DESIGN OPTIMIZATION OF TRUSS STRUCTURES USING GENETIC ALGORITHMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DİLEK ÜNALMIŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
AEROSPACE ENGINEERING

SEPTEMBER 2012

Approval of the thesis:

DESIGN OPTIMIZATION OF TRUSS STRUCTURES USING GENETIC ALGORITHMS

Submitted to **DİLEK ÜNALMIŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Aerospace Engineering, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Ozan Tekinalp _____
Head of Department, **Aerospace Engineering**

Prof. Dr. Altan Kayran _____
Supervisor, **Aerospace Engineering Dept., METU**

Examining Committee Members

Asst. Prof. Dr. Demirkan Çoker _____
Aerospace Engineering Dept., METU

Prof. Dr. Altan Kayran _____
Aerospace Engineering Dept., METU

Asst. Prof. Dr. Melin Şahin _____
Aerospace Engineering Dept., METU

Asst. Prof. Dr. Ercan Gürses _____
Aerospace Engineering Dept., METU

Dr. Muvaffak Hasan _____
Turkish Aerospace Industries, Inc.

Date: 20/09/2012

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Dilek ÜNALMIŞ

Signature :

ABSTRACT

DESIGN OPTIMIZATION OF TRUSS STRUCTURES USING GENETIC ALGORITHMS

Ünalmış, Dilek

M.Sc., Department of Aerospace Engineering

Supervisor: Prof. Dr. Altan Kayran

September 2012, 107 pages

Design optimization of truss structures is a popular topic in aerospace, mechanical, civil, and structural engineering due to benefits to industry. Common design problem for the structures is the weight minimization. Especially in aerospace engineering the minimization of the weight of the total structure gets the highest importance in the design.

This study focuses on the design optimization of 2D and 3D truss structures. The objective function is the total mass of the structure which is subjected to stress and nodal displacement constraints. To optimize the design, Genetic Algorithm (GA) is preferred due to its efficiency in dealing with problems with discrete design variables as in the case of truss structures. This technique yields more realistic results than linear programming methods.

In the thesis, a finite element code is developed for the analysis of planar and space truss structures. The developed finite element solver is coupled with a genetic algorithm optimization code which is also developed as a part of the thesis study. Different truss optimization case studies are performed to demonstrate the performance of the finite element solver and the genetic algorithm optimization code that are developed. It is shown that with the use of adaptive penalty function employing scaled fitnesses, the arbitrariness issue of the factor multiplying the error term in the augmented fitness function can be resolved. It is also shown that significant weight reduction can

be achieved by employing shape optimization together with size optimization compared to pure size optimization.

Keywords: Genetic Algorithms (GAs), Structural Optimization, Shape Optimization, Size optimization, Truss Systems Design, Finite Element Method

ÖZ

GENETİK ALGORİTMA KULLANARAK KAFES SİSTEMLERİNİN TASARIM OPTİMİZASYONU

Ünalmış, Dilek

Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Altan Kayran

Eylül 2012, 107 sayfa

Kafes sistemlerinin tasarım optimizasyonu, havacılık ve uzay, makine, inşaat ve yapısal mühendislikte, endüstriye faydalarından dolayı popüler bir konudur. Yapılar için ortak tasarım problemi ağırlık minimizasyonudur. Özellikle havacılık ve uzay mühendisliğinde toplam yapı ağırlığının azalımı tasarımda en önemli yeri alır.

Bu çalışma 2 boyutlu ve 3 boyutlu kafes sistemlerinin tasarım optimizasyonunu kapsar. Amaç fonksiyonu, gerilme ve düğüm noktalarının deplasman sınırlayıcılarına tabi olan toplam yapı ağırlığıdır. Tasarımı optimize etmek için kafes sistemlerinde de olduğu gibi ayrık tasarım değişkenlerine sahip problemlerin çözümündeki etkinliğinden dolayı Genetik Algoritma (GA) tercih edilmektedir. Bu teknik doğrusal programlama yöntemlerinden daha gerçekçi sonuçlar ortaya koyar.

Bu tezde düzlemsel ve uzay kafes yapılarının analizi için sonlu eleman kodu geliştirilmiştir. Geliştirilmiş olan sonlu eleman çözücüsü bu tezin bir parçası olarak geliştirilen bir genetik algoritma optimizasyon kodu ile birleştirilmiştir. Geliştirilmiş olan çözücü ve optimizasyon kodlarının performansını göstermek için farklı kafes yapılarının optimizasyon çalışmaları yapılmıştır. Uyarlanabilir ceza fonksiyonunun kullanımıyla, orantılanmış form fonksiyonunun faktör çarpımıyla ilgili belirsizlik konusunun çözüldüğü gösterilmektedir. Bununla birlikte boyut optimizasyonu ile birlikte yapılan şekil optimizasyonu, boyut

optimizasyonun yalnız yapıldığı durumla kıyaslandığında, önemli miktarda ağırlık azalımınım elde edilebildiği gösterilmiştir.

Anahtar Kelimeler: Genetik Algoritmalar, Yapı Optimizasyonu, Boyut Optimizasyonu, Şekil Optimizasyonu, Kafes Sistemlerinin Tasarımı, Sonlu Eleman Metodu

To my son Umut

ACKNOWLEDGEMENTS

I would like to express my respectfulness and special thanks to my supervisor Prof. Dr. Altan Kayran for his guidance and support throughout the completion of this thesis.

I would like to thank my company, Turkish Aerospace Industries (TAI) and my colleagues for their support and encouragement.

Also, I would like to express heartfelt thanks to my mom for her blessings, to my husband for his endless understanding and great moral support and to my son for his great heartening.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Structural design of trusses deals with systems comprised from a set of structural members. These members are bar elements, connected by pinned or fixed joints. Common structures include truss are bridges, frame buildings, race car, airplane space frames, crane arms, and power line truss towers.

Truss elements carry only axial loads due to their pin connections at nodes which are only allowed translational degrees of freedom. Only a cross sectional area (A) is needed to define its geometry due to the limitation of axial load.

Structural optimization has become important for engineers and designers in recent years. After the usage of high performance computing systems, optimization in engineering became a commonly used design tool. Structures are becoming lighter, stronger, and cheaper as the industry adopts higher forms of optimization. This type of problem solving and product improvement is now indispensable part of the design process in today's engineering industry.

The topic of optimization has its mathematical roots dating back to the 1670's with the introduction of differential calculus. Its primary purpose is to find the best result to a problem given a set of circumstances. It wasn't until the early 1950's that computer-based optimization launched itself into the engineering industry. This was due to the fact that the topic lends itself to numerical computation, which is the one task in which computers have superiority over humans. Programmers immediately began introducing new optimization methods such as nonlinear programming, unconstrained optimization, and multi-objective optimization.

One of the ways of numerical optimization methods is evolutionary computation. This category of optimization includes the genetic algorithm. These forms of computation have opened many possibilities never before achievable in optimization. The first work utilizing the genetic algorithms was done by evolving state machines in the 1960's [1].

Genetic algorithm applies Darwin's major principles of evolution to artificial systems. Main characteristics of this method are random actions, trial and error, survival of the fittest to evolve solutions to optimization problems and recombining parts of previous good solutions. It can be applied to a wide variety of problems which makes it very powerful tool for engineers and designers.

Complex structures with high variable interactions become difficult to optimize. Because of these interactions, classical optimization methods can produce results closer to the optimal solution. Genetic algorithms are best for handling global optimization problems with many local optima in a non-continuous fitness landscape.

1.1 Objective of the Thesis

The objective of this thesis is to determine the minimum mass of the truss structures by means of size and shape optimization. Typically the optimization problem has stress and displacement constraints. These problems deal with mixed continuous and discrete search spaces, which can create solution landscapes which are non-smooth and deceptive. To overcome this difficulty in finding the optimum solution of such a complex problem, as the optimizer genetic algorithm is chosen. Because of searching solution in a wide range of intervals, genetic algorithm has greater chance to reach to optimum result.

In this thesis, a genetic algorithm optimization code is developed to make size and shape optimization of the 2D and 3D truss structures. The developed GA provides tailoring the code to the specific needs of to the problem by being able to interfere at each step of the algorithm. At each generation, the GA code gives solutions to the problem which is equal to the population size. To perform the analysis of the truss structures, GA code is coupled with a finite element solver, which is also developed in this thesis. By this way, the

algorithm becomes more dynamic eliminating interface problems to integrate a professional analysis tool to the optimizer. Both codes are developed in MATLAB to allow the use of ready MATLAB functions in the calculations.

In the thesis, two example problems, 10-bar and 25-bar truss structures, from the literature, are studied to show the validity of the genetic algorithm and finite element codes developed. Also the codes developed are used to optimize a real engineering problem, the tailcone truss structure of the helicopter “Aerospatiale SA-318C (Alouette II)”.

1.2 Scope of the Thesis

The scope of this work is to develop a solver-optimization code to make size and shape optimization for 2D and 3D truss structures by using genetic algorithm.

1.3 Outline of the Thesis

In Chapter 2, the details of the developed 2D and 3D finite element code for truss structures are given.

In Chapter 3, a review of Genetic Algorithm based optimization is made, and the developed genetic algorithm based optimization code is introduced. Special emphasis is given to the use of adaptive penalty function employing scaled fitnesses. It is shown that the arbitrariness issue of the factor multiplying the error term in the augmented fitness function can be resolved with the use of the adaptive penalty function through the scaled fitnesses. In addition, definition of the optimization problem for the simultaneous shape and size optimization is described in detail.

In Chapter 4, applications of design optimization are given. Case studies are performed for verification purposes, and shape and size optimizations are performed for different truss structures. In this chapter, mass optimization of the tailcone truss structure of the helicopter “Aerospatiale SA-318C (Alouette II)” is also presented

In Chapter 5, concluding remarks are given.

In Appendix A, B and C, the detail inputs of the developed finite element code and genetic algorithm are given for the studies made in in Chapter 4.

1.4 Literature Survey

To get comprehension on the structural optimization especially related with the truss structures and Genetic Algorithm (GA), a detail literature survey has been made. Since the literature on the genetic algorithm is vast, in the literature survey some sample studies are highlighted. Sandıkcı [4] published a paper giving information about basic definitions, the details of the operators used in genetic algorithm. Also the comparison of the genetic algorithm with the other optimization methods is made. Dianati, Song and Treiber [20] represented the Genetic Algorithms and Evolution Strategies in their paper. The history, mathematical background and theory, and applications of these evolutionary algorithms are examined. Charbonneau [21] gave distinction between local and global optimization. After introducing general idea of the genetic algorithm, Charbonneau compared the performance of the GA in finding global optimum with hill climbing in his technical note. Said [17] published a paper giving information about GAs and their applications. In his paper, the basic concepts and functionality of genetic computation has been described. In 2004, McCall [2] gave general structure of the GA by describing how to construct GA and the theoretical approaches to genetic algorithms.

After performing literature survey on the concept of GAs, application of genetic algorithm on the optimization of the structures has been investigated. Throughout this investigation, it is observed that different concepts and definitions have been applied on the structural optimization with GA. Different ways of defining the objective function with the given structural and design constraints were especially examined. Rajeev and Krishnamoorthy [9] presented optimization of structural systems with discrete variables with a simple GA. They used penalty-based transformation method in their work. In 1994, Coello [5] proposed the use of genetic algorithms for the problems with discrete search space in his paper. Erbatur, Hasançebi, Tütüncü and Kılıç [14] reported the development of the optimization of discrete design of planar and space structures composed of one-dimensional elements with a computer-based systematic approach in their paper. Tong and Liu [12] presented an

optimization procedure for the weight optimization of the truss structures, with discrete design variables, subjected to constraints on stresses, natural frequencies and frequency responses. In 2000, Gil and Andreu [16] presented a method for the optimization of the shape and cross-sections of a plane truss structure under the stress and geometrical constraints in their paper. The method includes a new approach of merging these problems. Croce, Ferreira and Lemonge [3] proposed in their paper a genetic algorithm for weight optimization of industrial buildings evolving the structural configuration in which shape, topology and the number of the truss elements are allowed to change during the optimization process. Auer [1] proposed a customized genetic algorithm developed to aid in the structural design process for size and shape optimization in his thesis. Toğan, Seyhun and Daloğlu [10] compared their previously coded and tested algorithm based on Genetic Algorithms (GA) to solve the optimization problems in structural engineering with MATLAB Genetic Algorithm Tool. In 2006, Taşkınoğlu [18] proposed a design procedure incorporating GA for the structural design optimization in his thesis. Kutay [19] presented a genetic algorithm code optimizing the stacking sequence of a composite pressure vessel in his thesis. Sun, Li, Zheng, Zhang and Hou [13] proposed a hybrid genetic algorithm based on relative difference quotient method and improved genetic algorithm to deal with the shape optimization of the truss structure. Hultman [6] developed a genetic optimization algorithm for weight minimization of steel trusses under the constraints regarding material strength and buckling stability.

From this examining, it is realized that the main contribution to the genetic algorithm is made by the adaptive approaches. By the adaptive approaches, one can control the algorithm progress; at the same time take the advantage of the non-deterministic characteristics of the GAs. Nanakorn and Meesomklin [11] published a paper proposing the adaptive penalty function method by which the penalty factor can be adjusted during the evaluation, providing the desired degree of penalty. In 2006, Toğan, Seyhun and Daloğlu [15] discussed the adaptive approach in genetic algorithms by testing the effects of some improvements in the penalty function, mutation and crossover on the performance of GAs. In this thesis, the algorithm is structured as a simple GA but involving adaptive penalty approach referred in reference [11].

CHAPTER 2

DEVELOPMENT OF A 2D&3D FINITE ELEMENT CODE FOR TRUSS STRUCTURES

2.1 Truss Structures

Truss can be called as “discrete element” structure. It consists of individual bar elements. So there is no need for dividing the continuum into appropriate elements and idealizing the behavior of each element, which are the important finite element processes [8].

Each bar of a truss is assumed to be uniform, linearly elastic, pin-connected to nodes at its ends, and axially loaded. This also means that a truss node is only allowed translational degrees of freedom.

A truss element needs only a cross sectional area to define its geometry due to the axial load limitation, and its length is determined by the position of its end nodes.

A three-dimensional truss element has two local degrees of freedom and six global degrees of freedom, with three translational degrees of freedom at each end of the element. Figure 2.1 shows a three-dimensional truss element with its local and global coordinate systems, degrees of freedom, and allowable forces. The black capital symbols represent global objects, while gray lower case symbols represent local objects. It can be seen that a truss element has only one local coordinate axis (x) originating from one node and extending through the length of the element. The only forces (f1,f2) and displacements (u1,u2) allowed in this local system lie along the axis of the element, and the element has two degrees of freedom. The global coordinate system (X,Y,Z) that is used in the structural analysis then causes each local

object to be broken into three equivalent global components. It is then shown that the three-dimensional truss element has six global degrees of freedom, with one for each global coordinate at each end of the element [1].

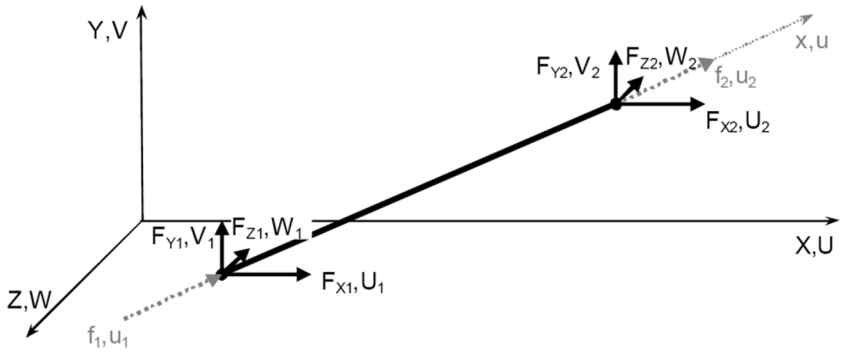


Figure 2.1. Truss element displaying local and global coordinate systems

2.2 Development of Finite Element Code

The developed finite element code for structural analysis can handle 2D and 3D truss structures. Boundary conditions should be placed on the nodes, and external loads are applied only at nodes. In addition to the external loads, finite element code also allows weight and thermal loading.

The principal steps of linear static stress analysis by the finite element method are listed below [8].

1. Input and Initialization

2. Computation of element properties; element stiffness matrices and element load vectors

3. Assembly of the structural matrices; structural stiffness matrix and structural load vector

4. Calculation of nodal displacements

5. Postprocessing; calculation of element strains and stresses

2.2.1 Input and initialization

In the finite element method, the number of nodes and elements, nodal coordinates, structure node numbers of each element, material properties, temperature changes, mechanical loads, and boundary conditions are given as input.

Finite element method starts with reserving storage space for structure arrays [K] and {R} to null arrays. [K] is the structural stiffness matrix. Following statement gives the physical meaning of [K]: "The j^{th} column of [K] is the vector of loads that must be applied to nodal degrees of freedom (d.o.f.) in order to maintain the deformation state associated with unit value of d.o.f. *j* while all other nodal d.o.f. are zero" [8].

[K] is a *nXn* dimensional matrix, where n is the number of d.o.f. of the structure meaning that to define the deformed configuration of the structure uniquely, n independent quantities are needed. For example, in a plane truss, n is d.o.f. of each node, which is 2, times number of nodes allowed to displace [8].

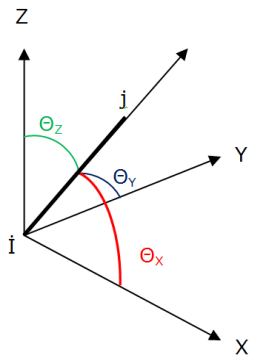
To manage boundary conditions, destination array (ID array), is initialized and converted to a table of equation numbers. ID array is *aXb* matrix, where *a* is the maximum d.o.f of each node and *b* is the number of nodes in the structure. It is filled with the numbers indicating the locations in [K] to which element stiffness coefficients k_{ij} are to be assigned [8].

2.2.2 Computation of the Element Properties

For each element, element stiffness matrix [k] and element load vector $\{r_e\}$ are computed.

2.2.2.1 Element Stiffness Matrix for 3D Space Trusses

In the calculation of the stiffness matrix for the 3D space trusses, direction cosines, the cosines of the angles between the vector and the three coordinate axes, are used as shown in Figure 2.2.



$C_x = \cos \theta_x$

$C_y = \cos \theta_y$

$C_z = \cos \theta_z$

Figure 2.2. Direction Cosines

In the calculation of the element stiffness matrix of the truss element, direct method is used. In the direct method, a degree of freedom is given a displacement while all the other degrees of freedom are kept fixed and nodal forces associated with such a displacement field is calculated. When the applied displacement is factored out, the resulting nodal force vector constitutes the column vector of the element stiffness matrix corresponding to the degree of freedom which is given a displacement.

For instance, referring to Figure 2.3, the displacement along element IJ due to displacement Uix is calculated as

$$e = U_{ix} \cos \theta_x \rightarrow e = U_{ix} C_x \qquad (2.1)$$

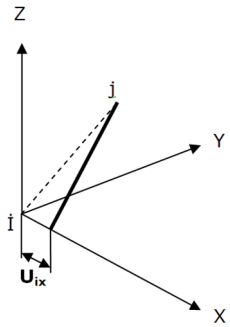


Figure 2.3. The displacement at node i in the x direction; U_{ix}

In Eq.(2.1), Uix is the displacement of node i in the global x direction.

The force acting on the element IJ due to small displacement Uix is then given by

$$F = ke = \frac{AE}{L} U_{ix} C_x \tag{2.2}$$

where axial stiffness of a truss element is given by AE/L with A representing the cross-sectional area, E is the Young's modulus and L is the length of the truss element IJ.

The components of the force F acting on node i in the X, Y and Z directions are calculated from Eqs.(2.3-2.5).

$$P_{ix} = FC_x = \frac{AE}{L} U_{ix} C_x^2 \tag{2.3}$$

$$P_{iy} = FC_y = \frac{AE}{L} U_{ix} \mathrm{C_x C_y} \tag{2.4}$$

$$P_{iz} = FC_z = \frac{AE}{L} U_{ix} \mathrm{C_x C_z} \tag{2.5}$$

The components of the forces acting at node J in the X, Y and Z directions are then calculated by requiring the equilibrium of the element IJ. Nodal forces at node j are given by Eqs.(2.6-2.8)

$$P_{jx} = -P_{ix} = -FC_x = -\frac{AE}{L} U_{ix} C_x^2 \tag{2.6}$$

$$P_{jy} = -P_{iy} = -FC_y = -\frac{AE}{L} U_{ix} \mathrm{C_x C_y} \tag{2.7}$$

$$P_{jz} = -P_{iz} = -FC_z = -\frac{AE}{L} U_{ix} \mathrm{C_x C_z} \tag{2.8}$$

Finally, the first column of the element stiffness matrix related to the displacement Uix is obtained. After factoring out the displacement Uix, Eq. (2.9) gives the column of the stiffness matrix associated with the nodal

displacement Uix. That is, the elements of the stiffness matrix, given by Eq.(2.9) are only multiplied by the nodal displacement Uix.

$$\begin{Bmatrix} P_{ix} \\ P_{iy} \\ P_{iz} \\ P_{jx} \\ P_{jy} \\ P_{jz} \end{Bmatrix}_{(1)} = \frac{AE}{L} U_{ix} \begin{Bmatrix} C_x^2 \\ \mathrm{C_x C_y} \\ \mathrm{C_x C_z} \\ -C_x^2 \\ -\mathrm{C_x C_y} \\ -\mathrm{C_x C_z} \end{Bmatrix} \tag{2.9}$$

In a similar manner, the second and third column of the element stiffness matrix is calculated by applying displacements U_{iy} and U_{iz} in the y and z directions, respectively.

The second column of the element stiffness matrix related to the displacement Uiy, shown in Figure 2.4,is is given by Eq. (2.10).

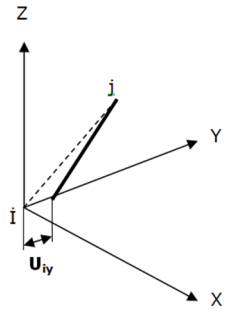


Figure 2.4. The displacement at node i in y direction; U_{iy}

$$\begin{Bmatrix} P_{ix} \\ P_{iy} \\ P_{iz} \\ P_{jx} \\ P_{jy} \\ P_{jz} \end{Bmatrix}_{(2)} = \frac{AE}{L} U_{iy} \begin{Bmatrix} \mathrm{C_x C_y} \\ C_y^2 \\ \mathrm{C_y C_z} \\ -\mathrm{C_x C_y} \\ -C_y^2 \\ -\mathrm{C_y C_z} \end{Bmatrix} \tag{2.10}$$

The third column of the element stiffness matrix related to the displacement Uiz, shown in Figure 2.5, is given by Eq. (2.11)

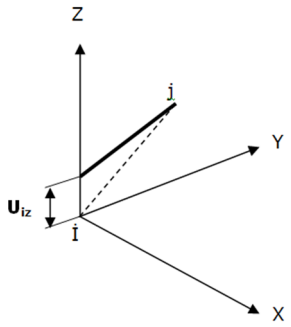


Figure 2.5. The displacement at node i in z direction; U_{iz}

$$\begin{Bmatrix} P_{ix} \\ P_{iy} \\ P_{iz} \\ P_{jx} \\ P_{jy} \\ P_{jz} \end{Bmatrix}_{(3)} = \frac{AE}{L} U_{iz} \begin{Bmatrix} C_x C_z \\ C_y C_z \\ C_z^2 \\ -C_x C_z \\ -C_y C_z \\ -C_z^2 \end{Bmatrix} \tag{2.11}$$

Using the symmetry property of the stiffness matrix, the total element stiffness matrix is obtained as follows:

$$\frac{AE}{L} \begin{bmatrix} C_x^2 & C_x C_y & C_x C_z & -C_x^2 & -C_x C_y & -C_x C_z \\ C_x C_y & C_y^2 & C_y C_z & -C_x C_y & -C_y^2 & -C_y C_z \\ C_x C_z & C_y C_z & C_z^2 & -C_x C_z & -C_y C_z & -C_y^2 \\ -C_x^2 & -C_x C_y & -C_x C_z & C_x^2 & C_x C_y & C_x C_z \\ -C_x C_y & -C_y^2 & -C_y C_z & C_x C_y & C_y^2 & C_y C_z \\ -C_x C_z & -C_y C_z & -C_y^2 & C_x C_z & C_y C_z & C_z^2 \end{bmatrix} \tag{2.12}$$

It should be noted that if the same procedure of obtaining nodal forces corresponding to the displacements applied at node j is followed, one would get the elements of the 4^{th}-6^{th} columns of the stiffness matrix. However, by taking advantage of the symmetry of the stiffness matrix, element stiffness matrix can be determined more easily.

The nodal equilibrium equations can be written as:

$$\begin{Bmatrix} P_{ix} \\ P_{iy} \\ P_{iz} \\ P_{jx} \\ P_{jy} \\ P_{jz} \end{Bmatrix} = \frac{AE}{L} \begin{bmatrix} C_x^2 & C_xC_y & C_xC_z & -C_x^2 & -C_xC_y & -C_xC_z \\ C_xC_y & C_y^2 & C_yC_z & -C_xC_y & -C_y^2 & -C_yC_z \\ C_xC_z & C_yC_z & C_z^2 & -C_xC_z & -C_yC_z & -C_y^2 \\ -C_x^2 & -C_xC_y & -C_xC_z & C_x^2 & C_xC_y & C_xC_z \\ -C_xC_y & -C_y^2 & -C_yC_z & C_xC_y & C_y^2 & C_yC_z \\ -C_xC_z & -C_yC_z & -C_y^2 & C_xC_z & C_yC_z & C_z^2 \end{bmatrix} \begin{Bmatrix} U_{ix} \\ U_{iy} \\ U_{iz} \\ U_{jx} \\ U_{jy} \\ U_{jz} \end{Bmatrix} \quad (2.13)$$

2.2.2.2 Element Stiffness Matrix for 2D Plane Trusses

For plane trusses, one angle θ is enough to define relation of the element to the x and y axes. Denoting cosθ as C and sinθ as S, and following the direct method of obtaining stiffness matrix defined for the 3D truss element, one can obtain the element stiffness matrix of the 2D truss element.

The displacement on element IJ due to displacement Uix is given by

$$e = U_{ix}\cos\theta \rightarrow e = U_{ix}C \quad (2.14)$$

Eq.(2.15) gives the force acting on the element IJ due to small displacement Uix.

$$F = ke = \frac{AE}{L}U_{ix}C \quad (2.15)$$

The forces acting on node i in reaction to the force F in X and Y directions are given by Eqs. (2.16-2.17)

$$P_{ix} = FC_x = \frac{AE}{L}U_{ix}C^2 \quad (2.16)$$

$$P_{iy} = FC_y = \frac{AE}{L}U_{ix}CS \quad (2.17)$$

The forces acting on node j in reaction to the force F in X and Y directions are given by Eqs. (2.18-2.19)

$$P_{jx} = -P_{ix} = -FC_x = -\frac{AE}{L}U_{ix}\mathrm{C}^2 \tag{2.18}$$

$$P_{jy} = -P_{iy} = -FC_y = -\frac{AE}{L}U_{ix}\mathrm{CS} \tag{2.19}$$

The first column of the element stiffness matrix related to the displacement Uix is obtained by Eq. (2.20)

$$\begin{Bmatrix} P_{ix} \\ P_{iy} \\ P_{jx} \\ P_{jy} \end{Bmatrix}_{(1)} = \frac{AE}{L}U_{ix}\begin{Bmatrix} C^2 \\ CS \\ -C^2 \\ -CS \end{Bmatrix} \tag{2.20}$$

The second column of the element stiffness matrix related to the displacement Uiy, obtained in a same manner, is given by Eq. (2.21)

$$\begin{Bmatrix} P_{ix} \\ P_{iy} \\ P_{jx} \\ P_{jy} \end{Bmatrix}_{(2)} = \frac{AE}{L}U_{iy}\begin{Bmatrix} CS \\ S^2 \\ -CS \\ -S^2 \end{Bmatrix} \tag{2.21}$$

Using the symmetry property of the stiffness matrix, the total element stiffness matrix is obtained as follows:

$$\frac{AE}{L}\begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix} \tag{2.22}$$

The equilibrium equations are given by Eq.(2.23)

$$\begin{Bmatrix} P_{ix} \\ P_{iy} \\ P_{jx} \\ P_{jy} \end{Bmatrix} = \frac{AE}{L} \begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix} \begin{Bmatrix} U_{ix} \\ U_{iy} \\ U_{jx} \\ U_{jy} \end{Bmatrix} \tag{2.23}$$

2.2.2.3 Additional Loads

There are two additional loads considered which are applied to the nodes from the sources other than element deformation. These are the thermal and the weight loading.

Thermal loading is the axial force sustained by a fully restrained bar element which is initially stress-free and uniformly heated T degrees [8]. For 3D and 2D trusses, element load vector for the thermal loading is defined by Eq.(2.24) and Eq. (2.25) respectively. In Eqs. (2.24) and (2.25), $\boldsymbol{\alpha}$ is the thermal expansion coefficient.

$$\{r_T\} = \alpha EAT \begin{Bmatrix} -C_x \\ -C_y \\ -C_z \\ C_x \\ C_y \\ C_z \end{Bmatrix} \tag{2.24}$$

$$\{r_T\} = \alpha EAT \begin{Bmatrix} -C \\ -S \\ C \\ S \end{Bmatrix} \tag{2.25}$$

Weight loading is the force applied to the nodes due to gravity [8]. For 3D and 2D trusses, element load vector for the weight loading is defined by Eq.(2.26) and Eq. (2.27) respectively. In Eqs.(2.26) and (2.27) W is the element weight, and in the present formulation, weight of the element is assumed to be equally shared with the nodes of the element.

$$\{r_w\} = \frac{W}{2}\begin{Bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ -1 \end{Bmatrix} \tag{2.26}$$

$$\{r_w\} = \frac{W}{2}\begin{Bmatrix} 0 \\ -1 \\ 0 \\ -1 \end{Bmatrix} \tag{2.27}$$

Combination of these additional element loads $\{r_e\}$ is given by Eq. (2.28)[8]. Equation (2.28) gives the element load vector and in the assembly process, the nodal loads associated with the element load vector are superimposed on the external nodal loads applied.

$$\{r_e\} = \{r_w\} + \{r_T\} \tag{2.28}$$

2.2.3 Assembly of the Element Matrices

The element stiffness and load matrices are assembled in structural stiffness and load matrices. In stress analysis, assembly of the elements can be considered as a process in which equilibrium equations are written for each node of the structure under all loads applied to it [8].

In this process, each of the element stiffness matrix [k] is added into structural stiffness matrix [K] and element load vector $\{r_e\}$ and external loads {P} are added into structural load vector {R}. The assembled structural stiffness and load matrices are given by Eqs (2.29, 2.30).

$$[K] = \sum_{n}^{numel} [k]_n \tag{2.29}$$

$$\{R\} = \{P\} + \sum_{n}^{numel} \{r_e\}_n \tag{2.30}$$

It should be noted that in the construction of the structural stiffness matrix and the structural load vector, theoretically element stiffness matrices and element load vectors should be expanded to structure size and added up, as shown in Eqs. (2.29 and 2.30). However, for large systems such a summation is practically impossible because element stiffness matrices and element load vectors contain many zeros. Therefore, in the present study, ID array concept [8] is used to locate the position of the element stiffness matrices and element load vectors in the structural stiffness matrix and structural load vector without expanding the element stiffness matrices and element load vectors to structure size.

2.2.4 Strain and Stress calculation

By using definitions of [K] and {R}, the static equilibrium of equations for the structure are stated as;

$$[K]\{D\} = \{R\} \tag{2.31}$$

where {D} is structure displacement vector.

In the Eq. (2.31), structure stiffness matrix [K] is singular since there is no unique solution for this equation if the structure is not supported, and rigid body motion of the structure is not prevented.

At this point, boundary conditions for the structure are introduced through which the fixed d.o.f. of the nodes are defined. By partitioning matrices, Eq. (2.31) is redefined by Eq. (2.32) [8].

$$\begin{bmatrix}[K_{11}] & [K_{12}]\\ [K_{21}] & [K_{22}]\end{bmatrix}\begin{Bmatrix}D_x\\ D_c\end{Bmatrix} = \begin{Bmatrix}R_c\\ R_x\end{Bmatrix} \tag{2.32}$$

or, in a expanded form

$$[K_{11}]\{D_x\} + [K_{12}]\{D_c\} = \{R_c\} \tag{2.33}$$

$$[K_{21}]\{D_x\} + [K_{22}]\{D_c\} = \{R_x\} \tag{2.34}$$

where D_c and R_c are known d.o.f. and loads respectively, and D_x and R_x are unknown d.o.f. and loads respectively.

The Eq. (2.33) is solved for D_x which represent the unknown displacements of the nodes. After finding D_x, by using Eq. (2.34), reaction force vector Rx is calculated.

For the stress calculation of the elements, strain in each element is calculated by using the nodal d.o.f. {d} of each element which is extracted from $\{D_x\}$. The computation of elongation in for 3D and 2D truss elements and mechanical strains is given by Eqs. (2.35, 2.36, 2.37) respectively [8].

$$e = (u_j - u_i)C_x + (v_j - v_i)C_y + (w_j - w_i)C_z \tag{2.35}$$

$$e = (u_j - u_i)cos\beta + (v_j - v_i)sin\beta \tag{2.36}$$

$$\epsilon = e/L \tag{2.37}$$

Where u_i, v_i, w_i and u_j, v_j, w_j are displacements in X, Y, Z directions at node i and node j of the element, respectively.

Stress calculation for each element is given by Eq. (2.38)

$$\sigma = E\epsilon \tag{2.38}$$

All of these steps are handled in the finite element code which is developed in MATLAB. The flow chart for the code is shown in Figure 2.6 [1].

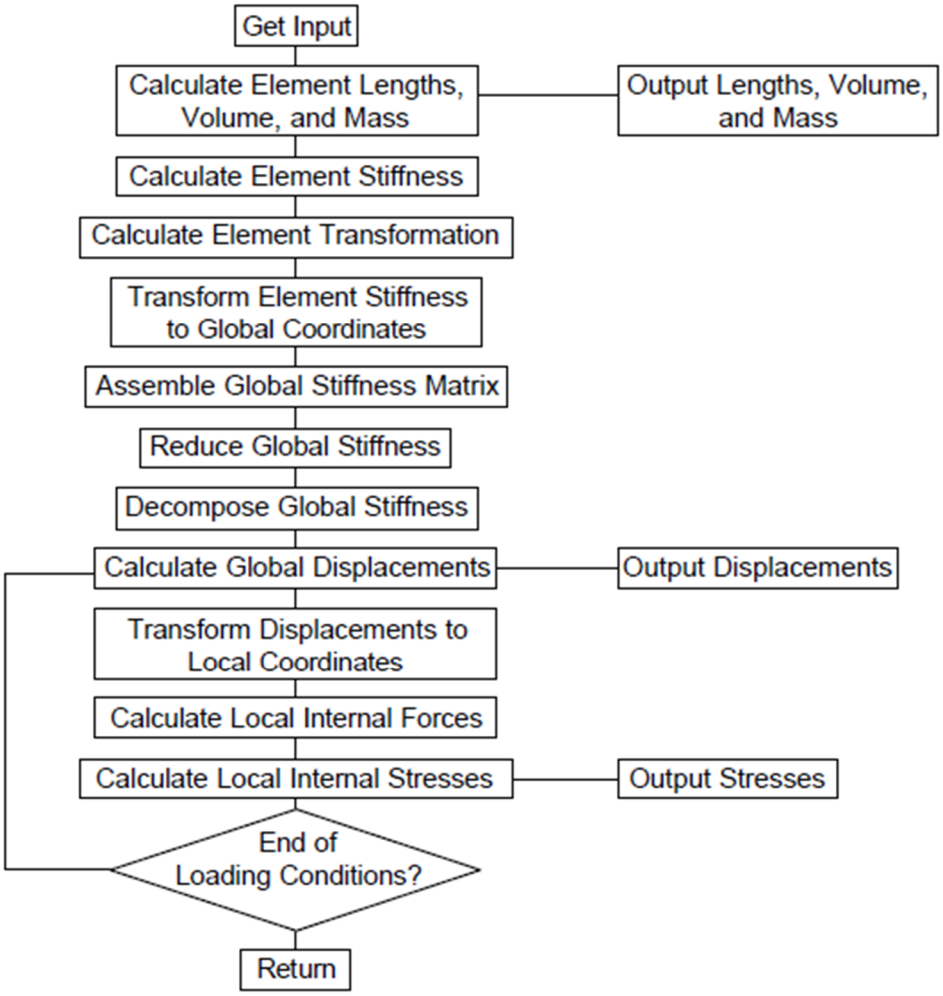


Figure 2.6. Flow chart for finite element method.

2.3 Verification Study for Finite Element Method

In the verification of the finite element code, a 3-bar space truss shown in Figure 2.7 is used. In the analysis, the nodes 1, 2 and 3 are taken fixed. A load P=12 kN is applied at node 4 in X-direction.

The modulus of elasticity for the bars is taken E=200 GPa. The area of the elements 1 and 3 is set to be 0.001 m^2 and for the second element, it is taken as 0.002 m^2.

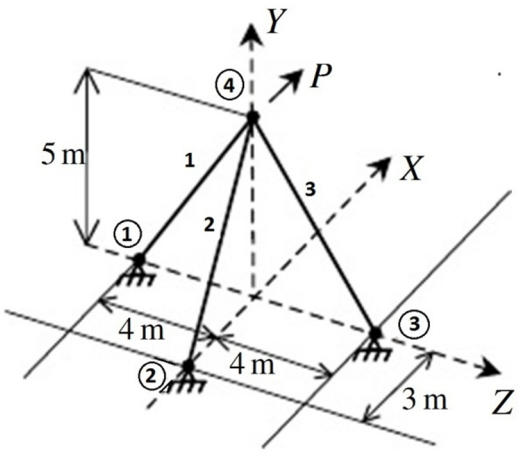


Figure 2.7. 3-Bar space truss

The stress outputs of the program are compared by the results calculated by hand and the same results are obtained from both. The results are given by Table 2.1. Since the truss shown in Figure 2.7 is a typical 3D truss, the stress comparison given in Table 2.1 shows that the developed finite element code is reliable and error free.

Table 2.1. Stress results for 3-bar space truss

	σ_{EL1}	σ_{EL2}	σ_{EL3}
	Pa		
Hand calculation	-12,806	11,662	-12,806
Finite Element Code	-12,806	11,662	-12,806

CHAPTER 3

GENETIC ALGORITHM BASED STRUCTURAL OPTIMIZATION

3.1 Structural Optimization Methods

Optimization of structures can be grouped in three categories which are topology, size, and shape optimization. Generally, in the structural optimization the objective is the mass minimization subject to stress and/or displacement constraints. These categories can be very briefly defined as:

Topology optimization: Element-node connectivity variation to find an optimal layout design.

Size optimization: Element cross sectional properties variation to find an optimal sizing

Shape optimization: Movements of nodes to change the shape of the structure without changing the topology.

In this thesis, size and shape optimization is studied by changing size and shape variables simultaneously.

In Figure 3.1, examples of the three types of structural optimization can be seen [1].

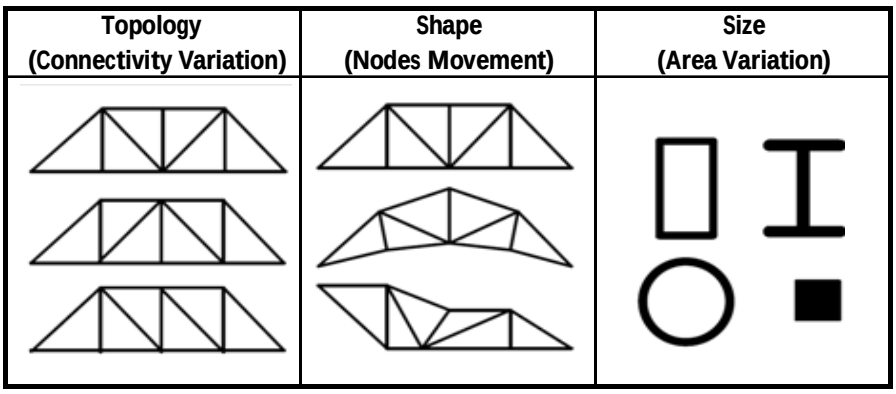


Figure 3.1. Examples of structural optimization

The optimization problem is generally defined as minimizing the mass of the structure under stress and displacement constraints. The objective function of the problem is the mass of the structure, and for the truss structure total mass is defined by

$$f(X) = \sum_{i=1}^{n} \rho A_i L_i \tag{3.1}$$

where n is the total number of elements in the truss structure, x is the objective function variables, Ai is the cross-sectional area of the i^{th} element, Li is the length of the i^{th} element, and ρ is the weight density of the material.

Additionally, the truss is subject to the following set of stress and displacement constraints;

$$\sigma_i \leq \sigma_a \qquad \textit{for } i=1 \textit{ to } n \tag{3.2}$$

$$U_i \leq U_a \tag{3.3}$$

where σ_i is the stress in element i, σ_a is the maximum allowable stress for all elements, U_i is the displacement of each node (horizontal and vertical

displacements), and U_a is the maximum allowable displacement for all nodes. These constraints can be expressed in normalized form as

$$\frac{\sigma_i}{\sigma_a} - 1 \leq 0 \tag{3.4}$$

$$\frac{U_i}{U_a} - 1 \leq 0 \tag{3.5}$$

3.2 Genetic Algorithm

The idea of evolutionary computing was introduced by I. Rechenberg. The genetic algorithm, which is an area of evolutionary computation, was invented by John H. Holland [4]. After the contribution of the high performance computing systems, the deal with genetic algorithms increased and it became more popular as an optimization tool.

Genetic algorithms (GAs) utilize the evaluation theory and natural selection for solution of the problem. It is nondeterministic stochastic search and optimization method, meaning that it uses some form of pseudorandom number generation, and as a result, this makes the solution path and results nondeterministic. Stochastic property gives opportunity to search complex landscapes filled throughout with local optima and deceptive solution paths [1].

Genetic algorithm is based on Darwin's five main principles of evolution which are population, crossover (recombination), mutation (variation), selection, and heredity [1].

In a basic GA, algorithm starts with a randomly generated population of individuals which are named as chromosomes. Every individual has finite number of design variables which are called as genes. From this population, parents for the recombination process are chosen by fitness based selection. After recombination, mutation is usually applied at a small rate. The child chromosomes then pass into the successor population which is the new population. As this process is iterated, the average fitness of the

chromosomes tends to increase until some stopping criterion is achieved. Thus, genetic algorithm evolves to the best solution for a given problem [2].

Superior characteristics of genetic algorithm over traditional optimization methods are [3][9]:

- Specific knowledge about the problem is not required for genetic algorithms. So it is not necessary to have continuous and/or differentiable objective function. The search can be performed over non-convex and even disjunct sets, and variables can be different types (e.g. continuous, discrete, boolean).
- GAs works on a population which has set of solutions for the objective functions while classical methods use single-point approach. In other words, GAs process a number of designs at a time.
- GAs are not deterministic opposite to the traditional methods. Instead, randomized operators are used which improve the search process, giving chance to any point to be a solution in a given range. So GAs are not sensitive to the starting point and less prone to entrapment in local optima.

Genetic operators, being randomized, working on coded design variables and processing on a population, make GAs powerful. Today, many kinds of genetic operators are used, such as reproduction, crossover, mutation, dominance, segregation, migration, translocation, deletion.

Having adaptable characteristics, GAs have been applied to a wide variety of problems in science, engineering, finance, etc.

3.3 The Coding Procedure

In this thesis, a genetic algorithm is coded for size and shape optimization of the planar and space trusses. The algorithm steps are very similar to the basic flowchart of a typical GA as shown in Figure 3.2. But in the present study, emphasis is given to the adaptive penalty function approach and the elimination of the arbitrariness issue of the factor multiplying the error term in the augmented fitness function to find solutions which satisfy constraints. Following sections explain the steps of the algorithm in more detail.

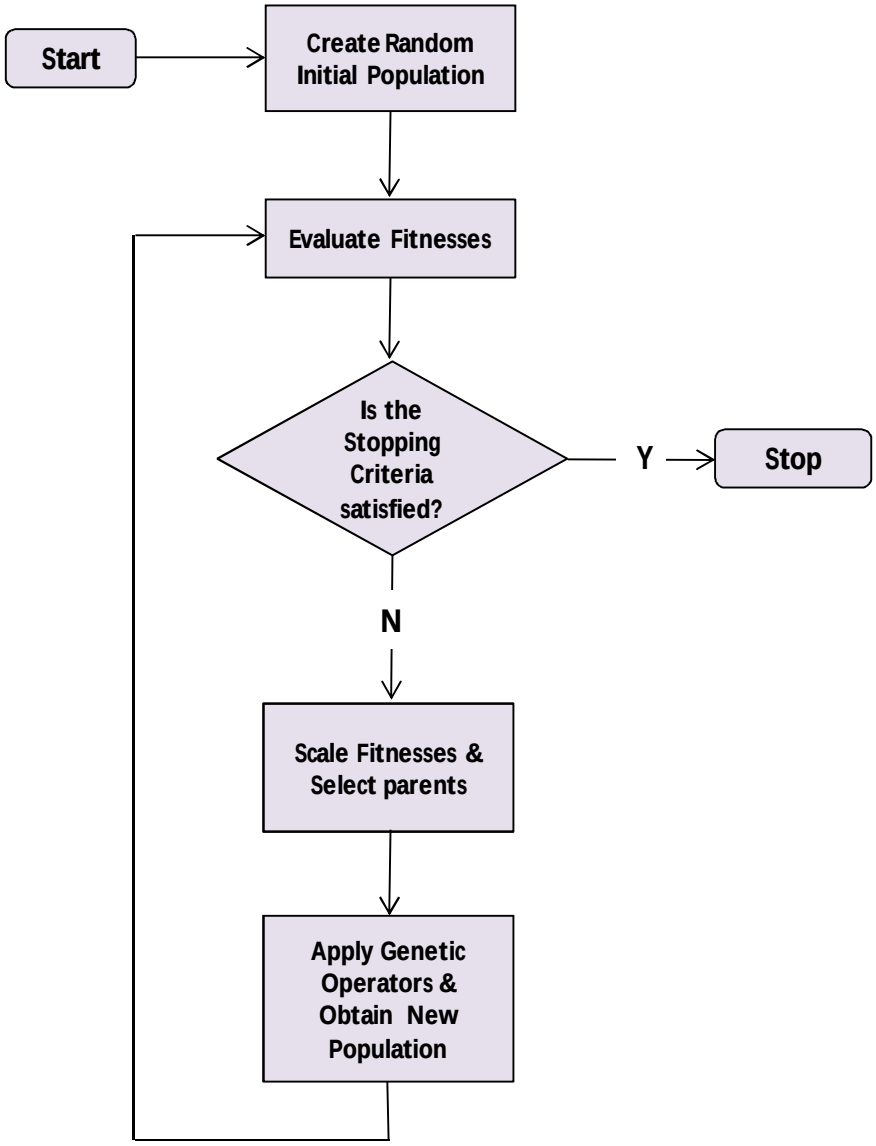


Figure 3.2. Genetic Algorithm Flowchart

3.3.1 Initial Population Creation

The algorithm starts by creating randomized initial population. In code, this is a matrix with dimension mXn, where m is the size of the population and n is the number of variables.

Population size gives the number of individuals in the population. It is user defined and can change for different problems. For GAs, population size is very is important in the effective implementation of the algorithm. If the population size is smaller than needed, the algorithm search capability decreases and if it is too much, convergence time and the number of run to find the optimum solution increase. For every specific problem, the population size should be determined properly. The complexity of the problem is an important criterion for the determination of it. In the literature, some recommendations are given for the population size. For instance, if binary coding is used in operations, the population size should be in the same order as the length of the string [6][19]. In Chapter 4, studies are performed to find the best population size range for each problem.

3.3.2 Fitness Evaluation

The aim of the GAs is to find best (i.e. maximum) fitness value in the population. So, in Gas we deal with a maximization process. In structural optimization problems, however, the objective is to minimize the mass. So in the algorithm, fitness function is used instead of objective function as shown in Eq. (3.1) [11].

$$F(Xi) = \frac{1}{f(Xi)} \tag{3.6}$$

where f(Xi) is objective function which is the mass of the i^{th} individual. By taking mass to the denominator, as the algorithm tries to find maximum fitness, the minimum mass is also found.

The optimization problem is then defined as:

Maximize

$$F(\boldsymbol{x}) = F[f(\boldsymbol{x})], \qquad \boldsymbol{x} = (x_1, x_2, \dots, x_N) \in R^N \tag{3.7}$$

Under constraints

$$\begin{aligned} g_i(\boldsymbol{x}) &\leq 0, \qquad i = 1,2,\dots,K \\ h_i(\boldsymbol{x}) &= 0, \qquad i = 1,2,\dots,P \end{aligned} \tag{3.8}$$

Problem variables are defined by **x** which is a N-dimensional design vector, and $g_i(\boldsymbol{x})$ and $h_i(\boldsymbol{x})$ are inequality and equality constraints, respectively.

For GAs, there should be only one equation to operate on. The presence of constraints makes it impossible to solve the problem directly using the fitness function *F(**x**)*. It is compulsory to redefine the fitness function by taking the constraints into account. Generally to handle constraints, the concept of penalty functions is used. The main idea in introducing penalty function is penalize infeasible solutions in the population. The definition of the fitness function in case of infeasible solutions is given by Eq. (3.9) variables are defined by **x** which is a N-dimensional design vector, and $g_i(\boldsymbol{x})$ and $h_i(\boldsymbol{x})$ are inequality and equality constraints, respectively.

$$\begin{aligned} F^a(\boldsymbol{x}) &= F(\boldsymbol{x}) && if\ \boldsymbol{x} \in \tilde{F}, \\ F^a(\boldsymbol{x}) &= F(\boldsymbol{x}) - P(\boldsymbol{x}) && otherwise, \end{aligned} \tag{3.9}$$

where $\tilde{F}$ is feasible search space, P(**x**) is the penalty function which is greater than zero, and $F^a(\boldsymbol{x})$ is an augmented fitness function after the penalty.

In the literature, the usual way of expressing the penalty function is [1][3][14][16][18][19]:

$$\sum_{j=1}^{K} (\lambda_G)_j [G_j(\boldsymbol{x})]^\beta + \sum_{j=1}^{P} (\lambda_H)_j [H_j(\boldsymbol{x})]^\beta, \tag{3.10}$$

where

$$G_j(\boldsymbol{x}) = \max[0, g_j(\boldsymbol{x})],$$
$$H_j(\boldsymbol{x}) = \text{abs}[h_j(\boldsymbol{x})] \tag{3.11}$$

$G_j(x)$ and $H_j(x)$ are degrees in violation of inequality and equality constraints, respectively. $(\lambda_G)_j$, $(\lambda_H)_j$ and β are constants. $(\lambda_G)_j$ and $(\lambda_H)_j$ are taken as same value, and the value of β is generally set to either 1 or 2.

By determining the values of the constants $(\lambda_G)_j$ and $(\lambda_H)_j$, the degree of penalty can be controlled. However, these coefficients, have no physical meaning, and cannot be selected judiciously. It is not easy to decide which values are really effective on the problem studied, because constants $(\lambda_G)_j$ and $(\lambda_H)_j$ have a very wide range such 0.000001 to 10000 depending on the problem studied [11].

In this thesis, the adaptive penalty function is used [11]. The expression of the new penalty function is given by Eq. (3.7).

$$F_i^a = F^a(\mathbf{x}_i) = F(\mathbf{x}_i) - \lambda(t)E(\mathbf{x}_i) \tag{3.12}$$

where F_i^a is the fitness function of the ith individual after penalty and $\lambda(t)$ is a factor multiplying the error term $E(\mathbf{x}_i)$. It is noted that $\lambda(t)$ changes in every iteration and t indicates the generation number. The error term $E(x_i)$ is defined by [11]

$$E(\boldsymbol{x}_i) = \sum_{j=1}^{K} G_j(\boldsymbol{x}_i) + \sum_{j=1}^{P} H_j(\boldsymbol{x}_i), \tag{3.13}$$

The key point in the adaptive penalty function implementation is the proper selection of the factor $\lambda(t)$ multiplying the error term which represents the infeasible solutions. It should be noted that if this factor is too small, then there is a chance of penalized fitnesses of some infeasible individuals may become higher than the fitnesses of feasible individuals. On the other hand, if the multiplying factor is too large, then some individuals with high fitness function values may be penalized heavily such that in the next generations

these individuals may have no chance to survive. Thus, good characteristics of these individuals may be lost.

At this point, a new definition for a factor $\phi(t)$ is introduced, as shown in Eq.(3.14)

$$F^a(\mathbf{x}_i) \leq \phi(t) F_{avg}^{a,\tilde{F}} \quad for\ \forall \boldsymbol{x_i} \in \tilde{U} \tag{3.14}$$

Here, $\tilde{U}$ is the infeasible search space and $F_{avg}^{a,\tilde{F}}$ is the average fitness value of the feasible individuals in the population. The equation sets that the fitness of the infeasible individuals cannot be more than $\phi(t)$ times the average fitness of the feasible individuals. An appropriate value for $\lambda(t)$ is determined by setting the penalized fitness value of the infeasible individual to $\phi(t)F_{avg}^{a,\tilde{F}}$. For each infeasible individual, $\lambda(t)$ is calculated, and maximum value is selected as the value of $\lambda(t)$ to be used as the multiplier factor in the expression for the augmented fitness function. Thus, $\lambda(t)$ can be defined as

$$\lambda(t) = \max\left(0, \max_{\forall \boldsymbol{x_i} \in \tilde{\boldsymbol{u}}} \left[\frac{F(\boldsymbol{x}_i) - \phi(t) F_{avg}^{a,\tilde{F}}}{E(\boldsymbol{x}_i)}\right]\right) \tag{3.15}$$

It should be noted that Eq. (3.15) assures that Eq. (3.14) is satisfied. Thus, to determine $\lambda(t)$, first, one has to obtain $\phi(t)$.

$\phi(t)$ is determined through the use of a bilinear scaling function that is introduced to scale the fitnesses [11]. In this scaling method, there are two linear curves to define two different intervals of the fitnesses as illustrated in Figure 3.3. The horizontal axis of the Figure 3.3 gives the augmented fitness values F^a, i.e. fitness values after penalty. Vertical axis of the Figure 3.3 is the scaled fitnesses F^s. The first linear curve scales the augmented fitness interval of $[F_{min}^a, F_{avg}^{a,\tilde{F}})$ to [0, 1). In other words, if the fitness of an individual is below the average fitness of the feasible individuals, then scaled fitness of the particular individual is between zero and one depending on the linear relation shown in Figure 3.3. In a similar manner, the second linear curve scales augmented fitness interval of $[F_{avg}^{a,\tilde{F}}, F_{max}^{a,\tilde{F}}]$ to [1, C] [11].

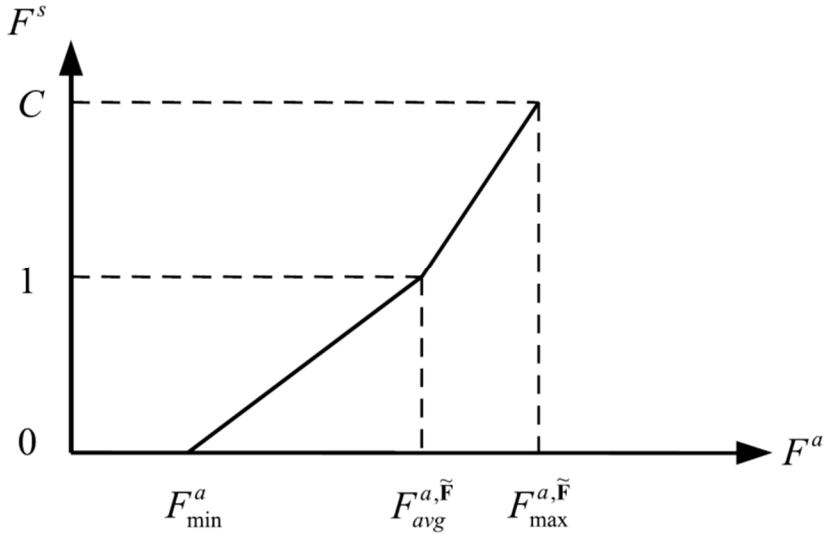


Figure 3.3. Bilinear Fitness Scaling

By this scaling method, for best feasible individuals, the chance to be selected as a parent for reproduction is set to be C times that of the average feasible members. The worst individual after penalty, which has fitness F^a_{min}, has no chance to be selected since the scaled fitness of the worst individual is zero. In the following, selection of the individuals is based on the scaled fitness values rather than the actual physical fitnesses.

In the scaled fitness approach, Eqn. (3.14) is written in terms of scaled fitnesses as shown in Eqn. (3.16).

$$F^s(\boldsymbol{x}) \leq (\varphi F^{s,\tilde{F}}_{avg} = \varphi) \quad for\ \forall \boldsymbol{x_i} \in \tilde{U} \tag{3.16}$$

where, $\tilde{U}$ represents the infeasible search space, $F^s(x)$ is the scaled fitness function, $F^{s,\tilde{F}}_{avg}$ is the average scaled fitness value of the feasible individuals which is equal to 1. According to this definition, the maximum value of the best infeasible individual is set to be φ which is constant throughout the algorithm for all generations.

The expression for *(t)* in Eqn. (3.16) can be expressed in terms of φ as shown in Eqn.(3.17) [11].

$$\phi(t) = \begin{cases} \dfrac{CF_{avg}^{a,\tilde{F}} + F_{max}^{a,\tilde{F}}(\varphi - 1) - \varphi F_{avg}^{a,\tilde{F}}}{(C-1)F_{avg}^{a,\tilde{F}}} & for\ \varphi \geq 1 \\ \dfrac{F_{min}^{a} + \varphi F_{avg}^{a,\tilde{F}} - \varphi F_{min}^{a}}{F_{avg}^{a,\tilde{F}}} & for\ \varphi < 1 \end{cases} \quad (3.17)$$

Having a very clear physical meaning and not having any unit, coefficient φ can be determined by experience. So the degree of the penalty $\lambda(t)$ is obtained, because it is defined in terms of $\phi(t)$ which is directly related to φ by Eq. (3.17).

It should be noted that in obtaining the expressions for *(t),* equation of linear curves shown in Figure 3.3 is utilized with F^s, being replaced by *(t)* and F^a, being replaced by $\phi(t)F_{avg}^{a,\tilde{F}}$.

Here, it should be stressed that coefficient φ has a very clear physical meaning, and it does not have any unit. Physical meaning of φ is that it gives the chance of best infeasible individual being selected into the mating pool compared with the chance of the average feasible individuals. It should be noted that coefficient has units and depending on the problem being solved at hand, its value can take on any value. However, coefficient φ does not have a unit and it can be determined by little experience. Once φ is selected, *(t* can be expressed in terms of φ from Eqn (3.17). Finally, the degree of the penalty $\lambda(t)$ is obtained, because it is defined in terms of $\phi(t)$ by Eqn. (3.15).

When there is no feasible solution in the population, another scaled function is used. This time, scaled fitness values are based on the values of error terms denoted by $E(\boldsymbol{x})$, which is the total constraint violation for each individual.

As shown in Figure 3.4, scaled function is set to Z for the individuals which violate the constraints the least (E_{min}).On the other hand, scaled function is set to 1 for the individuals which violate constraint at the average (E_{avg}), and is set to 0 for the individuals which violate constraint most (E_{max}).

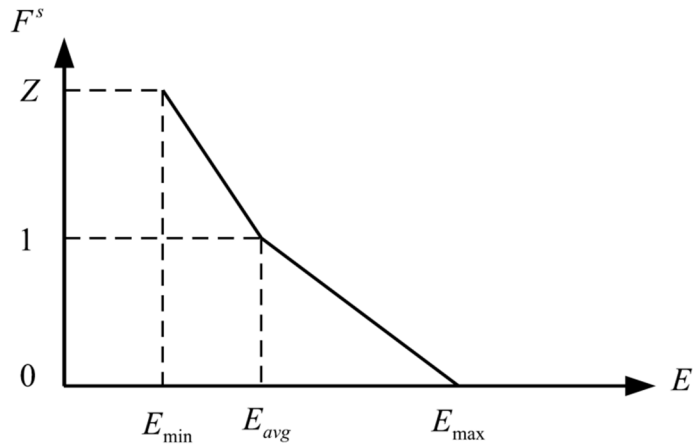


Figure 3.4. Bilinear Fitness Scaling for Totally Non-Feasible Populations

In the algorithm, the values of φ, C and Z are initially set to 0.75, 2 and 5, respectively, as indicated in reference [11]. At the beginning; the algorithm can search the total search space by these problem inputs. Because, the linearly scaled fitness values change in an interval of 0 and 2 (C value), every individual, even infeasible ones, has a chance of being selected in the mating pool. To use the variety of the randomized initial population, this starting condition is very meaningful. Thus, algorithm can catch valuable gene information which leads to the better optimum points.

However, after generating a certain number of feasible individuals in the population, the algorithm should reduce the size of the search space to feasible region. Because it is time consuming to select infeasible individuals with the same probability when the feasible region for the solution of the problem has been already defined.

To work on feasible search space, the value of C is increased. So the probability of the feasible individuals being chosen as parent by the selection operation becomes higher. At the same time, infeasible individuals are automatically eliminated from the population by losing their fitnesses rapidly.

By making this improvement in the methodology given in reference [11], algorithm is made more efficient by finding the optimum solution in less time.

3.3.3 The Selection Scheme

The basic logic in selection methodology comes from the Darwin's evolution theory; the best ones should survive and create new offspring. There are many methods used to select the best chromosomes, such as, roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others. A brief explanation is given for roulette wheel selection.

3.3.3.1 Roulette Wheel Selection

Roulette wheel selection method is best explained through an example.

Consider a scaled fitness matrix for a population size of 5.

FS= [1, 1.5, 0, 2, 0.5]

- First cumulative sum of the scaled fitnesses is obtained.

  CSFS= [1, 2.5, 2.5, 4.5, 5]

- Then by dividing every element by maximum value of the CSFS, the portions of the individuals in the total interval of [0, 1] are obtained;

  [0.2, 0.5, 0.5, 0.9, 1]

- Normalized interval implies that the individuals 1, 2, 3, 4 and 5 belong to intervals [0, 0.2], (0.2, 0.5], [0.5, 0.5], (0.5, 0.9], (0.9, 1], respectively, as shown in Figure 3.5.

- To select an individual as a parent, first a random number is generated in the interval [0-1]. Random number generated points a slice, and the individual belonging to that slice is chosen as the parent. In the given example, the 4^{th} individual, having the highest fitness, has the highest chance of 40% while 3^{rd} individual have no chance to be selected for reproduction.

In the genetic algorithm developed, the roulette wheel selection is chosen as the selection algorithm.

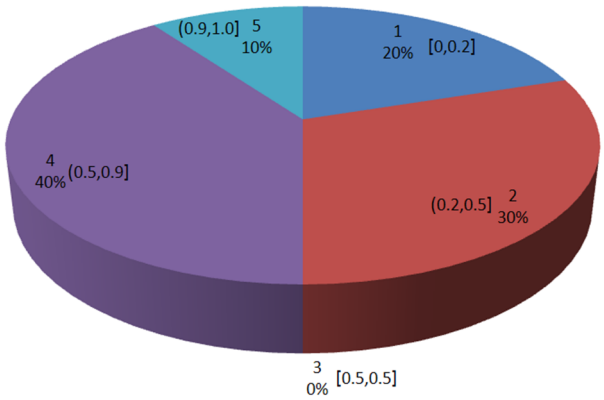


Figure 3.5. Roulette Wheel

3.3.4 Genetic Operators

Genetic algorithm gets the optimum result of a given problem by exploiting useful information contained in the population to generate new population containing better fitness values. This process is done by elitism, crossover (or recombination) and mutation operations in the algorithm. These operations are the most classical genetic operations.

In the genetic algorithm, the individuals are represented as a string which consists of variables of the problem. In the present study of structural optimization of truss structures, these variables are the areas of truss elements (bars) for size optimization, and the coordinates of the nodes which are allowed to move in an interval specified in the problem definition for the shape optimization.

To explain the operators used in the algorithm, a 3-bar-planar-truss structure, shown in Figure 3.6, is given as an example. This structure has 3 bar elements and 4 nodes. Nodes 1, 2, and 3 are fixed while node 4 can move in x and y directions. The load is applied at node 4.

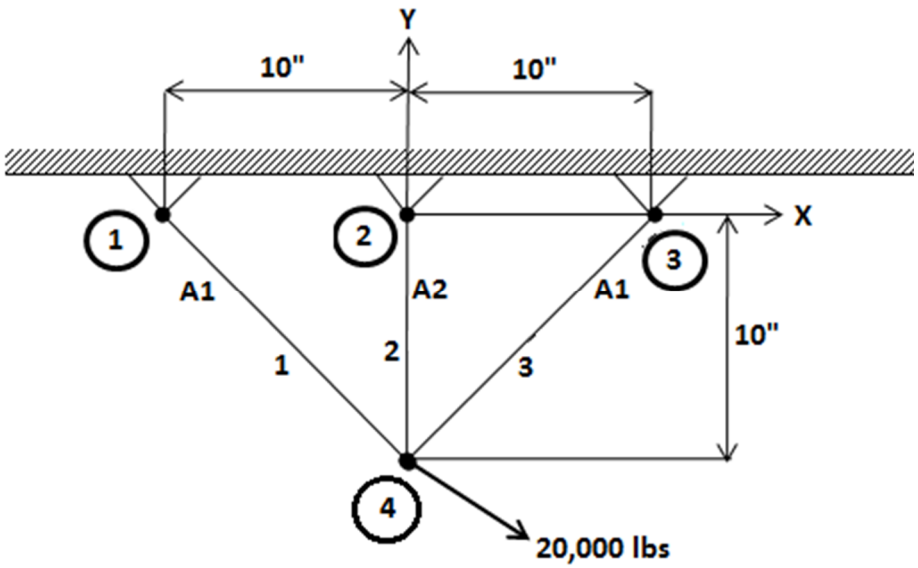


Figure 3.6. 3-Bar planar truss structure

Consider the size and shape optimization of this example structure. The area variables are denoted as A1 and A2. The areas of the elements 1 and 3 are equal, and node 4 has X and Y coordinate variables which are defined as design variables. Other nodes have no shape variable, because they are fixed. Coding these variables as a string for each individual is made in the sequence given in Figure 3.7.

A1	A2	X	Y

Figure 3.7. Sample string for an individual

3.3.4.1 Elitism

Elitism is a method by which a user defined number of the best individuals survive directly and are copied to the new population. Thus, they still have a chance of being selected as parent for reproduction. Elitism can very rapidly increase the performance of GA, because it protects best solution.

Population consists of the same type of the individuals. For the truss structure example with a population size of 5, Table 3.1gives the values of the size and shape design variables, associated fitness values, weight and scaled fitness results for a particular iteration step.

Table 3.1. Sample iteration results for 3-bar planar truss

ind#	A1	A2	X	Y	Mass	Fitness	Total Const. Violation ($E(X_i)$)	Penalty Constant (λ(t))	Fitness After Penalty (F^a)	Scaled Fitness (F^s)
1	0,33	1,30	-4	-7	1,86	0,54	2,49	0,09	0,32	0,80
2	0,34	1,42	-8	-2	1,92	0,52	2,80	0,09	0,28	0,43
3	0,74	0,46	9	-8	2,71	0,37	0	0	0,37	1,29
4	1,01	0,33	5	-3	2,31	0,43	0	0	0,43	2,00
5	1,23	1,22	-6	-7	4,43	0,23	0	0	0,23	0,00

In this sample iteration, first by using values of the design variables (cross sectional areas and nodal coordinates) the mass of the each truss element is calculated. Then the corresponding fitness values are obtained by using fitness function presented in Eq. (3.6). After computing error terms for each truss element and penalty constant λ(t) for the related iteration *t*, augmented fitness function F^a can be computed by the equation Eq. (3.12). As presented in Figure 3.3, by using bilinear fitness scaling method, scaled fitness values are obtained.

If the elite individual number of is 1, only one individual having the best fitness selected as elite individual. In our example, the fourth individual is selected as an elite individual and passes to the new population because its scaled fitness value is the maximum. It should be noted that although there are individuals with less mass, since they have higher error terms, i.e., total constraint violation value, their scaled fitness values are less.

3.3.4.2 Crossover

Just selecting better individuals is not enough to improve the results of the algorithm. There should be new individuals produced in the population with

the hope of finding better solutions. Crossover is the one of the most popular way of creating child chromosomes from the parents. Because the selection method chooses parents from the better individuals, the chance of getting improvement in next generation increases by recombining better gene information.

There are different cross-over operators; the most popular ones are one point crossover, two point crossover and uniform crossover. In this algorithm, one-point crossover is used. The main idea is that selected two parents exchange some portion of their strings to create one child [6].

Crossover operation is done with a probability. It is called crossover rate and taken as 0.8 in the algorithm developed. Before crossover operation, a random number is generated between 0 and 1, and if this number is smaller than the crossover rate, the crossover occurs. If not, new random number between 0 and 1 is generated. If the random number is less than 0.5, the first parent is selected as the crossover child. Otherwise, second parent is accepted as the crossover child. The procedure is continued until achieving the population size.

In the algorithm, the genes are redefined by binary coding. In other words, every gene is represented by a number of bits defined by gene length. In example truss, each truss element has 4 variables i.e. 4 genes. This means that, the chromosome length holding the total design information of the element, has length of 4X4=16 bit. After working on binary strings for genetic operators, algorithm is to transform these strings to real values which are used in calculation of the mass and constraints.

Depending on the variable types, discrete or continuous, different transformations are to be made. In the present work, both of the variable types are used. For example, in the size optimization of the benchmark study of 10-bar truss structure, discrete size variables, from standard areas, are used. On the other hand, for the shape optimization studies, continuous variables are used.

The transformation from binary strings to real valued variables is described below in detail.

- First the decimal equivalents of the binary strings are found.

- If the variable type is discrete, these decimal numbers are matched with the values of the standard areas given as input for the size optimization. To be able to make appropriate matching, the number of the areas in the standard area list should be less than or equal to the maximum number that each gene can get.
- For example, if the standard area list contains 10 areas, the minimum number of bit of each gene must be higher than or equal to 4. If the gene length is 4, then the maximum number that can be defined with a four digit binary number is 15 ((1111)bin=(15)dec). Thus, 10 areas can be assigned to any of the 15 decimal numbers. If the binary string length is 3, then maximum number that can be defined is 7, and 7 different numbers is not enough to define 10 areas in standard list.

  The real value equivalent of the 15 different decimal numbers to the 10 standard area values is given in Table 3.2.

Table 3.2. Indices of the standard area list for discrete variables

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	2	3	4	4	5	6	6	7	8	8	9	10

These values are found by the simple calculation,

N=D*10/15 and I=floor(N),

where D is the one of the possible 15 different numbers coming from the transformation of the binary string to decimal number, and I denotes the I^{th} area in the standard area list. I is found by rounding N down to the next integer. It should be noted that since the number of the standard areas is less than the maximum integer number that can be obtained with four digit binary number, some areas are selected more than once. This may be a drawback of using binary strings to represent design variables. A possible remedy could be to use the discrete variables as is without resorting to binary strings to represent genes.

- If the variable type is continuous, the transformed numbers are assigned to the interval given as input. These intervals can be for areas in both size and shape optimization and for coordinates of the nodes which are free to move in shape optimization.

  As an example, the area alternatives for a gene length of 4 and area interval of [0.2, 2], and for a step size of (2-0.2)/(15-1)=0.129, are given in Table 3.3. For example, for the assigned value of 9, the program uses 1.229 as the area value.

Table 3.3. Area alternatives for continuous variables

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0,2	0,329	0,457	0,586	0,714	0,843	0,971	1,1	1,229	1,357	1,486	1,614	1,743	1,871	2

As an illustrative example for the crossover operation, again the 3-bar-planar truss can be used. The parent chromosomes before crossover are represented as shown in Figure 3.8.

parent_1	15	7	3	8
	1111	0111	0011	1000
parent_2	6	8	3	11
	0110	1000	0011	1011

Figure 3.8. Parents before crossover

Consider that randomly generated crossover point is 5, in this example. The child chromosome is obtained by getting the values 1 to 5 from first parent and the values from 6 to 16 from second parent. Figure 3.9 shows the one point crossover operation.

Figure 3.9. One Point Crossover Operation

With the crossover operation, the search space is explored more effectively making it possible to try alternative generation of individuals. With high crossover rate, the probability to gain new individuals increases. But if it is too large, the good genetic characteristics may be destroyed and good individuals are demolished. But if the rate is chosen too small, the search process becomes slower and depending on the maximum generation number, the algorithm converges before finding optimal solution. So choosing appropriate crossover rate is very important and should be determined problem specific [13].

3.3.4.3 Mutation

After crossover, mutation operation is applied to each child, obtained by crossover operation, individually. It randomly alters each bit in the chromosome with a smaller probability compared with the crossover rate.

In the mutation operation, first a random number, between 0 and 1, is generated. If the random number of the gene is smaller than the mutation rate, mutation operation is performed. To make mutation on chromosome, mutation point is selected randomly, and the value of the bit at the mutation point is changed from 1 to 0, or vice versa.

Application of mutation operation can be examined by the child chromosome obtained for the sample truss structure. In this example, the mutation rate is taken as 0.05. If the random number is 0.04, then the chromosome is mutated. To perform the mutation operation, the value of the bit at the randomly chosen mutation point (take 3) is changed from 1 to 0, or vice versa, as shown in Figure 3.10.

Figure 3.10. Mutation operation

After mutation operation, child chromosome is transformed from binary string to decimal number. The Table 3.4 gives the results of this transformation.

Table 3.4. Child chromosome

child	1101	0000	0011	1011
	14	1	3	11

Mutation provides a small amount of random search so that it helps ensure that every point in the search space has a chance of being examined [3]. Mutation maintains the diversity in the population and prevents getting stuck in a local minimum [6].

It should be noted that mutation rate should not be too high, especially at the beginning of the iterations. Because, randomly initiated population cannot converge by the diversity property of mutation. However, after a stage that population starts to converge in a feasible region, higher mutation rate is needed to increase the search capability in the algorithm, because crossover itself cannot provide different gene information from the parents which are almost same [21].

3.3.5 Stopping Criteria

After a new population is created, algorithm turns to fitness evaluation step. And this routine continues up to a point at which at least one of the stopping criteria is satisfied. Two stopping criteria used in algorithm are listed below [6][7];

- Maximum generation number: When algorithm comes to maximum generation number, it stops the iteration. This value should be large enough for the algorithm to search the solution space properly.
- Stall generation: This is the maximum number of generations through which optimum solution does not improve. Usually, this condition stops the algorithm when the maximum generation number is too much for the given problem or the crossover and/or mutation rate is too small.

3.3.6 The Detail Steps of the Developed GA

The basic flow chart of the developed genetic algorithm given in Figure 3.2 is explained in detail by the following steps:

1. The algorithm starts with the randomly created initial population. The number of individuals, which is called as population size and initially given as input to the program, is constant through the iterations.
2. Then fitness values of the individuals are calculated by Eq.(3.6)
3. By using bilinear scaling method the fitness values are scaled.
   - 3.1. The number of best individuals identified by elite number passes directly to the new population which has still having chance to be selected as parent.
   - 3.2. The rest of the individuals are determined by processing the GA operators, crossovers and mutation.
     - 3.2.1. First crossover operation is applied at a probability of crossover rate on two parents selected by roulette wheel selection method. Until achieving the number of the individuals which is population size minus elite number, this procedure is repeated.
     - 3.2.2. Then the individual reproduced by crossover is mutated at a rate of mutation.
     - 3.2.3. Until achieving the number of the individuals which is population size minus elite number, the steps 3.2.1 and 3.2.2 are repeated.
   - 3.3. New population is created by the combination of the elite individuals and the new individuals obtained by crossover and mutation.
4. The step 2 and 3 are repeated until one of the stopping criteria is reached and algorithm stops by providing the solution for the optimization problem

3.4 Genetic Algorithms in MATLAB

The result obtained for the studies in Chapter 4 are compared with the references and also with the solutions obtained by the “Genetic Algorithm Toolbox” in MATLAB which enables the use of GA on a wide range of problems.

General view of the Genetic Algorithm Toolbox is presented in Figure 3.11. The left side of the tool is used to define inputs of problem to be optimized. These are the fitness function, number of variables, the nonlinear constraint function and upper and lower bounds for the variables. An example problem definition can be seen in Figure 3.12.

The right size of the toolbox is allocated to the options for the genetic algorithm shown in Figure 3.13. The toolbox includes many different options, e.g. different selection, crossover and mutation operators, and has a built in graphical interface. Due to fact that it is written in open MATLAB language, the user is free to inspect and modify the algorithms, or create own, custom functions [7].

To apply the GA toolbox on an optimization problem, the MATLAB functions has to be implemented with a problem specific representation, genotype/phenotype mapping, fitness evaluation and penalty function.

In this thesis, for the Genetic Algorithm Toolbox, the selected options are same with those options used in the developed algorithm. These options are roulette wheel selection, one point crossover and mutation. In the genetic algorithm, for the constraint problems there is a special mutation option, adaptive feasible mutation. In the developed GA, there is no operation matching with this mutation operator, but the other operators selected are the same. So that the results with the usage of the same operators can be seen and the performance of the developed GA can be compared with the Genetic Algorithm Toolbox.

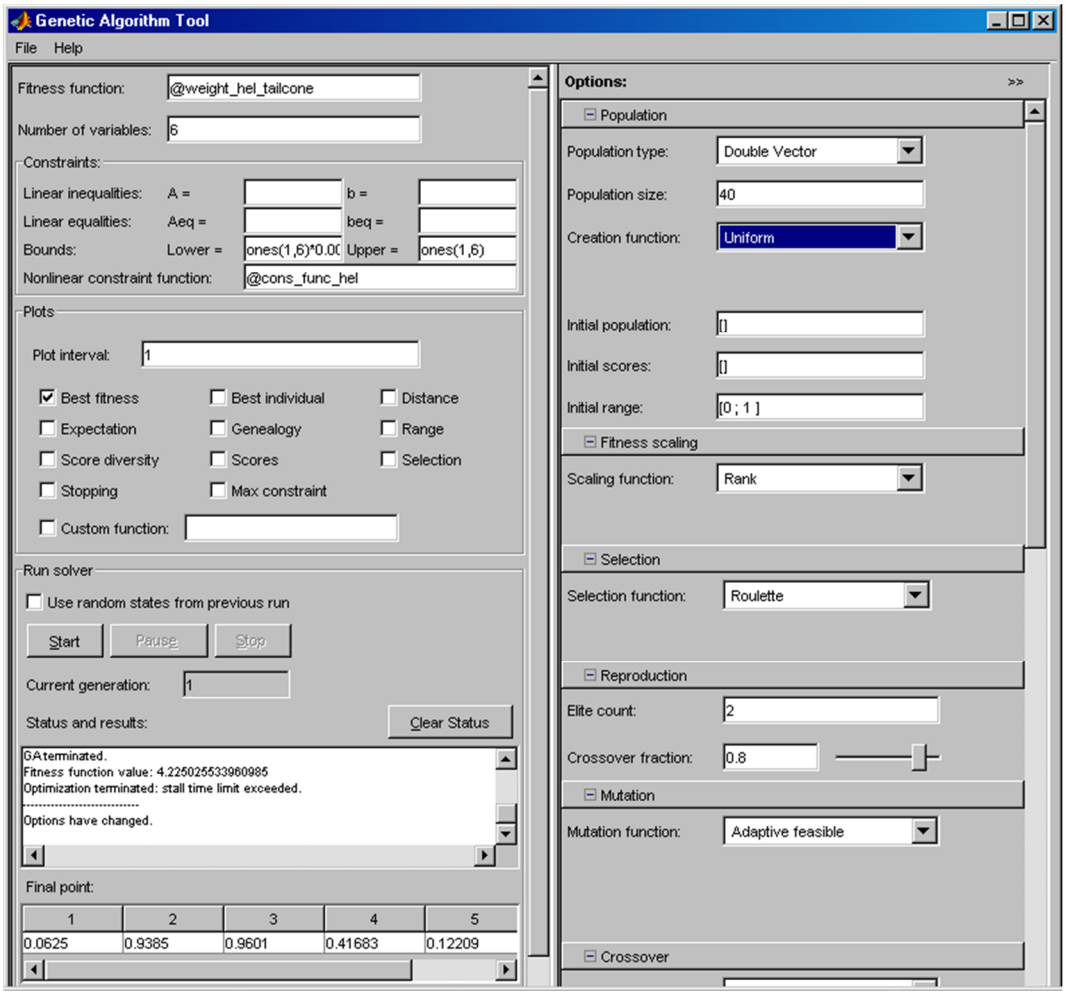


Figure 3.11. Genetic Algorithm Toolbox

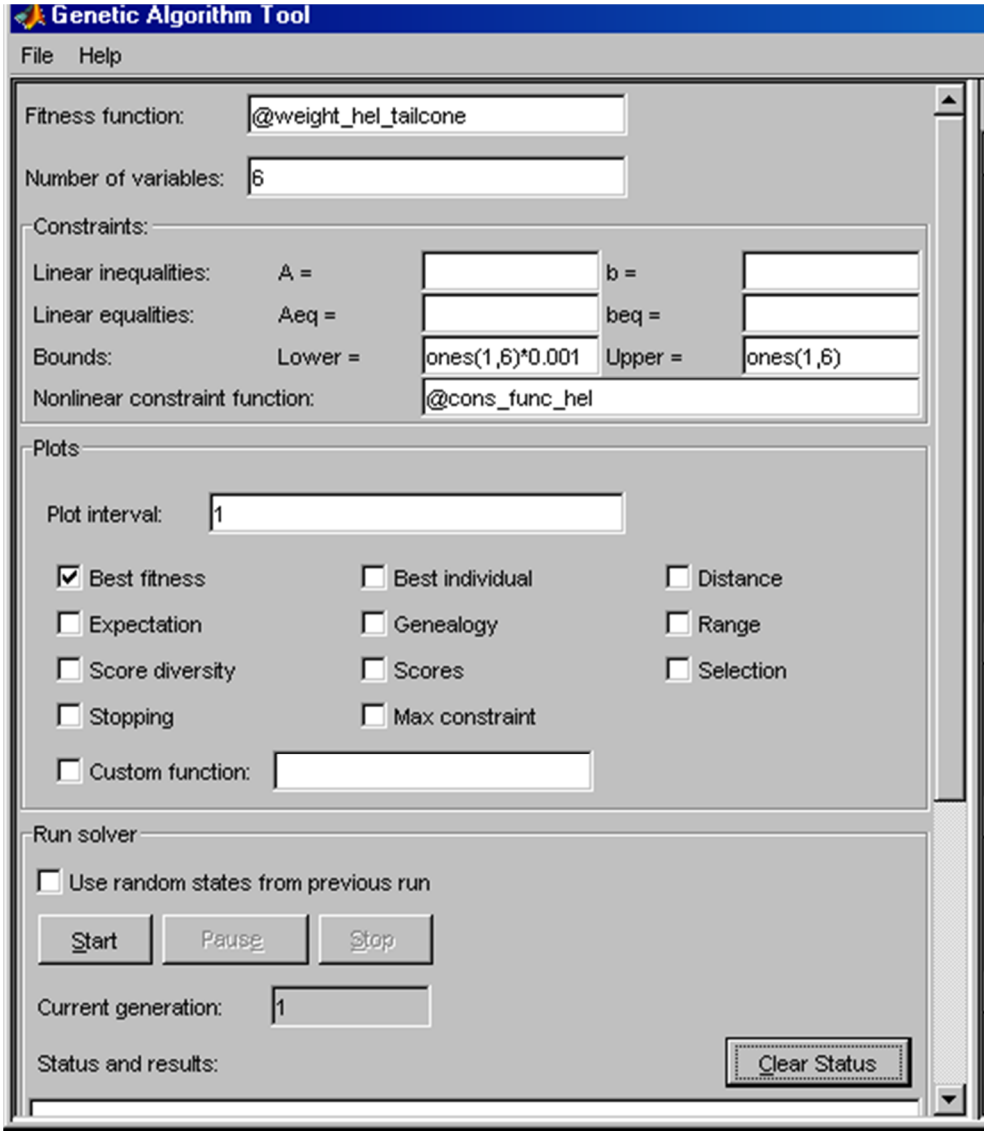


Figure 3.12. Problem inputs

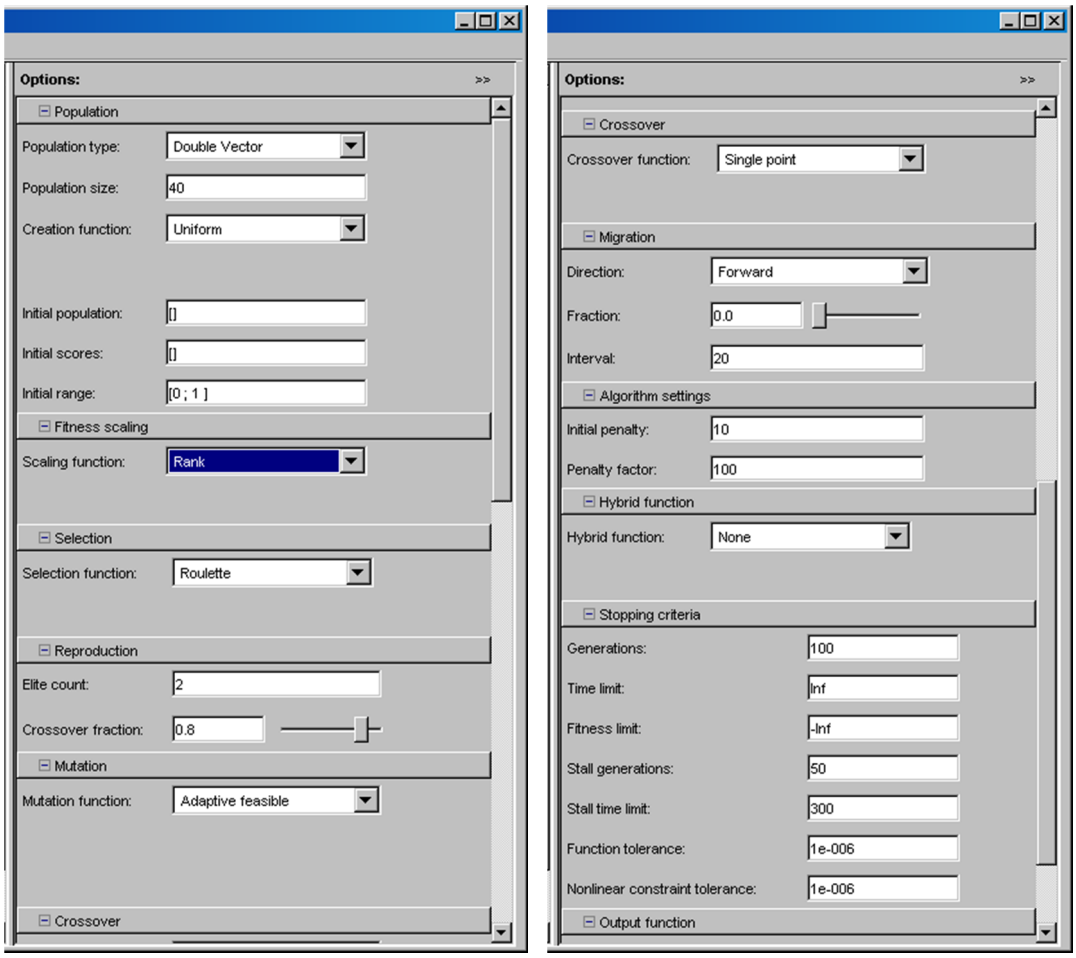


Figure 3.13. Genetic Algorithm Toolbox Options.

CHAPTER 4

APPLICATIONS OF TRUSS DESIGN OPTIMIZATION

The genetic algorithm code that is developed is tested with 2 benchmark studies, a 10-bar planar truss and a classic 25-bar space truss problems. These truss structures are commonly used to verify new structural optimization algorithms developed by researchers.

Both size and shape optimization is performed and optimized design configurations are compared with the references which had already studied these structures [1][5][9][11][12] and “Genetic Algorithm and Direct Search Toolbox” in MATLAB.

As final case study, size optimization of the tailcone truss structure of the helicopter “Aerospatiale SA-318C (Alouette II)” is also performed. Results of the present study are checked with the results obtained by the “Genetic Algorithm Toolbox” in MATLAB.

In the optimization procedure, following steps are taken;

- First algorithm is run twenty times for three different population size, chosen in the order of chromosome length.
- The population size which gives the best optimum and average mass results is accepted and used in the rest of the calculations. The low average mass implies that results obtained from any run can give a result which is close to the optimum mass. In a way, having low fluctuation of the results of the individual runs with respect to the average mass shows the stability of the algorithm at this population

size. Thus, independent from the initial randomly created population, the algorithm converges to the optimum result.

- By analyzing each problem, maximum generation number, crossover rate, mutation rate and elite number are modified.

After the determination of the inputs for the operators, algorithm is run twenty times and the best result is chosen as the optimum solution for the problem.

4.1 10-Bar Truss

The 10-bar truss benchmark is a well-defined structure with few variables and constraints. This structure is optimized in size and in size & shape for minimization of mass with stress and displacement constraints applied to every member and node respectively. Displacement constraints are not considered for the shape optimization. Also the weight of the truss itself is not taken into consideration in the calculation of the stress in both optimizations.

The dimensions, node numbers, element numbers, and loaded nodes are shown in Figure 4.1. The load values are presented in Table 4.1 and the material properties for the entire structure are given in Table 4.2.

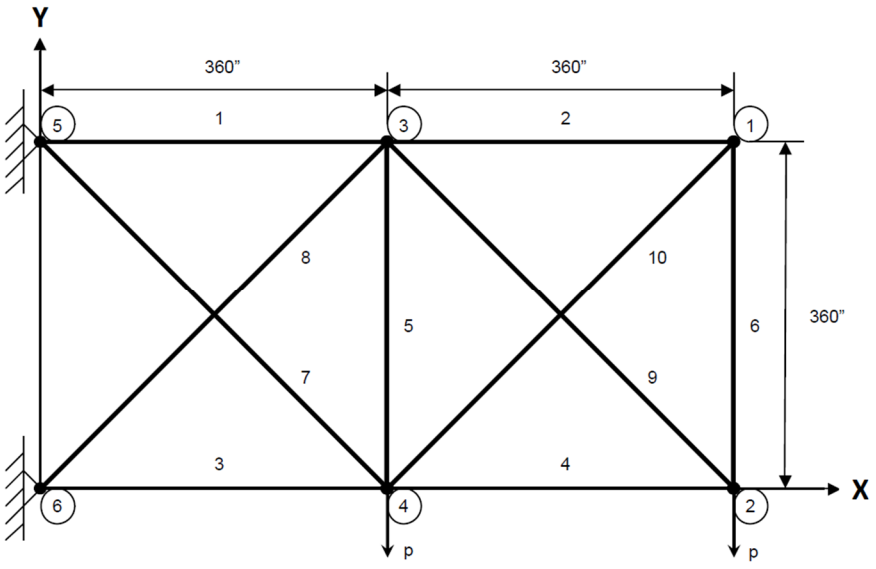


Figure 4.1. 10- Bar truss

Table 4.1. 10-Bar Truss Loads

Node	Load Case (lb)		
	x	y	z
2	0	100000	0
4	0	100000	0

Table 4.2. 10-Bar Truss Material

Material	*E (psi)*	v	σ_a *(psi)*	γ *(lb/in^3)*
Aluminum	10000000	0.334	25000	0.1

The inputs for the developed finite element code and genetic algorithm are given in Appendix A.

4.1.1 Size Optimization

In the size optimization, design variables are the cross sectional areas of 10 bars. The compressive and tensile stress limit for the truss members are specified as 25000 psi (172.37 MPa) and the maximum displacement limit for the nodes in the X and Y directions is taken as 2 in (50.8 mm).

Optimization is made by the use of discrete variables. The possible areas are taken from the American Institute of Steel Construction Manual, which are given in the following list [5];

S = (1.62, 1.80, 199, 2.13, 2.38, 2.62, 2.63, 2.88,2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49,4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.5, 13.5, 13.9, 14.2, 15.5, 16.0,16.9, 18.8, 19.9, 22.0, 22.9, 26.5, 30.0, 33.5) (in^2).

The advantage of using discrete area variables is that the result obtained from the GA gives more realistic and meaningful solutions to the problem having standard bars which can be available for the manufacturing.

In this list, there are 42 different areas. To be able to define these areas, the length of the gene must be at least 6. Because, is we use gene length of 5, for example, algorithm gives maximum 31 different possible values for each variable ((11111)binary=(31)decimal).

So, with 10 design variables (cross sectional areas) and the binary string length of 6 for each variable (gene), the chromosome length becomes 60. Three population size values are selected are 40, 60 and 80 from which the most appropriate population size is selected to be used in further analyses.

Determination of the population size

For every population size, minimum mass values and the average of the optimum mass results from all runs are shown in Table 4.3.

Table 4.3. 10 Bar size optimization results for 3 population size

Population size	Minimum mass (lb)	Average mass (lb)
40	5556,60	5677,32
60	5526,60	5641,23
80	5553,60	5610,48

Figures 4.2-4.4 show the comparison of the optimum mass result with the average mass obtained for each population size for 20 different runs performed.

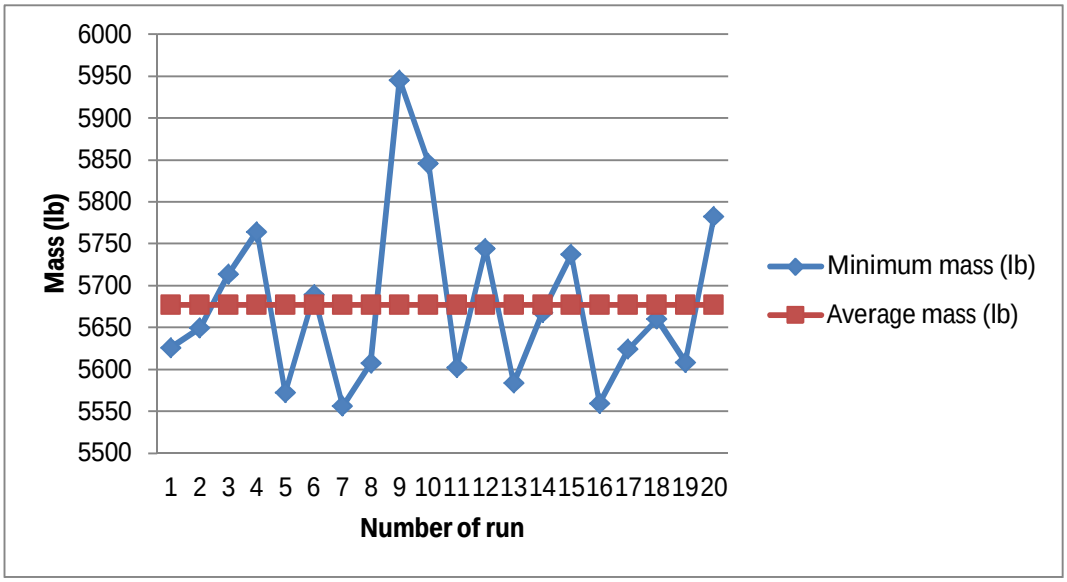


Figure 4.2. 10 Bar Planar Truss for population the size 40

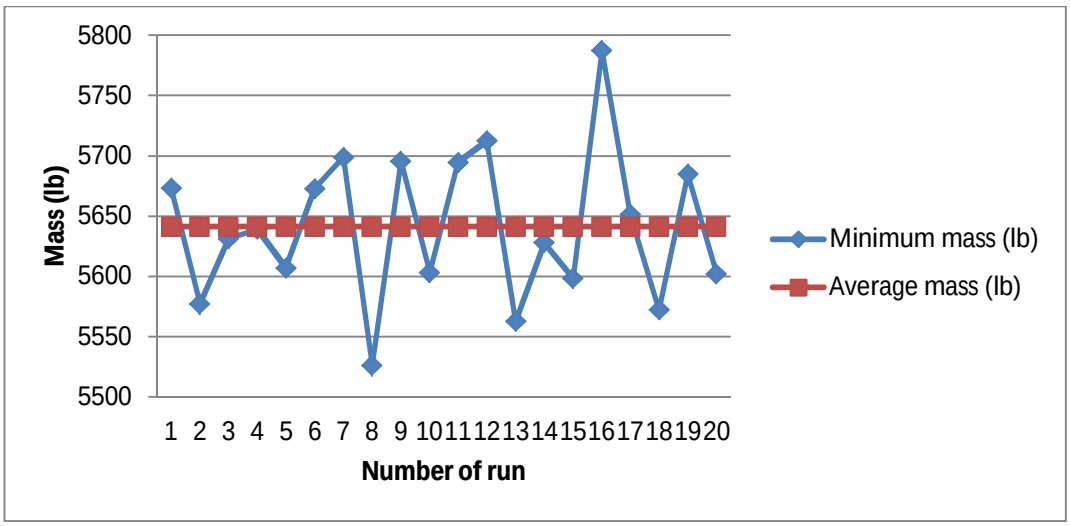


Figure 4.3. 10 Bar Planar Truss for the population size 60

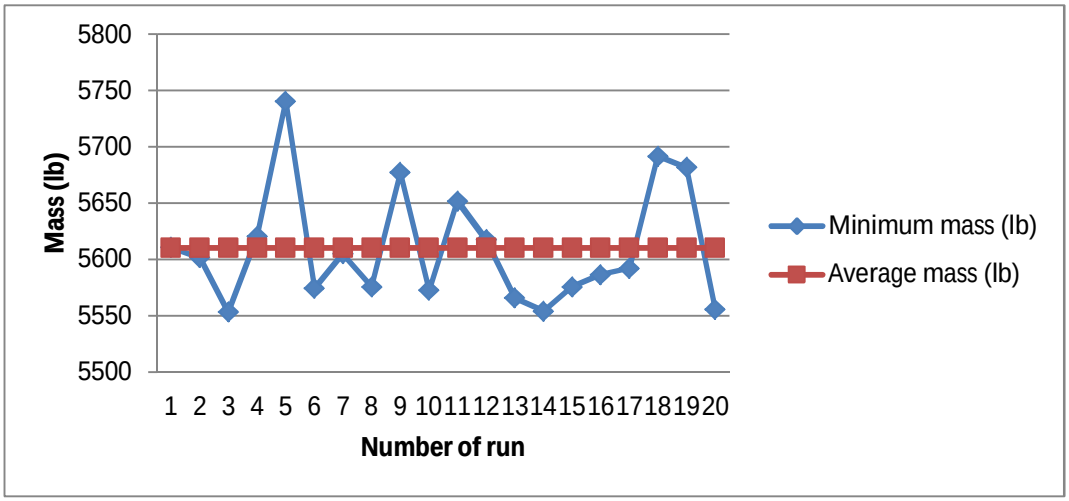


Figure 4.4. 10 Bar Planar Truss for the population size 80

Among these three populations, population size of 60 is chosen, because this population size gives the best optimum mass and considerably low average mass.

Optimization results

After the selection of the population size, by making a number of trials, the optimization parameters in algorithm for the size optimization of the 10 bar planar truss are determined. These parameters are given in Table 4.4.

Table 4.4. Optimization parameters for 10-bar-truss size optimization

Population Size	60
Gene Length	6
Maximum Generation Number	500
Crossover Rate	0,8
Mutation Rate	0,01
Elite Number	1

Figures 4.5 and 4.6 show the results of the 20 runs.

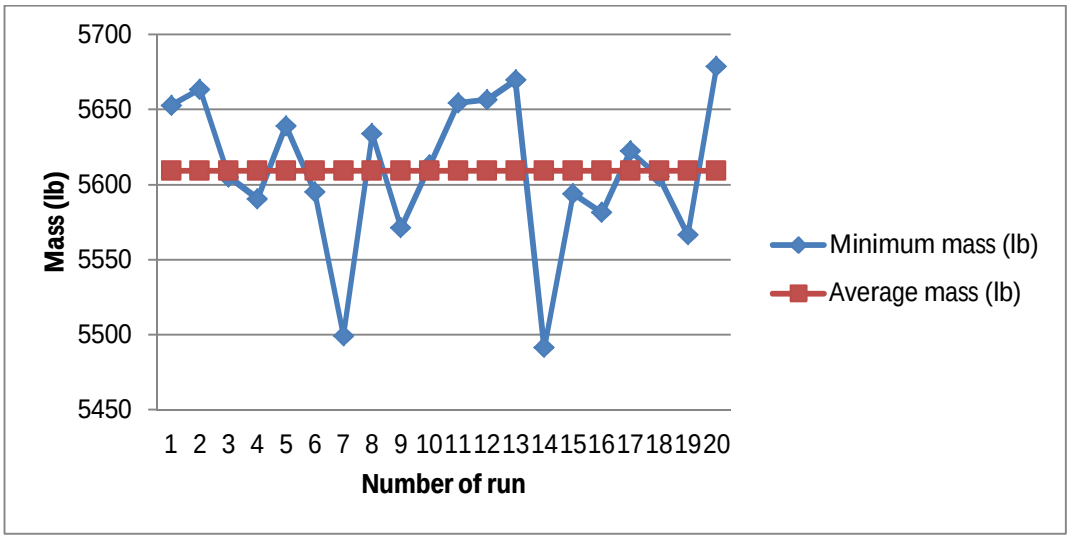


Figure 4.5. 10-bar-truss size optimization results for 20 runs

The optimum mass is found as 5491.70 lb (2490.99 kg). For the optimum mass solution, the mass change through iterations is shown in Figure 4.6. The mass of the structure decreases from the initial value of 7884 lb.

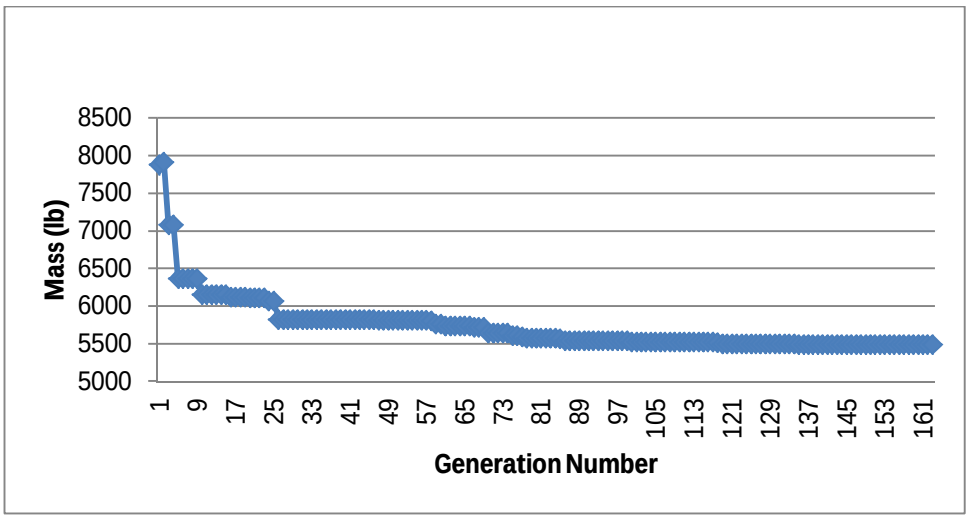


Figure 4.6. 10-Bar-truss size optimization, change in mass through generations

Table 4.5 compares the optimum cross-sectional areas obtained by the present analysis with the results of other studies.

Table 4.5. Comparison of the results with references

	Mass (lb)	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
		(in^2)									
Present	5491,71	33,50	1,62	22,90	15,50	1,62	1,62	7,97	22,00	22,00	1,62
MATLAB GATOOL	5543,00	33,50	1,80	26,50	14,20	1,80	1,80	7,97	19,90	18,80	1,80
Ref [5]	5563,00										
Ref [9]	5613,84	33,50	1,62	22,00	15,50	1,62	1,62	14,20	19,90	19,90	2,62
Ref[11]	5499,30	33,50	1,62	22,90	15,50	1,62	1,62	7,22	22,90	22,00	1,62
Ref[12]	5491,71	33,50	1,62	22,90	15,50	1,62	1,62	7,97	22,00	22,00	1,62

From Table 4.5, it is seen that the developed genetic algorithm gives one of the best results compared to the other references.

4.1.2 Size and Shape Optimization

In the second study, the 10–bar-truss structure is optimized both in size and shape simultaneously. To be able to perform shape optimization, some of the nodes are set free to move in a given design boundary. In this study, the nodes 1 and 3 are let free to move in both X and Y directions, and node 5 is allowed to move in Y direction. So there are 5 shape variables defined in the optimization problem. The design boundaries given for these nodal movements are given in Appendix A.

So in addition to the 10 size variables (cross sectional areas), 5 shape variables (the coordinates of the nodes which are allowed to move in the x and y directions) are added to the design variables.

In order to be able to compare the results of the present analysis with the independent study in the literature [1], design variables are taken as continuous variables instead of discrete. In the usage of the continuous design variables, there are intervals given as input for each variable from which solution for the related variable is chosen. So any number in the interval can be set to be value of the variable. This makes it possible to obtain better optimum results compared to the optimization made by discrete variables which constrains the variables to be assigned only to the given list

of standard areas. However, the optimum solutions obtained by continuous variables are not always possible to manufacture, i.e., practically infeasible.

It should be noted that in this study, displacement constraints are not taken into consideration, and only stress constraints are applied to each bar.

Determination of population size

In total, the optimization problem has 15 variables. The binary string length of each gene is 6 by providing maximum 63 different values to be assigned for each variable, which gives to chance of searching considerably large solution space. Thus, the chromosome length for each design becomes 90. Taking the value of the chromosome length as reference, in the selection of the population size, population sizes of 60, 90 and 120 are tried.

For every population size, minimum mass values and the average of the optimum mass results from all runs are presented in Table 4.6.

Table 4.6. 10 Bar size& shape optimization results for the 3 population sizes

Population size	Minimum mass (lb)	Average mass (lb)
60	1341,40	1483,81
90	1323,40	1429,70
120	1307,70	1442,27

Figures 4.7-4.9 show the comparison of the optimum mass result with average mass for the population sizes 60, 90 and 120, respectively.

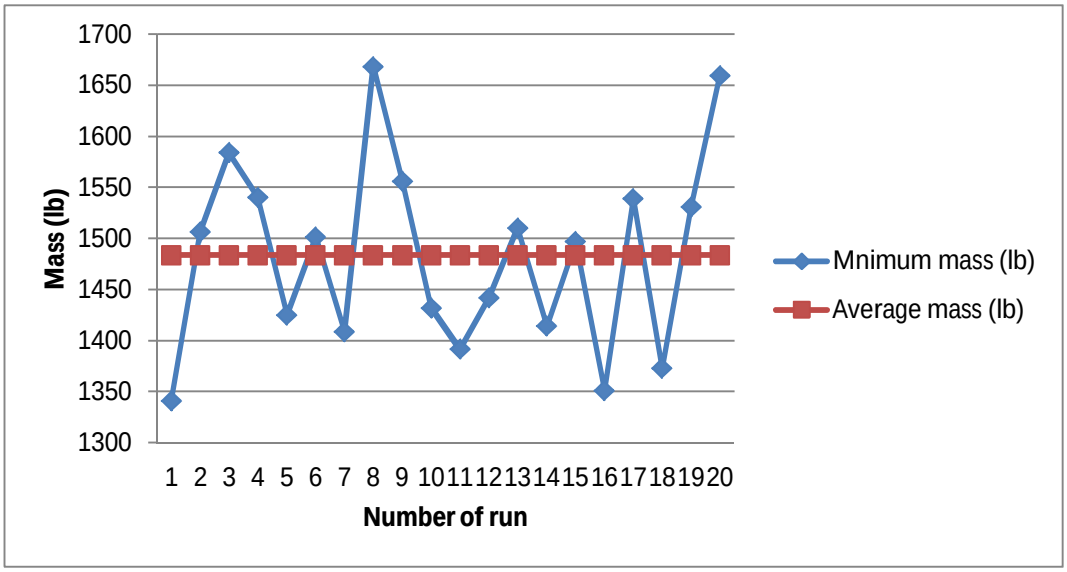


Figure 4.7. 10 Bar truss size& shape optimization for the pop. size 60

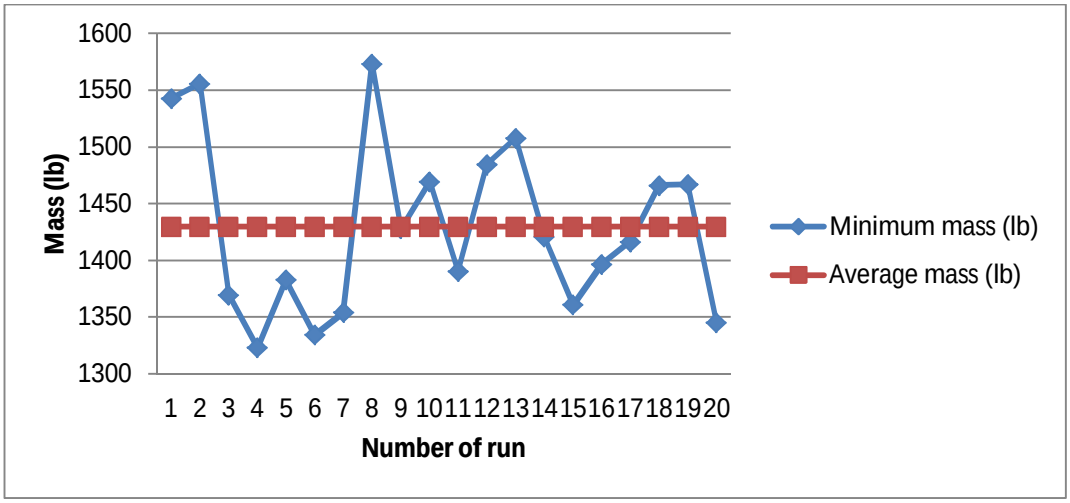


Figure 4.8. 10 Bar truss size& shape optimization for the pop. size 90

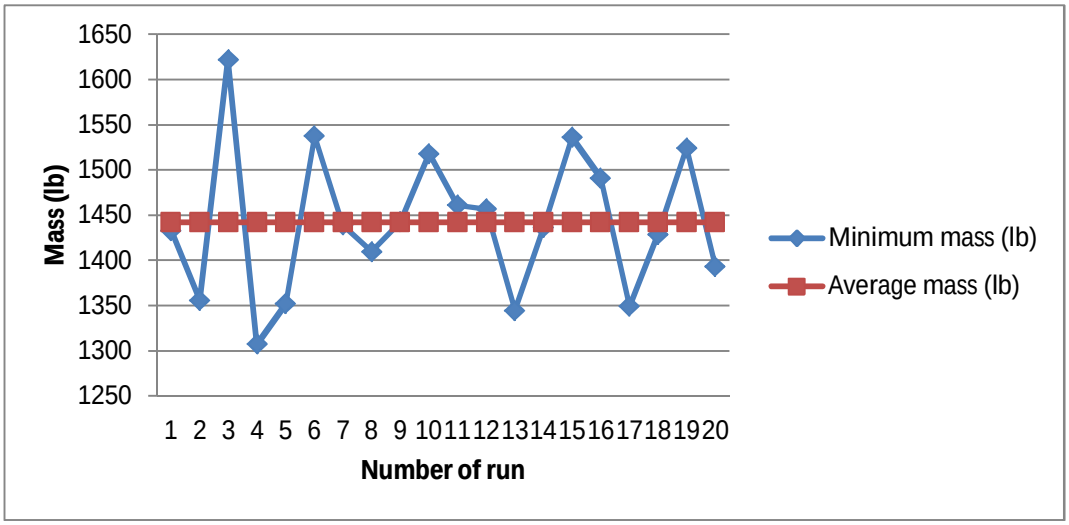


Figure 4.9. 10 Bar truss size& shape optimization for the pop. size 120

From Figs. 4.7-4.9 it is seen that the performance of the population size 90 gives the best optimum mass with the least average mass and with least fluctuation of the individual results with respect to the average mass. So this population size is chosen for the rest of the analysis.

Optimization results

The optimization parameters used in algorithm for the size & shape optimization of the 10 bar planar truss is given in Table 4.7.

Table 4.7. Optimization parameters for 10 bar size& shape optimization

Population Size	90
Gene Length	6
Maximum Generation Number	1000
Crossover Rate	1
Mutation Rate	0,02
Elite Number	1

A separate 20 different optimization runs are performed with a population size of 75, and the results obtained from these 20 runs are shown in Figure 4.10. The optimum mass for the structure is found as 1308.30 lb (593.43 kg).

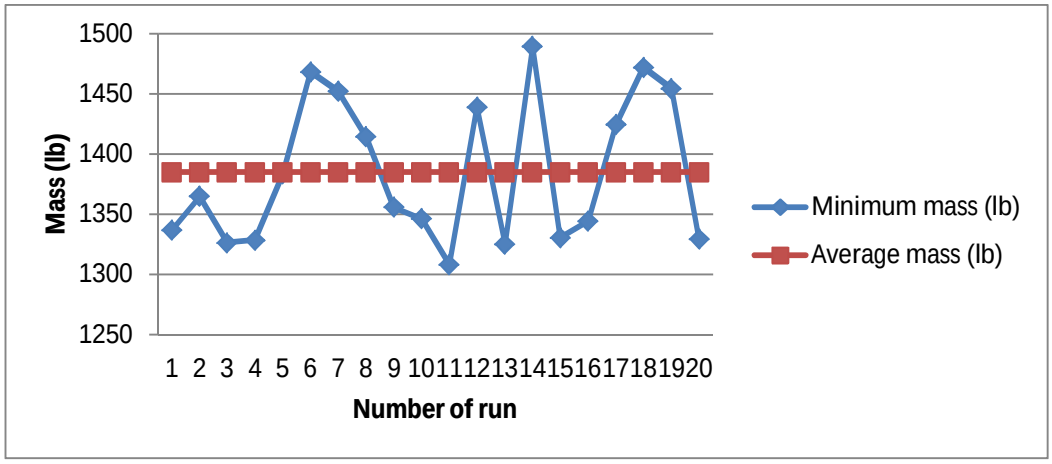


Figure 4.10. 10-bar-truss size& shape optimization results for 20 runs

For the optimum mass configuration obtained as a result of 20 different runs, change in the mass from 2463 lb to final value of 1308.30 lb is given in Figure 4.11.

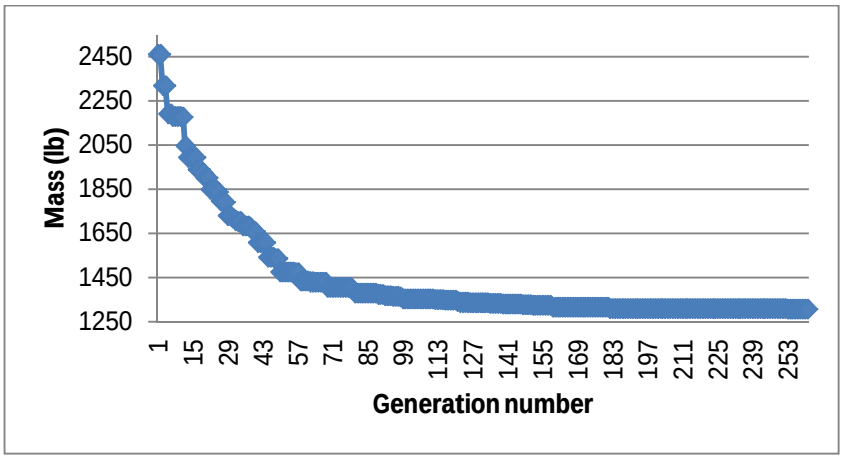


Figure 4.11. The mass change through generations

Table 4.8. Comparison of the area and mass results with references

	Mass (lb)	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
		(in^2)									
Present	1308,43	5,60	0,73	4,50	1,99	0,18	0,41	4,81	3,56	4,19	0,12
MATLAB GATOOL	1320,41	3,88	0,65	5,81	1,94	4,52	0,65	8,39	0,65	3,88	0,65
Ref [1]	1236,46	4,88	0,10	4,17	2,12	0,10	0,10	4,48	2,52	4,43	0,10

Optimum mass determined by the developed code is very close to the optimum mass determined by the genetic algorithm toolbox of Matlab but there is some discrepancy with the optimum mass obtained in Reference [1] as shown in Table 4.9. It can be improved by increasing the gene length and population size. By increasing gene length, the maximum number value of each gene is increased, and this gives the chance of searching solution space in more detail. In the same manner, if the population size increases, search space becomes larger, but this will cause spending more computation time.

The results for the new coordinates of the nodes are presented in Table 4.9.

Table 4.9. The new nodal coordinates after size& shape optimization

Node	Present		MATLAB GATOOL		Ref [1]	
	X	Y	X	Y	X	Y
1	508,95	261,59	716,16	228,92	642,43	143,56
2	720,00	0	720,00	0	720,00	0
3	569,05	307,94	593,22	359,58	523,36	371,96
4	360,00	0	360,00	0	360,00	0
5	0	568,57	0	704,89	0	694,25
6	0	0,00	0	0	0	0

4.2 25-Bar Space Truss

The 25-bar truss benchmark is a simple, space truss structure with many variables, constraints, and loading conditions. This structure is optimized for minimization of mass subject to loads given in Table 4.10, with stress constraints applied to every member and displacement constraints of 2.0 inches applied at each node in all three coordinate directions [1]. The weight of the truss structure is neglected.

Table 4.10. 25-Bar Space Truss Loads

Node	Load Case (lb)		
	x	y	z
1	0	20000	-5000
2	0	20000	-5000

The material for the entire structure is aluminum, properties of which are given in Table 4.11.

Table 4.11. 25-Bar Truss Material

Material	E (psi)	v	σ_a (psi)	γ (lb/in^3)
Aluminum	10000000	0.334	35294	0.1

The dimensions, node numbers, element numbers, and constraints for this structure are shown in Figure 4.12. The design variables consist of element cross sectional areas for size and nodal positions for shape optimization. Continuous design variables are used for both size and size & shape optimization of the 25 bar space truss as in the case of reference [1].

Inputs for the finite element analysis and the optimization algorithm for 25-bar truss structure are given in Appendix B.

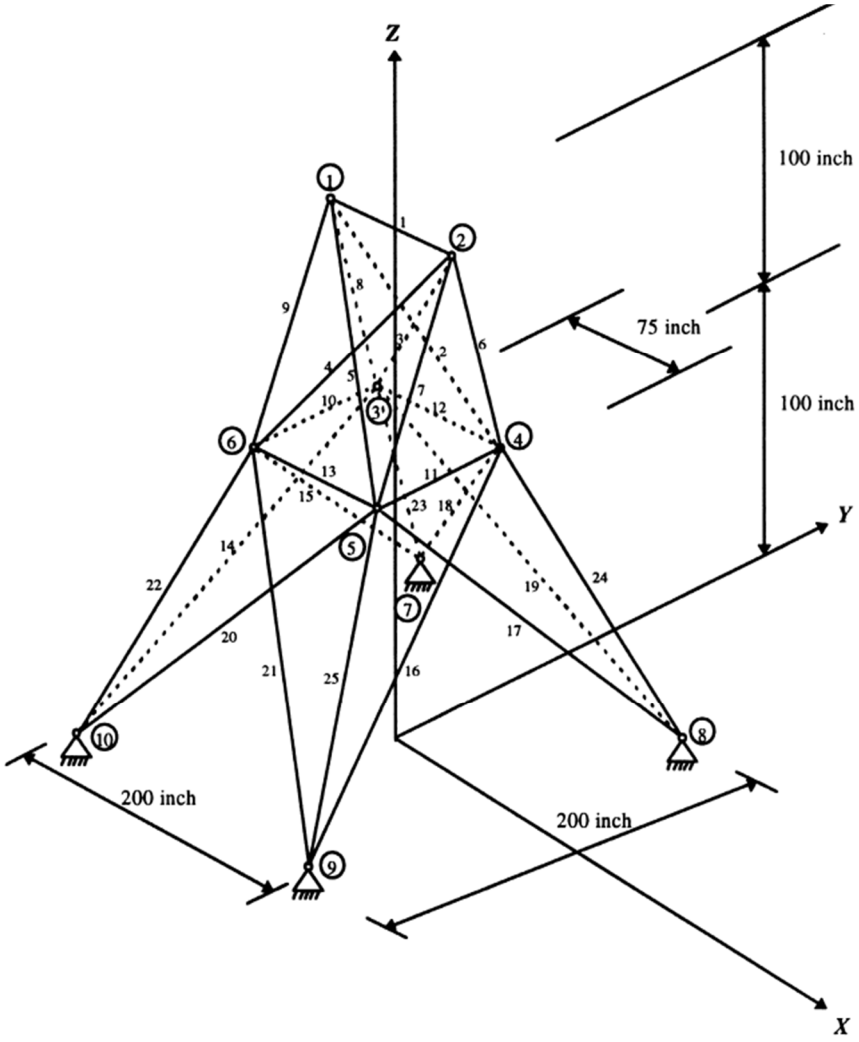


Figure 4.12. 25-Bar Space Truss[1]

4.2.1 Size Optimization

The structure is composed of 25 bars. But to reduce the computation time, the symmetry of the structure is used. The cross sectional areas are grouped according to Table 4.12. By using the symmetry of the structure, the number of the size variables decreases from 25 to 8, meaning that every element in each group has the same cross sectional area, but the cross sectional areas of elements in each group are allowed to vary independently.

Table 4.12. Size Symmetry Variables

Variable	Element
1	1
2	2-5
3	6-9
4	10-11
5	12-13
6	14-17
7	18-21
8	22-25

Determination of population size

The binary length of each variable is taken 6 setting maximum number of possible area values to 63. So chromosome length is calculated as 48, with 8 design variables. The population sizes are selected as 30, 50 and 70 considering the chromosome length, and 20 different runs are executed for each population size to decide on an appropriate population size to perform the size optimization.

For each population size, minimum mass values and the average of the optimum mass results from 20 different runs are shown in Table 4.13.

Table 4.13. 25 Bar size optimization results for 3 population sizes

Population size	Minimum mass (lb)	Average mass (lb)
30	163,89	182,01
50	164,85	173,52
70	163,89	172,57

The least minimum and average mass are obtained by the population size 70. But to overcome the high computation time, the population 50 with very close values to the results of the population size 70, is chosen as the population size in the further optimization analyses.

Figures 4.14-4.16 show the comparison of the optimum mass result obtained in each run with average mass determined for each population size.

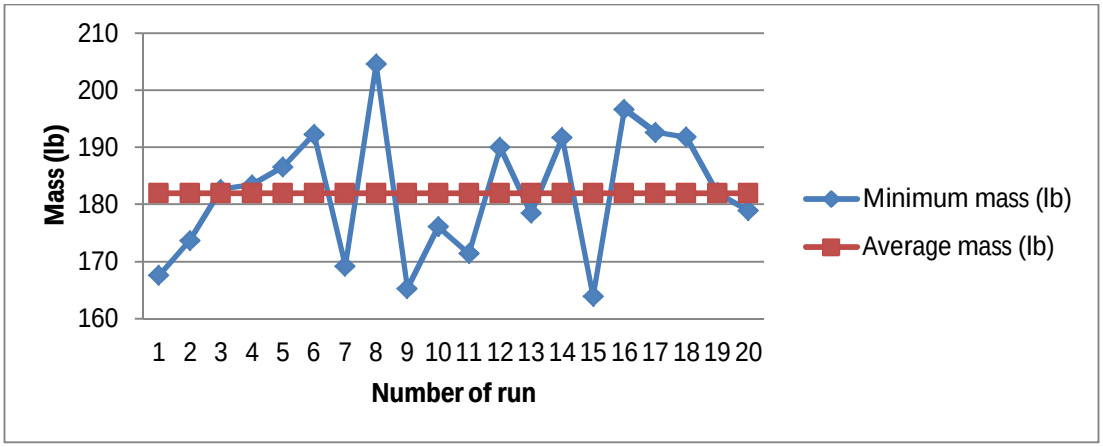


Figure 4.13. 25 Bar truss size optimization for the population size 30

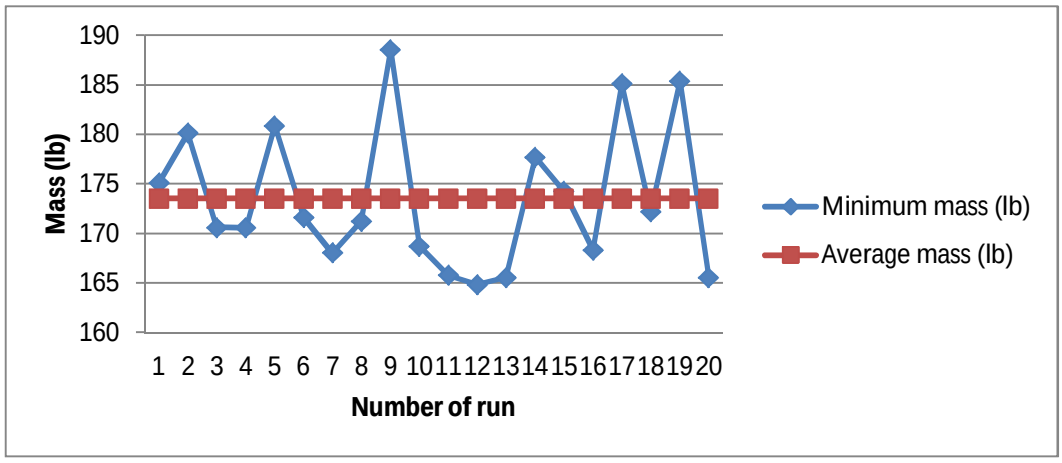


Figure 4.14. 25 Bar truss size optimization for the population size 50

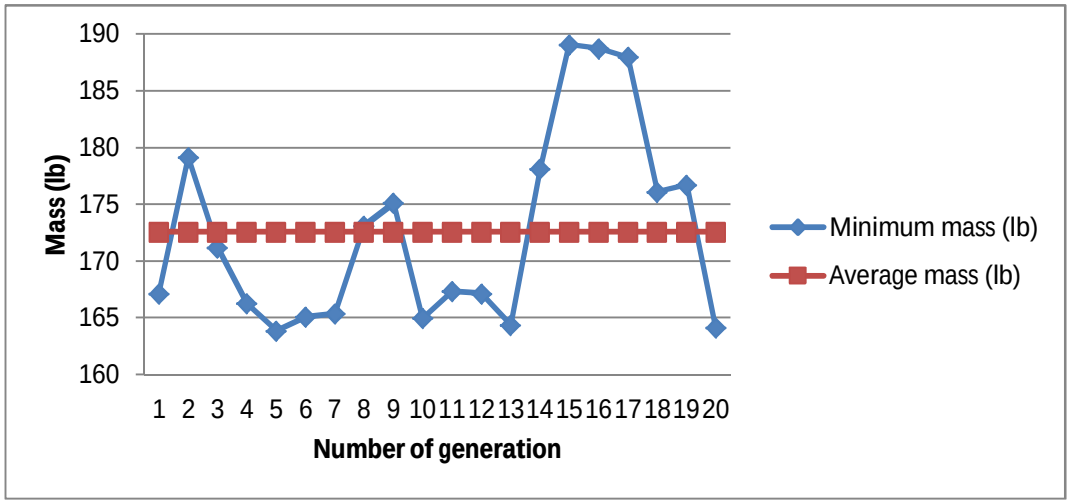


Figure 4.15. 25 Bar truss size optimization for the population size 70

Optimization results

The optimization parameters used in the genetic algorithm for the further size optimization of the 25 bar space truss is summarized in Table 4.14.

Table 4.14. Optimization parameters for size optimization of 25-bar-truss

Population Size	50
Gene Length	6
Maximum Generation Number	200
Crossover Rate	0,8
Mutation Rate	0,01
Elite Number	1

For the population size of 50, the optimum mass is determined to be 164.85 lb (74.77 kg) as seen in Table 4.13. The mass variation of the optimum configuration is shown in Figure 4.16. The optimization starts from 473.7 lb.

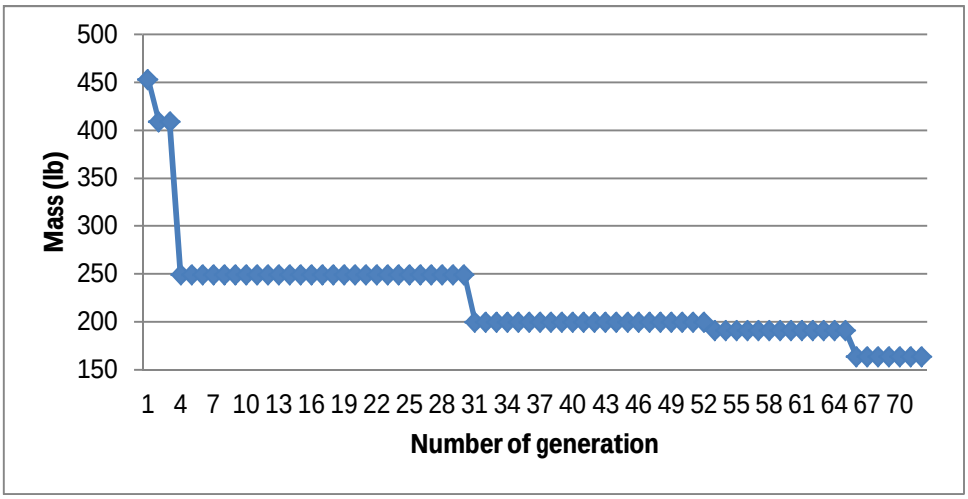


Figure 4.16. The mass change through generations

The optimum mass is compared with the findings of the other studies in the literature in Table 4.15. Table 4.14 shows that in the present study, the genetic algorithm developed gives very reasonable result which is very close to the value calculated by the MATLAB Genetic Algorithm Tool and the optimum mass determined in Reference [1].

Table 4.15. Comparison of the optimum mass and cross-sectional area results

	Mass (lb)	A1	A2	A3	A4	A5	A6	A7	A8
		(in^2)							
Present	164,84	0,07	0,07	1,30	0,06	0,57	0,29	0,06	1,30
MATLAB GATOOL	162,26	0,43	0,43	0,55	0,51	0,82	0,49	0,47	0,43
Ref [1]	162,28	0,01	0,03	1,30	0,01	0.71	0,36	0,01	1,27

4.2.2 Size and Shape Optimization

In In this study, in addition to the size optimization, the developed genetic algorithm also performs shape optimization at the same time.

In shape optimization, the symmetry of nodes is also used. Shape symmetry is made as described in Table 4.16. Nodes 1 and 2 remain as fixed nodes. Nodes 3 through 6 are permitted to move in any direction parallel to the xy-plane at the fixed elevation of 100 inches. Nodes 7 through 10 are permitted to move in any direction along the xy-plane. There are two shape variables for each node, which are the x and y displacements of the nodes. Thus, the shape symmetry reduces the number of variables from 16 to 4. So the problem can be defined with 4 independent shape variables and 12 dependent shape variables [1].

Table 4.16. Shape Symmetry Rules

Node	Symmetric with	About Plane
3	4	yz
5	4	xz
6	3	xz
7	8	yz
9	8	xz
10	7	xz

Determination of the population size

From the combination of the independent size and shape variables, 8 and 4 respectively, the total number of the variables is set to be 12. With a gene length of 6, the chromosome becomes 72. For this problem, in the initial phase population sizes of 40, 70 and 100 are selected to carry out runs to decide on a proper population size.

Optimized mass results obtained for the three different population sizes are given in Table 4.17.

Table 4.17. 25 Bar size& shape optimization results for 3 population sizes

Population size	Minimum mass (lb)	Average mass (lb)
50	72,90	85,77
70	67,60	80,18
90	66,97	79,71

Minimum mass and average masses obtained using population sizes of 50, 70 and 90 are presented in Figs. 4.18-4.20.

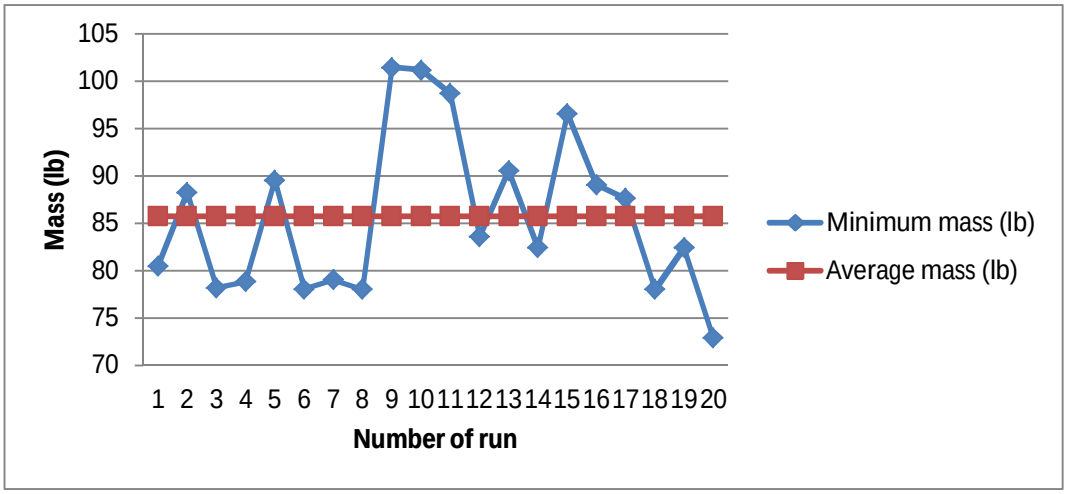


Figure 4.17. 25 Bar truss size& shape optimization for the pop. size 50

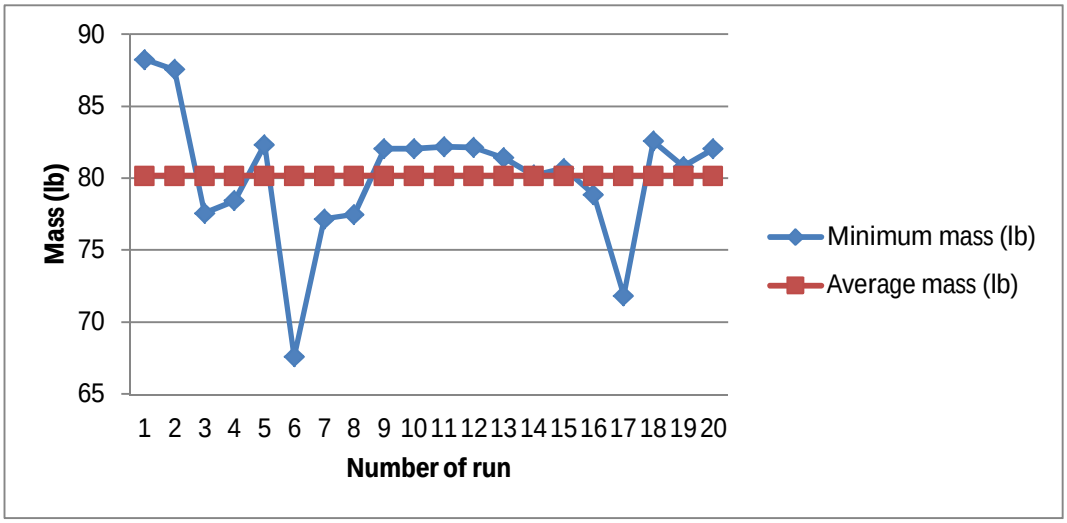


Figure 4.18. 25 Bar truss size& shape optimization for the pop. size 70

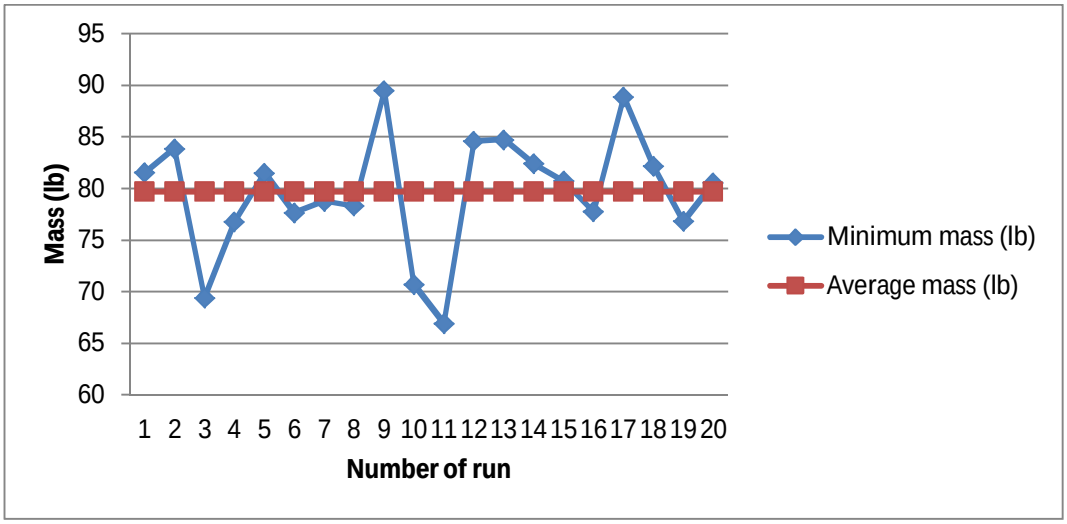


Figure 4.19. 25 Bar truss size& shape optimization for the pop. size 90

Looking at the Table 4.17 and in Figs. 4.18-4.20, the population size 100 gives the best minimum and average mass values. On the hand, the population size of 70 gives a very close minimum mass and average mass value with the corresponding values obtained with a population size of 100.

Leading to less computation time, the population size of 70 is selected to be used in the further optimization analyses.

Optimization results

The optimization parameters used in algorithm for the size and shape optimization of the 25 bar space truss is given in Table 4.18.

Table 4.18. Optimization parameters for size& shape optimization of the 25-bar-truss

Population Size	70
Gene Length	6
Maximum Generation Number	400
Crossover Rate	0,8
Mutation Rate	0,01
Elite Number	1

The optimum mass is determined as 66.97 lb (30.38 kg) as shown in Figure 4.20. The mass variation of the optimized configuration with the generations is shown in Figure 4.21. The mass of the total structure starts to decrease from 328.4 lb.

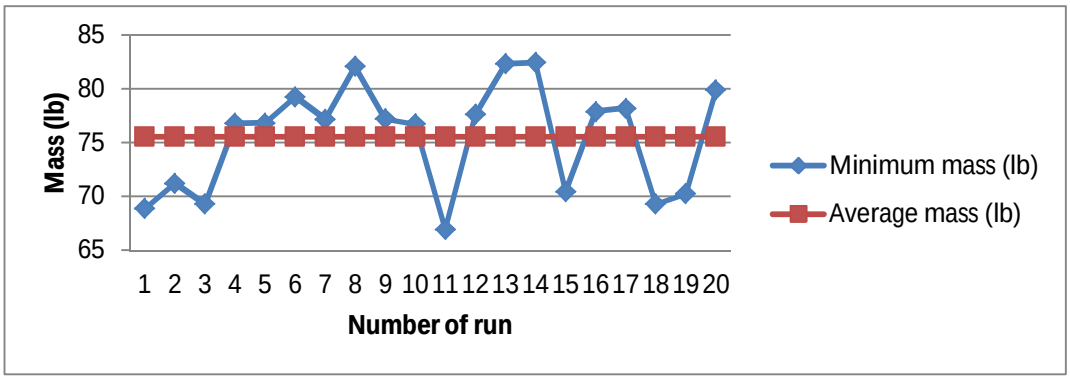


Figure 4.20. 25-bar-truss size & shape optimization results for 20 runs

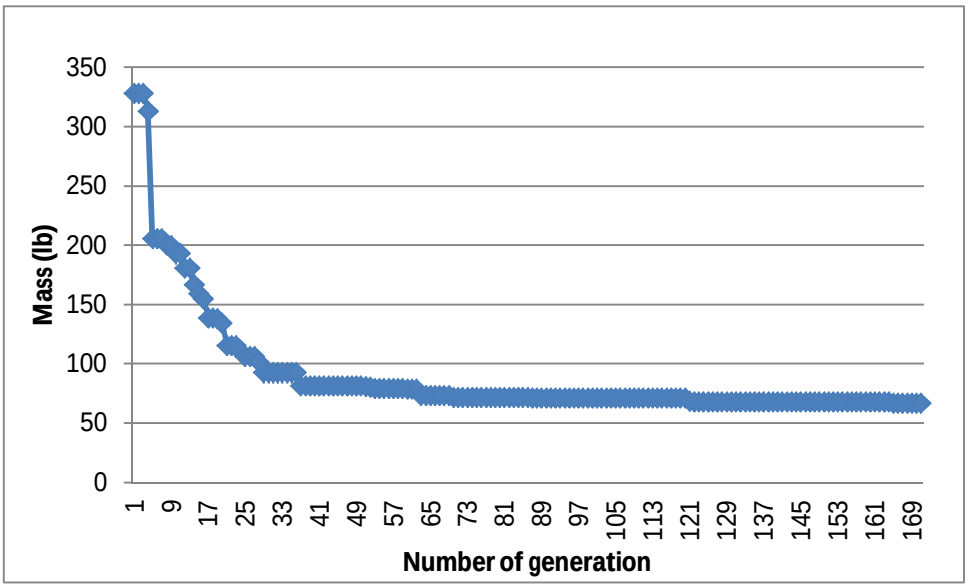


Figure 4.21. The mass change through generations

The optimum mass and cross-sectional areas determined by the genetic algorithm developed in the present study are compared with the results obtained by the genetic algorithm toolbox of Matlab, and results of Reference [1] in Table 4.18. From Fig. 4.18, it can be seen that present study gives better result than the genetic algorithm toolbox of Matlab. But the result is a bit higher from the result given by reference [1].

Table 4.19. Comparison of the area and mass results with references

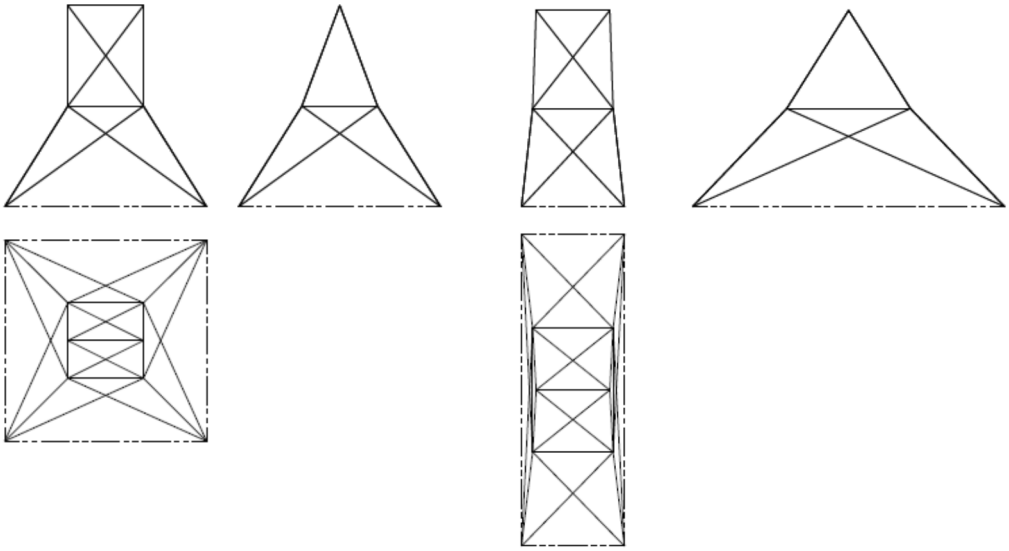
	Mass (lb)	A1	A2	A3	A4	A5	A6	A7	A8
		(in^2)							
Present	66,97	0,058	0,058	0,541	0,058	0,058	0,058	0,058	0,540
MATLAB GATOOL	83,23	0,310	0,310	0,320	0,310	0,310	0,310	0,340	0,340
Ref [1]	62,85	0,043	0,020	0,600	0,010	0,010	0,040	0,027	0,595

Final coordinates obtained by size & shape optimization is given by Table 4.20

The final shape can be seen in Figure 4.22.

Table 4.20. Comparison of the new nodal coordinates with references

Node	Present		MATLAB GATOOL		Ref [1]	
	X	Y	X	Y	X	Y
1	-37,5	0	-37,5	0	-37,5	0
2	37,5	0	37,5	0	37,5	0
3	-40,97	72,58	-14,50	37,66	-46,59	65,20
4	40,97	72,58	14,50	37,66	46,59	65,20
5	40,97	-72,58	14,50	-37,66	46,59	-65,20
6	-40,97	-72,58	-14,50	-37,66	-46,59	-65,20
7	-52,42	129,84	-26,66	56,58	-56,62	120,77
8	52,42	129,84	26,66	56,58	56,62	120,77
9	52,42	-129,84	26,66	-56,58	56,62	-120,77
10	-52,42	-129,84	-26,66	-56,58	-56,62	-120,77



Before optimization Optimized shape

Figure 4.22. Final shape for size& shape optimization of 25 bar truss

Results of the size only optimization and size & shape optimization for the 25-bar truss structure show that as long as the design boundaries give allowance to the nodal movements, there is a significant mass reduction in the structure. Under the same loading condition, size & shape optimization gives considerably low mass values.

4.3 Case Study

After trials on benchmark structures, the developed genetic algorithm code is used to optimize the size of the tail cone truss structure of the helicopter “Aerospatiale SA-318C (Alouette II)”with the objective of minimization of mass.

Tailcone structure has 62 bar elements and 24 nodes, as shown in Figure 4.23. The dimensions of the elements of truss structure are obtained by taking measurements on the helicopter which is in the METU Aerospace Engineering’s Hangar building. The basic dimensions of the helicopter are available in Reference [23]. The measured data for the structure and the inputs for the developed algorithms are given in Appendix C.

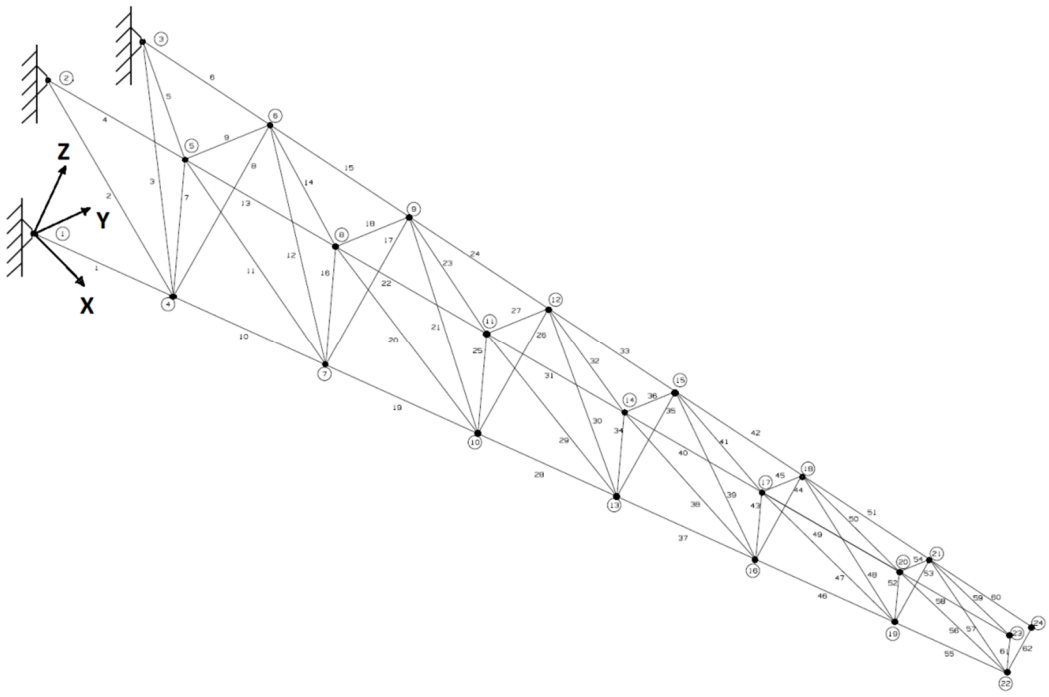


Figure 4.23. Elements and nodes of the tailcone structure

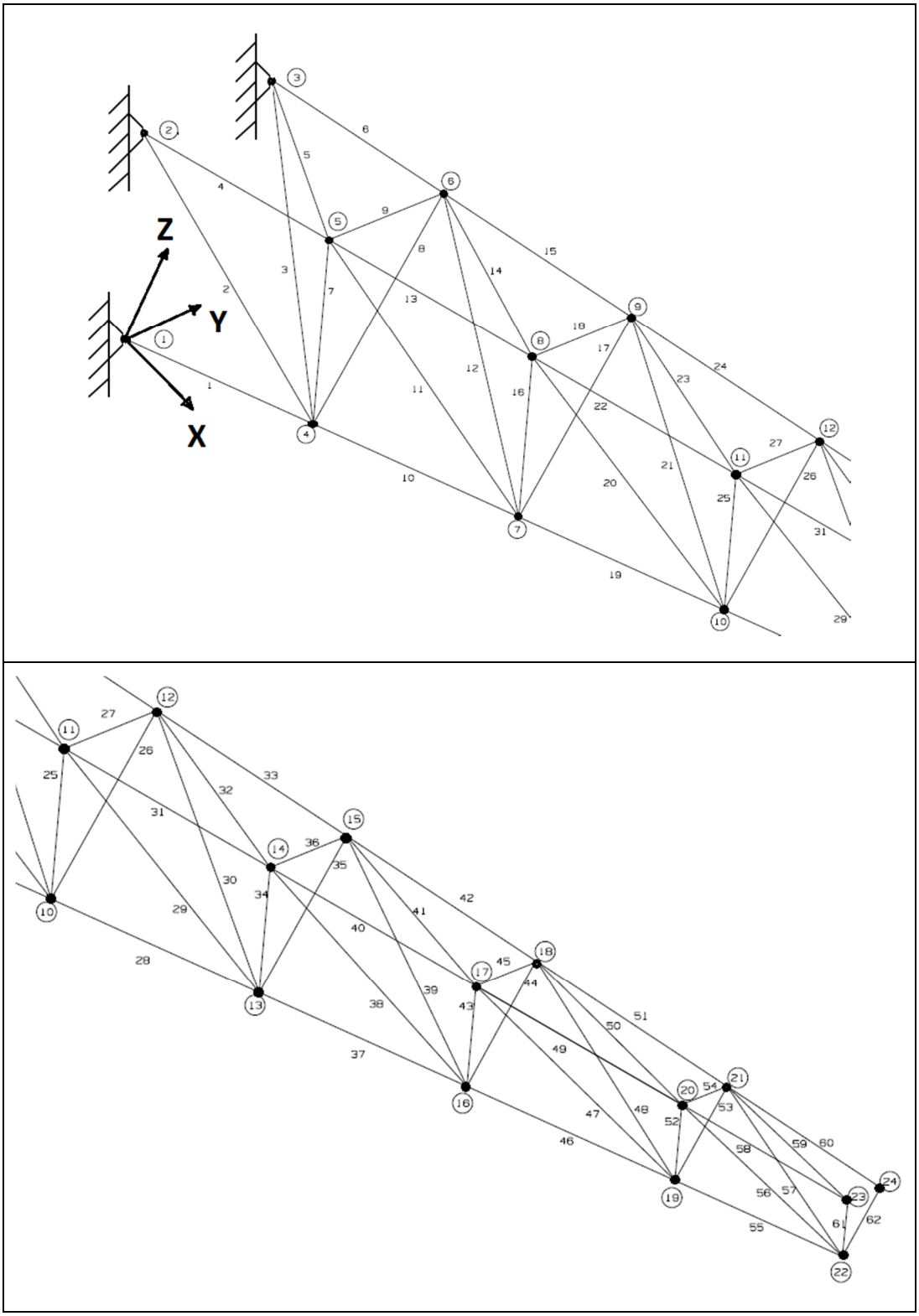
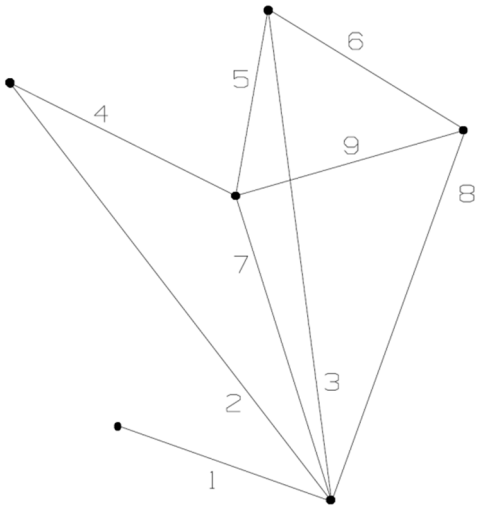


Figure 4.24. Elements and nodes of the tailcone structure (detail view)

The truss structure mainly has 7 sub-partitions which have the similar configuration for the elements and nodes, presented in Figure 4.25.

Based on the actual the measurement of the diameters of the bar elements of the truss structure, it is assumed that there are 6 groups of the bars in truss structure which have the same cross sectional area. The groups are listed for the total truss assembly in Table 4.21.



Variable	Element
1	1
2	4,6
3	5
4	9
5	7,8
6	2,3

Figure 4.25. Tailcone sub-partition

Table 4.21. Group of elements for size variables

Variable	Element
1	1,10,19,28,37,46,55
2	4,6,13,15,22,24,31,33,40,42,49,51,58,60
3	5,14,23,32,41,50,59
4	9,18,27,36,45,54
5	7,8,16,17,25,26,34,35,43,44,52,53,61,62
6	2,3,11,12,20,21,29,30,38,39,47,48,56,57

In this study, it is assumed that only external load is the force, coming from the tail rotor, acting on the tip of the tail cone. The force created by the tail rotor is calculated by the formula given in [22].

$$l * T_Y = C_R \tag{4.1}$$

where, C_R is the reaction torque of the main rotor, T_Y is the thrust of the tail rotor and l is the lever arm as illustrated Figure 4.26. C_R is specified in reference [24] as 488.9 Nm. Since no dimension could be found for lever arm from the references listed for the helicopter, lever arm length is determined by making proportion calculations from the drawings shown in Figure 4.27 (dimensions are in mm), and it is determined to be approximately 6 meters. Thrust of the tail rotor is taken 81.5 N, and half of this force is assumed to be acting on nodes 23 and 24, shown in Figure 4.23.

For this structure, two analyses are made; one is the case when truss weight is neglected and the other case is that the element weight is considered in the stress calculations. By this analysis, the effect of the weight of the structure, which is generally is not taken into consideration, is analyzed on the mass optimization of the structure.

There is a part of the tail rotor mounted at the nodes 23 and 24. This part is assumed to be an element of truss with a cross sectional area much higher than the truss elements. Therefore, the distance between these nodes is assumed to be fixed.

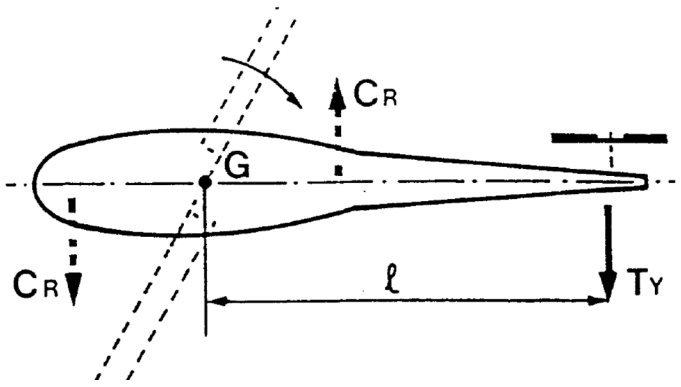


Figure 4.26. Force diagram for the tail rotor thrust [22]

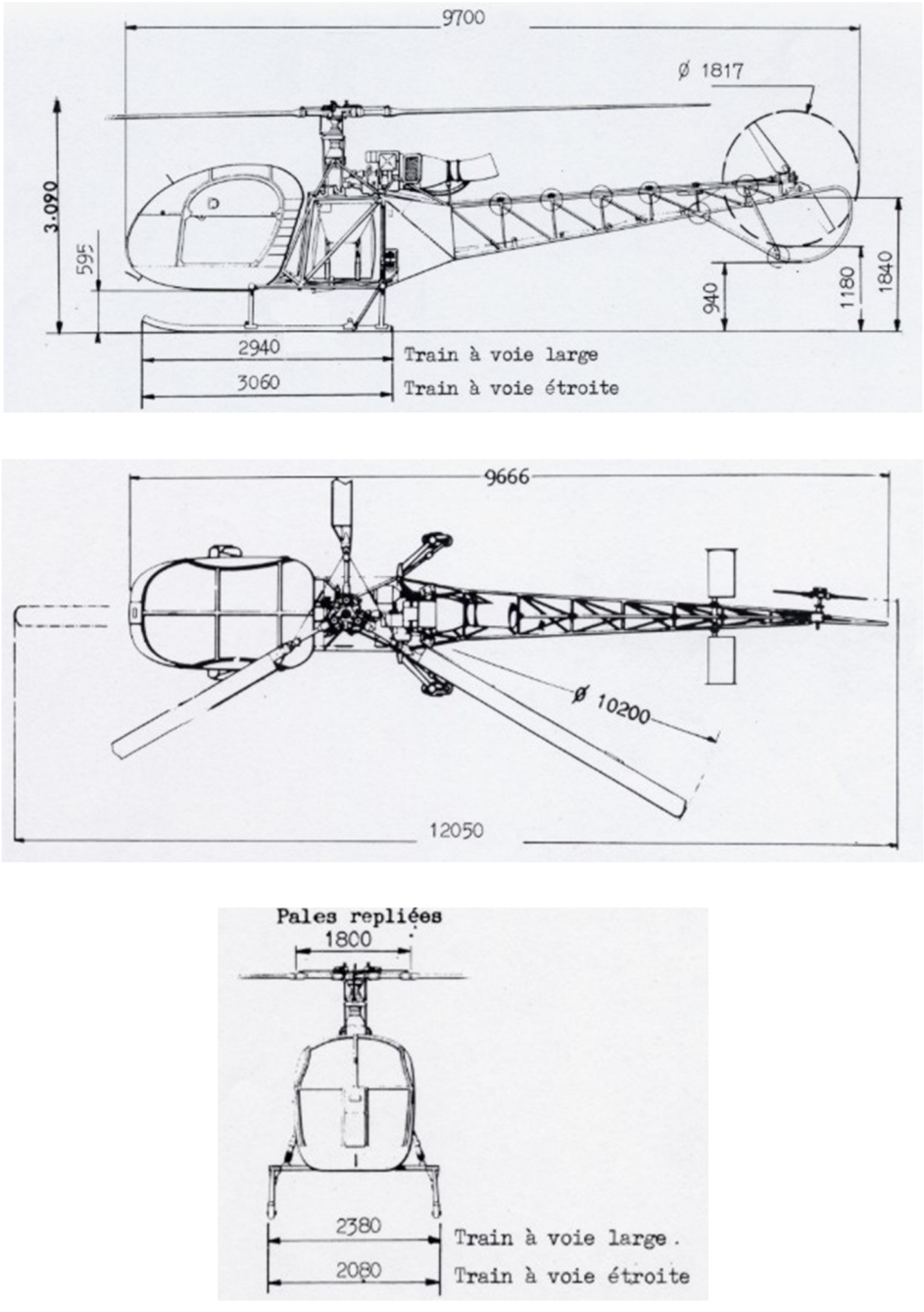


Figure 4.27. Side, top and front views of AerospatialeSA-318C [23]

The tailcone structure is assumed to be fixed at nodes 1, 2 and 3 to the helicopter fuselage. The size optimization of the structure is performed with the objective of minimizing mass under the stress constraints applied to each member and displacement constraints applied to each node of the truss structure. The maximum displacement constraint is taken as 5 mm.

The material of the elements is assumed to be Aluminum. Properties of the Aluminum is given in Table 4.22 where σ_a is maximum allowable stress for aluminum.

Table 4.22. Material properties of the tailcone truss structure

Material	*E*	σ_a	γ
Aluminum	70 GPa	172.37 MPa	2768 kg/m^3

4.3.1 Size optimization

The tailcone truss structure has 6 independent, 56 dependent size variables (cross sectional areas) as indicated in Table 4.21. The gene length is taken 6. Thus, total chromosome length becomes 36. From the trials made in benchmark studies, the results of which verified that population size should be in the order of chromosome length, population size is determined as 40. Other optimization parameters used are listed in Table 4.23.

In the optimization, continuous variables are used not to be restricted with the standard area variables. If desired, the final result for variables for the optimum mass configuration can be assigned to the nearest standard area. By this way, advantage of continuous variables making use of maximum number of different values defined by the length of the gene is preserved.

Table 4.23. Optimization parameters for the size optimization of the tailcone truss structure

Population Size	40
Gene Length	6
Maximum Generation Number	100
Crossover Rate	0,8
Mutation Rate	0,02
Elite Number	1

Case 1: Optimization without element weight loading

The optimum mass of the tailcone, without considering the element weight loading, is determined as 3.196 kg after 20 separate runs. The results of the 20 runs are shown in Figure 4.28.

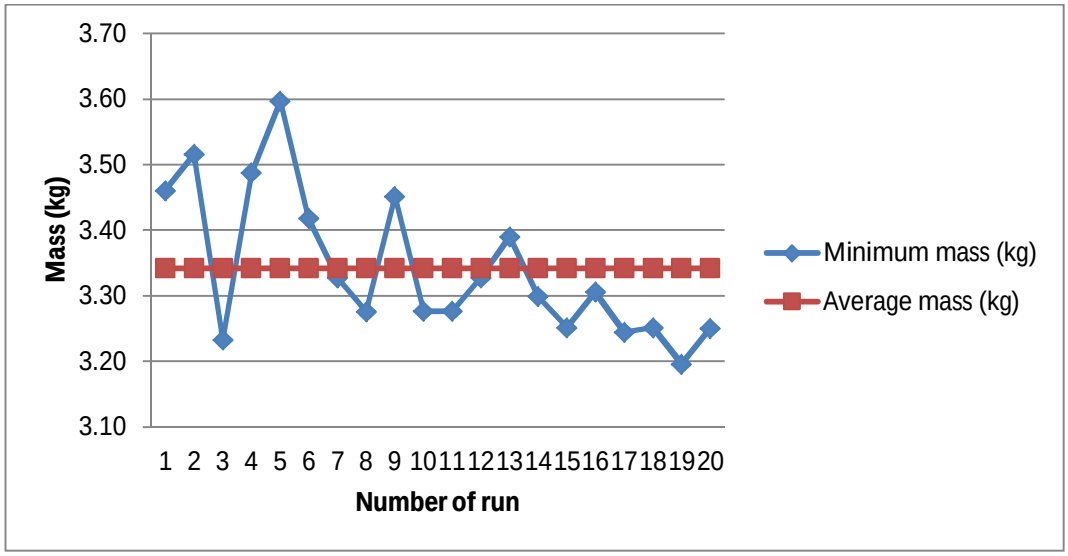


Figure 4.28. Tailcone truss structure size optimization results for 20 runs

The variation of the mass of the tailcone with the generations for the run which yielded minimum mass is given in Figure 4.29. As can be seen from the figure, the optimization starts from 5.286 kg.

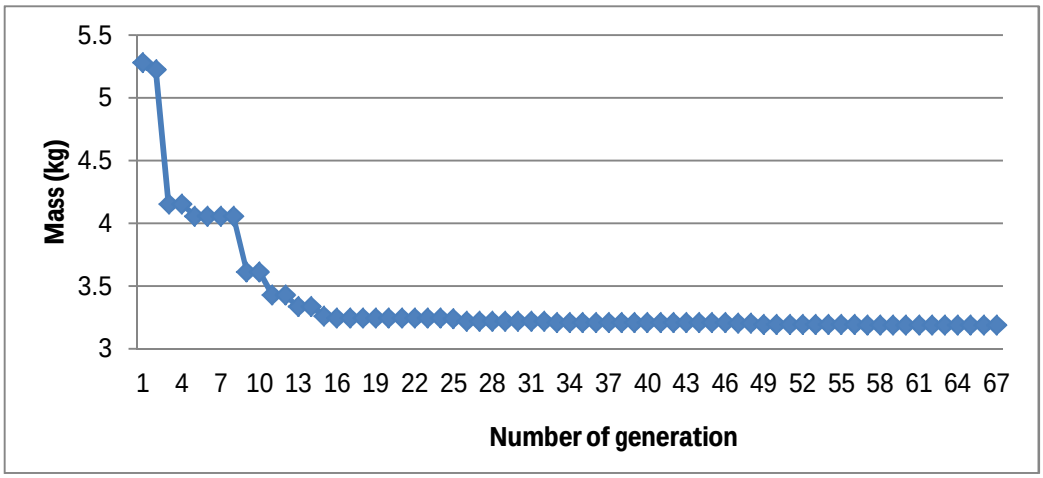


Figure 4.29. The mass change through generations

The results obtained by the genetic algorithm optimization code developed in the present study are compared with the results of obtained by the MATLAB Genetic Algorithm Tool in Table 4.24.

Table 4.24. Comparison of the optimized mass and cross sectional areas

	Mass (kg)	A1	A2	A3	A4	A5	A6
		(mm^2)					
Present	3,196	16,05	81,05	17,10	16,05	16,05	15,05
MATLAB GATOOL	3,198	15,65	79,68	19,36	15,65	15,65	15,65

From Table 4.24, it is seen that optimized mass and cross-sectional areas obtained in present study and by Matlab results are almost same.

Case 2: Optimization with element weight loading

In this case, by the addition of the element weight, the tailcone truss structure size optimization is repeated.

The optimum mass result is 3.23 kg as shown in Figure 4.30 and Figure 4.31 shows the change in the mass from 4.815 kg to final optimum value.

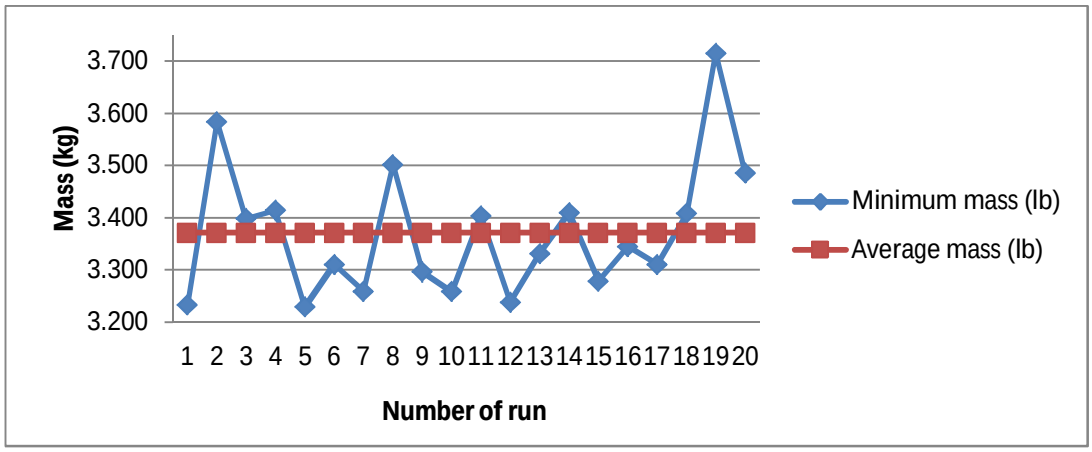


Figure 4.30. Tailcone truss structure size optimization results for 20 runs

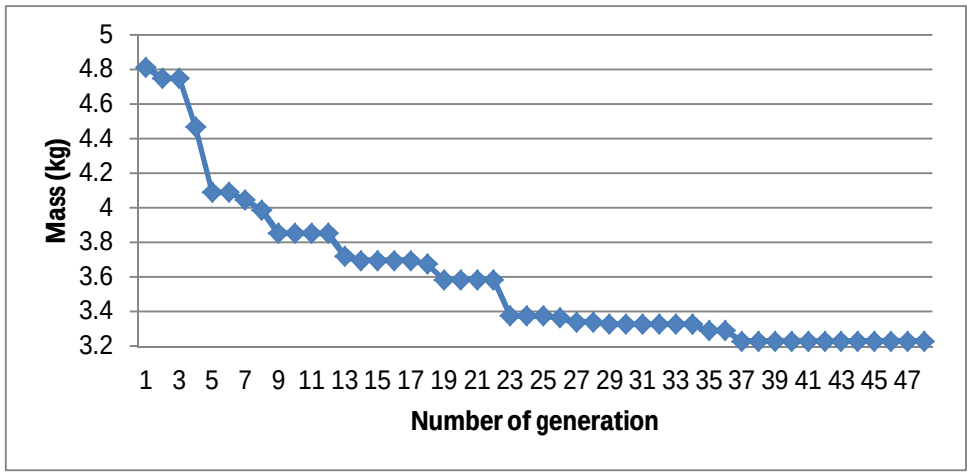


Figure 4.31. The mass change through generations

The results obtained from this analysis is compared with the the MATLAB Genetic Algorithm Tool in Table 4.24.

Table 4.25. Comparison of the optimized mass and cross sectional areas

	Mass (kg)	A1	A2	A3	A4	A5	A6
		(mm^2)					
Present	3,229	16,05	78,95	21,29	15,43	16,05	16,05
MATLAB GATOOL	3,203	15,07	79,95	18,57	22,66	15,20	15,14

The results obtained are very close values to the optimum mass obtained by reference.

Other result that can be concluded is that the addition of element weight values to the loading does not make much difference in the stress analysis supporting the general approach taken in the literature in neglecting the weight of the truss members in the calculations.

4.3.2 Buckling Analysis of the Tailcone Truss Structure

The developed finite element code is also used for the buckling analysis of the tailcone truss structure. In order to show that buckling constraint implemented in the genetic algorithm code works fine, the load applied at nodes 23 and 24 is increased to a value of 750 N.

In the buckling formulation, the Euler beam buckling formula for the pinned end beams is used. Eqn.(4.2) gives the critical buckling load for the pinned end beams under compression.

$$P = \frac{E * I * \pi^2}{L^2} \tag{4.2}$$

The critical buckling stress is determined by dividing the critical buckling load by the cross-sectional area of the annular truss members.

In addition, to get feasible solutions from the size optimization, maximum displacement constraint is set to be 70 mm.

The optimum result after 20 runs is found as 4.757 kg. Figure 4.32 shows the variation of the tailcone mass with the different runs performed.

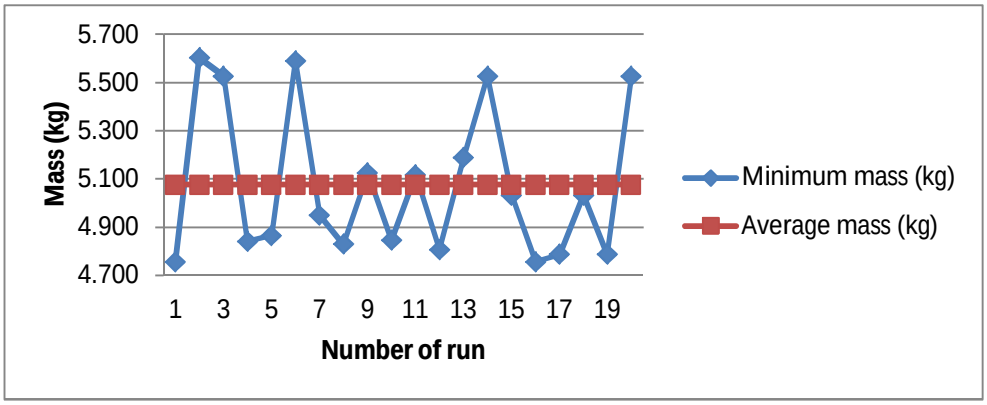


Figure 4.32. Tailcone truss structure size optimization results for 20 runs

The mass change through generations for the optimum solution is given by Figure 4.33.

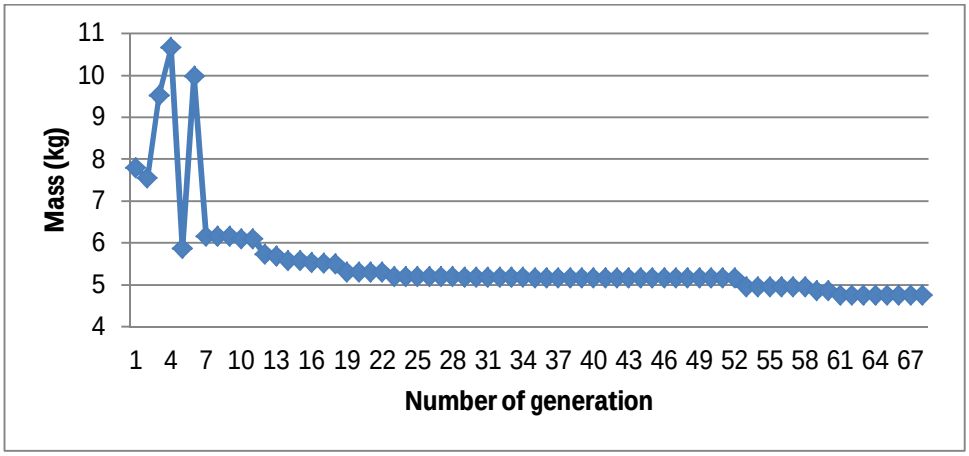


Figure 4.33. The mass change through generations

In this study, the comparison of the stress values for the elements, which have stress values above the critical buckling stress in the optimum solution are made to demonstrate the proper working of the buckling constraints

applied. Table 4.26 shows the result from an arbitrary run which gives the stresses in the elements which violate the buckling stress constraints in the initial population and the stresses in the same elements in the final optimum population. As it can be seen from Table 4.26, elements 13 and 22, which violate the buckling stress constraints in the initial population, have lower stress values than the critical stress values when the feasible optimum solution is found in the final population. This example demonstrates that buckling constraints implemented in the genetic algorithm code works fine.

Table 4.26. Comparison of the stress values

	Initial Population			Final Population		
# El	abs(σ_{el}/σ_{cr})-1	σ_{el} (Pa)	σ_{cr} (Pa)	abs(σ_{el}/σ_{cr})-1	σ_{el} (Pa)	σ_{cr} (Pa)
13	0.0576	-79,459,000	75,130,000	-0.0036	-73,544,000	73,810,000
22	0.0004	-75,160,000	75,130,000	-0.0575	-69,566,000	73,810,000

4.4 Adaptive penalty function results

After all these studies, the developed algorithm is verified by the results obtained which are very close to the results of other reference studies. The algorithm generally finds the optimum mass in a small interval and the average mass results from different runs are very close to the optimum solution. This property of the algorithm makes it possible that independent from the initial population, algorithm can converge to a point which is very close to the real optimum solution.

This also proves the methodology of the adaptive penalty function. Penalty factor,which is calculated for each generation, adapts to the current stuation of the population. Thus, the method can tailor the degree of the penalty for the constraints.As an example, one of the runs is taken, and the penalty factor variation through the generations is demonstrated in Figure 4.34

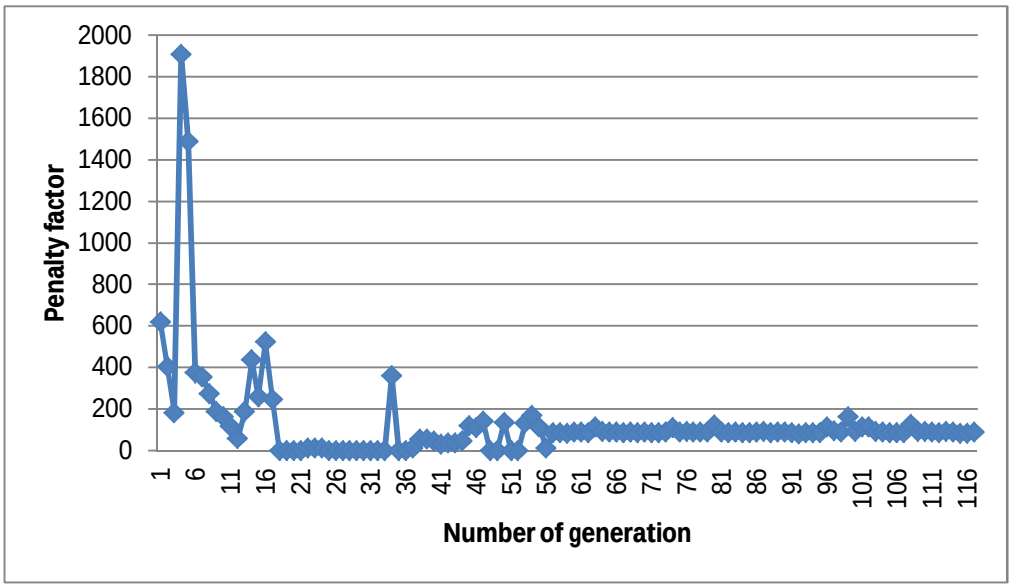


Figure 4.34. The penalty factor variation through the generations

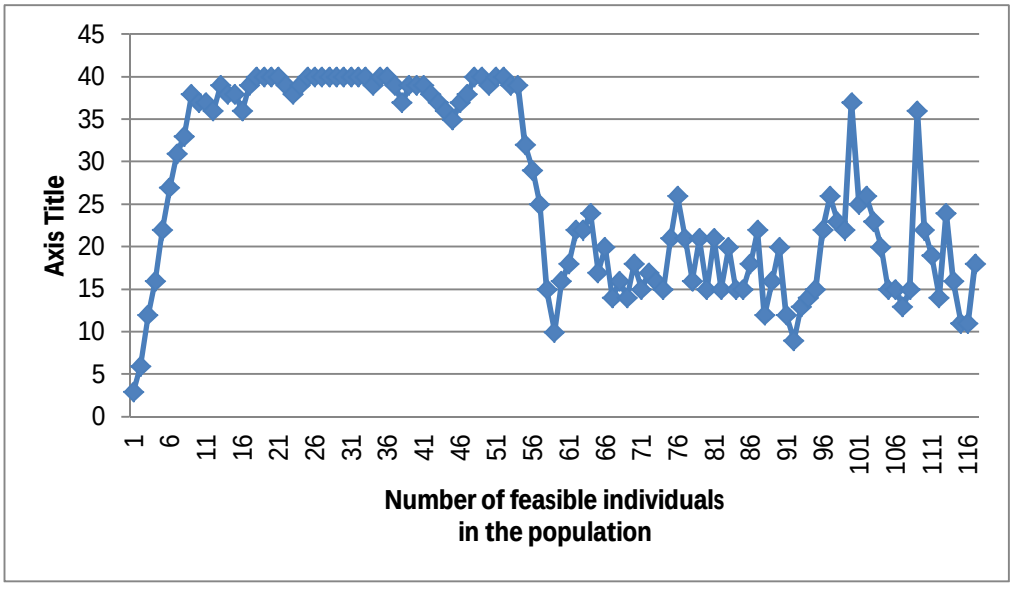


Figure 4.35. The number of feasible individuals in the population through the generations

The Figure 4.34 and Figure 4.35 give the change in the value of the penalty factor depending on the number of the feasible individuals in the population

through the generations. In this example, the population size is 40. From the starting point, as the number of the feasible individuals increases, the penalty factor gets smaller and as expected, when all the population becomes feasible the factor is set zero. And depending on the small changes in feasible portion, the penalty factor responds with small values.

This result shows that setting the penalty factor from the beginning of the algorithm, as in the case of the other penalty methods, is not a logical way to follow.

CHAPTER 5

CONCLUSIONS

In this thesis study, a genetic algorithm optimization code is developed for the size and shape optimization of the two dimensional and space truss structures. Structural solver part of the optimization code is a truss finite element solver which is also developed in the thesis study for general truss structures. Loading applied on the truss members is allowed to be nodal forces, weights of the members and thermal loading. The hearth of the thesis is devoted to the development of the genetic algorithm based optimization code for truss structures. Coming from the randomness characteristics of the genetic algorithm, every run gives unique solution to the problem bypassing the local optima, which is the superiority of the genetic algorithm over the gradient based optimization techniques.

In the developed genetic algorithm code, the simple basic genetic algorithm steps and operators are used. However, special emphasis is given to the penalty approach. It is shown that with the use of adaptive penalty function, without using complex genetic algorithm operators, very reasonable results can be obtained. In the adaptive penalty approach, the penalty constant by adapting to the current state in the population fitness is calculated at each iteration. In the calculation, an important constant φ is given as input. This constant is the most deterministic parameter of the adaptive penalty method. It identifies the chance of best infeasible individuals to be selected into the mating pool. This is very clear meaning for the penalty issue, and can be determined by using experience. This is the point in which this method is separated from the other penalty methods in which the constants are selected arbitrarily which can cause too weak or too strong penalty result during algorithm.

The optimization parameters get very important role in finding the optimum result. To determine the best population size, for every problem, results of 3 population sizes are compared. Working with the population size which is more suitable for the problem, provides effective use of search space preventing unnecessary computation time.

Best individuals always directly pass to the new population by the elitism method. They also have a chance of being parents. Elite number specifies how many individuals pass to the new population. To protect the best gene information in the population, this number is very important especially at the beginning of the algorithm. It is set to one in the algorithm

Crossover is an indispensable operator for the GAs. In the developed code, the rate of the crossover is set to a high probability of 0.8. Crossover operation carries the randomly initiated population to feasible region by recombining the parents chosen from the fitter individuals.

Mutation gives the algorithm a chance to search the design space in more detail. Mutation operation is not needed too much especially in the beginning of the algorithm. Because of fact that the population already is a mix of feasible and infeasible individuals, at the beginning the rate of mutation is taken as a very small, such as 0.01.

However, after approaching the feasible region, crossover operations do not make much work in getting different gene probabilities. At this stage, by decreasing the crossover rate and increasing mutation rate, it is seen that rapid convergence to a point which is local optimum, not global optimum, is avoided. Because by the increased mutation probability, the diversity of the population increased, giving chance of searching solution space in detail.

It is observed that, to make assurance that the results obtained from the code are correct solutions for the problem or not, making reasonable amount of runs is necessary. By these trials, the most effective parameters are determined.

The importance of the population size which gives low deviations from the average optimum mass, controlling the stability of the algorithm in getting optima is presented.

To verify the developed code, 2 benchmark studies are made for the size and shape optimization. By the given design boundaries, it is shown that with same loading condition, size& shape optimization of the 25-bar space truss structure achieves significant mass reduction when compared with the size-only optimization. So, with an appropriate study on design boundaries on the problem, it is concluded that the structures with great mass reduction are possible to design with the aid of size& shape optimization methodology.

In the developed code, optimization can be made by both discrete and continuous design variables. By the usage of discrete design variables, practically feasible solutions are created. On the other hand, their usage resulted with values which are initially set as input, restricting the algorithm from searching any point in the solution space.

Thus, in the size optimization of the tailcone structure, continuous variables are preferred to be sure that every point in the search has a chance of being the solution to the problem. If preferred, the results of the optimum solution can be used as a reference for the standard areas, by taking the standard area with the nearest higher value to these optimum values.

The performance of the genetic algorithm depends on the determination of many optimization parameters which are generally problem specific. It needs time to gather information about best values of these parameters by making sufficient number of trials. This is one of the disadvantages of the genetic algorithm.

Another disadvantage of the GA observed is that the result obtained by the genetic algorithm generally is not the real optimum value. It gives acceptably good results in an acceptably less time. It takes time to get considerably better optimum values, and more runs are needed to get better results. So the solutions obtained from the code, regarding the GA's nondeterministic characteristics, should be taken as a design guide not an exact solution. GA should be used when there is no specific solution methodology for the problem or the problem is in the complexity of NP-complete and NP-hard where NP stands for nondeterministic polynomial time [4].

5.1 Future Work

In this thesis study, every structure is optimized considering only one loading case. But these optimum solutions are not always valid for another load case. So to get practically feasible solutions, different loading conditions are to be studied and the final decision for the optimum solution should be given by gathering the results from these analyses.

In this thesis study, binary strings are used in the formation of the chromosomes to get more effective usage of the genetic operators with longer chromosome length. But some drawbacks of using binary strings are observed.

When discrete design variable are used, if the maximum number, the binary string of the gene can take, is more than the number of the values in the standard area list, resulting from the matching procedure, some of the standard areas have more chance to be selected than the others have. This disturbs the randomness of the genetic algorithm.

In the case of the continuous variables, the length of the binary string constraining the maximum number of the values to be set as variables limits the effective search in the solution space.

So if the number of the variables is enough to make meaningful crossover and mutation operations, the variables can be directly used in the chromosome. For example, in the size and shape optimization of the 10-bar truss structure, the number of the variables is 15. So the variables can be used in the chromosome without making binary transformation. This also eliminates the process of coding and decoding between binary and decimal numbers, providing less computation time.

In the genetic algorithm, the most important issue is the determination of the parameters, crossover rate, mutations rate and the penalty factor, which are generally constant given as inputs to the algorithm. In this study, by using adaptive penalty function, the penalty factor is determined by adjusting to get the desired degree of the penalty at each iteration.

In a same way, the developed GA can be improved by introducing adaptive functions for the crossover and mutation rates. This leads to control the

progress in the algorithm and by the GA parameters adjusted to the current situation of the population. The most important gain by this improvement is the elimination of the decision process to set the proper GA parameters derived by a great number of trials for every specific problem.

REFERENCES

[1] Auer,B.J., “Size And Shape Optimization Of Frame And Truss Structures Through Evolutionary Methods”, University of Idaho, April 2005

[2] McCall, J., “Genetic Algorithms for Modelling and Optimisation”, Journal of Computational and Applied Mathematics 184 (2005) 205–222, 2004

[3] Croce, E.S., Ferreira E.G., Lemonge A.C.C., “A Genetic Algorithm For Structural Optimization Of Steel Truss Roofs”, Federal University of Juiz de For a, 2004

[4] Sandıkcı, B., “Genetic Algorithms”, Bilkent University, 2000

[5] Coello, A.A.C, "Discrete Optimization of Trusses Using Genetic Algorithms", Tulane University, 1994

[6] Hultman, M., “Weight Optimization of Steel Trusses by a Genetic Algorithm”, Lund University, 2010

[7] The Math Works (http://www.mathworks.com/), Last accessed January 2012

[8] Cook, R.D., Malkus, D.S., Plesha, M.E., “Concept and Applications of Finite Element Analysis”, 3rd E., University of Wisconsin-Madison, 1989

[9] Rajeev, S., Krishnamoorthy, C.S., “Discrete Optimization of Structures Using Genetic Algorithms”, Indian Institude of Technology, 1992

[10] Toğan, V., Seyhun, M.O., Daloğlu, A., “A Comparative Study for the Optimum Design of Structures Using Genetic Algorithm”, Karadeniz Technical University, 2006

[11] Nanakorn, P., Meesomklin, K., “An Adaptive Penalty Function in Genetic Algorithms for Structural Design Optimization”, Thammasat University, 2001

[12] Tong, W.H., Liu, G.R., “A Optimization Procedure for Truss Structures with Discrete Design Variables and Dynamic Constraints”, National University of Singapore, 2000

[13] Sun, F., Li, S., Zheng, C., Zhang, B., Hou, S., “A Hybrid Genetic Algorithm for Optimization of the Truss with Discrete Variables”, Shandong University & Shandong Janzhu University, 2009

[14] Erbatur, F., Hasançebi, O., Tütüncü, İ., Kılıç, H., “Optimal Design of Planar and Space Structures with Genetic Algorithms”, Middle East Technical University, 1997

[15] Toğan, V., Daloğlu, A.T., “Optimization of 3D Trusses with Adaptive Approach in Genetic Algorithms”, Karadeniz Technical University, 2006

[16] Gil, L., Andreu, A., “Shape and Cross-section Optimisation of a Truss Structure”, Technical University of Catalonia UPC, 2000

[17] Said, Y.H., “On Genetic Algorithms and Their Applications”, Handbook of Statistics, Vol 24, 2005

[18] Taşkınoğlu, E.E., “A Genetic Algorithm for Structural Optimization”, Middle East Technical University, 2006

[19] Kutay, H., “Stacking Sequence Optimization of a Composite Pressure Vessel by Genetic Algorithm”, Middle East Technical University, 2007

[20] Dianati, M., Song, I., Treiber, M., “An Introduction to Genetic Algorithms and Evolution Strategies”, University of Waterloo, 2002

[21] Charbonneau, P., “An Optimization to Genetic Algorithms for Numerical Optimization”, National Center for Atmospheric Research, 2002

[22] Raletz, R., “Basic Theory of the Helicopter”, Cepadues Editions, 1990

[23] Helimat, helimat.free.fr, Last visited August 29, 2012

[24] European Aviation Safety Agency, “TURBOMECA ARTOUSTE II Series Engines-Certificate Data Sheet”, 2008

APPENDIX A

TEN BAR TRUSS STRUCTURE DESIGN DATA

A.1. Inputs for Finite Element Code

Table A.1. Element nodes

element	node1	node2
1	3	5
2	1	3
3	4	6
4	2	4
5	3	4
6	1	2
7	4	5
8	3	6
9	2	3
10	1	4

Table A.2. Coordinates of the nodes

node	X	Y
1	720	360
2	720	0
3	360	360
4	360	0
5	0	360
6	0	0

Table A.3. Material properties

Material	*E (psi)*	v	σ_a *(psi)*	γ *(lb/in^3)*
Aluminum	10000000	0.334	25000	0.1

Table A.4. Nodal displacement constraints

node	1	2	3	4	5	6
X (*in*)	2	2	2	2	0	0
Y(*in*)	2	2	2	2	0	0

Table A.5. Identification array (ID Array)

node	1	2	3	4	5	6
X	0	0	0	0	1	1
Y	0	0	0	0	1	1

Table A.6. Load matrix

node	1	2	3	4	5	6
X	0	0	0	0	0	0
Y	0	-100000	0	-100000	0	0

A.2. Inputs for Genetic Algorithm

Table A.7. Standard area list for size optimization

[1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63,
2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55,
3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49,
4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97,
11.5, 13.5, 13.9, 14.2, 15.5, 16.0, 16.9,
18.8, 19.9, 22.0, 22.9, 26.5, 30.0, 33.5]

Table A.8. Area intervals for size& shape optimization

Element	Lower bound	Upper bound
1	0,1	10
2	0,1	10
3	0,1	10
4	0,1	10
5	0,1	10
6	0,1	10
7	0,1	10
8	0,1	10
9	0,1	10
10	0,1	10

Table A.9. Shape optimization boundaries for nodal coordinates

node	X		Y	
	Lower bound	Upper bound	Lower bound	Upper bound
1	501	720	50	360
2	0	0	0	0
3	300	600	200	400
4	0	0	0	0
5	0	0	360	720
6	0	0	0	0

APPENDIX B

25-BAR TRUSS STRUCTURE DESIGN DATA

B.1. Inputs for Finite Element method

Table B.1. Coordinates of the nodes

node	X	Y	Z
1	-37,5	0	200
2	37,5	0	200
3	-37,5	37,5	100
4	37,5	37,5	100
5	37,5	-37,5	100
6	-37,5	-37,5	100
7	-100	100	0
8	100	100	0
9	100	-100	0
10	-100	-100	0

Table B.2. Material properties

Material	*E (psi)*	v	σ_a *(psi)*	γ *(lb/in^3)*
Aluminum	10000000	0.334	35294	0.1

Table B.3. Nodal displacement constraints

node	1	2	3	4	5	6
X (*in*)	2	2	2	2	0	0
Y(*in*)	2	2	2	2	0	0

Table B.4. Identification array (ID Array)

node	1	2	3	4	5	6	7	8	9	10
X	0	0	0	0	0	0	1	1	1	1
Y	0	0	0	0	0	0	1	1	1	1
Z	0	0	0	0	0	0	1	1	1	1

Table B.5. Load matrix

node	1	2	3	4	5	6	7	8	9	10
X	0	0	0	0						
Y	20000	20000	0	0						
Z	-5000	-5000	0	0						

B.2. Inputs for Genetic Algorithm

Table B.6. Area intervals for size and size& shape optimization

Element	Lower bound	Upper bound
1	0,01	3
2	0,01	3
3	0,01	3
4	0,01	3
5	0,01	3
6	0,01	3
7	0,01	3
8	0,01	3
9	0,01	3
10	0,01	3
11	0,01	3
12	0,01	3
13	0,01	3
14	0,01	3
15	0,01	3
16	0,01	3
17	0,01	3
18	0,01	3
19	0,01	3
20	0,01	3
21	0,01	3
22	0,01	3
23	0,01	3
24	0,01	3
25	0,01	3

Table B.7. Shape optimization boundaries for nodal coordinates

node	X		Y		Z	
	Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound
1	0	0	0	0	0	0
2	0	0	0	0	0	0
4	37,5	100	37,5	100	0	0
8	50	200	50	200	0	0

APPENDIX C

TAILCONE TRUSS STRUCTURE DESIGN DATA

C.1. Inputs for Finite Element Code

The design data of the helicopter "Aerospatiale SA-318C (Alouette II)" is obtained by taking measurements on the helicopter which is in the METU Aerospace Engineering's Hangar building as presented in Figure C.1. After getting diameter and length of the each bar element, the truss structure is modeled. From the model, the node coordinates are found.

To decide upper and lower limits of the design variables, two thickness values for the bar elements are assumed, which are 2 mm and 0.4 mm. and the minimum and maximum values are taken as reference for the area interval for the bars. Table C.1 presents the measured data and calculated area values according to these two thickness values.

Table C.1. The measured diameters of the bar elements

Variable	**Diameter** (mm)	**Area** (mm^2) t=2 mm	**Area** (mm^2) t=0.4 mm
1	25,4	147,0	31,4
2	22,2	126,9	27,4
3	16,3	73,9	20,0
4	14,5	64,8	17,7
5	14,4	64,3	17,6
6	16,2	73,4	19,9

Table C.2. Element nodes

ELEMENT NO	NODE_1	NODE_2
1	1	4
2	2	4
3	3	4
4	2	5
5	3	5
6	3	6
7	4	5
8	4	6
9	5	6
10	4	7
11	5	7
12	6	7
13	5	8
14	6	8
15	6	9
16	7	8
17	7	9
18	8	9
19	7	10
20	8	10
21	9	10
22	8	11
23	9	11
24	9	12
25	10	11
26	10	12
27	11	12
28	10	13
29	11	13
30	12	13
31	11	14
32	12	14

ELEMENT NO	NODE_1	NODE_2
33	12	15
34	13	14
35	13	15
36	14	15
37	13	16
38	14	16
39	15	16
40	14	17
41	15	17
42	15	18
43	16	17
44	16	18
45	17	18
46	16	19
47	17	19
48	18	19
49	17	20
50	18	20
51	18	21
52	19	20
53	19	21
54	20	21
55	19	22
56	20	22
57	21	22
58	20	23
59	21	23
60	21	24
61	22	23
62	22	24
63	23	24

Table C.3. Coordinates of the nodes

NODE	COORDINATE		
	X (mm)	Y (mm)	Z (mm)
1	0	0	0
2	0	-285	750
3	0	285	750
4	620	0	114.3
5	620	-253.6	782.9
6	620	253.6	782.9
7	1300	0	239.6
8	1300	-219.1	818.9
9	1300	219.1	818.9
10	1980	0	365
11	1980	-184.6	854.9
12	1980	184.6	854.9
13	2600	0	479.3
14	2600	-153.2	888.9
15	2600	153.2	887.8
16	3220	0	593.5
17	3220	-121.8	920.6
18	3220	121.8	920.6
19	3840	0	707.8
20	3840	-90.3	953.5
21	3840	90.3	953.5
22	4340	0	800
23	4340	-65	980
24	4340	65	980

Table C.4. Material properties

Material	E	σ_a	γ
Aluminum	70 GPa	172.37 MPa	2768 kg/m^3

Table C.5. Nodal displacement constraints

NODE	COORDINATE		
	X (mm)	Y (mm)	Z (mm)
1	0	0	0
2	0	0	0
3	0	0	0
4	5	5	5
5	5	5	5
6	5	5	5
7	5	5	5
8	5	5	5
9	5	5	5
10	5	5	5
11	5	5	5
12	5	5	5
13	5	5	5
14	5	5	5
15	5	5	5
16	5	5	5
17	5	5	5
18	5	5	5
19	5	5	5
20	5	5	5
21	5	5	5
22	5	5	5
23	5	5	5
24	5	5	5

Table C.6. Identification array (ID Array)

NODE	COORDINATE		
	X	Y	Z
1	1	1	1
2	1	1	1
3	1	1	1
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0

Table C.7. Load matrix

NODE	COORDINATE		
	X	Y	Z
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	-40,5	0
24	0	-40,5	0

C.2. Inputs for Genetic Algorithm

Table C.8. Area intervals for size& shape optimization

Element #	Lower bound (mm^2)	Upper bound (mm^2)
1	15	150
2	15	150
3	15	150
4	15	150
5	15	150
6	15	150
7	15	150
8	15	150
9	15	150
10	15	150
11	15	150
12	15	150
13	15	150
14	15	150
15	15	150
16	15	150
17	15	150
18	15	150
19	15	150
20	15	150
21	15	150
22	15	150
23	15	150
24	15	150
25	15	150
26	15	150
27	15	150
28	15	150
29	15	150
30	15	150
31	15	150

Element #	Lower bound (mm^2)	Upper bound (mm^2)
32	15	150
33	15	150
34	15	150
35	15	150
36	15	150
37	15	150
38	15	150
39	15	150
40	15	150
41	15	150
42	15	150
43	15	150
44	15	150
45	15	150
46	15	150
47	15	150
48	15	150
49	15	150
50	15	150
51	15	150
52	15	150
53	15	150
54	15	150
55	15	150
56	15	150
57	15	150
58	15	150
59	15	150
60	15	150
61	15	150
62	15	150

Figure C.1. Aerospatiale SA-318C (Alouette II) photographs