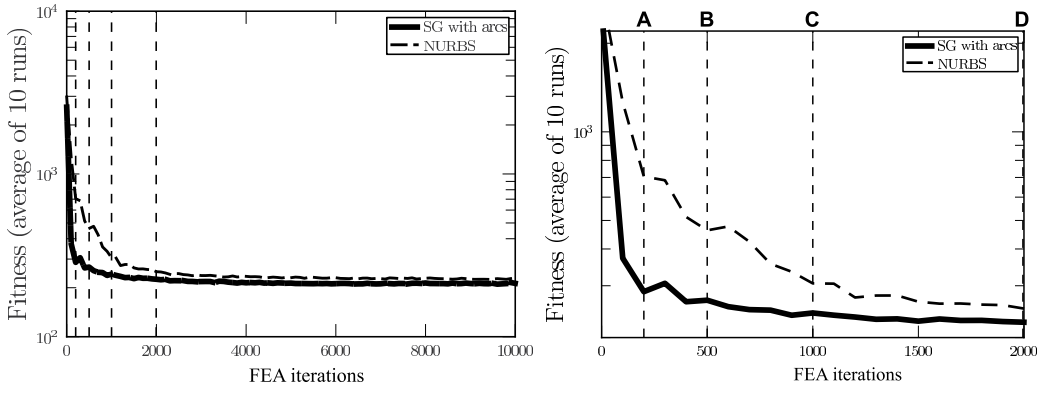
Designs obtained for each weight w using automated SG are shown in Fig. 4.11(a). Similar hook designs, for each w, found using NURBS parameterisation are shown

TABLE 4.5: Averaged optimum results from 10 runs

	SG with arcs		NURBS	
w	σ^{vm} MPa	$A \times 10^2$ [mm^2]	σ^{vm} MPa	$A \times 10^2$ mm^2
0.1	409.58	103.02	446.49	100.43
0.2	263.67	132.49	279.18	132.97
0.3	217.42	148.74	238.25	138.21
0.4	173.15	179.68	181.81	164.18
0.5	153.03	194.34	155.99	184.21
0.6	134.37	224.71	142.4	224.26
0.7	121.71	237.38	115.01	242.05
0.8	117.99	259.37	111.14	273.0
0.9	110.07	279.11	94.45	310.04

in Fig. 4.11(b). A visual comparison of these designs reveals some similarities such as the increased amount of material around stress concentration regions. Numerical comparison between these hook designs in terms of values for the two objectives are listed in Table 4.5.

The efficacy of the two shape description methods in an optimisation context is compared against the convergence speed in Fig. 4.12. Since the GA is a stochastic search algorithm, the results in this figure are averaged for 10 runs, where $w = 0.6$. It can be seen that SG with arc primitives converges much faster in the beginning. A hook design, close to the globally optimal design is discovered using SG parameterisation in the first 200 FEA evaluations. In comparison, the NURBS based shape optimisation requires an additional 1,000 iterations to reach this level. The convergence remains almost unchanged for both methods after 2,000 evaluations.

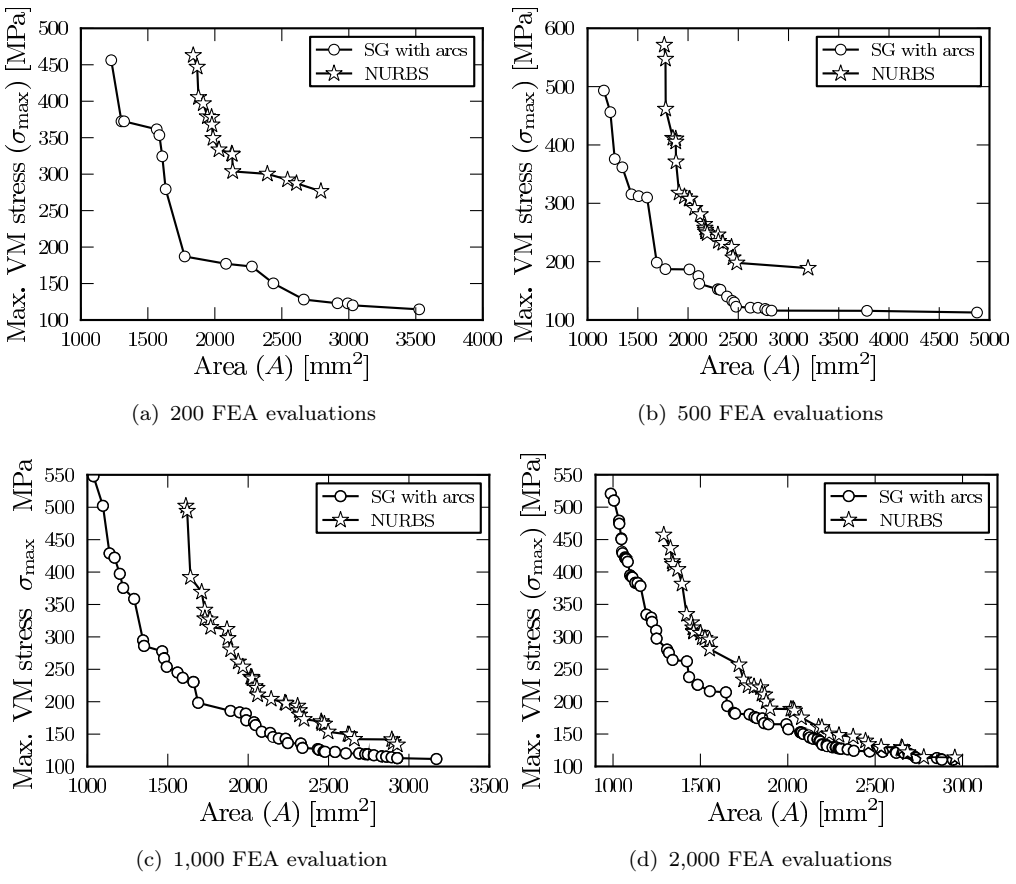


(a) All 10,000 runs (b) First 2,000 runs.

FIGURE 4.12: The averaged convergence history of 10 runs with SG and NURBS method for the bi-objective fitness calculation weight $w = 0.6$

The differences in convergence speed are masked in Fig. 4.12(a). This can be highlighted by zooming in and presenting the convergence at the initial phase – up to 2,000 FEA evaluations. The differences are remarkable given the logarithmic scale of fitness, see Fig. 4.12(b).

Looking back at the Pareto-front presented previously in Fig. 4.10 the closeness of the fronts indicates that both shape parameterisation schemes (SG and NURBS) were given adequate opportunity to converge by providing perhaps excessive computation resources. For complex engineering problems, this may not be possible. In such a scenario, a superior shape description strategy is one that leads to faster convergence. However, the fact that both schemes led to very close Pareto-fronts highlights that the two shape description methods have very similar *reach* within the search space. If this were not the case, then the converged front for the intrinsically constrained method of geometry description would have led to a poorer



(a) 200 FEA evaluations

(b) 500 FEA evaluations

(c) 1,000 FEA evaluation

(d) 2,000 FEA evaluations

FIGURE 4.13: Pareto-front evolution comparison for the hook design search problem solved using SG with arcs using GE and NURBS representation with GA search strategy. The four figures show the evolution of the Pareto-front after 200, 500, 1000 and 2000 FEA iterations during the design search with both methods

result even after being provided with excessive computation resources.

The evolution of the Pareto-front with increasing number of FE evaluations is presented in Fig. 4.13 (a) through to (d) using the two methods of shape parameterisations. Since the most significant part of the total computational budget is the effort associated with FE runs, each of the four stages of the Pareto-front evolution can be regarded as those corresponding to fixed computational resource allocated at each step. Clearly, the fronts obtained using SG shape description exhibit superiority in terms of convergence throughout as they dominate the fronts obtained from NURBS-based shape description almost completely.

4.6.2 Discussion

Several limitations and issues with the automated SG with arc primitive shapes proposed in this chapter were noticed during the numerical experiments. First of all, effective SG rule selection appears to be dependent on the formulation of the BNF syntax and especially the initial sentence in this BNF syntax. For this reason, sometimes, the automated SG may fail to increase the number of arcs in a synthesised piecewise curve due to biased BNF syntax, insufficient formulation of penalty function or initial sentence. This is because the penalty function used in the crane hook example does not tolerate increases in the arc primitives in the piecewise curves, since this is likely to result in high curvature curves intersecting with each other.

The second issue is related to the genotype length (μ). It is not very clear yet what the optimal genotype length should be and whether this is generic or problem sensitive. For example, a short binary string could constrain the search space, while a long one could increase the convergence time.

The evolutionary search usually requires a significant number of generations before poorly performing members of the population evolve to produce better results. From a structural design optimisation perspective, this is probably the biggest drawback of the evolutionary design synthesis and optimisation framework presented here. Each binary string represents a particular design in a structural optimisation study. The performance of each new design in terms of mechanical response is obtained as a separate FEA run. Simulations of complex 3D structures often require many hours of computation for fitness calculation of a particular genotype to generate a feedback. This may increase the run time of a design optimisation process. Nevertheless in such a practical scenario, the methodology may

perform better than NURBS based parameterisation using a GA search strategy because computation resources are limited.

4.7 Conclusions

A general methodology for synthesising complex piecewise planar curves by manipulating a finite number of arc primitives with the design description and optimisation framework proposed in the previous chapter was presented. A specific BNF syntax containing four SG rules with arc primitives of variable size was developed. The proposed rules were then used in a bi-objective design optimisation study of a planar crane hook shape. The concept of synthesising the design boundaries of a planar crane hook as three separate piecewise curves using GE was demonstrated. The efficacy of the proposed shape synthesis and optimisation methodology based on SG rules with arcs was compared to that of NURBS control points manipulation strategy based on a GA.

The development of an automated SG was initiated by the need to investigate superior shape exploration techniques in the structural design optimisation context. The approach developed here to automate the SG is driven by evolutionary intelligence and, therefore, can be used as an intelligent design search tool in a variety of design problems. In the shape optimisation of the crane hook studied here the design performance measure was readily obtainable in terms of the area and the maximum von Mises stress. Some of the key conclusions drawn during this comparison are:

- The SG could be successfully used for the synthesis of sophisticated free-form shapes using explicitly and/or implicitly defined primitive shapes.
- When the computational resources are limited, the proposed SG with arc primitive shapes of variable size finds a better solution compared to conventional NURBS based control point manipulation for automated shape explorations. This is because it leads to faster convergence during the search. Given excessive computational resources, the two methods converge to very similar design solutions. This indicates that the reach of the two shape description methods within the search space is very similar.
- A drawback of design search with an automated SG using GE is that sometimes the search process converges prematurely to a local optima. Nevertheless this is an artefact primarily contributed by the GA process.

- The control point design search strategy based on NURBS is particularly useful if controlled manually by an experienced designer; it provides an intuitive control. On the other hand, shape description and its use within a design search scenario requires less human intervention which is an attraction when a design needs to be driven in an automated way but is a drawback if a designer's intuitive experience needs to be incorporated in the process.

Chapter 5

Optimisation of planar trusses using SG syntax

In the previous chapter, shape synthesis of 2D geometry using an automated Shape Grammar (SG) was presented. Following this, a case study of a crane hook design was undertaken. The optimisation was carried out by applying specific SG rules with arc primitives represented in a BNF syntax. The syntax was categorised as a generic shape synthesis tool which is applicable in a range of multidisciplinary shape optimisation problems. In this chapter, another class of structural design problems frequently encountered in engineering practice is considered. The proposed methodology in Chapter 3 is developed here for optimisation of planar trusses. The chapter covers topology, sizing, and configuration aspects of truss design. Various innovative BNF syntaxes are developed to represent a set of grammar rules designed for truss optimisation. The results of the truss optimisation problems considered here are compared with results available in the literature. A comparison of convergence speed is also presented.

5.1 The truss optimisation problem

A truss is a network of structural members connected at a set of points. Because of the topology of trusses, elastic deformation in the members is stretch dominated. This simplifies the stress analysis. The total weight of a truss structure is a sum of the product of cross-section area a_i, length l_i, and material density ρ_i over the

entire structure

$$f_{\min} = \sum_{i=1}^{m} a_i l_i \rho_i, \tag{5.1}$$

where i refers to a member and m is the total number of members in the truss. The optimisation variables here are a_i, l_i, ρ_i, and m. It is assumed here that the material of all the truss members is the same. For all the numerical implementations presented in this chapter ρ_i is chosen as constant 2.768×10^{-6} kg $\times$ mm^{-3}. Taking this into account, the above optimisation problem can be divided into 3 stages:

1. *Topology optimisation.* Consider a truss having m members of which some could be removed safely so as to reduce the total weight of the truss. Topology optimisation is aimed at removing unnecessary members in an optimal way. Numerically, this is carried out by assigning an accept/reject type binary value to each member. An important requirement at this stage of optimisation is to eliminate kinematically mobile topologies from consideration. Therefore, during optimisation, kinematically unstable topologies are penalised.

2. *Sizing optimisation.* The cross-sectional area of truss members often needs optimising. This stage of optimisation is employed to reduce weight by finding the optimal cross-section for each member (a_i) such that stress or deflection constraints are met. Here, the optimisation variables are taken as the values of the cross-section of all members. A simultaneous sizing and topology optimisation is achieved by defining an additional optimisation constraint to sizing optimisation, expressed as a minimum threshold value of cross-section area $a_{\min}$. Using this constraint, members with area value less than the minimum threshold $a_i \leq a_{min}$ are removed.

3. *Configuration optimisation.* The third stage of optimisation is used to reduce the weight by varying the length of truss members l_i. This is possible when the co-ordinates of the joints of the truss structure are treated as free optimisation variables.

5.2 Topology description and truss optimisation using GE

In the previous chapter, the use of SG rules expressed as a BNF syntax to explore the description of shapes and optimisation of the resulting structure was presented. The problem of truss optimisation is significantly different so that a new grammar to be developed. The effectiveness of the combination of GE for this new class of structural problems is investigated in this chapter.

Topology optimisation of truss structures with GE. A new BNF syntax, which could be applied to describe the topology of a truss and can subsequently be used for the topology optimisation problem is presented next.

Suppose that a truss structure has n joints. The total number of members in a n node truss is $m = \binom{n}{2}$, i.e. when each joint is connected to all the others. If these joints are represented numerically as a set $\{1, 2, \cdots, n\}$, the representation of the truss members could be given by the set

$$\mathbf{v} = \{(1,2),(1,3),\cdots,(1,n);(2,3),(2,4),\cdots,(2,n);\cdots;(n-1,n)\}. \tag{5.2}$$

Taking this into account the definition of a BNF to solve the connectivity problem of an n node truss requires only one component:

(i) An expression for accepting or rejecting a connection (truss members) between two nodes e.g. ⟨exist⟩::= accept | reject

Let $\mathrm{V_T} = \{1, 2, \cdots, n, \text{accept}, \text{reject}\}$ and $\mathrm{V_N} = \{\langle\text{exist}\rangle\}$. The grammar rules in Table 5.1 expressed in BNF syntax can be used for topology optimisation of such truss structures, where (S) is the initial sentence.

In this BNF syntax, the presence of the non-terminal ⟨exist⟩ in the initial sentence triggers the rule selection routine. The BNF syntax represents two unique grammar rules (⟨exist⟩ → accept) and (⟨exist⟩ → reject). When the first grammar rule

TABLE 5.1: A BNF used to solve truss optimisation problems

⟨exist⟩ ::= accept | reject	
S	={$(1,2)\langle\text{exist}\rangle, (1,3,)\langle\text{exist}\rangle, \cdots, (1,n,)\langle\text{exist}\rangle$; $(2,3)\langle\text{exist}\rangle, (2,4)\langle\text{exist}\rangle, \cdots, (2,n)\langle\text{exist}\rangle$; $\cdots; (n-1,n)\langle\text{exist}\rangle$}

is selected, the truss member is accepted within the structure. When the second rule is selected the truss member is rejected from the structure.

For example, if $n = 4$ the initial sentence would be

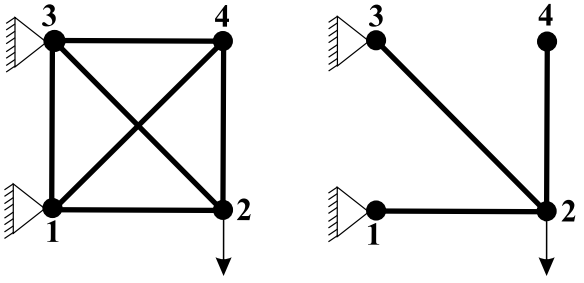
$$\mathrm{S} = \{(1,2)\langle\text{exist}\rangle, (1,3)\langle\text{exist}\rangle, (1,4)\langle\text{exist}\rangle, (2,3)\langle\text{exist}\rangle, (2,4)\langle\text{exist}\rangle, (3,4)\langle\text{exist}\rangle\} \tag{5.3}$$

The non-terminals $\langle\text{exist}\rangle$ are used to select a rule for each truss member. The number of members in a 4 node truss is $m = \binom{4}{2} = 6$, therefore a total of 6 rules are to be selected (one for each non-terminal $\langle\text{exist}\rangle$).

Considering the proposed BNF syntax in Table 5.1 the final sentence could be of the form

$$\mathrm{S} = \{(1,2)\text{accept}, (1,3)\text{reject}, (1,4)\text{accept}, (2,3)\text{accept}, (2,4)\text{accept}, (3,4)\text{reject}\}. \tag{5.4}$$

The resulting truss structure is shown in Fig. 5.1(b), where members given by nodes (1,3) and (3,4) are removed. The minimum codon length for this problem is $\beta = 1$ bit, since the maximum number of rules grouped in a BNF expression is $\xi = 2$. This is because 1 bit provides two Boolean positions i.e. 0, 1 (accept, reject). If the genotype wrapping is disabled during the mapping process, a sufficient binary string length to select 6 grammar rules would be $\mu = 6 \times 1 = 6$ bits or $\lambda = 6$ codons, where each codon contains 1 bit. Obviously, the proposed method, presented in this way, is overcomplicated compared to the standard GA, where a binary string of 6 bits is also sufficient. From a GA perspective, one can associate the topology optimisation problem presented above with Goldbeg's one armed



(a) Knematically stable truss (4-node, 6-member)

(b) Kinematically unstable truss (4-node, 3-member)

FIGURE 5.1: An illustration of truss problems. The boundary conditions are given as displacement constraints at nodes 1 and 2. A force load is applied at node 2

bandit optimisation example (Goldberg, 1989), where the variables take only the values of 0, and 1. Nevertheless, the proposed BNF syntax is a good example of topology optimisation of truss structures with GE, because it demonstrates an alternative mapping technique of optimisation variables and, provides faster convergence in cases considered here.

Topology and sizing optimisation of truss structures with GE. Consider that a truss structure has n nodes, however the optimisation parameters are given as the connectivity between nodes as well as the cross-section area of each truss member.

In order to have grammar rules that are simple yet general for size optimisation the following two BNF components are needed:

(i) An expression for accepting or rejecting a connection (truss members) between two nodes e.g. $\langle \text{exist} \rangle ::= \text{accept} \mid \text{reject}$

(ii) If truss member is accepted an expression for synthesising a value for its cross-section area. Here, an expression similar to that for synthesis of normalised optimisation variables presented in Chapter 3 can be used e.g.

$$\langle \bar{a}_i \rangle ::= 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle, \tag{5.5}$$

where $\langle \bar{a}_i \rangle$ is normalised value for i-th member cross-section with η digits after the decimal point.

The required components lead to definition of BNF syntax in Table 5.2. The vocabulary sets are extended to: $\mathrm{V_T} = \{1, 2, \cdots, n, \text{accept}, \text{reject}, 0, .\}$ and $\mathrm{V_N} = \{\langle \text{exist} \rangle, \langle \text{int} \rangle, \langle \bar{a}_i \rangle\}$, where $i = 1, 2, \cdots, m$. Similar to the BNF syntax form Table

TABLE 5.2: A proposed BNF syntax efficient for simultaneous sizing and topology optimisation – BNF3

$\langle \texttt{int} \rangle$	::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
$\langle \bar{a}_i \rangle$	::= 0.$\langle \texttt{int} \rangle \langle \texttt{int} \rangle \cdots \langle \texttt{int} \rangle$ | reject
$\bar{a}_i$	$=\langle \bar{a}_i \rangle$

5.1, the two grammar rules

$$\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \tag{5.6}$$

and

$$\langle \bar{a}_i \rangle \rightarrow \text{reject} \tag{5.7}$$

are used to define the presence or absence of the i-th truss member. This BNF syntax is adequate for simultaneous size and topology optimisation.

The additional information required is that if the rule

$$\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \tag{5.8}$$

is selected (i.e. the truss member is present), a normalised value for its cross-section area $0 \leq \bar{a}_i < 1$ is synthesised. This value is normalised for the interval $a_\Delta = a_{\max} - a_{\min}$, where $a_{\max}$ and $a_{\min}$ are the allowable upper and lower values of real cross-section a_i. Similar to numerical optimisation of mathematical functions presented in Chapter 3, $a_i = a_{\min} + \bar{a}_i \times a_\Delta$.

The number of non-terminals (digits), $\langle \text{int} \rangle$ after the decimal point in the rule $(\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle)$ is of variable size and is defined arbitrarily. It represents the desired search step η of the normalised variables $\bar{a}_i$. The search step κ of real variables a_i is calculated as follows

$$\kappa = \eta \times a_\Delta. \tag{5.9}$$

For example, a double digit precision $\eta = 0.01$ is achieved using two non-terminals $\langle \text{int} \rangle$ after the decimal point$(\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle)$, therefore $\kappa = 0.01 \times a_\Delta$. The search step κ can be adjusted by appending or removing non-terminals $\langle \text{int} \rangle$ after the decimal point. For example, if $\bar{a}_i = 0.\langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle = 0.001$, then $\kappa = 0.005$ and $a_i = a_{\min} + \kappa$.

The price to pay for increasing the number of non-terminals $\langle \text{int} \rangle$ in the rule $(\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle)$ and thus refining the search resolution, is a longer binary string length μ. This is because each non-terminal $\langle \text{int} \rangle$ requires an additional codon to select a grammar rule (a digit in this case). Recall that a binary string of length μ provides a set of λ codons, where each codon is of length β bits. The decoded decimal codon values represent variables in λ-dimensional search space for the interval $[0, (2^\beta - 1)]$.

Suppose that a binary string (1011|0101|1010) of length $\mu = 12$ is provided by the GA. If $\beta = 4$ bits, this string is decoded to a decimal set of codons (11, 5, 10), where each value is a variable in a three dimensional search space in the interval $[0, 15]$. The grammar rules selected by the set of codons represent one of the solutions in the solution space. This means that refining the search resolution comes at the expense of increased dimensionality of the search space, i.e. each additional codon required to select a grammar rule will increase the dimensionality

of the search space by 1. Therefore, calculation of optimal binary string length μ is crucial for achieving good convergence speed of optimisation.

An efficient calculation of μ for simultaneous topology and size optimisation of truss problems using GE and BNF syntax listed in Table 5.2 is not straightforward due to the expression ($\langle \bar{a}_i \rangle ::= 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \mid \text{reject}$) and needs further explanation.

Suppose that a η digit precision after the decimal point is desired to synthesise a variable ($\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle$). If the wrapping operation is disabled, the required binary string is of length

$$\mu = \beta(k \times \eta + m), \tag{5.10}$$

where k is the number of members identified as active, m is the total number of members in the initial structure and β bits is the codon length.

Consider the truss structure shown in Fig. 5.2. This truss is similar to the one shown in Fig. 5.1(a) with one difference – the cross-section area of each member of the truss is a variable denoted by a_i. The total number on members in Fig. 5.2 is $m = 6$, if all members on this truss problem are active $k = 6$, $\eta = 2$ because double digit precision after the decimal point is required ($\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle$) and $\beta = 4$ then $\mu = 4(6 \times 2 + 6) = 72$ bits or $\lambda = 18$ codons, i.e. the search space is 18-dimensional.

The value of $\beta = 4$ is calculated by taking into account the maximum number of rules $\xi = 10$ grouped in any of the expressions in Table 5.2. In Chapter 3, it was shown that a codon of β bits provides selection of up-to 2^β grammar rules grouped in one BNF expression. The above calculation of μ assumes that $k = m = 6$, which means that the rule ($\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle$) is selected 6 times and 6 codons are used for the purpose. Additional $k \times \eta = 6 \times 2 = 12$ codons are required to select a rule for each non-terminal $\langle \text{int} \rangle$ as to synthesise variable $\bar{a}_i = 0.\langle \text{int} \rangle \langle \text{int} \rangle$ for each

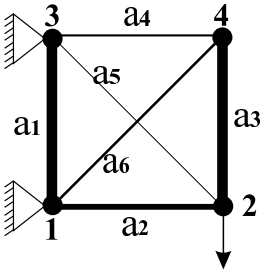


FIGURE 5.2: An illustration of truss problem with variable cross-section of truss members

truss member identified as active. The complication comes from the fact that the optimal topology of a structure might have members identified as removed, i.e. the rule ($\langle \bar{a}_i \rangle \rightarrow$ reject) is selected for some of the initial members. In this case $k < m$.

Consider the truss shown in Fig 5.1(b), where only three members ($k = 3$) out of six ($m = 6$) are identified as active by selecting the rule ($\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle$). The other three members are rejected by selecting the rule ($\langle \bar{a}_i \rangle \rightarrow$ reject). Here the most efficient binary string length would be $\mu = 4(3 \times 2 + 6) = 48$ or $\lambda = 12$ (12 dimensional search space). This is because only 3 normalised variables are required, which needs 3×2 codons. Unfortunately k (the optimal number of active members) is of optimisation interest and is previously unknown, therefore it is safe to assume μ as if all truss members are active i.e. $k = m$. This assumption ensures that the binary string contains enough information to synthesise a variable $\bar{a}_i$ for all members in the initial structure. If the optimal topology has $k < m$ active members there will be some unused binary information in the search space represented by unused codons – part of the binary string. This means that the search space will be λ-dimensional, but some of the dimensions will be dummy. This is not necessarily detrimental to GA based search.

An alternative to the BNF syntax from Table 5.2 for simultaneous topology and sizing optimisation with GE is presented in Table 5.3. The only difference between BNF in Table 5.2 and that in 5.3 is that the rule ($\langle \bar{a}_i \rangle \rightarrow$ reject) is replaced by the rule ($\langle \bar{a}_i \rangle \rightarrow -0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle$). This means that a normalised value for i-th member $\bar{a}_i$ is synthesised regardless of the fact that the member is identified as removed. Recall that a member can be also identified as removed if $a_i \leq a_{\min}$. Now, a synthesis of normalised area values for removed members is clearly unnecessary, but one is synthesised as a negative value. This appears to be beneficial because all codons are used in the rule selection process (even when $k < m$) i.e. all information provided by the binary string is used in sequential order. Evidence to support this statement is provided in Section 5.3.1, where a numerical comparison of the results obtained by using BNF listed in Tables 5.2 and 5.3 is presented. This is further discuses in Section 5.4.

TABLE 5.3: A proposed BNF syntax efficient in simultaneous sizing and topology optimisation – BNF4

$\langle \text{int} \rangle$	::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
$\langle \bar{a}_i \rangle$	::= 0.$\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle$ | -0.$\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle$
$\bar{a}_i$	=$\langle \bar{a}_i \rangle$

Shape optimisation of truss structures with GE. Having developed the BNF syntax rules for topology and sizing optimisation of trusses, the approach is explored for a new problem – that of shape description of trusses and its further use in shape optimisation of trusses.

All BNF components listed in sizing in topology optimisation with GE are needed. In addition normalised values for x and y position co-ordinates of truss nodes are synthesised by the BNF syntax listed in Table 5.4. This syntax is standard variable synthesis technique, which was developed in Chapter 3.

The variables x_i and y_i are the co-ordinates of the i-th joint in the truss, where $i = 1, 2, \cdots, n$. The initial sentence

$$\begin{aligned} \bar{x}_i &= 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \\ \bar{y}_i &= 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \end{aligned} \tag{5.11}$$

is used to synthesise a pair of x and y co-ordinates for each node. The normalised variables $\bar{x}_i$ and $\bar{y}_i$ are synthesised for the intervals $x_\Delta = x_{max} - x_{\min}$ and $y_\Delta = y_{\max} - y_{\min}$ respectively. Again the number of non-terminals $\langle \text{int} \rangle$ is determined arbitrary based on desired search resolution (number of digits after the decimal point). Note that the nodal co-ordinates can be fixed in any direction. This is required in practical circumstances to implement certain design requirements especially regarding support points in the structure. It is achieved by assigning to x or y co-ordinate a constant value. In BNF syntax this can be represented by the initial sentence

$$\begin{aligned} \bar{x} &= 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \\ y &= \text{constant}, \end{aligned} \tag{5.12}$$

where y is a specified constant value. Considering the BNF in Table 5.4, the movement of the node is fixed in y direction because it consists only of terminal primitives i.e. it is not a variable.

TABLE 5.4: A proposed BNF syntax efficient in shape optimisation

$\langle$`int`$\rangle$	`::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`
$(\bar{x}_i, \bar{y}_i)$	`=` $(0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle, 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle)$.

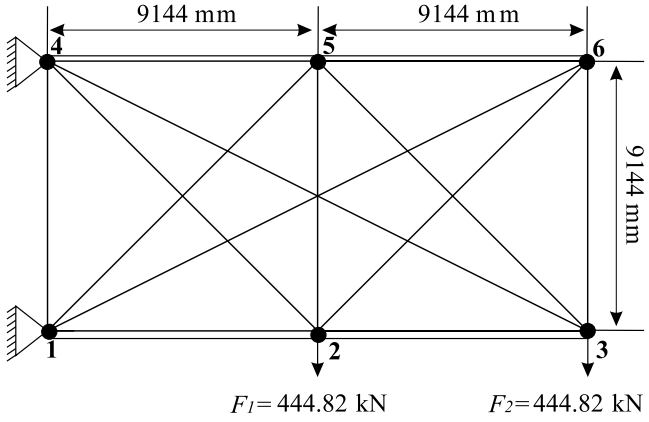


FIGURE 5.3: An illustration of a 6-node 15-member truss problem. The two parallel lines with a small gap in between illustrate all possible overlapping connections between collinear nodes (1, 2, 3) and (4, 5, 6). The truss is constrained at nodes 1 and 4. A force of $F = 444.82$kN is applied at nodes 2 and 3

5.3 Illustrative examples

In this section, several examples to demonstrate the effectiveness of the BNF syntax based approach to describe a truss structures in an optimisation scenario are presented. Results are also compared with those from other existing methods implemented on a range of problems previously available in the literature.

5.3.1 A 6-node, 15-member structure.

Consider the 2D truss problem shown in Fig. 5.3. This problem is well studied and is discussed in Hajela and Lee (1995); Deb and Gulati (2001); Kawamura et al. (2002). The structure consists of $m = 15$ truss members connected in a network of $n = 6$ nodes, when each node is connected to every other node. The overlapping members connecting the nodes (1-2, 2-3, 1-3) and (4-5,5-6, 4-6) are sketched using parallel lines with a small gap between. This structure is completely constrained at nodes 1 and 4. Point forces of magnitude $F = 444.82$ kN are applied at the two nodes in the y-direction as shown. The performance of the truss optimisation scheme based on GE proposed here will be compared with that of Deb and Gulati (2001), which is based on GA. For simplicity, the alternative rival method proposed will be called DEB.

The truss optimisation problem described here is posed as one with a single objective and two constraints. The objective is to minimise the total weight of the structure calculated using Eq. 5.1, which is the sum of the product of cross-section

TABLE 5.5: Problem specific truss optimisation parameters

Point mass load (F_i)	=	444.82 kN
Young's modulus (E)	=	68947.57 MPa
Density – Aluminium (ρ)	=	2.77×10^{-6}kg $\times$ mm^{-3}
Allowed displacement ($\delta_{\max}$)	=	50.8 mm
Allowed axial stress ($\sigma_{\max}$)	=	172.37 MPa
Minimum cross-section area ($a_{\min}$)	=	58 mm^2
Maximum cross-section area ($a_{\max}$)	=	22580.6 mm^2

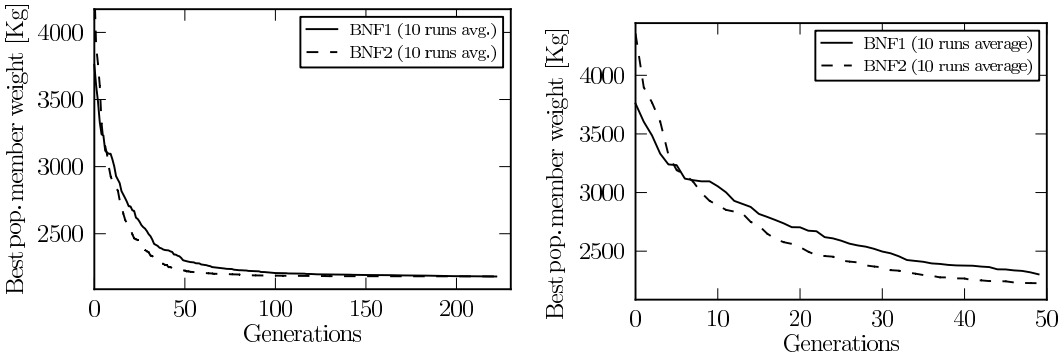
area a_i, length l_i, and material density ρ_i. The two constraints are the maximum allowed displacement $\delta_{\max}$ at any of the nodes and maximum allowed axial stress $\sigma_{\max}$. The value of σ_i could be positive or negative (compression or tension), therefore its absolute value is to be kept below $\sigma_{\max}$; the same is true for δ_i at the nodes. In the literature (Deb and Gulati, 2001), the parameters of this problem are given in imperial units. These parameters, when converted to metric units, are listed in Table 5.5.

The design optimisation of the truss problem is performed using GE with BNFs listed in Tables 5.2 and 5.3. The optimisation run with BNF in Table 5.2 will be referred as BNF3 while that in Table 5.3 as BNF4.

An average of 10 runs is taken. The number of non-terminals $\langle \text{int} \rangle$ after the decimal point is selected as four ($\langle \bar{a}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle$) for both BNFs. This provides search resolution of $\bar{a}_i = 0.0001$ or $\kappa = 0.0001 \times a_\Delta = 2.25$ mm^2, where $a_\Delta = 22,580.6 - 58 = 22,522.59$ mm^2. Wrapping is disabled, therefore the binary string length is $\mu = \beta(k \times 4 + m) = 300$ or $\lambda = \mu/\beta = 75$ codons, where $k = m = 15$ and $\beta = 4$ bits. The optimal topology of the truss problem is known to have fewer members than the initial truss (Deb and Gulati, 2001). When using BNF3 some of the codons will be dummy. All GE specific parameters used in BNF3 and BNF4 runs are listed in Table 5.6.

TABLE 5.6: GE specific parameters in truss optimisation

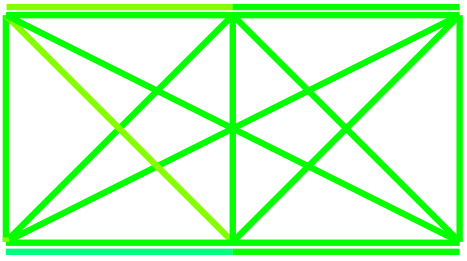
Generations	225
Population size	300
Crossover probability(p_c)	0.8
Crossover type	multiple
Crossover points	10
Mutation type	multiple
Mutation probability (p_m)	0.035
Codon length (β)	4



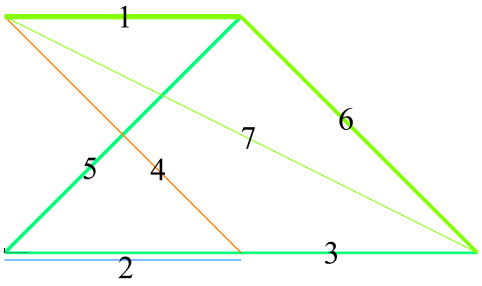
(a) Convergence history of 225 generations

(b) A zoomed view of first 50 generations

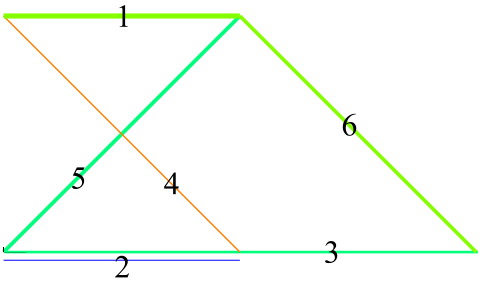
FIGURE 5.4: An averaged convergence comparison between truss optimisation with GE using BNF3 and BNF4. The zoomed view illustrates the convergence during the first 50 generations. On average the figure shows slightly better performance of BNF4 runs over BNF3



(a) Initial design (Weight = 12075 kg)



(b) Optimised design using BNF3 (Weight = 2147.99 kg)



(c) Optimised design using BNF4 & DEB (Weight = 2146.96 kg)

FIGURE 5.5: Stress contours for the initial and optimised truss design of 6-node 15-member truss problem. The different thickness of the lines represents the size of the cross-sectional area of each member. The overlapping truss members are illustrated as parallel lines with a small gap between

Fig. 5.4(a) presents the convergence history averaged over 10 runs by the best population member in a generation for BNF3 and BNF4 based search. The zoomed view for the first 50 generations (Fig. 5.4(b)) shows a quicker convergence of BNF4 over BNF3. In a reduced computational resources scenario, BNF4 would provide a much better solution. However, the first four generations show the opposite trend.

The optimal truss designs found by the best run of optimisation with BNF3 and BNF4 are shown in Fig. 5.5(b) and 5.5(c), respectively. The truss design shown in Fig. 5.5(c) is also reported by Deb and Gulati (2001) as optimal and is taken as the reference. The optimal topology found by BNF3 is slightly different than the reference design. This is because it has an additional truss member (7) shown in Fig 5.5(b) i.e. only 8 members are identified as removed and rejected. On the contrary, the optimal topology found by BNF4 is the same as the reference one, where 9 members are identified as removed. Note that the reference topology is obtained with equal number of generations, but with larger population size (450). This means that additional computation resources are required to reach the same solution.

A comparison of the cross-sectional area of the members from the best optimisation run is presented in Table 5.7 along with the total weight and the maximum displacement over the nodes within the structure. Unfortunately the averaged results for the method of Deb and Gulati (2001) are not available in the literature therefore the comparison is performed only on a single run. This comparison reveals significant size differences between BNF3 generated structure and the reference structure, while the size differences of design obtained by BNF4 with respect to the reference design are much smaller.

The weight of the initial structure shown in Fig. 5.5(a) is 12,075 kg ($a_i = 22,580.6$ mm$^2, i = 1, 2, \cdots, 15$). Considering this, the weight of the optimal truss found by the best run of BNF3 is reduced by approximately 10,000 kg to 2147.99 kg. It is only slightly heavier than that found by the best run of BNF4 (0.043%) or 2146.96 kg. The structure found by the best run of BNF4 is slightly lighter than

Table 5.7: Optimised truss structure results comparison

Member No	Cross-section area (mm^2)		
(refer to Fig. 5.5)	BNF3	BNF4	Deb and Gulati (2001)
1	19019.79	18195.46	18185.13
2	3555.81	3299.06	3367.1
3	9292.30	9346.36	9414.82
4	4877.89	5069.33	5014.18
5	12803.57	13573.84	13103.2
6	19019.79	12920.68	13322.56
7	78.34	–	–
Weight(kg)	2147.99	2146.96	2147.46
Disp. max.(mm)	2.0	2.0	2.0

TABLE 5.8: Stress comparison by truss members obtained from 6-node 15 member structure for BNF3, BNF4 and Deb and Gulati (2001)

Member No	Stress (MPa)		
(refer to Fig. 5.5)	BNF3	BNF4	Deb and Gulati (2001)
1	47.88	48.85	48.922
2	**-125.10**	**-134.83**	**-132.11**
3	-48.01	-47.59	-47.24
4	**128.97**	**124.09**	**125.46**
5	-48.98	-46.35	-48.00
6	46.63	48.69	47.22
7	3.82	–	–

the reference (0.023%) or 2147.46 kg. This means that both methods converge to the same solution. Averaged weight of the designs obtained using BNF3 is 2181.37 kg, while that of the designs obtained using BNF4 is 2182.00 kg. This reveals that the optimal truss configurations obtained by some of the runs with BNF3 are identical to the reference configuration.

Considering the allowed displacement constraint at any of the nodes ($\delta_{\max} \leq 2$ mm), the optimal truss designs found using BNF3 and BNF4 (Fig. 5.5) lie on the constraint boundary of $\delta_{\max} = 2$ mm in the design search space. This is because the global optimum is outside the constrained region and the best solution for the considered optimisation problem is at the constraint boundary.

A comparison of the axial stress at each member is listed in Table 5.8, where $\max_i |\sigma^{\text{BNF3}}| = 128.97$ MPa, $\max_i |\sigma^{\text{BNF4}}| = 134.83$ MPa and $\max_i |\sigma^{\text{DEB}}| = 132.11$ MPa. Taking these results into account, if the truss problem was posed as bi-objective problem (simultaneous minimisation of the total weight and the maximum axial stress) the optimal truss shown in Fig. 5.5(b) would lie on a Pareto-front due to its much lower maximum axial stress. This means that the solution provided by BNF3 is less feasible to that afforded by BNF4 and the reference solution in this single objective study, but would be competitive in a bi-objective study, where the total weight and the maximum axial stress are given as objectives.

5.3.2 A 6-node, 13-member structure.

The optimal truss design shown in Fig. 5.5(b) and Fig. 5.5(c) have two overlapping members – these are members 2 and 3. The overlapping of truss members is

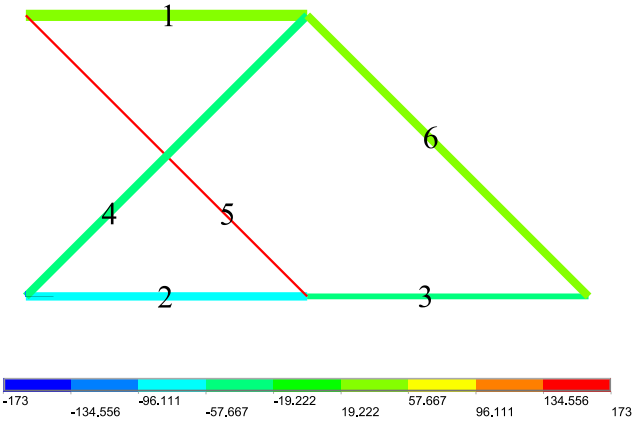


FIGURE 5.6: Stress contours for optimised truss design of 6-node 13-member truss problem. The different thickness of the lines represents the size of the cross-sectional area of each member. There are no overlapping truss members in this design

not a practical solution in many real engineering problems. For this reason members connecting nodes 1-3 and 4-6 (Fig. 5.3) are removed from the initial truss structure. The resulting initial truss has 13 unique members and six nodes.

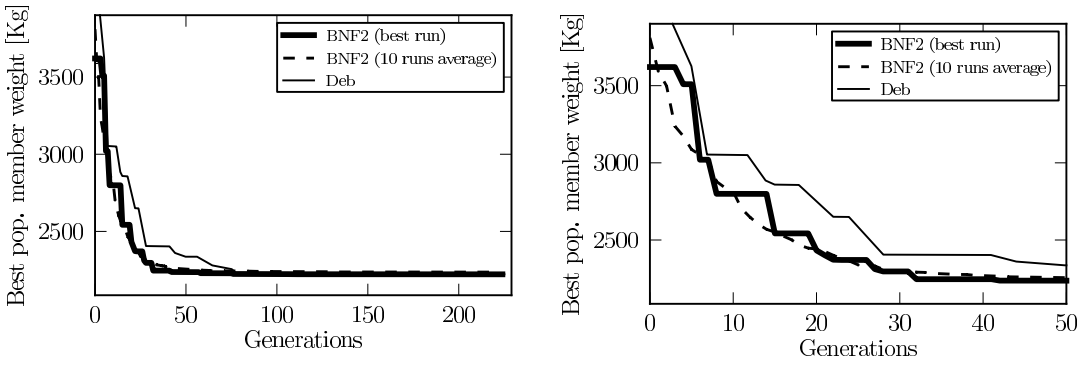
An additional run of the BNF4 based optimisation with the modified initial truss is considered next. The resolution of the search step for each normalised variable is again 4 digits after the decimal point. The binary string length is reduced to $\mu = 260$ or $\lambda = 65$ codons of length 4 bits, because $m = 13$. The population size is 220. All other GE specific parameters are unchanged and kept the same as those in Table 5.6. The truss design obtained by this run is shown in Fig. 5.6. Its topology is exactly the same as the one reported by Deb and Gulati (2001); Ringertz (1985); Hajela et al. (1993) and using alternative truss optimisation methodologies based on GA. The optimal structure is constructed by 6 members and weighs 2,222.01 kg.

TABLE 5.9: Results comparison based on several truss optimisation methods

Member No	Cross-section area (mm^2)			
(refer to Fig. 5.6)	BNF4	Deb and Gulati	Ringertz	Hajela and Lee
1	19245.01	19148.35	19419.32	18064.48
2	14123.39	14238.68	14193.52	15483.84
3	9745.01	9870.95	9677.4	10322.56
4	3931.94	3929.02	3922.57	3870.96
5	13873.39	13832.23	13741.91	13548.36
6	13785.55	13735.46	13741.91	14193.52
Weight(kg)	2222.01	2222.29	2219.87	2241.95

A quantitative comparison is presented in Table 5.9, where the cross-sectional area of each member is listed. From these results, it can be concluded that the optimised structure is almost identical to the solution reported by Deb and Gulati (2001) and Ringertz (1985). Also, it is 20 kg (0.89%) lighter than the structure reported by Hajela et al. (1993).

The comparison of the performance can be best made by studying the convergence history for various methods – i.e. by plotting the evolution of the objective function with generations. This is taken up next. The convergence history averaged over 10 runs is shown in Fig. 5.7(a). The figure also shows the convergence comparison from best run of design search using BNF4 and that using the method of Deb and Gulati (2001) (DEB). The convergence history reveals that the objective function moves towards the minimum in fewer generations than what is achieved by the use of DEB. The objective settles at the same value eventually; however, when the computational resources are limited, the proposed methodology would provide better solutions. To highlight this, a zoomed view of the convergence history for the first 50 generations is presented in Fig. 5.7(b). It can be seen that the proposed grammar based methodology using GE with BNF4 outperforms the DEB method within 50 generations. Unfortunately averaged optimisation comparison between BNF4 and DEB is not possible since the averaged results based on DEB method are not available in the literature.



(a) Convergence history of 225 generations

(b) A zoomed view of first 50 generations

FIGURE 5.7: A convergence comparison between BNF4 and DEB (results interpreted from (Deb and Gulati, 2001)) for the 6-node 13-member truss problem

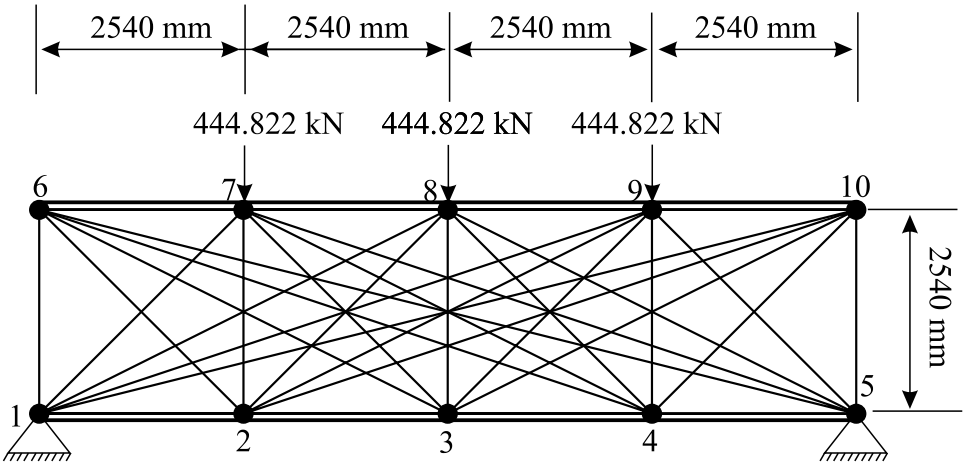


FIGURE 5.8: A 10-node 45-member truss problem. All possible connections between the 10 nodes are present. The two parallel lines with a small gap in between illustrate all possible overlapping connections between collinear nodes (1,2,3,4,5) and (6,7,8,9,10). The truss is constrained at nodes 1 and 5. A force of $F = 444.82$kN is applied at nodes 7, 8 and 9

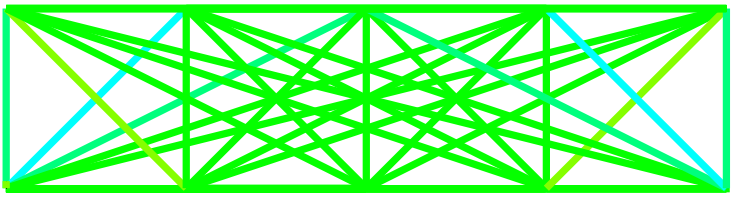
5.3.3 A 10-node 45-member structure

Another truss problem is now considered in order to compare the performance of the proposed BNF4 with the results obtained from all 2D truss optimisation conducted by Deb and Gulati (2001). Consider the optimisation of the 10-node planar truss structure shown in Fig. 5.8. The total possible number of members in a ten node 2D truss structure connecting each node to every other node is $m = \binom{10}{2} = 45$. This number includes all overlapping members connecting the collinear nodes (1,2,3,4,5) and (6,7,8,9,10) highlighted in the same figure as two parallel lines with a small gap in between. The two parallel lines represent all possible connections between the collinear nodes (1,2,3,4,5) and (6,7,8,9,10). Note that not all members are shown so as to reduce the clutter of parallel lines in the sketch. The truss is constrained at nodes 1 and 5 in both directions. The loading is applied in the y-direction at nodes 7, 8 and 9.

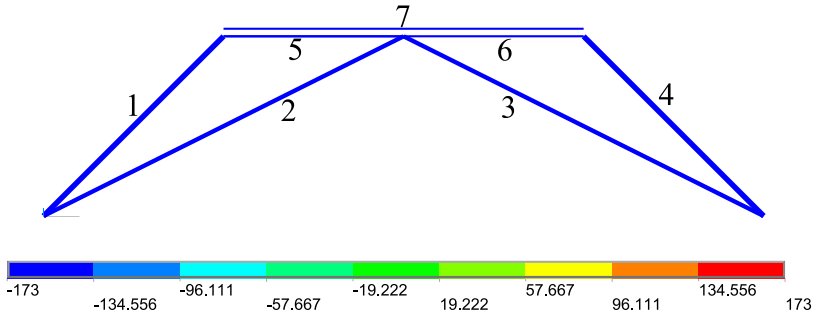
A BNF4 grammar with search step of three digits after the decimal point e.g.

$$\langle \bar{a}_i \rangle ::= 0.\langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle \mid -0.\langle \text{int} \rangle \langle \text{int} \rangle \langle \text{int} \rangle \tag{5.13}$$

is used. The binary string length is calculated as $\mu = 4(45 \times 3 + 45) = 720$ or $\lambda = 180$ codons. All GE specific parameters are kept the same as the previous examples (see Table 5.6) except that the number of binary string crossover points is increased to 30 (due to increased length μ) and the population size is adjusted to 220. All the loading and the structural parameters (F_i, $\delta_{\max}, \sigma_{\max}$, etc.) are the



(a) Initial truss design (Weight = 411.07 kg)



(b) Optimised truss design (Weight = 20.14 kg)

FIGURE 5.9: Stress contours for the initial and optimised truss design of 10-node 45-member truss problem. The different thickness of the lines represents the size of the cross-sectional area of each member. The overlapping truss members are illustrated as parallel lines with a small gap between

same as those in Section 5.3.1 (Table 5.5) except the maximum and the minimum allowable values of the cross-section area that are $a_{\max} = 645.16$ mm^2 and $a_{\min} = 32.258$ mm^2, respectively.

The optimal truss configuration obtained from the best run of BNF4 based search is shown in Fig. 5.9. It consist of 7 truss members and weighs 20.14 kg. Each repetitive run resulted in identical truss configurations with small variations in the weight due to different cross-section of truss members. The average weight is 20.55 kg with standard deviation of 0.137 kg. In comparison, the weight of the initial (unoptimised) truss structure is 411.07 kg. The cross-sectional area of each member from the best run is listed in Table 5.10. These are compared with reference results from a single run reported in Deb and Gulati (2001) for the same problem. The comparison shows insignificantly small difference. The reference optimal truss was obtained with a truss optimisation methodology based on GA, where the population size is 1,800. Unfortunately no information is available in the literature about the number of generations and averaged results from repetitive runs. The truss optimisation methodology BNF4 proposed in this chapter, was able to provide the same result with the population of size 220 members in 225 generations. The population size (related to the computational expense) in the proposed methodology is reduced by a factor of 9.

The values listed in Table 5.10 reveal that the members numbered as (1,4), (2,3) and (5,6) (see Fig. 5.9), have identical cross-section area. This means that the solution of the problem is symmetrical about the vertical axis passing through nodes 3 and 8; see Fig. 5.8. This is expected since the boundary conditions are applied symmetrically at nodes (1,5) and (7,8,9). Therefore, the number of optimisation variables can be reduced by approximately half if a symmetrical FEA is conducted. This is because the results obtained in terms of cross-sectional area, axial stress, displacement at nodes and their position can be mirrored using the symmetry axis. Next optimisation of a new bridge-type truss problem is considered, where this type of symmetry is imposed.

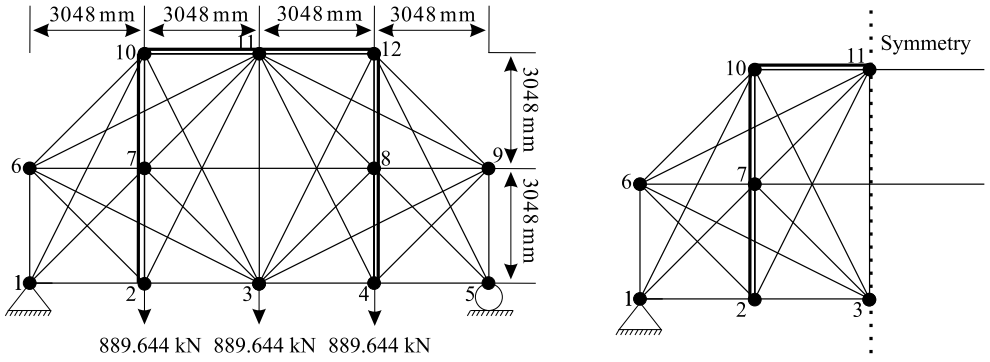
TABLE 5.10: Optimal cross-section area comparison by truss members for the 10 node 45 member truss problem

Member No	Cross-section area (mm^2)	
(refer to Fig. 5.9)	BNF4	Deb and Gulati
1	367.74	365.16
2	290.32	307.74
3	290.32	307.74
4	367.74	365.16
5	138.06	52.90
6	125.16	207.10
7	138.06	51.61
Weight(kg)	20.14	20.58

5.3.4 Bridge-type structure

The last problem considered here is characterised by the complexity frequently present in many real engineering structures. The number of nodes is increased to 12 and the boundary constraints are applied to simulate a bridge-type structure under loading. This problem is taken as a case study for two reasons. Firstly, to investigate if the proposed truss design methodology based on GE using BNF4 will converge to a solution available in the literature, but with less number of iterations. Secondly to demonstrate the application of the proposed methodology in a symmetrical truss design optimisation scenario.

The initial truss shown in Fig. 5.10(a) has a total of 39 members connected to 12 nodes. The two parallel lines represent all possible connections between the collinear nodes (2, 7, 10), (4, 8, 12) and (10, 11, 12). The boundary constraints



(a) All members

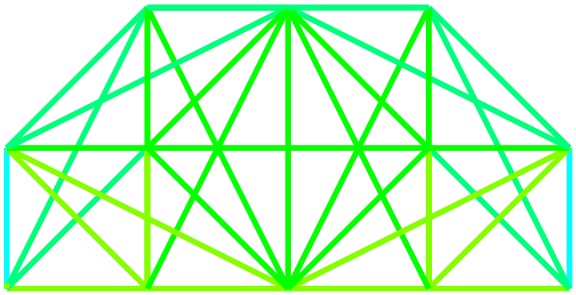
(b) Symmetry along the middle vertical axis passing though nodes 3 and 11

FIGURE 5.10: An illustration of a 12-node 39-member bridge-type truss problem. The parallel lines with a small gap in between illustrate all possible overlapping connections between collinear nodes (2, 7 10), (10, 11, 12) and (4, 8, 12). The truss is constrained at nodes 1 and 5. A force of $F = 889.64$kN is applied at nodes 2, 3 and 4. The problem is considered as symmetrical along the middle vertical axis passing though nodes 3 and 11

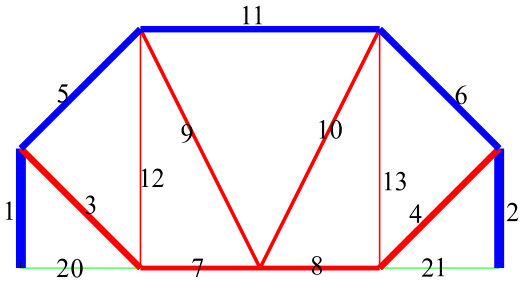
shown in Fig. 5.10(a) suggest that the problem can be solved by assuming symmetry along the vertical axis passing through nodes 3 and 11. Therefore, the truss members are reduced to 21 by assuming a symmetry along the central vertical axis of the structure shown in Fig. 5.10(b). In this way the number of optimisation variables is reduced to 21.

TABLE 5.11: Optimal cross-section area comparison by truss members for the 12 node 39 member truss problem

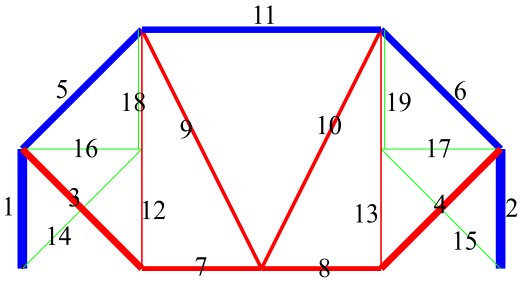
Member No	Cross-section area (mm^2)	
(refer to Fig. 5.11)	BNF4	Deb and Gulati
(1,2)	997.42	969.03
(3,4)	685.16	684.52
(5,6)	685.31	685.85
(7,8)	486.45	484.52
(9,10)	361.55	360.64
(11)	645.42	648.39
(12,13)	32.40	–
(14,15)	–	32.90
(16,17)	–	32.90
(18,19)	–	33.55
(20,21)	32.90	–
Weight(kg)	88.27	89.145



(a) Initial truss design (Weight = 740.98 kg)



(b) Optimised design using BNF4 (Weight = 88.27 kg)



(c) Optimised design using DEB (Weight = 89.14 kg)

FIGURE 5.11: Stress contours for the initial and optimised design of 12-node 39-member bridge-type truss problem. Two similar optimal designs obtained using GE with BNF4 and the method of Deb and Gulati (2001) are presented

The proposed BNF4 syntax is used again with a search step resolution of normalised variables as three digits after the decimal point

$$\langle \bar{a}_i \rangle ::= 0.\langle \text{int} \rangle\langle \text{int} \rangle\langle \text{int} \rangle \mid -0.\langle \text{int} \rangle\langle \text{int} \rangle\langle \text{int} \rangle. \tag{5.14}$$

All other GE specific parameters are unchanged except the number of generations is increased to 255. The binary string length for the initial truss having 21 members is calculated as $\mu = 4(21\times4+21) = 420$ or $\lambda = 105$ codons. All the loading and the structural parameters ($F_i, \delta_{\max}, \sigma_{\max}$, etc.) are the same as those in Section 5.3.1 (Table 5.5) except the value of the allowed maximum axial stress $\sigma_{\max} = 137.895$ MPa and the maximum and minimum cross-sectional area $a_{\max} = 1451.61$ mm^2, and $a_{\min} = 32.258$ mm^2.

The optimised truss configuration obtained from the best run is shown in Fig. 5.11(b). The optimised structure consists of 13 members and weighs 88.27 kg. In contrast, the optimised structure obtained by the method of Deb and Gulati (2001) is shown in Fig. 5.11(c). It consists of 19 members and weighs 89.14 kg. From these figures, it can be seen that the two structures have slightly different topologies. However, the main weight contributing members are the same in both cases. The averaged topology obtained from 10 repetitive runs using BNF4 weighs

91.16 kg with standard deviation of 3.08 kg, where some of the runs converged to slightly different topological configurations.

5.4 Discussions

The truss optimisation problems presented in Section 5.3 are highly multi-modal i.e. the solution space has many local optima. Also, the global optimum lies exactly on the displacement constraint boundary surrounded by many local optima close to each other. A solution space having these features was considered in Chapter 3 where optimisation of the 'bump' function was carried out.

The dimensionality of the solution space here, is given by the number of truss members m, which coincides with values for their cross-section area. Considering the BNF syntaxes proposed in Section 5.2, a member is removed if the following rules are selected

$$\begin{aligned} \bar{a}_i &= \text{reject} \\ \bar{a}_i &\leq a_{\min}. \end{aligned} \tag{5.15}$$

From the truss optimisation perspective, the difference between the GE and a pure GA is in the mapping procedure of the binary string. The accuracy of locating the global optimum depends on search step κ in the solution space. Recall that the mapping is carried out using the grammar rule $\bar{a}_i \rightarrow 0.\langle \text{int} \rangle$, where the number of non-terminals $\langle \text{int} \rangle$ after the decimal point is used to calculate κ. This is where the mapping of search space to the solution space occurs. In GE, the binary string represents a λ-dimensional search space, where λ is the number of codons.

5.5 Conclusions

In this chapter, the methodology proposed earlier to describe the shapes of solids is extended to the description of another class of problems involving the topology and sizing design of trusses. A new set of grammar rules are developed here in order to generate a family of trusses. Subsequently the truss topology description is incorporated into an optimisation scheme which is evolutionary.

The results are compared with previously reported examples of truss optimisation based on GA. The optimised structures match very well with the known results.

However, the convergence speed of the proposed methodology is slightly improved. Most of this improvement is due to BNF mapping of binary strings to grammar rules, which is an alternative technique for decoding binary information to real numbers (variables). The other not that significant contribution for an improved convergence could be a result of refinements and recent advances in GAs – the results chosen for comparison are a decade old.

The comparison reveals that the automated SG based on GE performs slightly better. It has better convergence speed – it requires much smaller population size and provides slightly better optimal results while using the same number of generations. This also suggests the usefulness of the proposed method the computational resources are limited. This may be very significant for large and complex structures such as those commonly encountered in engineering practice.

The use of GA in combination with BNF syntax mapping i.e. GE in the context of truss optimisation provides fast convergence if:

1. The solution space is relatively non-linear compared to the search space.
2. There are no dummy codons in the search space i.e. all codons are used to select grammar rules.
3. Multiple crossover points are used when the binary string length is particularly long.
4. Multiple mutations occur in a single binary string with probability p_m.

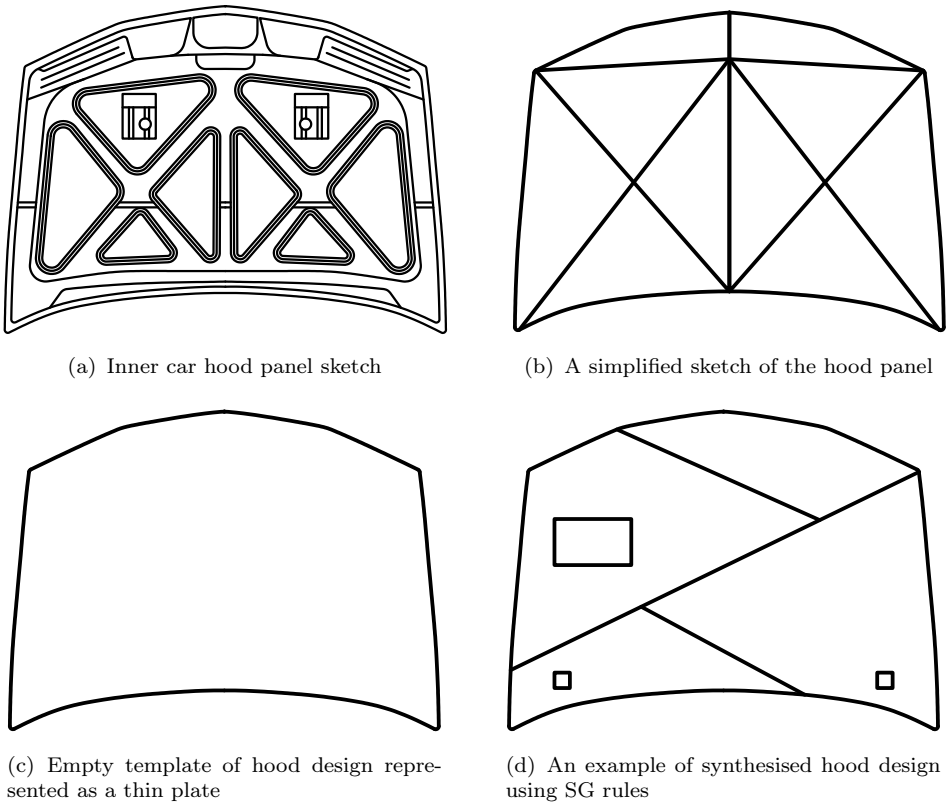
Most of the stated factors are related to characteristics of the GA rather than that of BNF mapping.

Chapter 6

Optimisation of stiffened plates under pressure loading

To demonstrate the applicability of the BNF4 syntax proposed in the previous chapter to engineering problems with reasonable complexity, a stiffened thin plate under pressure loading is considered in this chapter. The stiffeners are defined by a network of beams tied to a plate. The topology of the stiffeners is similar to the truss problems considered in Chapter 5 because they are networks of straight elastic members. However, the mechanics of trusses is computationally different from that of stiffeners because the former has member in axial tension and compression whereas the latter is in flexure.

A practical engineering situation having these structural features is an inner car hood panel design. McCormack and Cagan (2002a) propose a family of SG rules efficient in design search of such structures. An example of typical inner car hood panel design is shown in Fig. 6.1(a). In Fig. 6.1(b) the hood panel is simplified as a thin plate welded to a network of beams. The network structure is used to stiffen the thin plate under out-of-plane loading. The design search technique proposed by McCormack and Cagan (2008) is novel and interesting, but not very practical. This is because the SG rules are selected and have to be applied by the designer *manually*. For example, consider the initial design representing the hood panel as an empty template shown in Fig. 6.1(c). Depending on the SG rules selected, various network structures with alternative topologies can be synthesised by arranging the connectivity of the members. An example of a hood design thus obtained by manual intervention is shown in Fig. 6.1(d).



(a) Inner car hood panel sketch

(b) A simplified sketch of the hood panel

(c) Empty template of hood design represented as a thin plate

(d) An example of synthesised hood design using SG rules

FIGURE 6.1: An example of inner panel car hood design adapted from (McCormack and Cagan, 2002a)

The idea proposed by McCormack and Cagan (2002a) for using SG rules in design description and optimisation of the panel has not been automated. In this chapter, a realisation of the idea as a computational routine is demonstrated via numerical examples. The BNF4 syntax proposed in Chapter 5 is modified by adding new grammar rules used to synthesise alternative network configurations (topologies) taking into account one or more of the following objectives:

1. Minimise the displacement at any point under out of plane loading.
2. Minimise the total weight.
3. Minimise the worst von Mises stress.

Topology optimisation with GE is also considered in a different light. Here, a bi-objective design optimisation of a stiffened panel is considered and the results are presented.

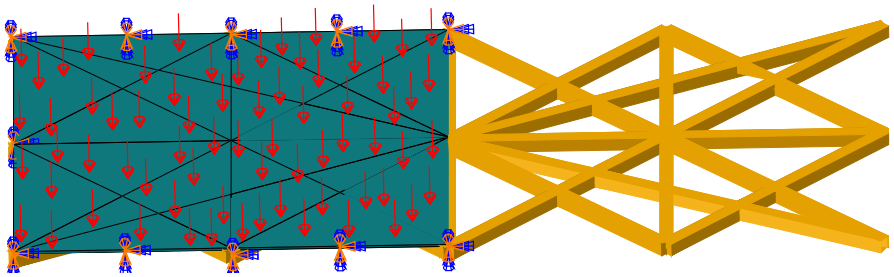


FIGURE 6.2: View cut of a 3D model representing a thin plate welded on top of network of beams. The plate is clamped along edges and transverse surface load of $P = 0.058$ N $\times\ mm^{-2}$ is applied on plate surface. The panel is shown in perspective where the topology of the stiffeners is visible

6.1 The design optimisation problem

The car hood panel design problem proposed by McCormack and Cagan (2002a) is solved here using GE where SG rules are given in BNF syntax. The design problem consists of a thin plate sheet which is stiffened by a network of beams. The panel is clamped along all edges, while an out-of-plane loading is applied on the surface.

The loading transverse to the stiffened panel can have a range of origins such as aerodynamic loading or an accidental localised force. It is not possible to design for such local loading prescribed at a given point because of the uncertainty in knowing this. Therefore the design problem will account for a uniform transverse force that is distributed over the panel. Also, the actual geometry in case of a car hood for example, has a complex outer contour. However, in this thesis, a problem that has the same generic mechanics as these structures but where the geometry has been deliberately simplified to a rectangle is considered. An example of such a structure is shown in Fig. 6.2. The plate is of rectangular shape clamped along its edges. The panel has stiffeners that are welded to it. A uniform pressure is applied at the top of the plate. Note that the view is cut through the middle of the panel for illustration.

There are close similarities between the stiffened panel problem under consideration and the stiffened skin structures commonly found in aircraft wings. There are generic similarities of this structure with many stiffened walls frequently used in marine applications too. Because such constructions are so ubiquitous in engineering, here a problem that is found across the sectors spanning the automotive, civil, aerospace and marine industries is considered.

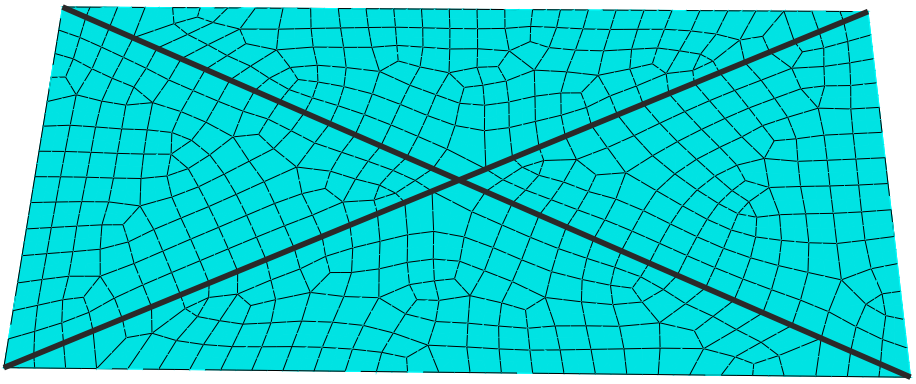


FIGURE 6.3: An example of a meshed panel stiffened by two beams. The beams are represented by two lines. General purpose shell elements S4R are used to mesh the plate. The beam are meshed by B32 beam elements

6.1.1 FEA set-up

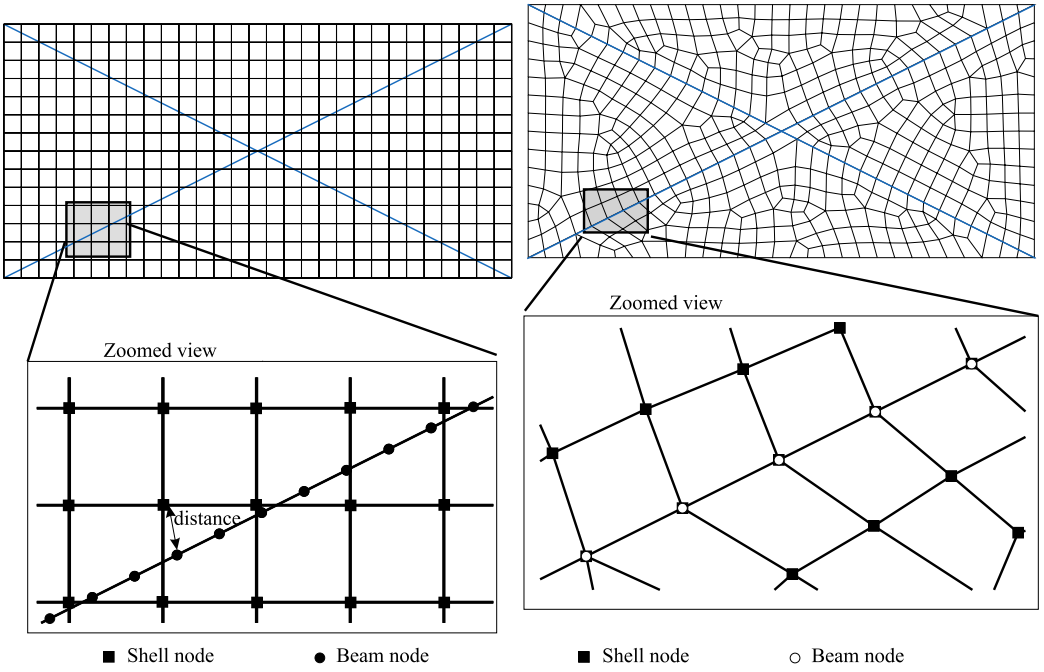
A stiffened plate is shown in Fig. 6.3. The skin structure is meshed using general purpose shell elements (S4R) provided by the ABAQUS software package. The FEA set-up requires two groups of finite elements:

1. The first group of element is used to mesh the plate containing a mixture of rectangular (S4R) and triangular (STRI65) conventional shell elements. The shell elements are used to model structures, where the thickness is significantly smaller than other two dimensions. The element type 'S4R' within the ABAQUS/Library is an 4-node stress/displacement shell (S) with reduced integration point (R). The element type 'STRI65' is a 6-node triangular (TRI) stress/displacement shell (S) with 5 degrees of freedom per node.

2. The second group of elements is used to mesh the stiffeners using beam elements (B32). The element type 'B32' is a 3-node quadratic beam element (B).

Tying the nodes between the plate and the stiffeners. As previously mentioned, the structure considered consists of two types of solid objects – stiffeners and the thin plate. It is, therefore, important to maintain continuity at the interface of the two types of finite elements. This is achieved by tying the nodes at the interface. In ABAQUS, the welding contact interaction is possible by "tying up" nodes that belong to beams and shells at the points that are approximately coincident. The tied nodes interact with each other obeying both equilibrium and compatibility.

The nodes to be tied are determined by the distance between the nodes that belong to one type of element and that to another type. Sometimes due to mesh



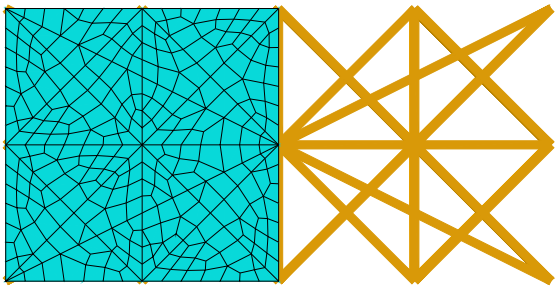
(a) Default meshing (b) Partitioned meshing

FIGURE 6.4: Two techniques for meshing stiffened plates with thick shell elements. Note that in partitioned meshing the beams element nodes match to those of shell element nodes. A zoomed view is presented in this illustration, where the position of nodes is visible

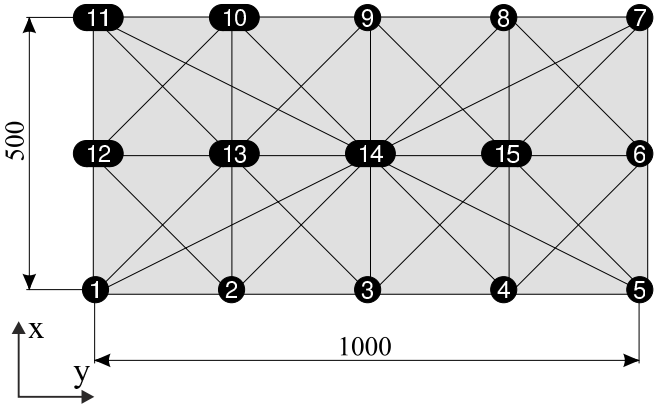
size differences (between the beam and shell nodes, for example) the closest planar distance may be significantly large (see Fig. 6.4(a)). Consequently, the structural interactions between the plate and the ribs could result in computational errors. To overcome this, an advanced meshing technique is used. In this technique, first the plate geometry is partitioned according to the topology of the network of beams. Then, each partition of the plate is meshed separately. This is followed by meshing the beams that lie on these partitions. Since the shell elements are selected to be of the same size as the beam elements their nodes coincide. The resulting mesh is illustrated in Fig. 6.4(b). Note the alignment of shell elements along the region of interaction with the beam elements – they match exactly to the nodes of the beam elements.

6.1.2 Problem definition

The stiffened plate shown in Fig. 6.5 is considered as the initial design. This structure consists of $m = 30$ beams connected at $n = 15$ nodes. Taking the boundary conditions into account, this problem can be considered as symmetric



(a) Meshed 3D model



(b) Sketched view with dimensions

FIGURE 6.5: An initial design of stiffened plate problem. The stiffeners are illustrated as a network of 30 members connected at 15 nodes

along the line of symmetry passing through nodes 3, 14 and 9. The objective of the optimisation is to find the topology of the stiffeners, their cross-sectional areas and the plate thickness that lead to the optimal performance on two counts – the weight and the maximum stress. In addition there is a constraint in terms of the maximum transverse displacement at any point within the structure.

The optimisation objectives. The optimisation problem is a bi-objective one with one constraint. The values of the overall weight and the maximum stress are restricted within a range of intervals. This is implemented early by assigning a penalty to designs that are outside this range. The objectives are to simultaneously minimise the total weight and the maximum von Mises stress, while keeping the displacement at any point in the structure below 2 mm.

The first objective is a sum of the weight of the constituents of the panel viz. the plate and the ribs. This can be expressed as

$$f_{min} = \sum_{i=1}^{m} a_i l_i \rho + tWB\rho. \tag{6.1}$$

TABLE 6.1: Fixed material and geometric parameters of optimisation

Surface pressure (P)	=	0.058 N $\times$ mm^{-2}
Young's modulus (E)	=	210 GPa
Density – Steel (ρ)	=	7.85×10^{-3} kg $\times$ mm^{-3}
Poisson's ratio (ν)	=	0.3
Plate breadth (H)	=	1000 mm
Plate width (W)	=	500 mm

The first term provides the weight of the stiffeners, where a_i is the cross-section, l_i is the length of each stiffener ρ is the material density. The second term is the expression for the weight of the panel, where t is the thickness, W is the width, H is the breadth and ρ is the material density.

The second objective (maximum von Mises stress within the panel) is calculated during the FEA run. The variables of the optimisation loop are:

- The topology of the ribs.
- The cross-sectional area of the ribs a_i.
- The thickness of the plate t.
- Position of the nodes (x_i, y_i co-ordinates).

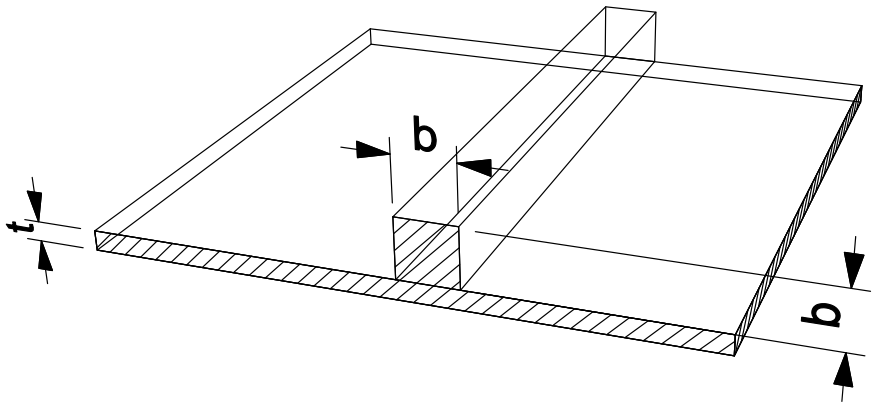


FIGURE 6.6: Panel drawing with dimensions. The optimisation variables are the beam cross-section b_i and plate thickness t

The maximum transverse displacement at any point in the structure is constrained in the range $\delta_{\max} \leq 2$ mm. Also, the Pareto-front is restricted in the range $\sigma_{\max} \leq 260$ MPa and $M \leq 25$ kg for the von Mises stress and the overall weight, respectively. Fixed material and geometric parameters are listed in Table 6.1 where material properties of the stiffeners and the plate are the same. The optimisation

TABLE 6.2: Optimisation variables for the proposed stiffened plate problem

Minimum beam area ($a_{i,\min}$)	=	10^2 mm^2
Maximum beam area ($a_{i,\max}$)	=	50^2 mm^2
Minimum plate thickness ($t_{\min}$)	=	1 mm
Maximum plate thickness ($t_{\max}$)	=	5 mm

variables are given in Table 6.2, where beam cross-section profile is square i.e. $a_i = b_i \times b_i$, see Fig. 6.6. In summary, the optimisation variables are: b_i, l_i and t, where $i = 1, 2, \cdots, m$.

6.1.3 The BNF syntax used

Two alternative BNF syntaxes are proposed considering the design problem presented above. The first BNF developed is efficient when the construction point co-ordinates (nodes) of the ribs are fixed and requires following components:

(i) A BNF expression to identify if the i-th network member is accepted or rejected.

(ii) If the i-th member is identified as accepted, a BNF expression to synthesise a normalised value for its cross-section.

(iii) A BNF expression to synthesise a normalised values for the plate thickness t with accuracy n digits after the decimal points.

The listed requirements lead to development of BNF shown in Table 6.3 and will be referred as BNF5. In BNF5, the expression

$$\langle \bar{b}_i \rangle ::= 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \mid -0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \tag{6.2}$$

represents two grammar rules. The first rule

$$\langle \bar{b}_i \rangle \rightarrow 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \tag{6.3}$$

TABLE 6.3: First BNF syntax developed for the stiffened plate problem – BNF5

$\langle$`int`$\rangle$	`::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`
$\langle \bar{b}_i \rangle$	`::= 0.`$\langle$`int`$\rangle\langle$`int`$\rangle\cdots\langle$int$\rangle$ `| -0.`$\langle$`int`$\rangle\langle$`int`$\rangle\cdots\langle$int$\rangle$
$\bar{b}_i$	= $\langle \bar{b}_i \rangle$
$\bar{t}$	= $0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle$

is used to identify the i-th network member as accepted. Following the application of this rule, a positive normalised value for the cross-sectional area of i-th beam $\bar{b}_i$ is synthesised. The second rule

$$\langle \bar{b}_i \rangle \rightarrow -0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle \tag{6.4}$$

identifies the i-th beam as rejected and a negative normalised value for $\bar{b}_i$ is synthesised. The negative values are used to identify the rejected members. The normalisation is carried out for the interval $b_\Delta = b_{\max}$ - $b_{\min}$, where $b_i = b_{\min} + \bar{b}_i \times b_\Delta$.

The initial sentence $\bar{b}_i$ is used to synthesise a cross-sectional area for each member as well as a normalised value for the plate thickness ($\bar{t}$). The normalisation of plate thickness value is carried out for the interval $t_\Delta = t_{\max}$ - $t_{\min}$, where $t = t_{\min} + \bar{t} \times t_\Delta$.

TABLE 6.4: Second BNF syntax developed for the stiffened plate problem – BNF6

$\langle \text{int} \rangle$	::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
$\langle \bar{b}_i \rangle$	::= $0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle$ | $-0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle$
$\bar{b}_i$	= $\langle \bar{b}_i \rangle$
$\bar{t}$	= $0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle$
$(\bar{x}_j, \bar{y}_j)$	= $(0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle, 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle)$

The second BNF considered is similar to BNF5 however, this syntax is used when the position of the joints in the beam network are also considered as optimisation variables. Therefore, additional expression to synthesise x and y position co-ordinates of nodes is required. This leads to development of BNF presented in Table 6.4 and will be referred as BNF6.

The only difference in BNF6 is the line

$$(\bar{x}_j, \bar{y}_j) = (0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle, 0.\langle \text{int} \rangle \langle \text{int} \rangle \cdots \langle \text{int} \rangle) \tag{6.5}$$

in the initial sentence. This sentence provides a synthesis of the x, y co-ordinates of the j-th node identified as movable. Again, the synthesised variables $\bar{x}_j$ and $\bar{y}_j$ are normalised for intervals $x_{\min} \leq x_i \leq x_{\max}$ and $y_{\min} \leq y_i \leq y_{\max}$.

6.2 Results

First, the design optimisation of the initial panel shown in Fig. 6.5 is conducted using BNF5. The optimisation variables are taken as the cross-section of the 16

TABLE 6.5: GE specific parameters applied to BNF5 and BNF6

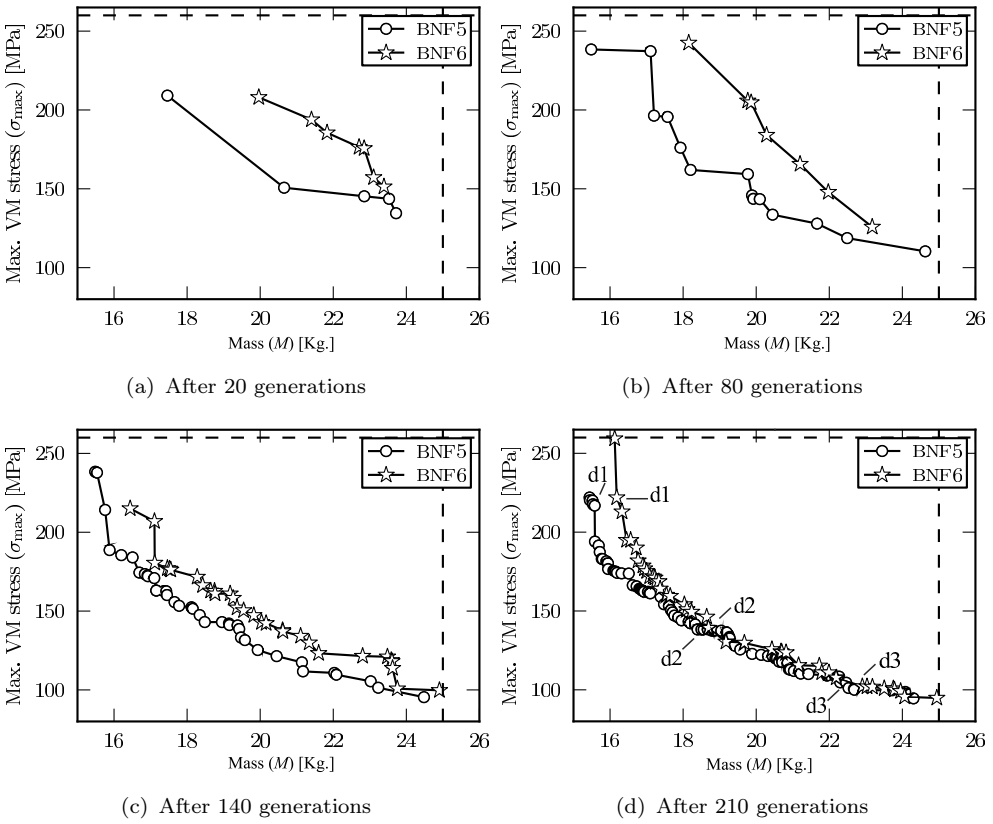
Generations	210
Population size	80
Crossover probability (p_c)	0.8
Crossover type	multiple
Crossover points	30
Mutation type	multiple
Mutation probability (p_m)	0.035
Codon length (β)	4

members and the plate thickness. All the 15 joints as labelled in Fig. 6.5(b) are fixed. The search resolution of $\bar{b}_i$ and $\bar{t}$ is taken as two digits after the decimal point by the sentence $0.\langle\text{int}\rangle\langle\text{int}\rangle$. Considering this, the search step for variables b_i is $0.01 \times b_\Delta = 0.4$ mm. Similarly the search step for the plate thickness t is $0.01 \times t_\Delta = 0.04$ mm. The optimal genotype length for this problem is calculated as $\mu = \beta(m \times 2 + m + 2) = 4(16 \times 2 + 16 + 2) = 200$ bits. This length is calculated taking into account the number of grammar rules to be selected which is triggered by the maximum possible non-terminal primitives during the rule selection process.

The second study is conducted using the BNF6 syntax, where the position of the 13-th node is kept as a variable free to move in the x and y-directions in the intervals $10 \leq x_j \leq 490$ and $10 \leq y_j \leq 490$, respectively. Also nodes 2 and 10 are allowed to move only in the x-direction – the y-direction is fixed. In contrast, node 12 is allowed to move only in the y-direction within the same interval. Note that the nodes 4, 6, 8 and 15 are moved accordingly based on the assumed symmetry, see Fig. 6.5(b). Taking into account the additional optimisation variables, the genotype length is increased by 40 bits to $\mu = 240$ bits, since 10 codons are now required to synthesise the nodal co-ordinates.

The GE specific parameters used in both runs (BNF5 and BNF6) are as listed in Table 6.5.

The Pareto-front obtained are superimposed and presented in Fig. 6.7. The four graphs show the evolution of the Pareto-fronts as the computational resources are increased. The figure shows that design optimisation using BNF5 (fixed nodes) finds panel structures with smaller weight. The Pareto-front for BNF5 dominates that of BNF6 in Fig. 6.7(d) for structures with weight less than 20 kg. Beyond this there is no clear superiority of one over the other. This figure also shows the Pareto-front evolution after 20, 80, 140 and 210 generations in Figures 6.7 (a)–(d), respectively. When the computational resources are limited the BNF5 syntax



(a) After 20 generations

(b) After 80 generations

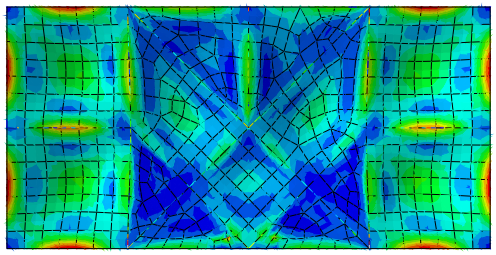
(c) After 140 generations

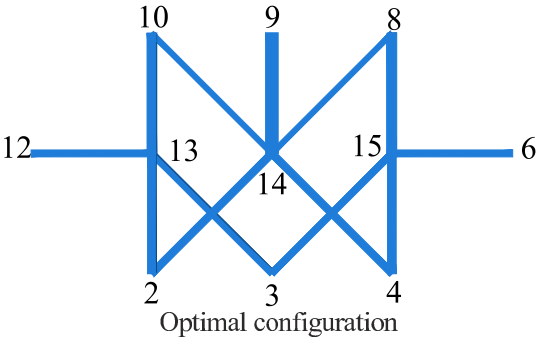
(d) After 210 generations

FIGURE 6.7: Pareto-front evolution for stiffened plate problem using the proposed method based on GE with BNF5 and BNF6, respectively. The evolution stages are given for the first 20, 80, 140 and 210 generations

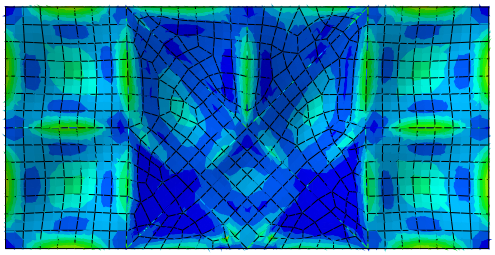
has an advantage over BNF6, which is evident in Fig. 6.7 (a) through to (c). This might be due to increased number of optimisation variables within the BNF6 syntax since some of the nodal co-ordinates are taken as additional optimisation variables. This, on the other hand, enlarges the dimensionality of the search space because additional 10 codons are required to synthesise the nodal co-ordinates. As a result, design optimisation using BNF6 requires more computation resources and converges slowly compared to BNF5. Comparison for averaged values was not conducted since each repetitive run requires significant computational time which was not available during the experiments.

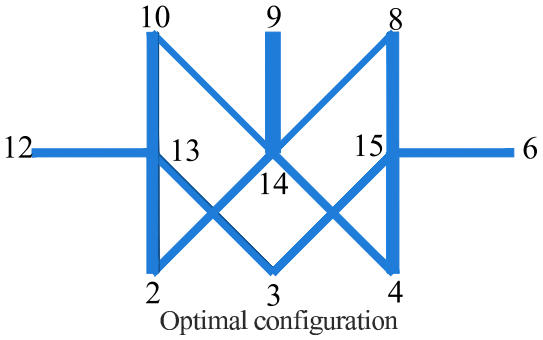
Earlier in this chapter it was mentioned that the optimisation problem considered is bi-objective i.e. the simultaneous minimisation of maximum von Mises stress σ_{vm} anywhere in the structure and its mass M. Three optimised designs picked up from the Pareto-front obtained using BNF5 (Fig. 6.7(d)) are shown in Fig. 6.8 (a) through to (c). In this figure, the cross-sectional size of beams is represented by the width of the lines. The selected designs correspond to the points on the Pareto-front labelled d1, d2 and d3, respectively. They are selected to illustrate



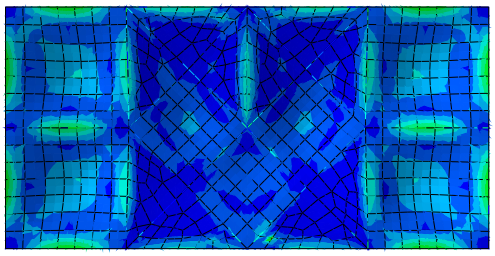


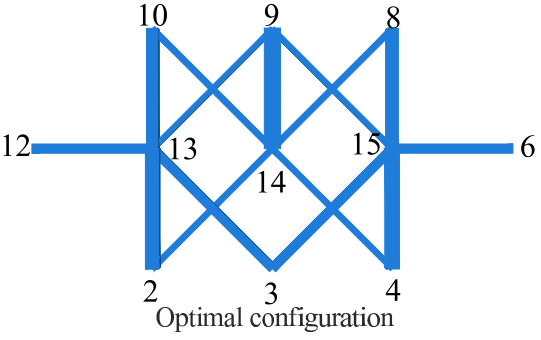
(a) d1





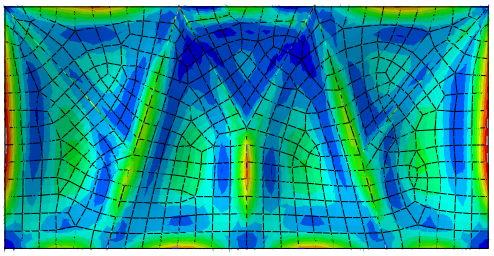
(b) d2

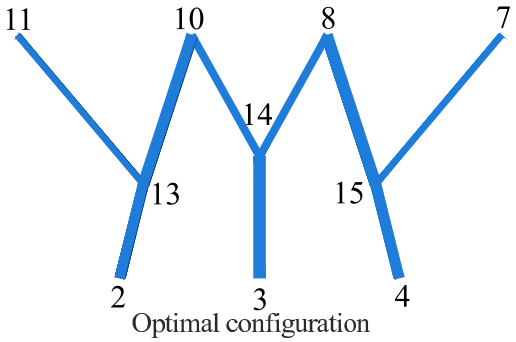




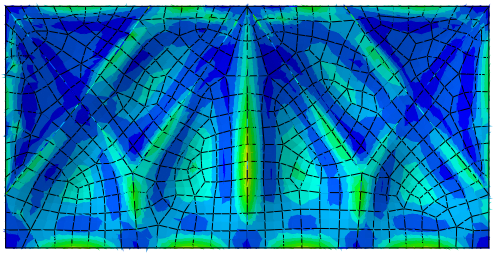
(c) d3

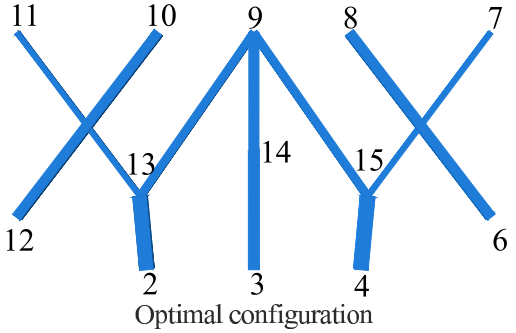
FIGURE 6.8: Optimised panel designs picked from Pareto-front obtained by BNF5. The three alternative designs illustrate the trade-off between panels with low stress and increased weight and those with high stress and reduced weight. The main difference can be seen in the number of stiffeners and their cross-section



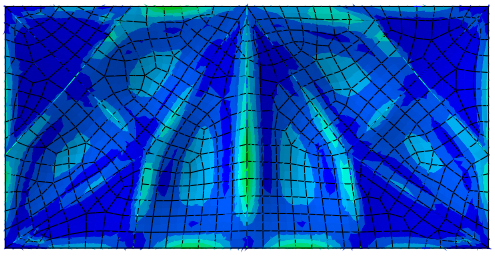


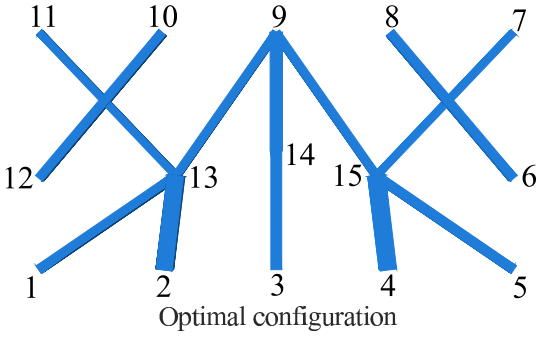
(a) d1





(b) d2





(c) d3

FIGURE 6.9: Optimised panel designs picked from Pareto-front obtained by BNF6. The three alternative designs illustrate the trade-off between panels with low stress and increased weight and those with high stress and reduced weight. The main difference can be seen in the number of stiffeners and their cross-section

the trade-off between maximum von Mises stress and weight of optimal panels. These designs are Pareto-optimal i.e. they represent a range of non-dominated solutions to the problem. For example, the panel design shown in Fig. 6.8(a) weighs less than that in Fig. 6.8(c), but has higher von Mises stress. This is mainly due to difference in plate thickness, cross-section of stiffeners and their number – the optimal configuration of panel design shown in Fig. 6.8(c) has two additional members compared to that in Fig. 6.8(a).

The asymmetric configuration of stiffeners considering the horizontal axis passing through nodes 12, 13, 14, 15 and 6 indicates that the design optimisation has not fully converged. This is because the computational time was limited to 210 generations, since the contact simulation between the plate and the stiffeners is a time consuming procedure and increases the overall search time significantly. However, the time for conducting this optimisation experiment was limited.

The next three optimised panel designs shown in Fig. 6.9 (a) through (c) are picked from Pareto-front BNF6 (Fig. 6.7(d)). Again, these designs are Pareto-optimal i.e. they represent a range of non-dominated solutions. They provide performance similar to panel designs presented in Fig. 6.8 respectively. The cross-section of the members as well as the plate thickness and the total weight can be found in the Appendix in Tables A.1 through to A.3 for designs presented in Fig. 6.8 (a) through to (c), respectively. Similarly, the cross-section of the members, the nodal co-ordinates as well as the plate thickness and the total weight are listed in Tables A.4 through to A.6 for panel designs presented in Fig. 6.9 (a) through to (c), respectively. The initial design weighs $M = 223.59$ kg, has a transverse displacement $\delta = 0.011$ mm and maximum von Mises stress $\sigma_{\max} = 6.87$ MPa, where the cross-sectional area of each member is $a_i = 50 \times 50$ mm^2 and the plate thickness is $t = 5$ mm.

6.3 Conclusions

The design description and optimisation framework proposed in this thesis is applied to class of design problems involving stiffened plates under transverse loading. An automated approach for the selection and application of SG rules similar to those proposed by McCormack and Cagan (2002a) was presented. Two specific BNF syntaxes were proposed. The first BNF (BNF5) was used in situations where the positions of nodes of the stiffener network are fixed and the design variables are given as the cross-section of the beams (stiffeners). The second BNF (BNF6)

was used in situations where the co-ordinates of the nodes were also taken as the design variables.

Experimental results from single run of BNF5 and BNF6 were presented. Interestingly, panel designs obtained by both BNF showed similar performance despite the fact that discovered optimal topology configurations are different. This is because the solutions are not fully converged. The BNF6 run would have provided better solutions than BNF5, although with greater computational resources, if more generations were allowed. This is because the nodal co-ordinates were taken as optimisation variables. Allowing the movement of network nodes increases the non-linearity of the optimisation problem. A possible solution to avoid the increased non-linearity was proposed as a subsequent run of BNF6 based on results obtained from preliminary optimisation run using BNF5.

Chapter 7

Conclusions & Future work

7.1 Conclusions

A general methodology for design description and optimisation based on an automated SG using GE has been proposed. Optimisation of single and multi-objective mathematical functions with GE were considered first. Following this, a method to synthesise planar shapes using SG strategy was presented. This general framework was then applied to the problem of optimising the shape of a crane hook. The methodology was further developed for two new classes of problems – those involving the topology description of elastic networks. Applications to truss structures in plane and stiffened plates under transverse pressure loading conditions were presented. The proposed methodology was supported by presenting experimental results for each of the structural design problems considered, where a problem specific BNFs syntax was developed and presented. The crane hook problem involved development of SG rules with arc primitive shapes, while the truss and stiffened plate problems were solved by development of SG with lines and the position of their joints. Each of these specific contributions is summarised, in turn, next.

An automation of SG using GE An algorithmic methodology to automate SG was proposed. The algorithm fills the gap between the original proposal of SG by Stiny and Gips (1971) and its actual realisation in automated design search studies using CAD modelling tools. The implementation fully automates the shape identification and grammar rule selection processes. This is carried out by combining the BNF syntax (phrase-structure grammar) and GA, in particular GE, where grammar sentences were composed with primitive shapes instead of linguistic words.

The GE is formed from a combination of BNF syntax and GA. In GE, the BNF syntax is used to define phrase-structure grammar rules. These rules were then selected using genetic information provided by GA, where through iterations of multiple generations the best sequence of selected grammar rules (genetic information) evolve.

In Chapter 3, it was demonstrated how GE can be used to automate the SG. Two techniques to represent SG rules in BNF syntax were introduced. The first technique relies on explicit representation of primitive shapes by using the co-ordinates of their construction points. The second technique relies on implicit representation of primitive shapes using mathematical functions.

The first technique was further developed to enable synthesis of primitives shapes with variable size. This was done by including their construction point co-ordinates as optimisation variables in the BNF syntax. Thus, the optimisation of the primitive shape parameters as well as their arrangement in a SG sentence was carried out by the GA. The proposed automation of SG was applied to structural design optimisation involving synthesis of SG sentences in order to improve structural designs.

Optimisation with GE From a numerical optimisation perspective, rather than automated programming and rule selection methodology, GE was characterised as an enhanced version of the standard GA. The enhancement was described as an alternative mapping (decoding) of genetic information by a means of BNF syntax (phrase-structure grammar) to the optimisation variables. It was demonstrated how the GE provides separation of search space (genotype) and solution space (optimisation variables) using the BNF syntax.

Numerical experiments using various BNF syntaxes – for synthesis of constrained optimisation variables, confirmed that the GE can be successfully used for single and multi-objective optimisation problems. The convergence speed of GE was further improved by a modification in the grammar rule mapping procedure. The modification was described as dynamic adjustment of codon bit string length β based on the defined BNF syntax. This led to efficient calculation of genotype length μ, which influenced the convergence speed of numerical optimisation in a positive way. Following this modification, the GE showed significant improvement.

The multi-objective concept of GE was realised by replacing the GA engine with NSGAII. Experimental results with several multi-objective test function matched with the results available in the literature.

During optimisation experiments with GE, it was noticed that the convergence speed depends on:

- The BNF syntax.
- Selected binary crossover type and the rate p_c i.e. single point crossover vs. multiple point crossover.
- Selected binary mutation type and rate p_m i.e. single point mutation vs. multiple point mutation.
- Calculation of the genetic code (binary sting) length μ.

The last three factors listed are related to the standard GA, however their influence is different in GE because of the BNF mapping procedure of optimisation variables.

An experimental finding was that correctly setting the crossover is an essential part of GE. In GE, the selection of crossover type and rate strongly depend on binary string length and the BNF syntax used, i.e. it is problem specific.

Shape optimisation In Chapter 4, it was demonstrated how SG can provide a means to generate complex shapes by manipulating a finite number of primitive shapes. The development of an automated SG was initiated by the need to investigate shape exploration techniques in the structural design optimisation context with improved efficiency. To illustrate this SG based planar shape synthesis was developed first. The shape optimisation of a planar crane hook using automated SG operations with arc primitive was then presented. Four unique SG rules using arc primitives with variable size were defined as a BNF syntax. These rules were then used to synthesise the design boundaries of a planar hook shape as three separate piecewise curves. The results, in terms of hook designs , given as shape sentences were compared to results from design optimisation using NURBS control points manipulation based on a GA search strategy. Some of the key conclusions drawn during this comparison are:

- The SG could be successfully used for the synthesis of sophisticated free-form shapes using explicitly and/or implicitly defined primitive shapes.
- When the computational resources are limited, the proposed SG with arc primitives of variable size finds a better solution compared to NURBS based

control point manipulation for automated shape explorations. This is because it leads to faster convergence during the search. Given excessive computational resources, the two methods converge to very similar design solutions. This indicates that the reach of the two shape description methods within the search space is very similar.

- The proposed automation of SG using arc primitive shapes can be applied to solve structural shape optimisation problems where little or no information is available regarding possible solutions of the problem e.g. for conceptual design. This may be important during the early stages of design.
- A drawback of design search with an automated SG using GE is that sometimes the search process converges prematurely to a local optima. Nevertheless this is an artefact primarily contributed by the GA process.
- The control point design search strategy based on NURBS is particularly useful if controlled manually by an experienced designer; it provides an intuitive control. On the other hand, for shape description and its use within an automated design search scenario this cannot be done. Automation has its own benefits for designs at early stages when a designer does not have a developed intuition. Speed of design search is another advantage.

Planar truss structures The methodology proposed in Chapter 4 to describe shapes of solids is extended to the description of a new class of problems involving the design of trusses in Chapter 5. A new set of grammar rules were developed in order to generate a family of trusses. Subsequently, the truss topology description is incorporated into an optimisation scheme which is evolutionary.

Several alternative BNF syntaxes were presented where all aspects (topology, sizing, and shape) of truss optimisation were considered. The results were compared with previously known examples of truss optimisation based on GA. The optimised structures match very well with the known results. However, the convergence speed of the proposed methodology is slightly improved. Most of this improvement is due to BNF mapping of binary strings to grammar rules, which quite frankly, is an alternative technique for decoding binary information to real numbers (variables). The other not that significant contribution for an improved convergence could be a result of refinements and recent advances in GAs – the compared results are a decade old.

The comparison reveals that the automated SG based on GE performs slightly better. It has better convergence speed – it requires much smaller population

size and provides slightly better optimal results while using the same number of generations. This also suggests the usefulness of the proposed method the computational resources are limited. This may be very significant for large and complex structures such as these commonly encountered in engineering practice.

The use of GA in combination with BNF syntax mapping i.e. GE in the context of truss optimisation provides fast convergence if:

1. The solution space is relatively non-linear compared to the search space.
2. There is no redundancy in the search space i.e. all codons are used to select grammar rules.
3. Multiple crossover points are used when the binary string length is particularly long.
4. Multiple mutations occur in a single binary string with probability p_m for each bit.

Most of the stated factors are related to characteristics of the GA rather than BNF mapping.

Stiffened plate under transverse pressure loading condition An application of the design description and optimisation framework proposed in this thesis to a stiffened thin plate class of design problem under transverse loading has been considered. An automated approach for selection and application of SG rules similar to those proposed by McCormack and Cagan (2002a) was presented. In particular, two specific BNF syntaxes were proposed. The first BNF (BNF5) was used in situations where the positions of network nodes are fixed and the design variables are given as the cross-section area of the beams (stiffeners). The second BNF (BNF6) was used in situations where the nodes co-ordinates were also taken as design variables.

Experimental results from single run of BNF5 and BNF6 were presented. Interestingly, panel designs obtained by both BNF showed similar performance despite the fact that discovered optimal topology configurations are different. This was explained by the fact that the solutions are not fully converged. The BNF6 run would have provided better solutions than BNF5, although slowly, if more generations were allowed. This is because the nodal co-ordinates were taken as optimisation variables. Allowing the movement of network nodes increases the non-linearity of the optimisation problem. A possible solution to avoid increased non-linearity was

proposed as a subsequent run of BNF6 based on results obtained from preliminary optimisation using BNF5.

In summary, the thesis demonstrated that the SG can be used as an automated design search tool involving grammatical operation with primitive shapes in a range of design problems encountered in engineering.

7.2 Future work

The present work was limited to geometries in 2D. Many practical engineering problems involve the design of truly 3D shapes. This would require reformulating the approach in this direction. The future work suggested here investigates development of specific BNF syntax for for synthesis of complex surfaces. Additionally, the future work plans listed below extend the application of a proposed intelligent design search tool to a range of multidisciplinary problems as follows:

- Design synthesis and optimisation of complex surfaces In general, parametric free-form surfaces are represented by mathematical functions. The idea is that SG rules could be defined using mathematical function primitives expressed implicitly e.g. $\sin(x)$, $\cos(x)$, $\tan(x)$, π, $+$, - , etc. These primitives could be used for synthesis of sophisticated functions representing parametric free-form surfaces. An example of a sphere and its transformation to various 3D geometries using grammar rules with function primitives is shown in Fig. 7.1 through to 7.4.

  Consider a set of functions representing parametric sphere geometry

$$\begin{aligned} x(u,v) &= \cos(u)\cos(v) \\ y(u,v) &= \sin(u) \\ z(u,v) &= \cos(u)\sin(v), \end{aligned} \tag{7.1}$$

  where $-\pi < u < \pi$ and $0 < v < 2\pi$. The two parameters u and v, were characterised as location, scale and rotation invariant shape representative parameters in Chapter 2. Such parametric surface possess a natural direction of traversal. By applying grammar rules with function primitives to an initial set of functions such as those in Eq. 7.1 a range of surfaces can be explored. For example, the initial geometry (a sphere) shown in Fig. 7.1 given by Eq. 7.1 is morphed into a bow tie surface shown in Fig. 7.2 by synthesising

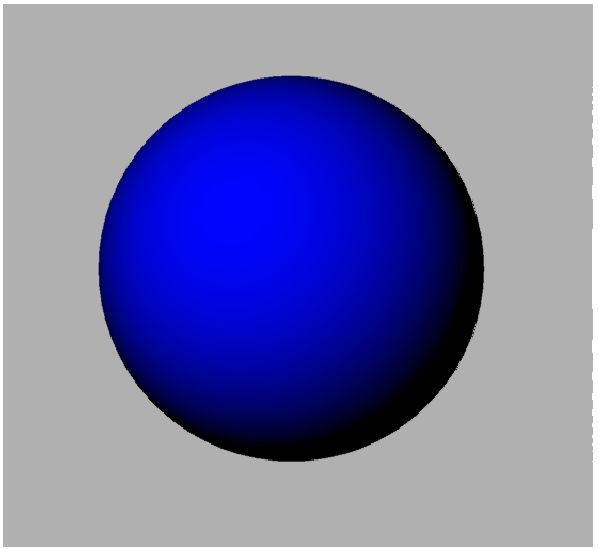


FIGURE 7.1: Functional representation of a parametric sphere

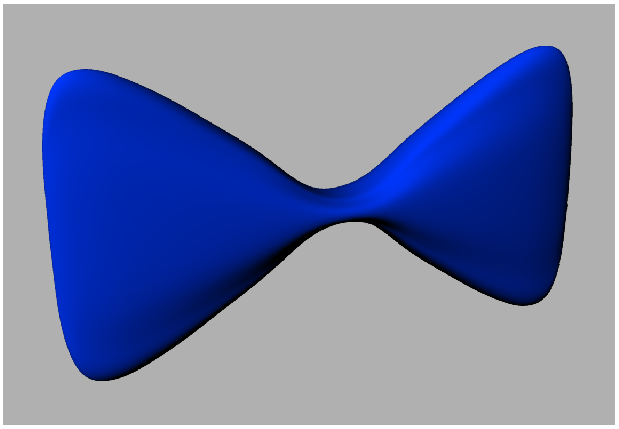


FIGURE 7.2: A bow tie geometry derived from an initial sphere geometry applying grammar rules with mathematical function primitives

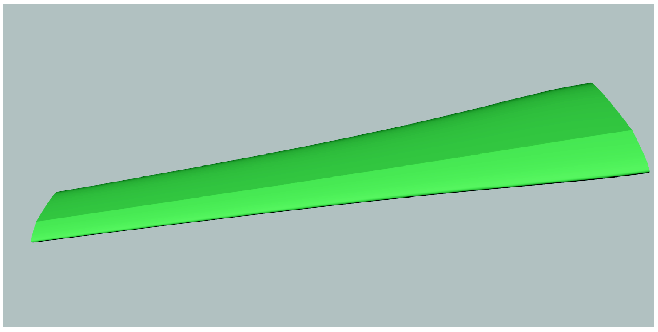


FIGURE 7.3: An air plane wing geometry derived from an initial sphere geometry applying grammar rules with mathematical function primitives

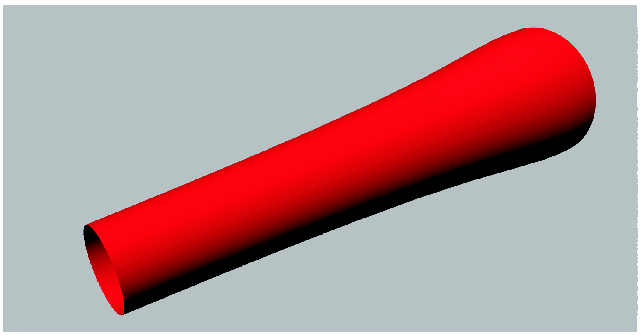


FIGURE 7.4: A tube geometry derived from an initial sphere geometry applying grammar rules with mathematical function primitives

functions

$$\begin{aligned} x(u,v) &= \frac{\cos(u)\cos(v)}{12} \\ y(u,v) &= \sin(u) \\ z(u,v) &= \cos(u)\sin(v)\frac{|\sin(u)|}{\cos(u)+\tan(0.3)}, \end{aligned} \tag{7.2}$$

where $-\pi/2 < u < \pi/2$, and $0 < v < 2\pi$.

Two other interesting surfaces synthesised using grammar rules applied to function primitives are an air-plane wing geometry shown in Fig. 7.3 represented by functions

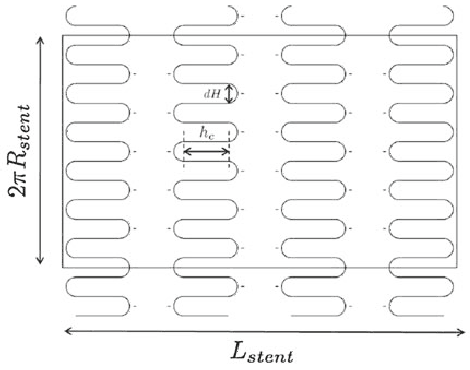
$$\begin{aligned} x(u,v) &= \cos(u)\frac{\cos(v)}{2} \\ y(u,v) &= \left(\frac{\tan(u)}{\cos(u)}\right)3 \\ z(u,v) &= \cos(u)\sin(v)v, \end{aligned} \tag{7.3}$$

where $-\pi/90 < u < \pi/3$ and $0 < v < 2\pi$ and a tube shown in Fig. 7.4 represented by functions

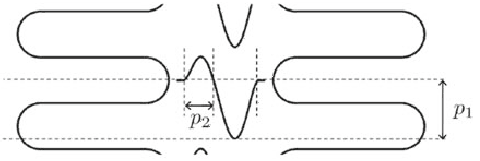
$$\begin{aligned} x(u,v) &= \cos(u)\cos(v) \\ y(u,v) &= \left(\frac{\tan(u)}{\cos(u)}\right)3 \\ z(u,v) &= \cos(u)\sin(v) \end{aligned} \tag{7.4}$$

where $-\pi/90 < u < \pi/3$ and $0 < v < 2\pi$.

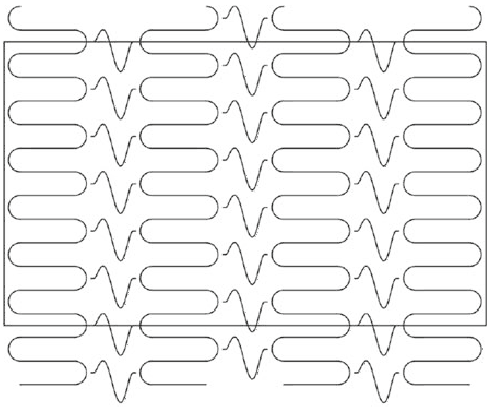
- Biomedical engineering In biomedical engineering, Coronary-artery stents are flexible pipe-type structures placed inside an artery to provide permanent support to its walls (Pant et al., 2011). The support is needed to allow the



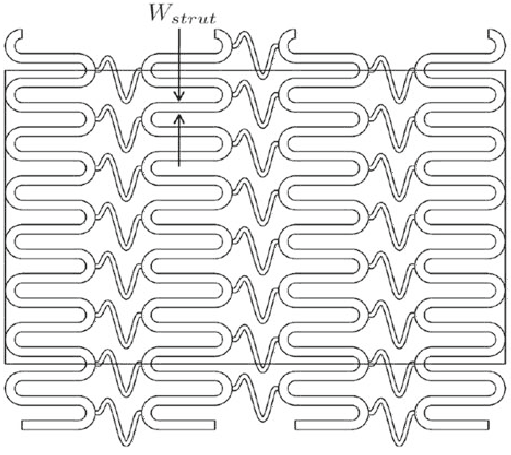
(a) The base shape for parameterization (h_c as the parameter)



(b) Parameterization for the link ($p1$ and $p2$ as parameters)



(c) Figure with links and the base structure



(d) Shape after offseting the curves(W_{strut} as the parameter)

(e) Final stent geometry

FIGURE 7.5: An illustration of design stages of Coronary-artery stent. The stent design consist of two curves: a base curve representing the base geometry and a link curve representing the connections between the base geometry. Of a design optimisation interest is to find the optimal shape of these two curves which meet several design objectives. From "Geometry parameterization and multidisciplinary constrained optimization of coronary stents", Pant et al., *Biomechanics and Modeling in Mechanobiology*, page 64. (Pant et al., 2011)

blood flow through otherwise clogged artery due to plaque accumulation known as Coronary-artery disease. The design optimisation of Coronary-artery stents requires consideration of structural as well as fluid dynamics performance. Of interest in design optimisation is the topology of stents (see Fig. 7.5). It is given by a complex network of strings along the length of the stent. Usually, this network is of periodical pattern but other configurations are also possible. The design description and optimisation framework could be used to synthesise a curve or a set of curves representing sting shapes, see Fig. 7.5. The string geometries thus obtained could be then seeded along the stent length is some pattern thus generating stent topology.

- Composite materials Another important aspect of structural optimisation is development of advanced materials, which weigh less and have improved mechanical properties. Nowadays, a very popular solution is to combine several layers of materials with different mechanical properties in a compact form. Such materials are known as composite materials. They find extensive application in a range of engineering problems encountered in aerospace, aeronautics, automotive, biomedicine, etc. The optimal geometrical configuration of material layers within a composite structure is equally important to selection of the material composition. Possible use of the automated SG to optimal geometry arrangement of composite material layers could be investigates. A specific BNF syntax would need to be developed which contains grammar rules for efficient arrangement of composite layers.

- Aerospace engineering The BNF syntax presented in Chapter 4 is a generic shape synthesis tool. Its application to shape optimisation of an airfoil geometry could be investigated. Arc primitives could be used for synthesis of two parametric piecewise curves describing an optimal airfoil geometry. The optimisation could be carried out without any major modifications to the algorithm presented in Chapter 4. The main difference would be that a CFD method would be used to obtain design performance objectives.

Appendix A

TABLE A.1: Design parameters of panel structure shown in Fig. 6.8(a)

Member by nodes see Fig. 6.8	Section area (a_i) mm^2
$(2, 13), (4, 15)$	$(16)^2$
$(2, 14), (4, 14)$	$(15.2)^2$
$(3, 13), (3, 15)$	$(12.4)^2$
$(9, 14)$	$(26)^2$
$(12, 13), (6, 15)$	$(14.4)^2$
$(10, 14), (8, 14)$	$(10.8)^2$
$(10, 13), (8, 15)$	$(18)^2$
Plate thickness (t) mm	2.12
Weight (f) kg	15.52
von Mises stress ($\sigma_{\max}$) MPa	125
Displacement (δ) mm	1.99

TABLE A.2: Design parameters of panel structure shown in Fig. 6.8(b)

Member by nodes see Fig. 6.8	Section area (a_i) mm^2
$(2, 13), (4, 15)$	$(22.4)^2$
$(2, 14), (4, 14)$	$(15.2)^2$
$(3, 13), (3, 15)$	$(14)^2$
$(9, 14)$	$(30.4)^2$
$(12, 13), (6, 15)$	$(16.8)^2$
$(10, 14), (8, 14)$	$(11.6)^2$
$(10, 13), (8, 15)$	$(21.6)^2$
Plate thickness (t) mm	2.56
Weight (f) kg	19.89
von Mises stress ($\sigma_{\max}$) MPa	122.81
Displacement (δ) mm	1.14

TABLE A.3: Design parameters of panel structure shown in Fig. 6.8(c)

Member by nodes see Fig. 6.8	Section area (a_i) mm^2
$(2,13),(4,15)$	$(27.2)^2$
$(2,14),(4,14)$	$(12)^2$
$(3,13),(3,15)$	$(17.2)^2$
$(9,14)$	$(33.6)^2$
$(12,13),(6,15)$	$(19.2)^2$
$(10,14),(8,14)$	$(12.4)^2$
$(10,13),(8,15)$	$(23.2)^2$
$(9,13),(9,15)$	$(10)^2$
Plate thickness (t) mm	3
Weight (f) kg	24.3
von Mises stress $(\sigma_{\max})$ MPa	94.56
Displacement (δ) mm	0.7

TABLE A.4: Design parameters of panel structure shown in Fig. 6.9(a)

(a) Node positions

Node	Position
2	$(211.6, 0.0)$
3	$(500, 0.0)$
4	$(788.4, 0.0)$
7	$(1000, 500)$
8	$(639.6, 500.0)$
10	$(360.4, 500.0)$
11	$(0, 1000)$
13	$(259.6, 192.4)$
14	$(500, 250)$
15	$(740.4, 192.4)$

(b) Section area by member

Member by nodes see Fig. 6.9(a)	Section area (a_i) mm^2
$(2,13),(4,15)$	$(18.8)^2$
$(3,14)$	$(23.6)^2$
$(11,13),(7,15)$	$(10.8)^2$
$(10,14),(8,14)$	$(14.4)^2$
$(10,13),(8,15)$	$(20.4)^2$
Plate thickness (t) mm	2.6
Weight (f) kg	16.18
von Mises stress $(\sigma_{\max})$ MPa	221.93
Displacement (δ) mm	1.9

TABLE A.5: Design parameters of panel structure shown in Fig. 6.9(b)

(a) Node positions

Node	Position
2	$(274.0, 0.0)$
3	$(500, 0.0)$
4	$(726.0, 0.0)$
6	$(1000.0, 110.8)$
7	$(1000, 500)$
8	$(702.0, 500.0)$
9	$(500, 500)$
10	$(298.0, 500.0)$
11	$(0, 1000)$
12	$(0.0, 110.8)$
13	$(259.6, 154.0)$
14	$(500, 250)$
15	$(740.4, 154.0)$

(b) Section area by member

Member by nodes see Fig. 6.9	Section area (a_i) mm^2
$(2, 13), (4, 15)$	$(28.0)^2$
$(3, 14)$	$(23.2)^2$
$(9, 14)$	$(21.2)^2$
$(11, 13), (7, 15)$	$(10.0)^2$
$(9, 13), (9, 15)$	$(15.6)^2$
$(10, 12), (6, 8)$	$(19.2)^2$
Plate thickness (t) mm	2.6
Weight (f) kg	19.17
von Mises stress $(\sigma_{\max})$MPa	130.42
Displacement (δ) mm	1.2

TABLE A.6: Design parameters of panel structure shown in Fig. 6.9(c)

(a) Node positions

Node	Position
1	$(0.0, 0.0)$
2	$(264.4, 0.0)$
3	$(500, 0.0)$
4	$(735.6, 0.0)$
5	$(1000.0, 0.0)$
6	$(1000.0, 192.4)$
7	$(1000, 500)$
8	$(740.4, 500.0)$
9	$(500, 500)$
10	$(259.6, 500.0)$
11	$(0, 1000)$
12	$(0.0, 192.4)$
13	$(288.4, 197.2)$
14	$(500, 250)$
15	$(711.6, 197.2)$

(b) Section area by member

Member by nodes see Fig. 6.9	Section area (a_i) mm^2
$(1, 13), (5, 15)$	$(19.2)^2$
$(2, 13), (4, 15)$	$(34.0)^2$
$(3, 14)$	$(22.8)^2$
$(9, 14)$	$(26.0)^2$
$(11, 13), (7, 15)$	$(14.8)^2$
$(9, 13), (9, 15)$	$(16.8)^2$
$(10, 12), (6, 8)$	$(18.0)^2$
Plate thickness (t) mm	3.08
Weight (f) kg	25
von Mises stress $(\sigma_{\max})$ MPa	94.56
Displacement (δ) mm	0.74

References

Adams D, Rohlf F, Slice D (2004) Geometric morphometrics: Ten years of progress following the "revolution". Italian Journal of Zoology 71(1):5–16

Arabas J, Michalewicz Z, Mulawka J (1994) GAVaPS – a Genetic Algorithm with Varying Population Size. In: Proceedings of the 1st IEEE Conference on Evolutionary Computation: IEEE World Congress on Computational Intelligence, IEEE, Orlando, USA, pp 73–78

Auger A, Bader J, Brockhoff D, Zitzler E (2009) Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. In: Proceedings of the 10th ACM SIGEVO workshop on Foundations of genetic algorithms, ACM, pp 87–102

Backus J (1959) The syntax and semantics of the proposed international algebraic language. In: Proceedings of the IFIP conference, ACM-GAMM, Zurich, Switzerland, pp 125–131

Backus J, Bauer F, Green J, Katz C, McCarthy J, Naur P, Perlis A, Rutishauser H, Samelson K, Vauquois B, et al. (1963) Revised report on the ALGOrithmic Language ALGOL 60. The Computer Journal 5(4):349–367

Banerjee P, Butterfield R (1981) Boundary element methods in engineering science, vol 17. McGraw-Hill, London, UK

Bendsøe MP, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Computer methods in applied mechanics and engineering 71(2):197–224

Blackith R, Reyment R (1971) Multivariate morphometrics. Academic Press, London, UK

Bookstein F (1984) A statistical method for biological shape comparisons. Theoretical Biology 107(3):475–520

Bookstein F (2002) Principal warps: Thin-plate splines and the decomposition of deformations. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(6):567–585

Brabazon A, O'Neill M (2006) Biologically inspired algorithms for financial modelling. Springer-Verlag, New York, USA

Castle T, Johnson C (2010) Positional effect of crossover and mutation in grammatical evolution. In: Genetic Programming, Springer, pp 26–37

Cervera E, Trevelyan J (2005a) Evolutionary structural optimisation based on boundary representation of NURBS. part I: 2D algorithms. Computers & Structures 83(23-24):1902–1916

Cervera E, Trevelyan J (2005b) Evolutionary structural optimisation based on boundary representation of NURBS. part II: 3D algorithms. Computers & Structures 83(23-24):1917–1929

Chen J, Shapiro V, Suresh K, Tsukanov I (2007) Shape optimization with topological changes and parametric control. International Journal for Numerical Methods in Engineering 71(3):313–346

Chen S, Torterelli D (1997) Three-dimensional shape optimization with variational geometry. Structural and Multidisciplinary Optimization 13(2):81–94

Chen Y, Bhaskar A, Keane A (2002) A parallel nodal-based evolutionary structural optimization algorithm. Structural and Multidisciplinary Optimization 23(3):241–251

Chomsky N (1956) Three models for the description of language. IRE Transactions on Information Theory 2(3):113–124

Clausen P, Pedersen C (2006) Non-parametric large scale structural optimization for industrial applications. In: Proceedings of the 3rd European Conference on Computational Mechanics, Solids, Structures and Coupled Problems in Engineering, Springer Netherlands, pp 482–482

Cook R, Malkus D, Plesha M (1989) Concepts and Applications of Finite Element Analysis, 3rd edn. John Wiley & Sons, New York, USA

Cramer N (1985) A representation for the adaptive generation of simple sequential programs. In: Proceedings of an International Conference on Genetic Algorithms and the Applications, pp 183–187

Deb K, Gulati S (2001) Design of truss-structures for minimum weight using genetic algorithms. Finite Elements in Analysis and Design 37(5):447–465

Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist Non-Dominated Sorting Genetic Algorithm for multi-objective optimization: NSGA-II. In: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, Springer, pp 849–858

Dryden I (2000) Statistical shape analysis in archaeology. In: Proceedings of the Workshop on Spatial Statistics in Archaeology

Dryden I, Mardia K (1998) Statistical shape analysis, 1st edn. John Wiley & Sons, Chichester, UK

DSSG (2011) The design synthesis and shape generation project. Retrieved on 15 March. 2011 from `http://www.engineering.leeds.ac.uk/dssg/`

Fonseca C, Fleming P (1993) Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, vol 423, pp 416–423

Fonseca C, Knowles J, Thiele L, Zitzler E (2005) A tutorial on the performance assessment of stochastic multiobjective optimizers. In: Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion, vol 216

Francone F, Conrads M, Banzhaf W, Nordin P (1999) Homologous crossover in genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, San Francisco, USA, vol 2, pp 1021–1026

Garner SW, et al. (2011) Design with vision. Retrieved on 23 Dec. 2011 from `http://design.open.ac.uk/DV/`

Goldberg D (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Harlow, England

Goldberg D, Samtani M (1986) Engineering optimization via genetic algorithm. In: Proceedings of the Conference on Electronic Computation, ASCE, pp 471–482

Gottlieb J, Julstrom B, Raidl G (2000a) Characterizing locality in decoder-based EAs for the multidimensional knapsack problem. In: Artificial Evolution, Springer, pp 38–52

Gottlieb J, Julstrom B, Raidl G (2000b) The effects of locality on the dynamics of decoder-based evolutionary search. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 283–290

Gottlieb J, Julstrom B, Raidl G (2001) Prüfer numbers: A poor representation of spanning trees for evolutionary search. In: In, Citeseer

Gunz P, Mitteroecker P, Bookstein F (2005) Semilandmarks in three dimensions. Modern morphometrics in physical anthropology pp 73–98

Hajela P, Lee E (1995) Genetic algorithms in truss topological optimization. Solids and Structures 32(22):3341–3357

Hajela P, Lee E, Lin C (1993) Genetic algorithms in structural topology optimization. Topology Optimization of Structures pp 117–133

Henning R (2008) Principal Component Analysis (PCA) & NIPALS algorithm. Retrieved on Dec. 2008 from `http://folk.uio.no/henninri/pca_module/pca_nipals.pdf`

Hicklin J (1986) Application of the genetic algorithm to automatic program generation. PhD thesis, University of Idaho

Horn J, Nafpliotis N, Goldberg D (1994) A niched pareto genetic algorithm for multiobjective optimization. In: Proceedings of the 1st IEEE Conference on Evolutionary Computation: IEEE World Congress on Computational Intelligence, IEEE, pp 82–87

Hsu W, Hughes J, Kaufman H (1992) Direct manipulation of Free-Form Deformations. In: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, ACM, pp 177–184

Hu S, Zhang H, Tai C, Sun J (2001) Direct manipulation of FFD: Efficient explicit solutions and decomposible multiple point constraints. The Visual Computer 17(6):370–379

Huang X, Xie Y (2008) A new look at ESO and BESO optimization methods. Structural and Multidisciplinary Optimization 35(1):89–92

Hugosson J, Hemberg E, Brabazon A, OŃeill M (2010) Genotype representations in grammatical evolution. Applied Soft Computing 10(1):36–43

ISO (1996) IEC 14977: 1996 (E). Information technology – Syntactic metalanguage – Extended BNF

Jadaan OA, Rajamani L, Rao C (2008) Non-dominated Ranked Genetic Algorithm for solving multi-objective optimization problems: NRGA. Theoretical and Applied Information Technology 4(1):61–68

Jin N, Mokhtarian F (2006) Human motion recognition based on statistical shape analysis. In: Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, IEEE, pp 4–9

Katre S, et al. (1989) Astādhyāyī of Pānini. Motilal Banarsidass Publ.

Kawamura H, Ohmori H, Kito N (2002) Truss topology optimization by a modified genetic algorithm. Structural and Multidisciplinary Optimization 23(6):467–473

Keane A, Nair P (2005) Computational approaches for aerospace design. The pursuit of Excellence, John Wiley & Sons, Chichester, UK

Keane AJ, et al. (2008) Optionsmatlab users' guide. Retrieved on 22 Aug. 2009 from `http://www.blog.ses.soton.ac.uk/OptionsMatlab`

Kendall D (1989) A survey of the statistical theory of shape. Statistical Science 4(2):87–99

Kendall D, Barden D, Carne T, Le H (1999) Shape and shape theory, vol 11. John Wiley & Sons

Kirsch U (1989) Optimal topologies of truss structures. Computer methods in applied mechanics and engineering 72(1):15–28

Knuth D (1964) Backus normal form vs. Backus Naur form. Communications of the ACM 7(12):735–736

Koza J (1994) Genetic programming as a means for programming computers by natural selection. Statistics and Computing 4(2):87–112

Koza J, Poli R (2005) Genetic Programming. In: Search Methodologies, Springer USA, pp 127–164

Lamousin H, WN Jr W (2002) NURBS-based Free-Form Deformations. Computer Graphics and Applications 14(6):59–65

Malyna D, Duarte J, Hendrix M, van Horck F (2006) Multi-objective optimization of power converters using genetic algorithms. In: Proceedings of the International Symposium on Power Electronics, Electrical Drives, Automation and Motion, IEEE, pp 713–717

McCormack J, Cagan J (2002a) Designing inner hood panels through a shape grammar based framework. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 16(04):273–290

McCormack J, Cagan J (2002b) Supporting designers' hierarchies through parametric shape recognition. Environment and Planning B: Planning & Design 29(6):913–932

McCormack J, Cagan J (2008) Parametric shape grammar interpreter. Patent US 7,415,156 B2, Appl. 10/350,428

McCormack J, Cagan J, Vogel C (2004) Speaking the Buick language: Capturing, understanding, and exploring brand identity with shape grammars. Design Studies 25(1):1–29

Mckay R, Hoai N, Whigham P, Shan Y, OŃeill M (2010) Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines 11(3-4):365–396

Meske R, Sauter J, Schnack E (2005) Nonparametric gradient-less shape optimization for real-world applications. Structural and Multidisciplinary Optimization 30(3):201–218

Meske R, Lauber B, Schnack E (2006) A new optimality criteria method for shape optimization of natural frequency problems. Structural and Multidisciplinary Optimization 31(4):295–310

Monteiro L, Bonato V, Reis SD (2005) Evolutionary integration and morphological diversification in complex morphological structures: Mandible shape divergence in spiny rats (rodentia, echimyidae). Evolution & Development 7(5):429–439

Mühlenbein H, Schomisch M, Born J (1991) The parallel genetic algorithm as function optimizer. Parallel Computing 17(6-7):619–632

Murphyand J, O'Neill M, Carr H (2009) Exploring grammatical evolution for horse gait optimisation. Genetic Programming pp 183–194

Nasuf A, Bhaskar A, Keane A (2011) Multi-objective optimisation using grammatical evolution. In: Proceedings of the Anniversary Scientific Conference: 40 Years Department of Industrial Automation, UCTM, Sofia, Bulgaria, pp 20–25

Ohsaki M (1995) Genetic algorithm for topology optimization of trusses. Computers & Structures 57(2):219–225

O'Neill M (2011) Grammatical Evolution. Retrieved on 05 Feb. 2011 from `http://www.grammatical-evolution.org/pubs`

O'Neill M, Ryan C (1998) Grammatical evolution: A steady state approach. In: Proceedings of the 2nd International Workshop on Frontiers in Evolutionary Algorithms, pp 419–423

O'Neill M, Ryan C (2002) Grammatical evolution. IEEE Transactions on Evolutionary Computation 5(4):349–358

O'Neill M, Ryan C (2003) Grammatical Evolution: Evolutionary automatic programming in an arbitrary language. Genetic programming, Kluwer Academic Publishers

O'Neill M, Collins J, Ryan C (2000) Automatic generation of robot behaviours using grammatical evolution. In: Proceedings of the AROB 2000: The 5th International Symposium on Artificial Life and Robotics

O'Neill M, Brabazon A, Ryan C, Collins J (2001a) Evolving market index trading rules using grammatical evolution. Applications of evolutionary computing pp 343–352

O'Neill M, Ryan C, Keijzer M, Cattolico M (2001b) Crossover in grammatical evolution: The search continues. Genetic Programming pp 337–347

O'Neill M, Swafford J, McDermott J, Byrne J, Brabazon A, Shotton E, McNally C, Hemberg M (2009) Shape grammars and grammatical evolution for evolutionary design. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, ACM, pp 1035–1042

O'Neill M, McDermott J, Swafford J, , Byrne J, Hemberg E, Brabazon A, Shotton E, McNally C, Hemberg M (2010) Evolutionary design using grammatical evolution and shape grammars: Designing a shelter. Design Engineering 3(1):4–24

Pant S, Bressloff N, Limbert G (2011) Geometry parameterization and multidisciplinary constrained optimization of coronary stents. Biomechanics and Modeling in Mechanobiology 11(1-2):1–22

Pearson K (1926) On the coefficient of racial likeness. Biometrika 18(1-2):105–117

Perry M, Koh C, Choo Y (2006) Modified genetic algorithm strategy for structural identification. Computers & Structures 84(8-9):529–540

Piegl L, Tiller W (1997) The NURBS book, 2nd edn. Springer-Verlag, New York, USA

Prats M, Garner S (2006) Observations on ambiguity in design sketches. Contemporary Drawing Research pp 1–20

Prats M, Lim S, Jowers I, Garner S, Chase S (2009) Transforming shape in design: Observations from studies of sketching. Design Studies 30(5):503–520, DOI 10.1016/j.destud.2009.04.002

de la Puente A, Alfonso R, Moreno M (2002) Automatic composition of music by means of grammatical evolution. In: Proceedings of the Conference on Array Processing Languages: Lore, Problems and Applications, ACM, pp 148–155

Querin O (1997) Evolutionary structural optimisation: stress based formulation and implementation. PhD thesis, Department of Aeronautical Engineering, University of Sydney, Australia

Rajeev S, Krishnamoorthy C (1992) Discrete optimization of structures using genetic algorithms. Structural Engineering 118(5):1233–1250

Ringertz U (1985) On topology optimization of trusses. Engineering optimization pp 209–218

Rohlf F, Marcus L (1993) A revolution in morphometries. Trends in Ecology and Evolution 8(4):129–132

Rohlf FJ (2008) Tpssplin software package to apply thin-plate spline deformation. Retrieved on 14 Sep. 2008 from `http://life.bio.sunysb.edu/morph/`

Rothlauf F, Goldberg D (1999) Tree network design with genetic algorithms – An investigation in the locality of the prüfernumber encoding. In: Late Breaking Papers at the Genetic and Evolutionary Computation Conference, pp 238–244

Rothlauf F, Goldberg DE (2000) Prüfer numbers and genetic algorithms: A lesson on how the low locality of an encoding can harm the performance of GAs. In: Parallel Problem Solving from Nature PPSN VI, Springer, pp 395–404

Rothlauf F, Oetzel M (2006) On the locality of grammatical evolution. In: Proceedings of the 9th European conference on Genetic Programming, Springer-Verlag, Berlin, Heidelberg, EuroGP'06, pp 320–330, DOI 10.1007/11729976_29, URL http://dx.doi.org/10.1007/11729976_29

Ryan C, Collins J, O'Neill M (1998) Grammatical evolution: Evolving programs for an arbitrary language. In: Proceedings of the 1st European Workshop on Genetic Programming

Scowen R (1998) Extended bnf-a generic base standard. Tech. rep., Technical report, ISO/IEC 14977

Sederberg T, Parry S (1986) Free-Form Deformation of solid geometric models. ACM SIGGRAPH Computer Graphics 20(4):151–160

Sharma C, Purohit K (2006) Theory of Mechanisms and Machines. Prentice-Hall of India, New Delhi, India

Shlens J (2009) A tutorial on principal component analysis. Systems Neurobiology Laboratory, University of California at San Diego

Stiny G, Gips J (1971) Shape grammars and the generative specification of painting and sculpture. In: Proceedings of the IFIP Congress, pp 125–135

Tang P, Chang K (2001) Integration of topology and shape optimization for design of structural components. Structural and Multidisciplinary Optimization 22(1):65–82

Thompson D (1917) On growth and form, Cambridge University Press, London, UK, chap XVII – On the theory of transformations, or the comparison of related forms, p 748

Topping B (1983) Shape optimization of skeletal structures: A review. Structural Engineering 109(8):1933–1951

Voutchkov I, Keane A, Fox R (2006) Robust structural design of a simplified jet engine model, using multiobjective optimization. In: Proceedings of the 11th Conference on Multidisciplinary Analysis and Optimization, American Institute of Aeronautics and Astronautics

Whigham P, et al. (1995) Grammatically-based genetic programming. In: Proceedings of the workshop on genetic programming: from theory to real-world applications, vol 16, pp 33–41

Wirth N (1977) What can we do about the unnecessary diversity of notation for syntactic definitions? Communications of the ACM 20(11):822–823

Young V, Querin O, Steven G, Xie Y (1999) 3D and multiple load case bi-directional evolutionary structural optimization (BESO). Structural and Multidisciplinary Optimization 18(2):183–192

Zhigljavsky A (1991) Theory of global random search. Kluwer Academic Publishers, Dordrecht, Netherlands

Zhou M, Rozvany G (1991) The COC algorithm, Part II: Topological, geometrical and generalized shape optimization. Computer Methods in Applied Mechanics and Engineering 89(1-3):309–336

Zhou Y, Han R (2005) A genetic algorithm with elite crossover and dynastic change strategies. Advances in Natural Computation pp 269–278

Zienkiewicz O, Taylor R, Zhu J (2005) The Finite Element Method: Its basis and fundamentals, vol 1. Elsevier Butterworth-Heinemann, London, UK

Zitzler E, Laumanns M, Thiele L (2002) SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Proceedings of the Conference on Evolutionary Methods for Design, Optimization and Control, vol 4