Evren Eyüp Taşkınoğlu

METU 2006

A GENETIC ALGORITHM FOR STRUCTURAL OPTIMIZATION

EVREN EYÜP TAŞKINOĞLU

DECEMBER 2006

A GENETIC ALGORITHM FOR STRUCTURAL OPTIMIZATION

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

BY

EVREN EYÜP TAŞKINOĞLU

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

DECEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Kemal İder
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Süha Oral
Supervisor

Examining Committee Members

Prof. Dr. Bülent Doyum	(METU,ME)	_____
Prof. Dr. Süha Oral	(METU,ME)	_____
Prof. Dr. Tuna Balkan	(METU,ME)	_____
Assoc. Prof. Uğur Polat	(METU,CE)	_____
Asst. Prof. Serkan Dağ	(METU,ME)	_____

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 : Evren Eyüp Taşkınoğlu

Signature :

ABSTRACT

A GENETIC ALGORITHM FOR STRUCTURAL OPTIMIZATION

Taşkınoğlu, Evren Eyüp
M.S., Department of Mechanical Engineering
Supervisor : Prof. Dr. Süha Oral

December 2006, 93 pages

In this study, a design procedure incorporating a genetic algorithm (GA) is developed for optimization of structures. The objective function considered is the total weight of the structure. The objective function is minimized subjected to displacement and strength requirements. In order to evaluate the design constraints, finite element analysis are performed either by using conventional finite element solvers (i.e. MSC/NASTRAN®) or by using in-house codes. The application of the algorithm is shown by a number of design examples. Several strategies for reproduction, mutation and crossover are tested. Several conclusions drawn from the research results are presented.

Keywords: Genetic Algorithm, Structural Optimization

ÖZ

GENETİK ALGORİTMA İLE YAPISAL OPTİMİZASYON

Taşkınoğlu, Evren Eyüp

Yüksek Lisans, Makina Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Süha Oral

Aralık 2006, 93 sayfa

Bu çalışmada, yapıların optimizasyonunda kullanılmak üzere Genetik Algoritma tabanlı bir tasarım prosedürü geliştirilmektedir. Amaç fonksiyonu yapıların toplam ağırlığıdır. Amaç fonksiyonu, deplasman ve dayanıklılık gerekleri baz alınarak azaltılmaktadır. Tasarım kısıtlamalarının değerlendirilmesinde, sonlu eleman analizleri, yaygın sonlu eleman çözücüleri (MSC/NASTRAN® gibi) yada mevcut yazılmış programlar kullanılarak yapılmaktadır. Algoritmanın uygulaması, bazı tasarım örnekleriyle gösterilmiştir. Birkaç reprodüksiyon, mutasyon ve crossover stratejisi denenmiştir. Araştırma sonuçlarından çıkarılan bazı yargılar sunulmuştur.

Anahtar Kelimeler: Genetik algoritma, Yapısal optimizasyon

To my parents and to my sisters who supported me through all the good times and the bad times.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my supervisor Prof. Dr. Süha Oral for his guidance and support throughout this research effort. I consider myself fortunate to have had a mentor with the strong work ethic and unyielding patience of Prof. Dr. Süha Oral.

I would like to thank to Dr. Mustafa Usta, Mustafa Açıkgöz, Özkan Murat, Umut Demirhisar and Umut Susuz for their technical and moral support. I also would like to thank to Dr. Muvaffak Hasan for his understanding and tolerance.

My love Kardem, I could not finalize this thesis without your help and motivation. Thanks for your endless support and trust.

TABLE OF CONTENTS

ABSTRACT iv
ÖZ v
ACKNOWLEDGMENTS vii
TABLE OF CONTENTS viii
LIST OF TABLES x
LIST OF FIGURES xi
CHAPTER
1. INTRODUCTION 1
1.1 Objective and Scope of the Study 1
1.2 Literature Survey 2
2. OPTIMIZATION 5
2.1 Objective function 6
2.2 Design Variables 6
2.3 Design Constraints 8
2.4 Classification of Optimization Problems 8
3. GENETIC ALGORITHM 10
3.1 Genetic Algorithm Description 10
3.2 History 11
3.3 Basic Structure of Genetic Algorithm 12
3.3.1 Chromosome Encoding 14
3.3.2 Fitness Evaluation 14
3.3.3 Selection 16
3.3.4 Crossover and Mutation 18
3.3.5 Termination criteria 20
3.3.6 GA Implementation 21
3.3.7 GA Comparison with Other Optimization Techniques 22
4. GABSO: Genetic Algorithm Based Structural Optimizer 25
4.1 Input.f90 25

4.2 Initial_Pop.f90 26
4.3 Fitness.f90 26
4.4 Truss.f90 27
4.5 Crossover.f90 28
4.6 Mutation.f90 28
4.7 Gabso.f90 29
5. DESIGN EXAMPLES 30
5.1 Test Problems 30
5.1.1 Test Problem 1 30
5.1.2 Test Problem 2 34
5.1.3 Test Problem 3 37
5.1.4 Test Problem 4 (Six-hump Camel Back Function) 41
5.2 Structural Optimization Problems 44
5.2.1 6-bar truss 44
5.2.2 25-bar space truss 48
5.2.3 72-bar space truss 53
5.2.4 18-bar truss 58
5.2.5 Simply supported bridge shape optimization 62
5.2.6 Stepped Cantilever Beam 66
5.2.7 Portal Frame 69
5.2.8 Bridge size, shape and topology optimization 73
5.2.9 Rajan's Truss 78
6. DISCUSSION AND CONCLUSION 87
6.1 Discussion 87
6.2 Conclusion 89
REFERENCES 91

LIST OF TABLES

Table 5.1 – Comparison of results with Ref [8] 45
Table 5.2 – Loading inputs for 25-bar space truss 49
Table 5.3 – Group membership for 25-bar space truss 50
Table 5.4 – Comparison of results with Ref [12] 50
Table 5.5 – Comparison of results with Ref [12] (2) 51
Table 5.6 – Loading inputs for 72-bar space truss 54
Table 5.7 – Group membership for 72-bar space truss 55
Table 5.8 – Comparison of results with Ref [13] 56
Table 5.9 – Group membership for 18-bar truss 59
Table 5.10 – Comparison of results with Ref [14] 60
Table 5.11 – Loading inputs for Bridge structure 62
Table 5.12 – Comparison of results with Ref [15] 63
Table 5.13 – Comparison of results with Ref [16] 67
Table 5.14 – Comparison of results with GENESIS results 70
Table 5.15 – Available Discrete Sizes for AISC Standard Weight Steel Pipe 74
Table 5.16 – Group membership for Bridge structure 74
Table 5.17 – Comparison of results with Ref [17] 75
Table 5.18 – Loads for Rajan's Truss Example 78
Table 5.19 – Available Discrete Areas 79
Table 5.20 – Group membership for Rajan's truss structure 79
Table 5.21 – Comparison of results with Ref [18] 80
Table 5.22 – Results obtained at consecutive runs 82

LIST OF FIGURES

Figure – 3.1 Outline of a basic genetic algorithm 13
Figure – 3.2 A binary encoded chromosome 14
Figure – 3.3 Roulette Wheel 17
Figure – 3.4 One-Point Crossover 19
Figure – 3.5 Two-Point Crossover 19
Figure – 3.6 Uniform Crossover 19
Figure – 3.7 Mutation 20
Figure – 5.1 Test Problem 1 30
Figure – 5.2 Genes History for Test Problem 1 32
Figure – 5.3 Design Variables History for Test Problem 1 32
Figure – 5.4 Objective Function History for Test Problem 1 33
Figure – 5.5 Test Problem 2 34
Figure – 5.6 Genes History for Test Problem 2 35
Figure – 5.7 Design Variables History for Test Problem 2 36
Figure – 5.8 Objective Function History for Test Problem 2 36
Figure – 5.9 Test Problem 3 37
Figure – 5.10 Genes History for Test Problem 3 39
Figure – 5.11 Design Variables History for Test Problem 3 39
Figure – 5.12 Objective Function History for Test Problem 3 40
Figure – 5.13 Test Problem 4 41
Figure – 5.14 Genes History for Test Problem 4 42
Figure – 5.15 Design Variables History for Test Problem 4 43
Figure – 5.16 Objective Function History for Test Problem 4 43
Figure – 5.17 6-bar truss 44
Figure – 5.18 Genes History for 6-bar truss 46
Figure – 5.19 Design Variables History for 6-bar truss 46
Figure – 5.20 Objective Function History for 6-bar truss 47

Figure – 5.21 25-bar space truss 48
Figure – 5.22 Design Variables History for 25-bar space truss 51
Figure – 5.23 Objective Function History for 25-bar space truss 52
Figure – 5.24 72-bar space truss 53
Figure – 5.25 Design Variables History for 72-bar space truss 57
Figure – 5.26 Objective Function History for 72-bar space truss 57
Figure – 5.27 18-bar space truss 58
Figure – 5.28 Final shape of the 18-bar space truss 60
Figure – 5.29 Design Variables History for 18-bar truss 61
Figure – 5.30 Objective Function History for 18-bar truss 61
Figure – 5.31 Simply supported bridge structure 62
Figure – 5.32 Final shape of the bridge structure 64
Figure – 5.33 Design Variables History for Bridge Structure 64
Figure – 5.34 Objective Function History for Bridge Structure 65
Figure – 5.35 Stepped Cantilever Beam 66
Figure – 5.36 Design Variables History for Cantilever Beam 68
Figure – 5.37 Objective Function History for Cantilever Beam 68
Figure – 5.38 Portal Frame 69
Figure – 5.39 Design Variables History for Portal Frame 71
Figure – 5.40 Objective Function History 72
Figure – 5.41 Bridge ground structure 73
Figure – 5.42 Final shape of the bridge structure 76
Figure – 5.43 Design Variables History for Bridge Structure-1 76
Figure – 5.44 Design Variables History for Bridge Structure-2 77
Figure – 5.45 Objective Function History for Bridge Structure 77
Figure – 5.46 Rajan's truss ground structure 78
Figure – 5.47 Final shape of the bridge structure 81
Figure – 5.48 Shape of the bridge structure in Ref [18] 81
Figure – 5.49 Shape of the bridge structure for runs 1, 2, 6, 7 and 9 82
Figure – 5.50 Shape of the bridge structure for run 3 83
Figure – 5.51 Shape of the bridge structure for run 4 83

Figure – 5.52 Shape of the bridge structure for run 5 84
Figure – 5.53 Shape of the bridge structure for runs 8 and 10 84
Figure – 5.54 Design Variables History for Rajan's truss-1 85
Figure – 5.55 Design Variables History for Rajan's truss-2 85

CHAPTER 1

INTRODUCTION

Structural design optimization is a critical and challenging activity that has received considerable attention in the last two decades. The main purpose in design optimization is to find the best ways so that a designer or a decision maker can derive a maximum benefit from the available resources. Genetic algorithm is one of the most popular optimization algorithms that is known for its robustness and ability to search complex and noisy search spaces.

1.1 Objective and Scope of the Study

The main aim of this study is to investigate Genetic algorithm as a structural optimization procedure. This has been done by considering several numbers of test and structural optimization problems. Test problems have been used for designing the algorithm and the final algorithm has been tested over structural optimization problems. All the results obtained have been compared with the results in literature or with the conventional optimization tools such as GENESIS.

The study also includes development of a code for GA implementation. The name of the code developed is, *GABSO: Genetic Algorithm Based Structural Optimizer*. It can be used for any kind of structural optimization problem as long as the finite element model of the structure is available. The code uses MSC/NASTRAN® as finite element solver. But it also contains a subroutine for finite element analysis for 2-D and 3-D truss structures. In this study, the structural optimization problems in consideration are mainly truss structures. The reason for this is the capability of the code to solve 2D and 3D truss structures. For other kind of structural problems, MSC/NASTRAN® has to be executed externally for fitness evaluation and this causes long runs that cannot be suitable for design parameter investigation.

1.2 Literature Survey

A detailed literature survey has been performed in order to get into structural optimization as well as Genetic algorithms. The study has started with detailed investigation about optimization concepts and definitions used (objective function, design variable, constraints etc) in optimization. The major characteristics of structural optimization have been identified. The details of the objective function design and constraint handling have been examined. Several numbers of publications are read in order to obtain the preliminary background for optimization as well as structural optimization.

In 1999, Belegundu and Chandrupatla [1] wrote a book on the implementation of optimization in engineering, offering a strong foundation and coverage of optimization theory.

In 1992, Kamat [2] edited a book written by experts documenting the state of the art in structural optimization with a view to establishing some of the most promising directions for future research in the field.

Haftka and Gürdal [3] wrote a book containing examples to present the application of various optimization methods. Optimization of both constrained and unconstrained problems are shown with emphasis on the variable types involved (i.e. continuous, discrete).

Moore [7] wrote a book discussing the theoretical details of design sensitivity and optimization.

After forming the necessary structural optimization background, a detailed investigation about Genetic algorithms has been started. The Genetic Algorithm concepts, operators and algorithm implementation are the main focus points for this part of the survey. The in-depth GA knowledge had been gained by examining the following papers and publications,

Dianati, Song, and Treiber [4] published a paper examining the history, theory and mathematical background, applications, and the current direction of both Genetic Algorithms and Evolution Strategies.

In 2002, Charbonneau [5] published a paper providing a detailed comparison of genetic algorithm based optimization schemes against other optimization schemes and describing in full detail the use of a genetic algorithm.

Said [6] published a paper describing the basic concepts and functionality of Genetic computation.

Gantovnik, Anderson-Cook, Gürdal and Watson [11] published a paper describing a new approach for reducing the number of the fitness function evaluations required by a genetic algorithm for optimization problems with mixed continuous and discrete design variables.

In 2005 McCall [19] published a paper demonstrating the structure of a Genetic Algorithm with simple examples and exploring the key advances that have been made in the theoretical understanding of how Genetic Algorithms operate.

After forming the algorithm and developing a code based on this algorithm, GABSO, several test and structural optimization problems are solved. The papers and publications introducing these optimization problems are as follows:

Charbonneau [5] wrote a paper demonstrating numerical optimization problems for testing the performance of optimization algorithms.

In 2001, Nanakorn and Meesomklin [8] published a paper demonstrating a new penalty scheme capable of adjusting itself during the optimization process.

Rajeev and Krishnamoorty [12] published a paper presenting a simple genetic algorithm for optimizing structural systems with discrete design variables using a penalty-based transformation method on 1992.

In 1992, Xicheng and Guixu [13] published a paper introducing a parallel iterative algorithm to solve constraint optimization problems in discrete design variable domains.

Yu, Johnson, and Zhang [14] published a paper discussing design of experiments (DOE) and conservative discrete design (CDD) approaches to deal with discrete variables with limited computational cost.

In 2002 Wang, Zhang and Jiang [15] published a paper presenting an evolutionary node shift method for truss shape optimization of weight minimization problems where the structure is subject to multiple displacement constraints under multiple load cases.

A paper reviewing available methods for discrete variable structural optimization by solving a stepped cantilever beam design problem was published by Thanedar and Vanderplaats [16].

Rajan [18] published a paper demonstrating a procedure for the combined size, shape and topology design of space trusses on 1995.

CHAPTER 2

OPTIMIZATION

The goal of an optimization problem can be formulated as follows: find the combination of parameters (design variables) which optimize a given quantity, possibly subject to some restrictions on the allowed parameter ranges.

The quantity to be optimized (maximized or minimized) is termed as objective function; the parameters which may be changed in the quest for the optimum are called design variables; the restrictions on allowed parameter values are known as constraints.

The general optimization problem may be stated mathematically as:

Maximize

$f(x)$, $\quad x = (x_1, x_2, \ldots\ldots\ldots\ldots x_N) \in R^N$,

Subject to

$g_i(x) \le 0$, $\quad i = 1, \ldots\ldots\ldots, K$,

$h_i(x) \le 0$, $\quad i = 1, \ldots\ldots\ldots, P$,

$f(x)$ is the objective function. $g_i(x)$ and $h_i(x)$ are inequality and equality constraints, respectively. They represent constraints, which the design must satisfy, such as stress and displacements limits.

2.1 Objective function

The objective function is a function returns a single value from which different designs can be compared. It is a scalar quantity that is either minimized or maximized by the optimizer. The optimal design is the design with a minimum (or maximum) value of the objective.

A minimum and maximum formulation may be interchanged by simply changing the sign of the objective. Optimization with more than one objective is generally referred to as Multiobjective optimization. For structural optimization problems, weight, displacements, stresses, vibration frequencies, buckling loads and cost or any combination of these can be used as objective functions.

When formulating the design objective, there are a couple of scaling-related issues that should be kept in mind since they affect overall performance. First, the design problem should be posed so that the objective function has sufficient sensitivity with respect to each of the design variables. The second item to consider is the absolute value of the response selected to be the objective function. Care should be taken so that this value is not too close to zero. If it is very close to zero, this will cause numerical difficulties in determination of weighting constants for constraint violations.

2.2 Design Variables

Design characteristics that are varied to achieve the objective are called as design variables. Design variables may take continuous or discrete values. Continuous design variables have a range of variation, and can take any value in that range. Discrete design variables can take only discrete values, typically from a list of permissible values.

In structural optimization, there are three types of design variable. These are;

- Size design variables
- Shape design variables
- Topology design variables

The notion of size design variable is related with cross-sectional quantities like area of bars, second moments of area of beams and thickness of plates and shells. The definition of size variable is related to the fact that the modeling domain is not changed. So, the line of the beam, rod or bar is unchanged, just like the reference surface of a plate or a shell is assumed unchanged when the concept of size design variable is used. The orientations of non-isotropic material can also be treated as size design variables.

The notion of shape design variable is related to the reference domain of the actual model. For beams, rods and bars, the length can be thought as a design variable, which is then a shape design variable. For truss structures node coordinates of the truss elements can also be treated as shape design variables. Also the curvature of the reference line for these one-dimensional models is a shape design variable. For 2D-models likewise the boundary curve or the curvature of the reference surface are shape design variables. For 3D-models the boundary surface (including internal boundaries like holes) is a shape design variable.

Finally, the notion of topology design variable is related to presence or absence of a certain design aspect. The complications in treating topology design variables are due to the fact that a change in topology results in a discontinuous change in the design response, while a continuous change in size or shape design variables normally results in continuous change in the design response.

2.3 Design Constraints

In optimization problems, there can be some constraints that have to be satisfied while minimizing (or maximizing) the objective function. Conditions that the designs must meet are called as design constraints**.**

If there is no constraint imposed on the optimization problem then it is called as unconstrained optimization otherwise it is called as constrained optimization problem.
In structural optimization problems, a constrained optimization problem arises in finding the minimum weight design of a structure subject to constraints on stress and deflection.

2.4 Classification of Optimization Problems

There are several classes of optimization problems. Knowing the type of optimization problem in consideration is critical, since the treatments of different class of optimization problems are not the same. The methodology to solve the optimization problem can be defined easily when the class of the problem is known. Optimization problems can be classified as follows:

Stochastic optimization refers to the minimization (or maximization) of a function in the presence of randomness in the optimization process. But in Deterministic optimization, the process followed to find the minimum (or maximum) for the given function is defined.

An optimization problem can have some constraints defined which have to be satisfied while minimizing (or maximizing) the objective. These types of optimization problems are called Constrained Optimization Problems whereas the problems are called Unconstrained Optimization Problems when there's no condition to be satisfied in the defined problem.

In some optimization problems, it is possible to have more than one objective. These kinds of optimization problems are called Multiobjective optimization problems. But in Single-objective problems, there is only one objective to be achieved.

CHAPTER 3

GENETIC ALGORITHM

There are several optimization techniques that are used in the context of engineering design optimization. Genetic Algorithm (GA) is one such technique that has been gaining substantial attention in recent years.

Genetic algorithm is a search strategy based on the rules of natural genetic evolution. It is well known for its robustness and ability to search complex and noisy search spaces, phenomena which are frequently encountered in design and optimization problems.

Genetic algorithm can be regarded as an expensive optimization tool that sometimes requires thousands of analyses to achieve convergence. However, there is a large amount of research work being done with GAs and it is continuing to grow, with many new ideas aimed at reducing computational cost.

3.1 Genetic Algorithm Description

Genetic Algorithms are nondeterministic stochastic search/optimization methods that utilize the theories of evolution and natural selection to solve a problem within a complex solution space.

A genetic algorithm emulates biological evolution to solve optimization problems. It is formed by a set of individual elements (the population) and a set of biological inspired operators that can change these individuals. It simulates evolution of individual structures via processes of selection, mutation, and reproduction that are referred to as search operators.

Each individual in the population receives a measure of its fitness in the environment. Only the individuals that are the more suited in the population are likely to survive and to generate offsprings, thus transmitting their biological heredity to new generations.

In computing terms, genetic algorithms map strings of numbers to each potential solution. Each solution becomes an individual in the population, and each string becomes a representation of an individual. There should be a way to derive each individual from its string representation. The genetic algorithm then manipulates the most promising strings in its search for an improved solution.

3.2 History

In the middle of the twentieth century some computer scientists worked on evolutionary systems with the notion that this will yield to an optimization mechanism for an array of engineering queries. GAs were invented and developed by John Holland, his students and his colleagues at the University of Michigan. His team's original intentions were not to create algorithms, but instead to determine exactly how adaptation occurs in nature and then develop ways that natural adaptation might become a part of computer systems.

This lead to Holland's book "Adaptation in Natural and Artificial Systems" published in 1975. GA, he stated, moves one population of bits (chromosomes and genes) to a new population using a type of natural selection along with genetic operators of crossover and mutation (all biological functions). These operators determine which chromosomes are the fittest and thus able to move on. Although some less fit chromosomes do move forward, on average the most fit chromosomes produce more offspring than their less fit counterparts. Biological recombination occurs between these chromosomes, and chromosomal inversion further completes the process of providing as many types as possible of recombination or crossover. This remarkable

quality that genetic algorithms have of focusing their attention on the fittest parts of a solution set in a population is directly related to their ability to combine strings, which contain partial solutions.

In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method genetic programming (GP).

3.3 Basic Structure of Genetic Algorithm

In nature, a combination of natural selection and procreation permits the development of living species that are highly adapted to their environments. A GA is an algorithm that operates on a similar principle.

The genetic algorithm (GA) is one of the probabilistic optimization algorithms generated on the basis of the theory of evolution. The optimization process is a model of the law of the survival of the fittest of actual creatures: the fittest adaptable individual can leave offspring. This survival-of-the-fittest process is modeled in a computer program. Those individual with the highest fitness within the given environment are selected at high probability for reproductions of next generation, and the rest of the individuals in the group are curtailed. From the selected elitist group, the genetic information of the next generation is produced by means of crossovers and mutations.

In order to solve the optimization problems by means of GA, design variables must be coded into a list of genes (chromosome) and a design example must correspond to a chromosome or chromosomes that represent an individual. The complexity of an organism can be controlled by the length and number of chromosome and gene strings, and the size and number of gene alphabets.

A group is made from these individuals, and the optimization is performed for the group using genetic procedures like fitness evaluations, selections, crossover and

mutation. A genetic algorithm is usually made up of a group of organisms commonly referred to as a population of organisms. Although there exist many different algorithms, the basic structure is still the same as shown in Figure 3.1.

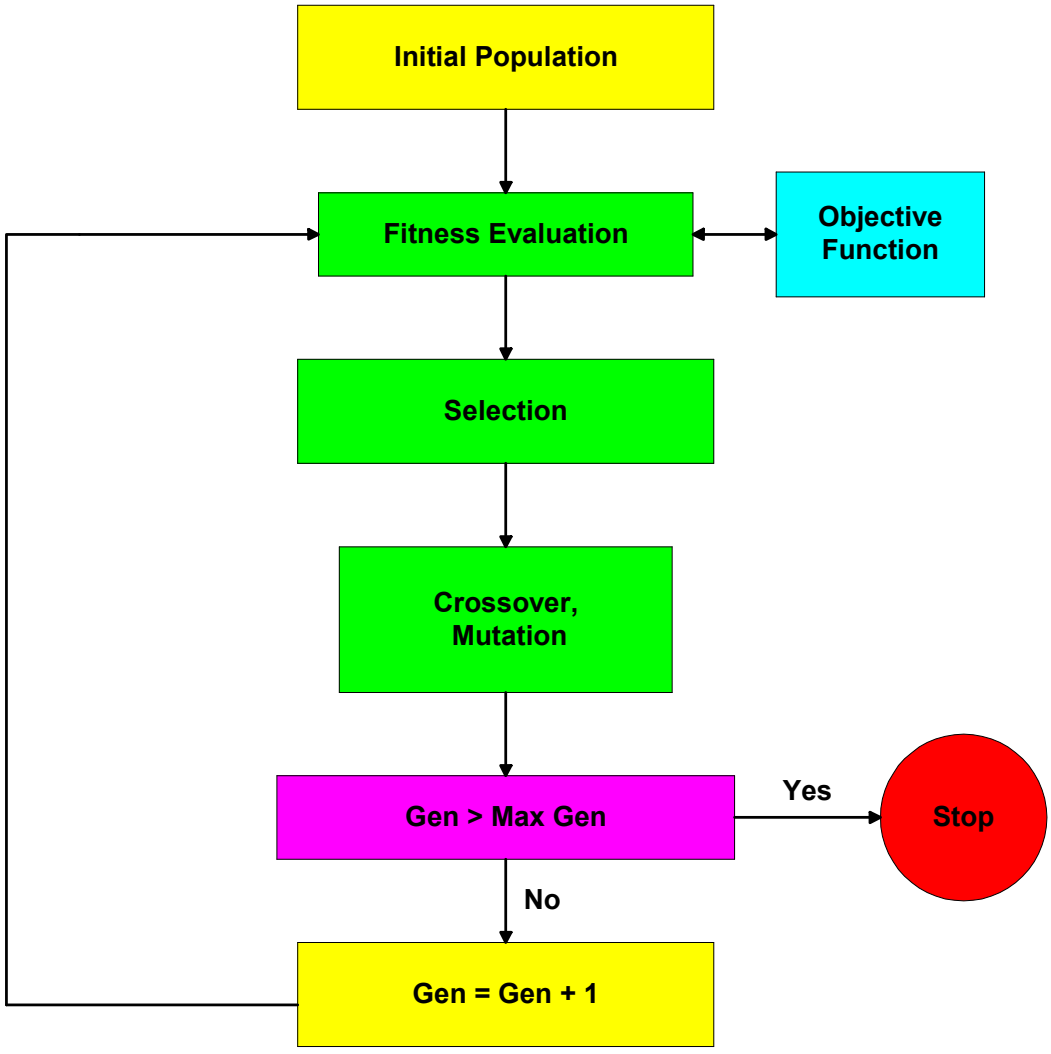


Figure – 3.1 Outline of a basic genetic algorithm

The main components of the basic genetic algorithms are the chromosome encoding, fitness evaluation, selection, crossover and mutation.

3.3.1 Chromosome Encoding

A GA manipulates populations of chromosomes, which are string representations of solutions to a particular problem. A chromosome is an abstraction of a biological DNA chromosome, which can be thought of as a string of letters from the alphabet {A,C,G,T}. A particular position or locus in a chromosome is referred to as a gene and the letter occurring at that point in the chromosome is referred to as the allele value or simply allele. Any particular representation used for a given problem is referred to as the GA encoding of the problem. The classical GA uses a bit-string representation to encode solutions. In binary encoding every chromosome is a string of bits, 0 or 1.

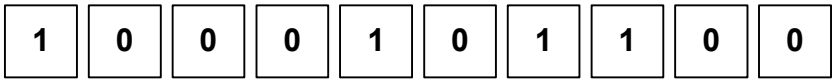


Figure – 3.2 A binary encoded chromosome

Encoding depends on the problem and also on the size of instance of the problem. There are many other ways of encoding.

3.3.2 Fitness Evaluation

GAs typically work by iteratively generating and evaluating individuals using an evaluation function. The fitness function is a computation that evaluates the quality of the chromosome as a solution to a particular problem

Generally, an optimization problem using GAs can be expressed as:

Maximize

$F(x) = F[f(x)], \qquad x = (x_1, x_2, \ldots\ldots\ldots\ldots x_N) \in R^N,$

under constraints defined as

$g_i(x) \le 0, \qquad i = 1, \ldots\ldots\ldots, K,$

$h_i(x) \le 0, \qquad i = 1, \ldots\ldots\ldots, P,$

For structural design optimization, x is an N- dimensional vector called the design vector, representing design variables of N structural components to be optimized, and $f(x)$ is the objective function. In addition, $g_i(x)$ and $h_i(x)$ are inequality and equality constraints, respectively. They represent constraints, which the design must satisfy, such as stress and displacements limits. Moreover, $F[f(x)]$ is the fitness function that is defined as a figure of merit.

It is not possible to directly utilize GAs to solve the constrained problems. In GAs, constraints are usually handled by using the concept of penalty functions, which penalize infeasible solutions. If any constraints are violated, a penalty is applied to the objective function, with the value of the penalty related to the degree in which the constraints are violated. The resulting penalized objective function quantitatively represents the extent of the violation of constraints and provides a relatively meaningful measurement of the performance of each solution string. Consider a problem where displacement and stress constraints are imposed. Each element is checked for stress violation, and each model node is checked for displacement violation. If no violation is found, then no penalty is imposed on the objective function. If a constraint is violated then the penalty is defined on the objective function.

There are several penalty methods. These are;

- Death Penalty
- Static Penalties
- Dynamic Penalties
- Annealing Penalties
- Adaptive Penalties
- Segregated GA
- Co-evolutionary Penalties

In this study, static penalty method is used. In this method, penalty parameters don't depend on the current generation number and a constant penalty is applied to unfeasible solutions. The individuals are simply evaluated by using

$$eval(x) = f(x) + \sum_{i=1}^{N} R_i \times C_i^{\,L}$$

where R indicates the penalty coefficient, N indicates the number of constraint types in consideration, C is number of constraint violation and L is the proportionality constant.

3.3.3 Selection

A GA uses fitness as a discriminator of the quality of solutions represented by the chromosomes in a GA population. The selection component of a GA is designed to use fitness to guide the evolution of chromosomes by selective pressure. Chromosomes are therefore selected for recombination on the basis of fitness. Those with higher fitness should have a greater chance of selection than those with lower fitness, thus creating a selective pressure towards more highly fit solutions. Selection is usually with replacement, meaning that highly fit chromosomes have a chance of being selected more than once or even recombined with themselves. There are many different selection schemes. Most common selection schemes are Rank selection and Tournament selection.

In Rank selection, all designs in the population must be ranked from best to worst according to the value of each designs fitness. A roulette wheel is implemented where the i_{th} ranked design in the population is given an interval $[\Phi_{i-1};\ \Phi_i)$, whose size depends on the population size, P, and its rank, i, in the population:

$$\Phi_i = \Phi_{i-1} + \frac{2\cdot(P-i+1)}{P\cdot(P+1)}$$

where $\Phi_0 = 0$; and $i = 1,.....,P$. A random number is generated between 0 and 1; design i is selected as a parent if the number lies in the interval $[\Phi_{i-1};\ \Phi_i)$.

When the wheel is spun (simulated by using a random number generator between 0 and 1, where the circumference of the wheel is normalized to be 1), those designs that occupy larger slices of the wheel have a better chance to be chosen as parent designs.

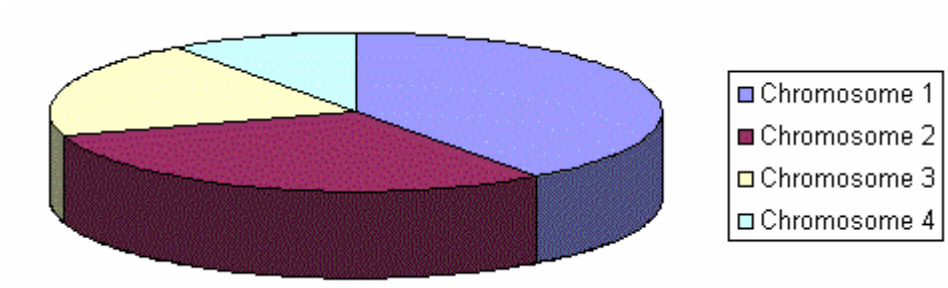


Figure – 3.3 Roulette Wheel

The Tournament selection is very simple and needs less processes. In this selection scheme, a number of individuals (typically between 2 and 7 individuals) are chosen randomly from the population and the best individual from this group is selected as parent. This process is repeated as often as individuals must be chosen. Tournament Selection is naturally elitist.

3.3.4 Crossover and Mutation

Selection alone cannot introduce any new individuals into the population (i.e., it cannot find new points in the search space). These are generated by genetically inspired operators, of which the most well known are crossover and mutation. Crossover is sometimes referred to as recombination.

The crossover and mutation are the most important part of a genetic algorithm. The performance of the algorithm is mainly influenced by these two operators. Usually, there is a predefined probability of procreation via each of these operators. Traditionally, these probability values are selected such that crossover is the most frequently used, with mutation being resorted to only relatively rarely. This is because the mutation operator is a random operator and serves to introduce diversity in the population. The kind of operator to be applied to each member of the gene pool is determined by random choice based on these probabilities.

The crossover operator functions on the breeding pool. Crossover is one of the genetic operators used to recombine the population genetic material. It takes two chromosomes and swaps part of their genetic information to produce new chromosomes. This operation is similar to sexual reproduction in nature. There are several types of crossovers that include single crossover also known as one-point crossover, two-point crossover, and uniform crossover among others.

In one-point crossover, a crossover point is selected randomly within a chromosome, and then the two parent chromosomes at this point are interchanged to produce two new offspring.

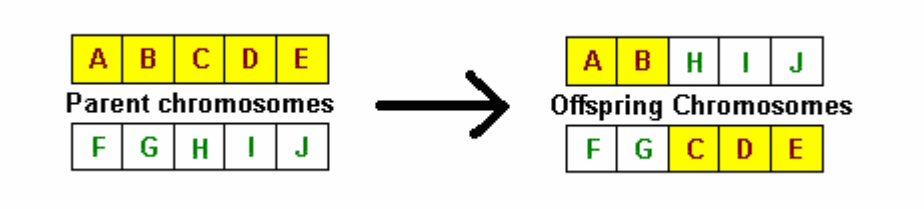


Figure – 3.4 One-Point Crossover

Similarly, in two-point crossover, two crossover points are selected randomly within a chromosome, then the two parent chromosomes between these points are interchanged to produce two new offspring.

Figure – 3.5 Two-Point Crossover

Uniform crossover is a crossover operator that decides (with some probability) which parent will contribute each of the gene values in the offspring chromosomes. This allows the parent chromosomes to be mixed at the gene level rather than the segment level (as with one and two point crossover).

Figure – 3.6 Uniform Crossover

In mutation, instead of exchanging cross-sections of a given two strings, the mutation operator randomly alters each gene with a small probability (i.e. 0.001). The main objective of the mutation operator is to produce a variety of different strings. The traditional view is that crossover is more important of the two techniques for rapidly exploring a search space. Mutation provides a small amount of random search, and helps ensure that no point in the search space has a zero probability of being examined.

Figure – 3.7 Mutation

3.3.5 Termination criteria

There are several termination criterias used in GAs .The most common termination criteria is to put a limit on the maximum number of generation. When the number of generations reaches to a predefined value then the optimization process stops. Another common criterion is based on the percentage of identical solutions in the population. If the percentage of identical solutions are higher or equal to the predefined percentage value then the optimization process is terminated.

A criterion based on no improvement tolerance can also be used in GAs. This criterion checks for the number of generations with no improvement in the best solution obtained and it terminates the optimization process based on the predefined tolerance.

3.3.6 GA Implementation

The pseudo-code for GA approach is as follows:

- Define the objective function (environment). Objective function is used in evaluating the designs fitnesses.

- Define the chromosome structure (genetic representation of the system) suitable for the problem in consideration. The characteristics of an organism are provided in the gene strings of each chromosome. All the design variables should be placed somewhere inside chromosome structure.

- Generate a random population of specific size (Initial population). The population size affects the efficiency and performance of GA. GA does poorly for very small size of populations and very large population size impacts performance of the algorithm. For typical applications, the suggested range is between 10-160 chromosomes.

- Evaluate the fitness of every solution over the objective function. Each organism is then placed into a common environment where it competes and breeds with other members of the population

- Select two parent chromosomes for mating from a population according to their fitness (the better fitness, the bigger chance to be selected) by a random selection method e.g. tournament selection and rank selection. The fittest organisms in the population are given the best opportunity to become parents of a child and may survive into the next generation.

- Apply crossover operation on the selected pairs if they have been chosen for crossover (based on probability of crossover). The main objective of crossover is to take good characteristics from organisms in the parent population and

create child organisms which will hopefully be better suited to their environment than their predecessors

- Based on the probability of mutation, mutate new offsprings at each locus (position in chromosome). Once offsprings have been created, they may be exposed to a mutation operator that allows for the introduction of new, random information that may aid the algorithm in creating stronger organisms.

- Replace the initial population with new generated population.

- Go through all the steps until the termination criteria met.

3.3.7 GA Comparison with Other Optimization Techniques

Under the umbrella of evolutionary computation also referred to as evolutionary algorithms (EAs) are the areas of evolutionary programming and evolution strategies. Each of these methods of evolutionary computation simulate the process of evolution through the mutation, selection, and/or reproduction processes and rely on perceived performance of individual structures assigned by the environment. Evolutionary algorithms support population structures, which progress to the rules of selection using genetic operators. Genetic operators determine which structures will move on to the next level and which will not. In essence, the individuals in the population obtain a degree of fitness from the environment. Reproduction concentrates on high fitness individuals. GAs are one of many that have been grouped in a class of search and optimization schemes called evolutionary algorithms (EAs).

GAs form a subset field of evolutionary computation, inspired by biological and evolutionary systems and provide an approach to learning that is based on simulated evolution. Genetic algorithms are a new generation for optimization and search

techniques. They have become very popular because of their efficiency, power, utility, and accessibility.

GAs function in a totally different way compared to the traditional search and optimization algorithms. There are major points that make the Genetic algorithms unique. First of all, The GA does not operate directly with the design variables, but with a coding of the design variables (typically string representations).

Secondly, The GA only requires evaluation of the objective function and does not require gradients of the object or constraint equations. Thus, GAs can be used to solve design problems in which discontinuous functions must be evaluated. Gradient-based optimization schemes are ideally suited to smooth functions with a single optimum point but would have difficulty with the non-smooth functions. Conditions on continuity and the existence of derivatives necessary for optimization and search methodologies limit the use of Derivative-based optimization techniques to a certain domain of problems. This brings us to the conclusion that calculus-based schemes lack the power to provide robust search through optimization techniques.

Thirdly, The GA evaluates the objective function for several randomly selected design points rather than sequentially stepping from one point to the next based on gradient information (i.e. hill climbing). . With this property, a genetic algorithm can search many areas of the design space at once. While one design area that contains good information is being exploited, other areas of the design space can be explored. Working with a population of designs also makes GAs less susceptible to difficulties encountered in problems with noisy design spaces. On the other hand, calculus-based methods work with a single point at a time and may have problems with functions with local optimum points.

Finally, The GA simulates the rules of natural genetic evolution by systematically applying selection, evaluation, crossover, and mutation operations. By making random moves in the design space, a GA can avoid local optima points easier.

However, it is important to note that GAs are not random searches, but use random information to help guide an organized search scheme toward an optimum point of coded parameter space. Unlike gradient-based methods, a GA can be initialized with the same population and converge to different solutions also.

CHAPTER 4

GABSO: GENETIC ALGORITHM BASED STRUCTURAL OPTIMIZER

GABSO is a genetic algorithm based optimization code. It is written in FORTRAN. It can be used for any kind of optimization problem. It is capable of using MSC/NASTRAN® or any in-house code for constraint evaluation.

All the examples demonstrated in the next section had been solved by using GABSO. The code gives two output files, a file containing the population information in each generations and a summary file containing the optimization parameters list, objective function history and the best design in each generation.

The code consists of one main code and five subroutines. And it also contains a finite element solver for 2D and 3D truss structures. Namely, the main code and these subroutines are;

- GABSO - main code
- INPUT
- INITIAL_POP
- FITNESS
- TRUSS
- CROSSOVER.
- MUTATION

4.1 Input.f90

This subroutine is used for input file reading. The input file have two parts, the first part contains detailed information about the Finite element model and mainly used for truss optimization. In this part, the coordinates of the nodes, DOFs, the element

connectivities, the element property coupling information, the element properties and also the loading conditions are listed.

The second part of the file contains detailed information about optimization parameters. The number of design variables, population size, selection type, crossover type, crossover and mutation probabilities, number of elite designs, no improvement tolerance, design variables weight constants and also the discrete value list are read through this section.

4.2 Initial_Pop.f90

In this subroutine, initial population of designs is formed. This procedure is a random procedure, in each run the code forms a new set of designs based on the time of the computer in use. The code obtains the seed number by combining the current hour, minute and seconds information.

According to number of design variables, initial population size and the number of discrete design variable values read from the input file, the code forms an initial population.

4.3 Fitness.f90

By using this subroutine, the fitnesses of all individuals in the population are evaluated. The fitness evaluation is done by considering the objective function value. The constraints are represented in the objective function by penalty terms defined according to the problem.

For constraint violations, the FEA must be performed. This is done either by using TRUSS.f90 subroutine, for truss structures or by using MSC/NASTRAN® finite element solver. This capability of the code makes it possible to solve all kinds of

structural optimization problems. The code updates the bdf file based on the design in consideration, MSC/NASTRAN® input file, and execute MSC/NASTRAN® by this bdf file. After waiting the finalization of MSC/NASTRAN® run, the code opens the f06 file and gets the necessary information to evaluate the constraint violation.

GA is an optimization algorithm that needs lots of function evaluation (or FEA). This is the reason of huge time consumption associated with GAs. In order to minimize the effect of this disadvantage; a cache system is introduced in the code. A database is formed by the elements for which function evaluation is performed and this database is used for the fitness evaluations of the repeated designs. This prevents the unnecessary function evaluations for the repeated designs and leads to a considerable reduction in runtime.

Mutation probability variation is a unique tool used in GABSO. The mutation rate can be varied during the optimization process according to the population fitness characteristics. There is an adaptive mutation probability algorithm in the code which is activated when the process stucks. Most of the time, this is a situation encountered when Rank selection and Elitisizm are used in the optimization process, the algorithm may converge to a local optimum very quickly, and stucks at that point. So in order to achieve the necessary variation in the population to jump from local optimum, the mutation probability is increased. After obtaining the variation, the mutation probability is set to its predefined value.

4.4 Truss.f90

This is a subroutine used for optimization of 2D and 3D truss structures. All the necessary details about the Finite element model are read from the FEM part of the input file. The subroutine calculates the element stresses and nodal displacements.

4.5 Crossover.f90

Crossover is one of the genetic operators used to recombine the population genetic material. It takes two chromosomes and swaps part of their genetic information to produce new chromosomes. This operation is similar to sexual reproduction in nature. There are several types of crossovers that include single crossover also known as one-point crossover, two-point crossover, and uniform crossover among others.

This subroutine used for crossover operation. In GABSO, It is possible to select any of these 3 types of crossover, one-point crossover, two-point crossover and uniform crossover. According to the type of the problem in consideration, the appropriate crossover type should be selected.

And also two type of selection scheme are available in GABSO, Rank selection and Tournament selection. The selection and crossover type is read from the input file.

4.6 Mutation.f90

In this subroutine, mutation is applied. Instead of exchanging cross-sections of a given two strings, the mutation operator randomly alters each gene with a small probability (i.e. 0.001). The main objective of the mutation operator is to produce a variety in the population.

In GABSO, there are two types of mutation, random mutation and creep mutation. The probability of using one of these mutation types is defined within the code. The probability is the same for both of these mutation types. So half of the mutation operations in the code is random mutation and the other half is creep mutation.

4.7 Gabso.f90

This is the main code. It controls all the optimization process. Also, The Elitisizm and the Tunning operations are done by the main code.

Simply, the Elitisizm can be defined as cloning some of the best designs to the next population directly. This leads to fast convergence but the algorithm generally stucks at local optimums when elitisizm is used.

Tunning is another unique tool of GABSO, and it searchs the neighborhood of the optimum, obtained by the algorithm and checks the possibility to have a better solution in the neighborhood. This utility is activated at the end of the optimization process.

CHAPTER 5

DESIGN EXAMPLES

In this section, first, four test problems are demonstrated in order to test the performance of the optimization algorithm. Secondly, structural optimization problems are solved and compared with the previous results in literature or with the results obtained by using conventional optimization programs (i.e. GENESIS)

5.1 Test Problems

The four test problems in this section are all very difficult global optimization problems, on which most conventional local optimization algorithms would fail miserably.

5.1.1 Test Problem 1

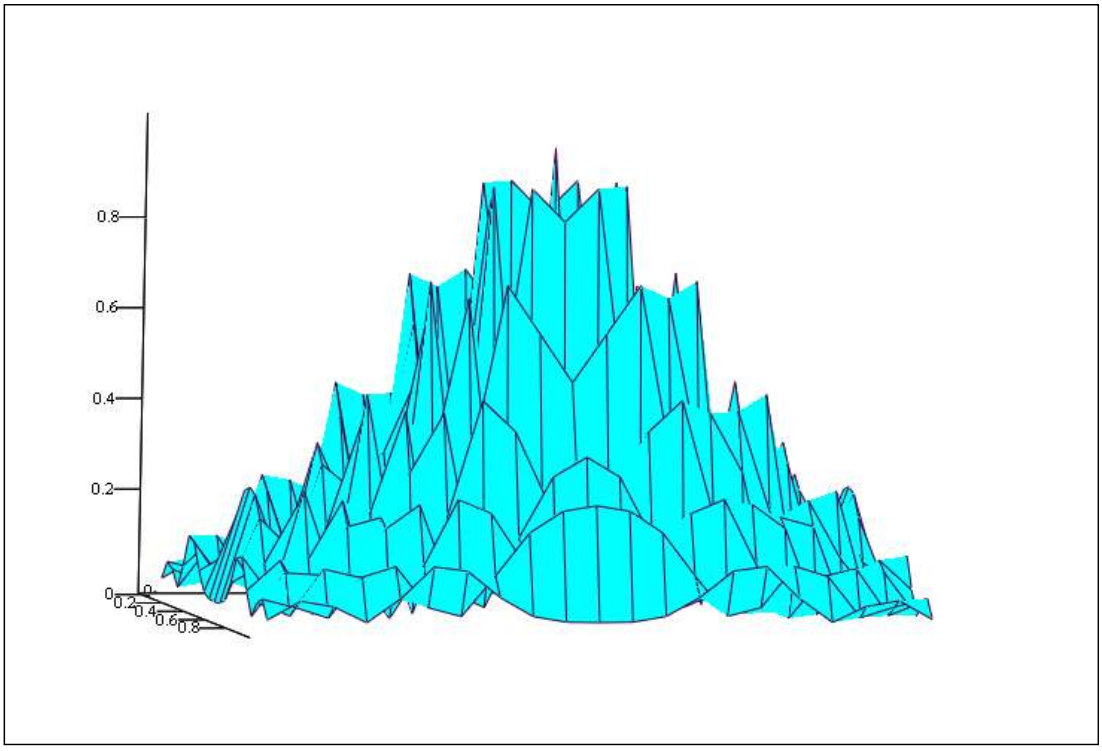


Figure – 5.1 Test Problem 1

The first test problem is defined as follows:

$$f(x,y) = \cos^2(n \cdot \pi \cdot r) \cdot \exp\left(\frac{r^2}{\sigma^2}\right)$$

$$r^2 = (x-0.5)^2 + (y-0.5)^2, \qquad x, y \in [0,1],$$

where, $n = 9$ and $\sigma^2 = 0.15$ are constants. The objective is to maximize the function defined above. The design variables are x and y.

The algorithm gives the maximum at point $(x,y) = (0.5,0.5)$ with $f(x,y) = 1$. This can also be observed from Figure 5.1. The optimization parameters used in the calculation are as follows:

Number of design variables : 2
Number of genes : 6
Initial population : 20
Selection type : Rank Selection
Crossover type : Single-Point crossover
Crossover probability: 0.9
Mutation probability : 0.05
Elitist selection : None

The genetic algorithms use discrete domains for the optimization problems. In order to solve this continuous optimization problem, 2 design variables in consideration are represented by using 6 genes giving an accuracy of 0.001.

The convergence is obtained at generation number 100. The search space is 10^6. The convergence is obtained by only considering less than $\frac{20 \times 100}{10^6} \cdot 100 = 0.2\%$ of the search space.

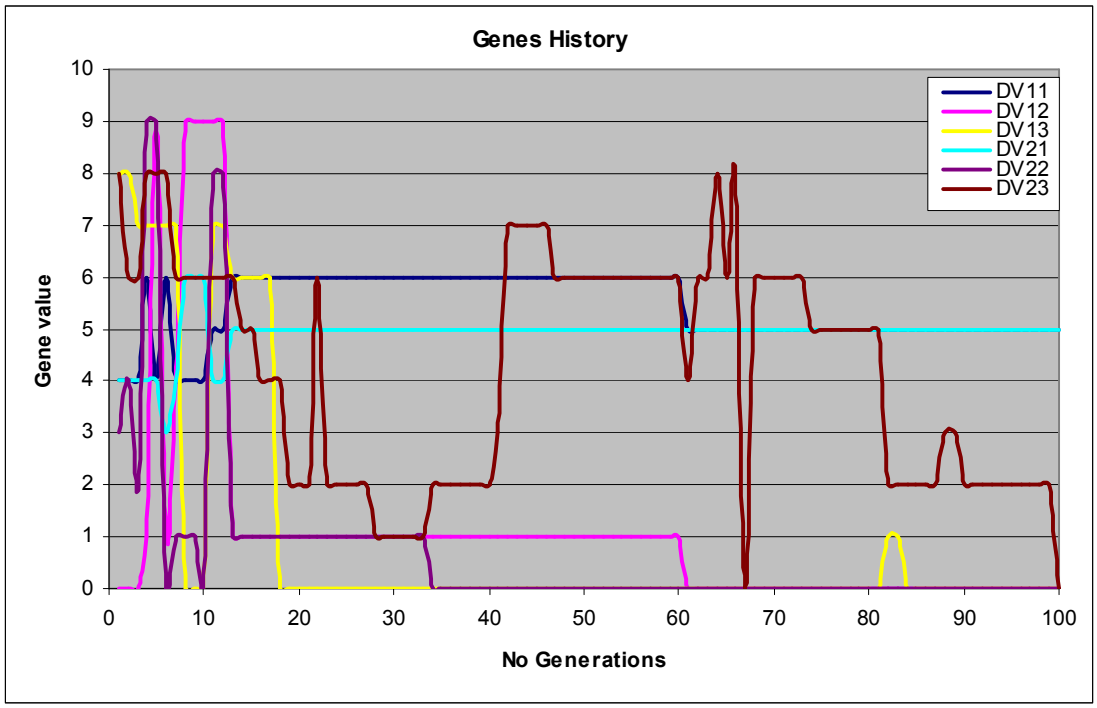


Figure – 5.2 Genes History for Test Problem 1

$$DesignVariable1 = DV11 \times 0.1 + DV12 \times 0.01 + DV13 \times 0.001$$

$$DesignVariable2 = DV21 \times 0.1 + DV22 \times 0.01 + DV23 \times 0.001$$

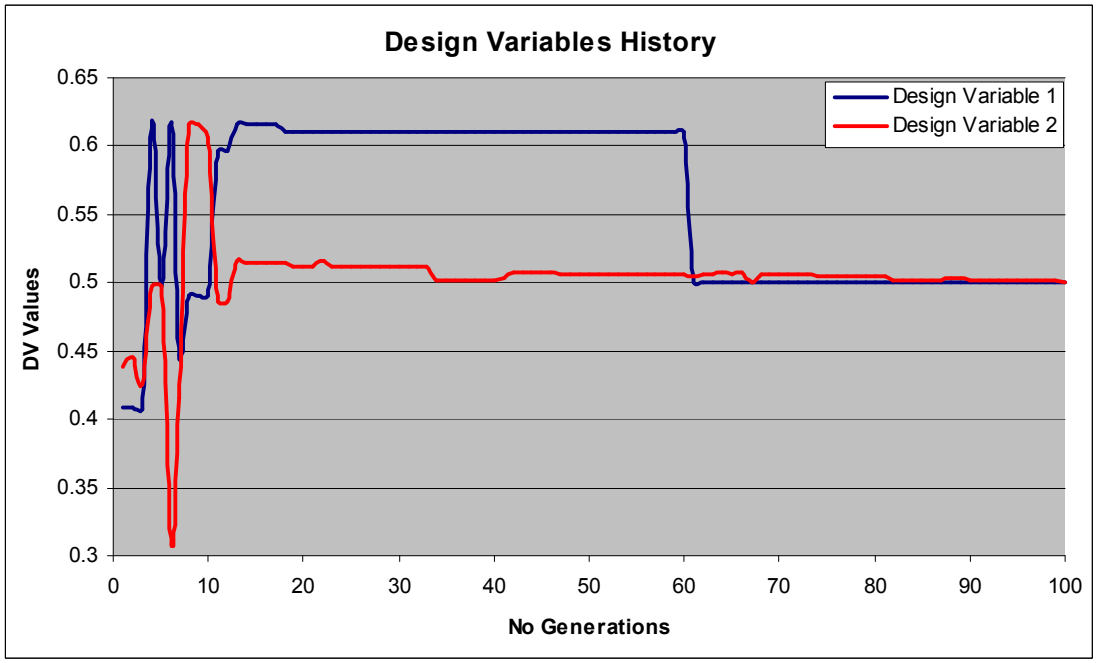


Figure – 5.3 Design Variables History for Test Problem 1

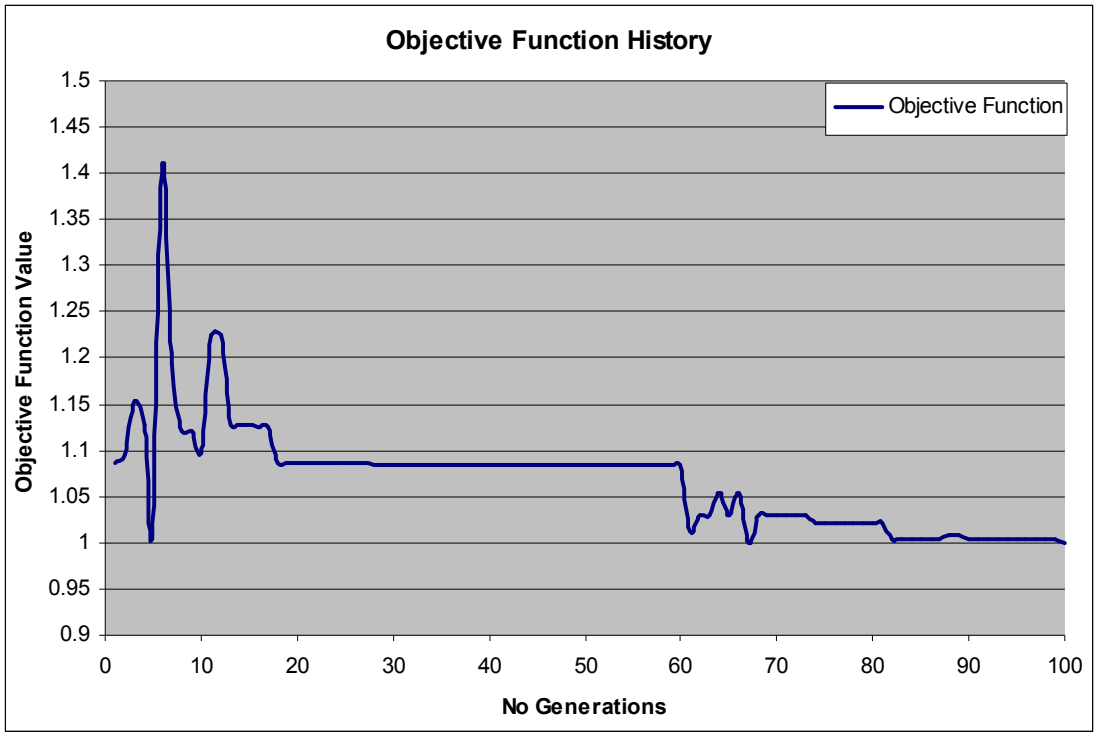


Figure – 5.4 Objective Function History for Test Problem 1

5.1.2 Test Problem 2

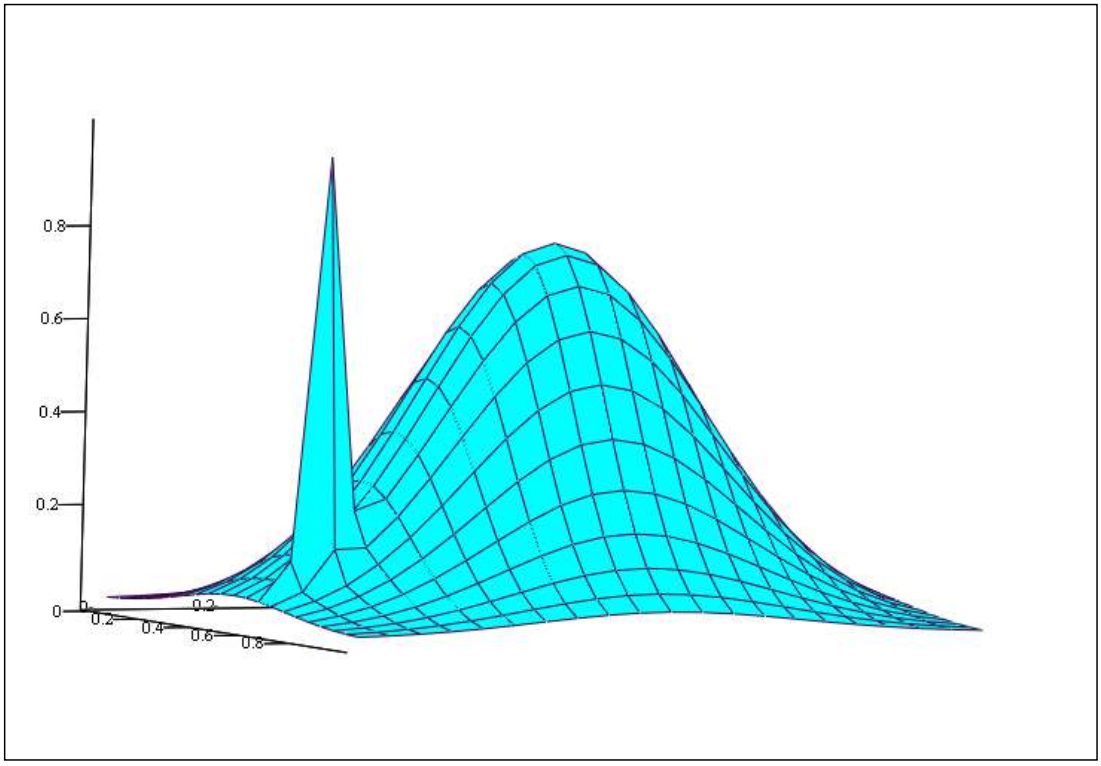


Figure – 5.5 Test Problem 2

The second test problem is defined as follows:

$$f(x,y) = 0.8 \cdot \exp\left(-\frac{r_1^{\,2}}{(0.3)^2}\right) + 0.879008 \cdot \exp\left(-\frac{r_2^{\,2}}{(0.03)^2}\right)$$

$$r_1^{\,2} = (x-0.5)^2 + (y-0.5)^2,$$

$$r_2^{\,2} = (x-0.6)^2 + (y-0.1)^2$$

The objective is to maximize the function defined above. The design variables are x and y.

The algorithm gives the maximum at point $(x,y) = (0.6,0.1)$ with $f(x,y) = 1$. The optimization parameters used in the calculation are as follows:

Number of design variables : 2

Number of genes : 6

Initial population : 20

Selection type : Tournament Selection

Crossover type : Single-Point crossover

Crossover probability: 0.9

Mutation probability : 0.06

Elitist selection : 1

The genetic algorithms use discrete domains for the optimization problems. In order to solve this continuous optimization problem, 2 design variables in consideration are represented by using 6 genes giving an accuracy of 0.001. The convergence is obtained at generation number 70. The search space is 10^6. The convergence is obtained by only considering less than $\frac{20 \times 70}{10^6} \cdot 100 = 0.14\,\%$ of the search space.

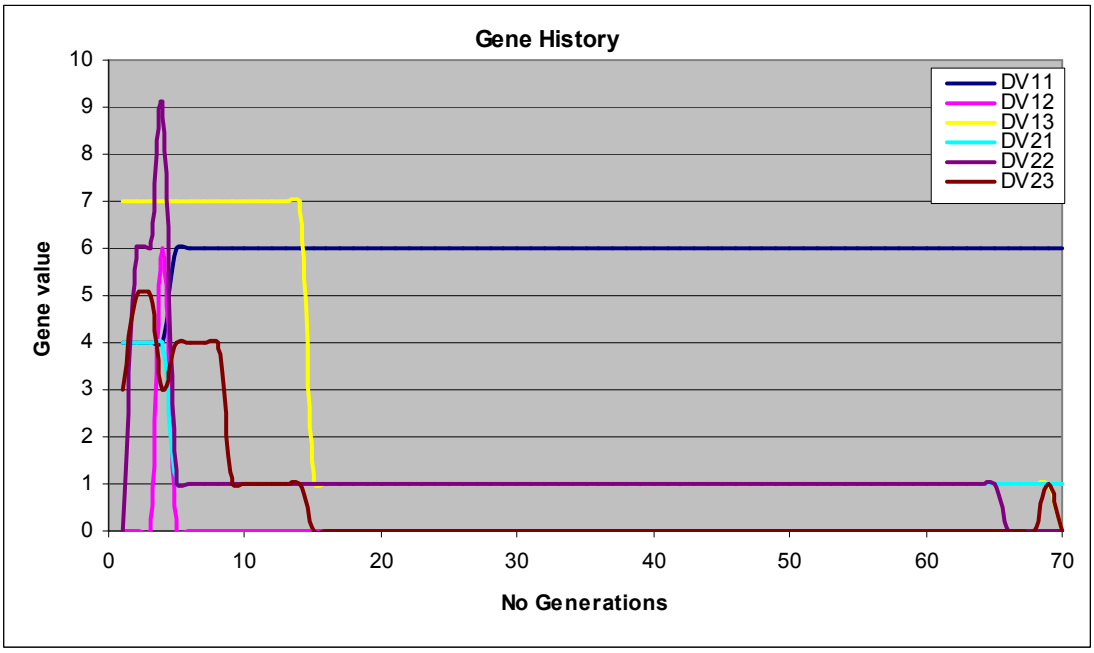


Figure – 5.6 Genes History for Test Problem 2

$$Design Variable1 = DV11 \times 0.1 + DV12 \times 0.01 + DV13 \times 0.001$$

$$Design Variable2 = DV21 \times 0.1 + DV22 \times 0.01 + DV23 \times 0.001$$

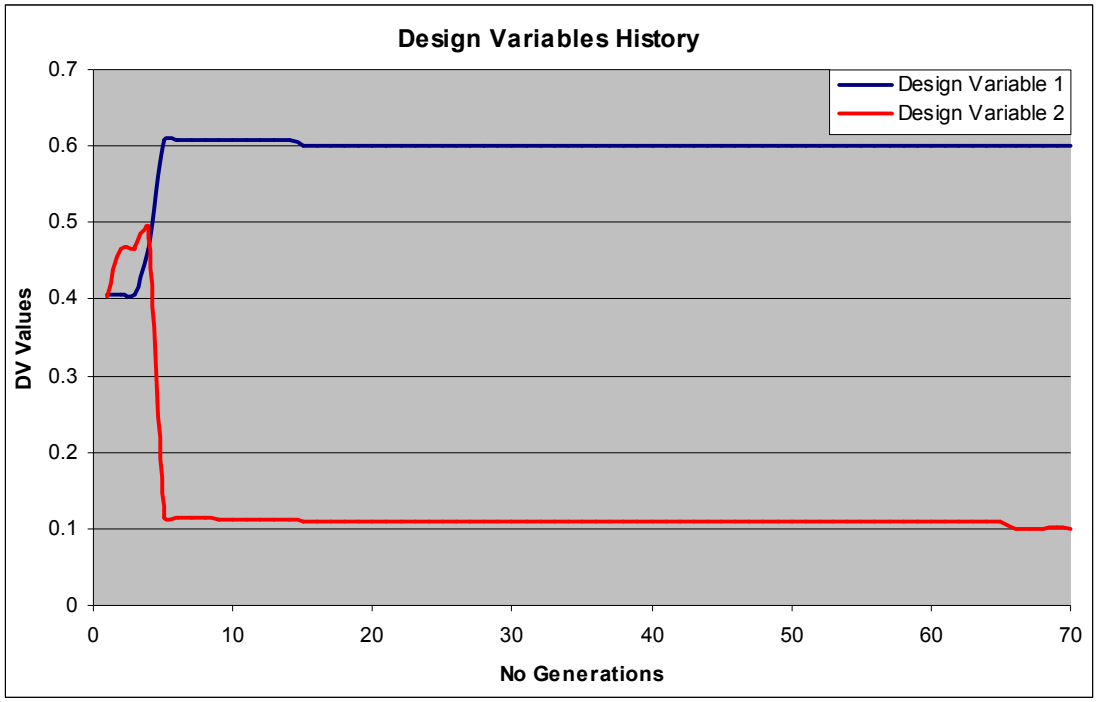


Figure – 5.7 Design Variables History for Test Problem 2

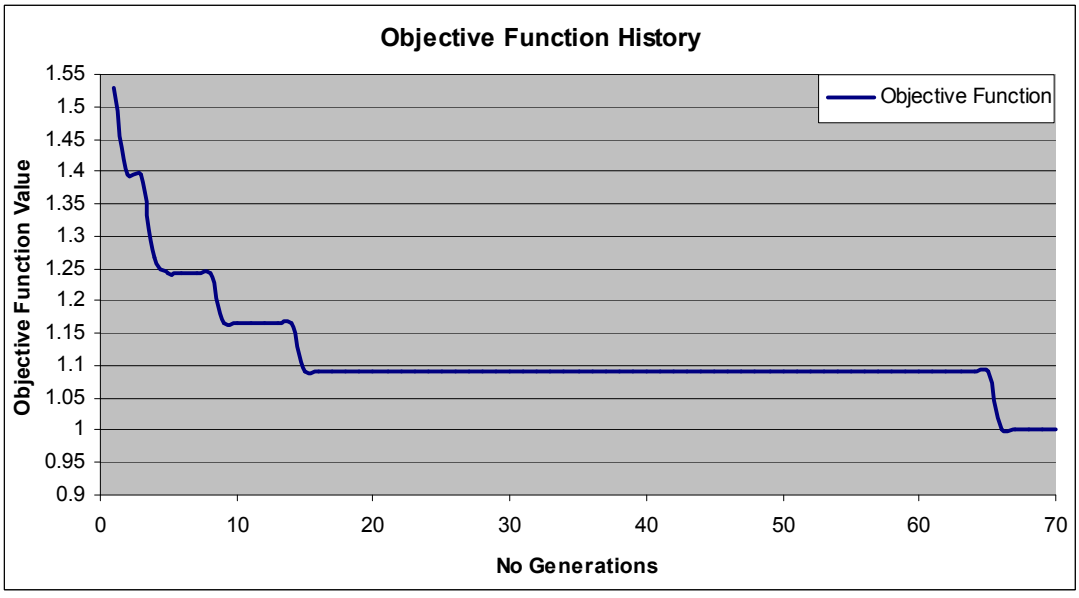
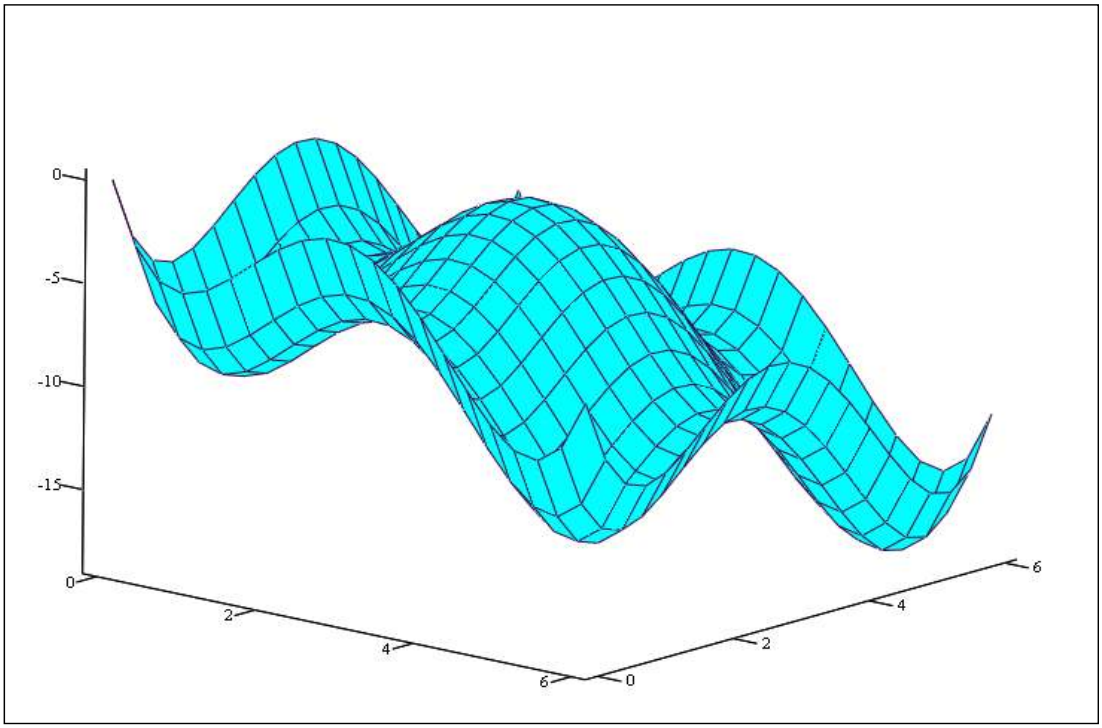


Figure – 5.8 Objective Function History for Test Problem 2

5.1.3 Test Problem 3



f

Figure – 5.9 Test Problem 3

The third test problem is defined as follows:

$$f(X_1, X_2) = 0.25 \cdot X_1^4 - 3 \cdot X_1^3 + 11 \cdot X_1^2 - 13 \cdot X_1 + 0.25 \cdot X_2^4 - 3 \cdot X_2^3 + 11 \cdot X_2^2 - 13 \cdot X_2$$

Subject to

$$g_1(X) \equiv 4 - X_1 - X_2 \leq 0$$

The objective is to minimize the function defined above. The design variables are X_1 and X_2.

The algorithm gives the maximum at point $(X_1, X_2) = (5.330, 5.330)$ with $f(X_1, X_2) = -18.568$. The optimization parameters used in the calculation are as follows:

Number of design variables : 2
Number of genes : 8
Initial population : 20
Selection type : Tournament Selection
Crossover type : Single-Point crossover
Crossover probability: 0.85
Mutation probability : 0.05
Elitist selection : None

The genetic algorithms use discrete domains for the optimization problems. In order to solve this continuous optimization problem, 2 design variables in consideration are represented by using 8 genes giving an accuracy of 0.001.

The convergence is obtained at generation number 81. The search space is 10^8. The convergence is obtained by only considering less than $\frac{20 \times 81}{10^8} \cdot 100 < 0.01\%$ of the search space.

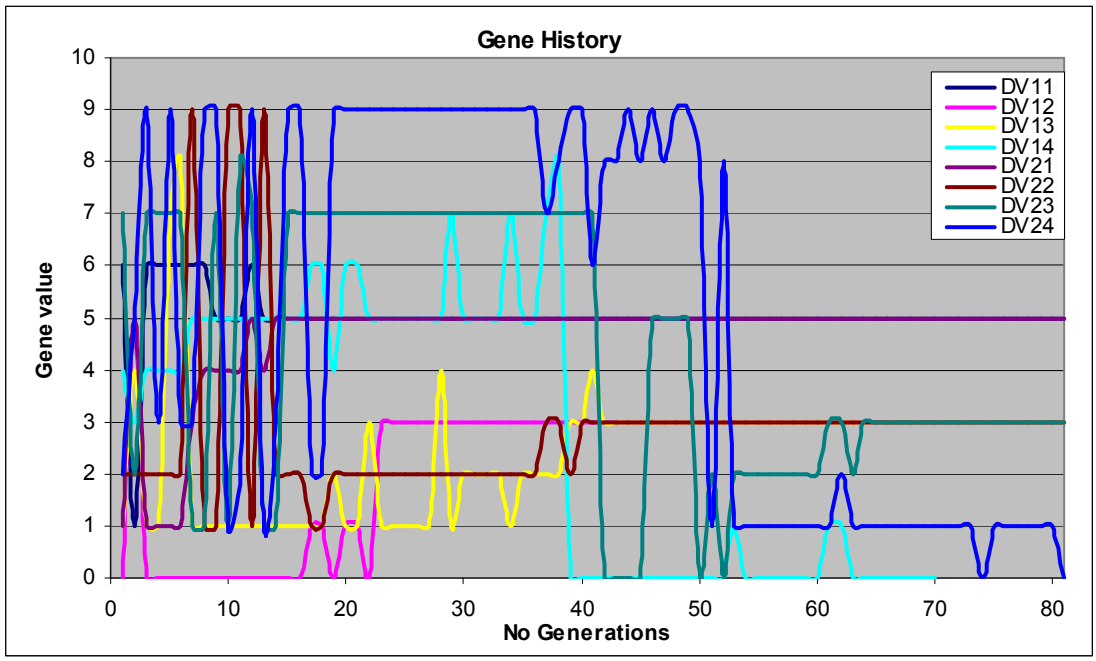


Figure – 5.10 Genes History for Test Problem 3

where design variables are defined as;

$$DesignVariable1 = DV11 + DV12 \times 0.1 + DV13 \times 0.01 + DV14 \times 0.001$$

$$DesignVariable2 = DV21 + DV22 \times 0.1 + DV23 \times 0.01 + DV24 \times 0.001$$

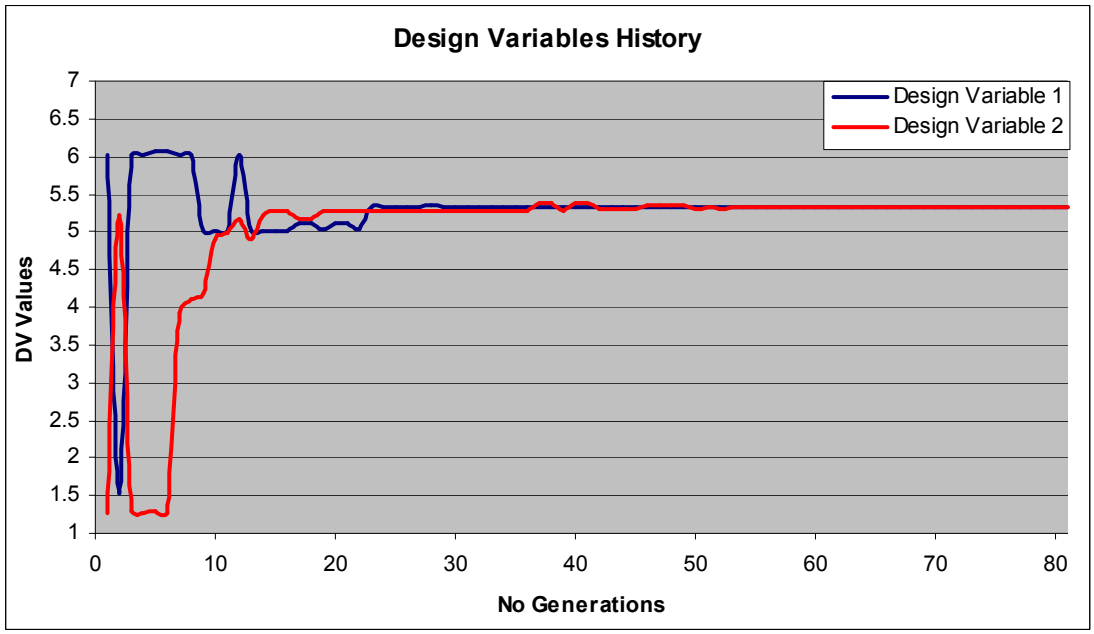


Figure – 5.11 Design Variables History for Test Problem 3

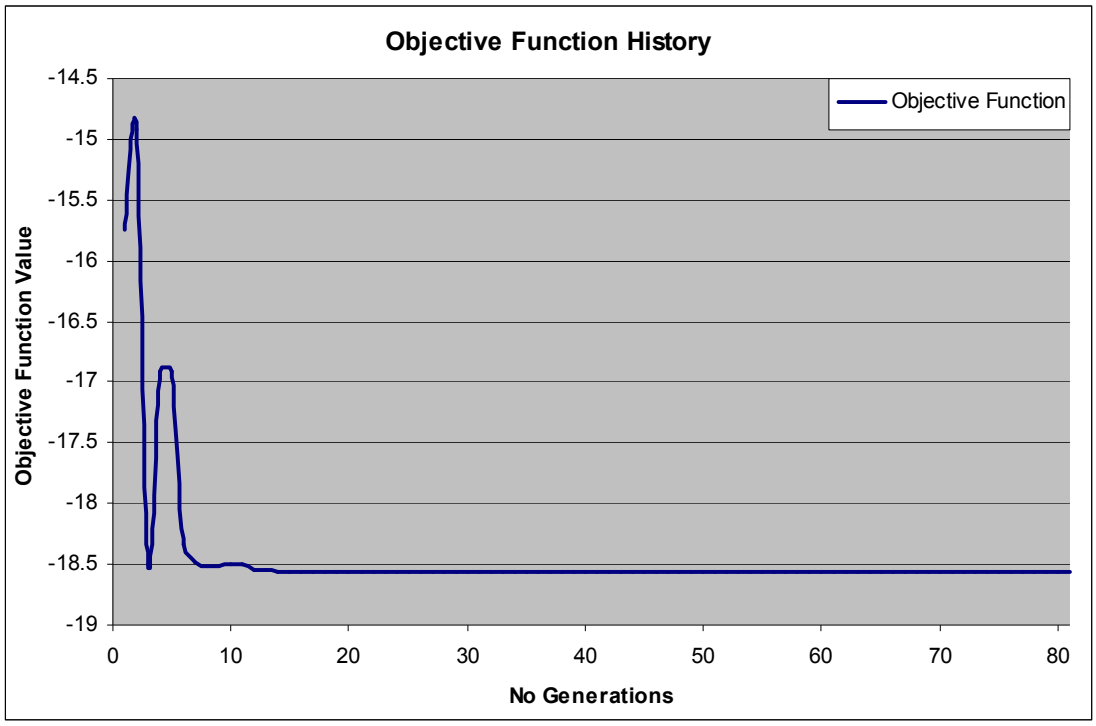
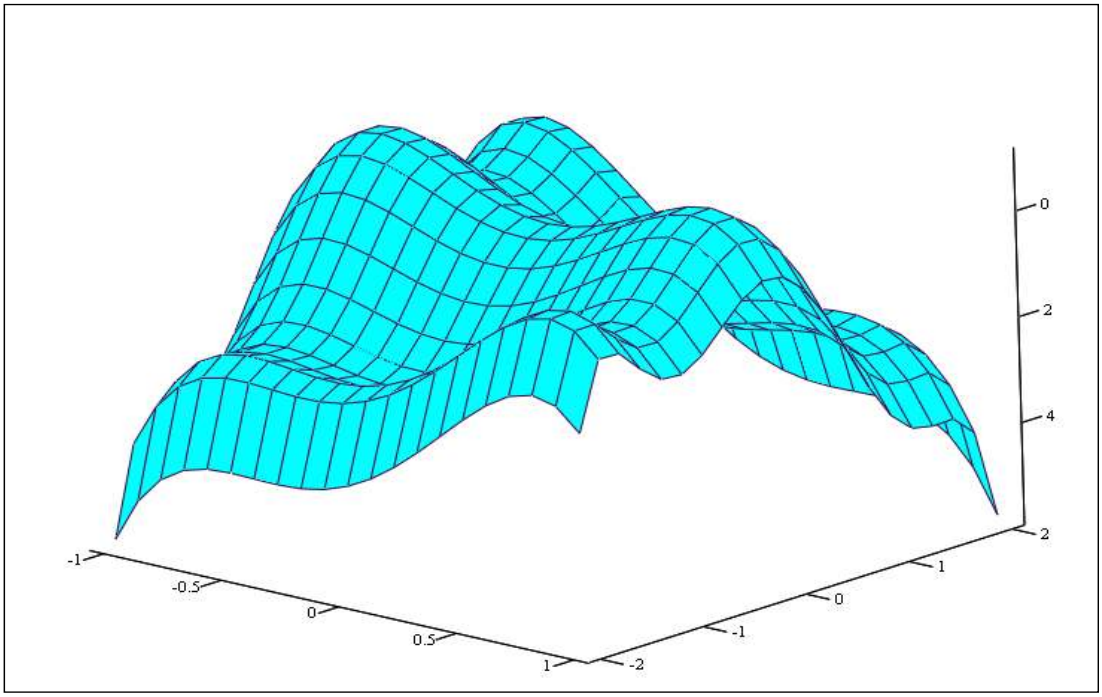


Figure – 5.12 Objective Function History for Test Problem 3

5.1.4 Test Problem 4 (Six-hump Camel Back Function)



f

Figure – 5.13 Test Problem 4

The last test problem is defined as follows:

$$f(x,y)=\left(4-2.1\cdot x^2+\frac{x^4}{3}\right)\cdot x^2+x\cdot y+\left(-4+4\cdot y^2\right)\cdot y^2;$$

where $-3\le x\le 3$ and $-2\le y\le 2$

Within the bounded region six local minima are located; two of them are global minima with $f(x,y)=-1.0316$

The objective is to minimize the function defined above. The design variables are x and y.

The algorithm gives the maximum at point $(x,y)=(0.0897,-0.7127)$ with $f(x,y)=-1.0316$. The optimization parameters used in the calculation are as follows:

Number of design variables : 2

Number of genes : 10

Initial population : 20

Selection type : Tournament Selection

Crossover type : Uniform crossover

Crossover probability: 0.96

Mutation probability : 0.08

Elitist selection : 1

The genetic algorithms use discrete domains for the optimization problems. In order to solve this continuous optimization problem, 2 design variables in consideration are represented by using 10 genes giving an accuracy of 0.0001. The convergence is obtained at generation number 62. The search space is 10^{10}. The convergence is obtained by only considering less than $\frac{20 \times 62}{10^{10}} \cdot 100 < 0.01\%$ of the search space.

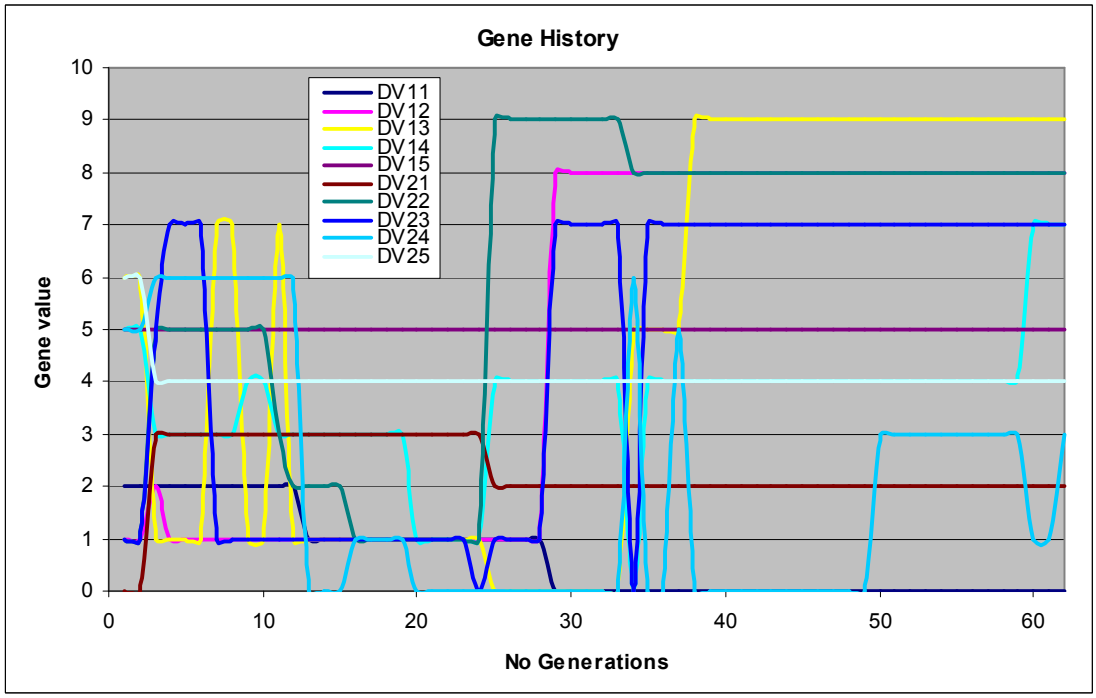


Figure – 5.14 Genes History for Test Problem 4

$$DesignVariable1 = DV11 \times 0.1 + DV12 \times 0.01 + DV13 \times 0.001 + DV14 \times 0.0001 + DV15 - 5$$

$$DesignVariable2 = DV21 \times 0.1 + DV22 \times 0.01 + DV23 \times 0.001 + DV24 \times 0.0001 + DV25 - 5$$

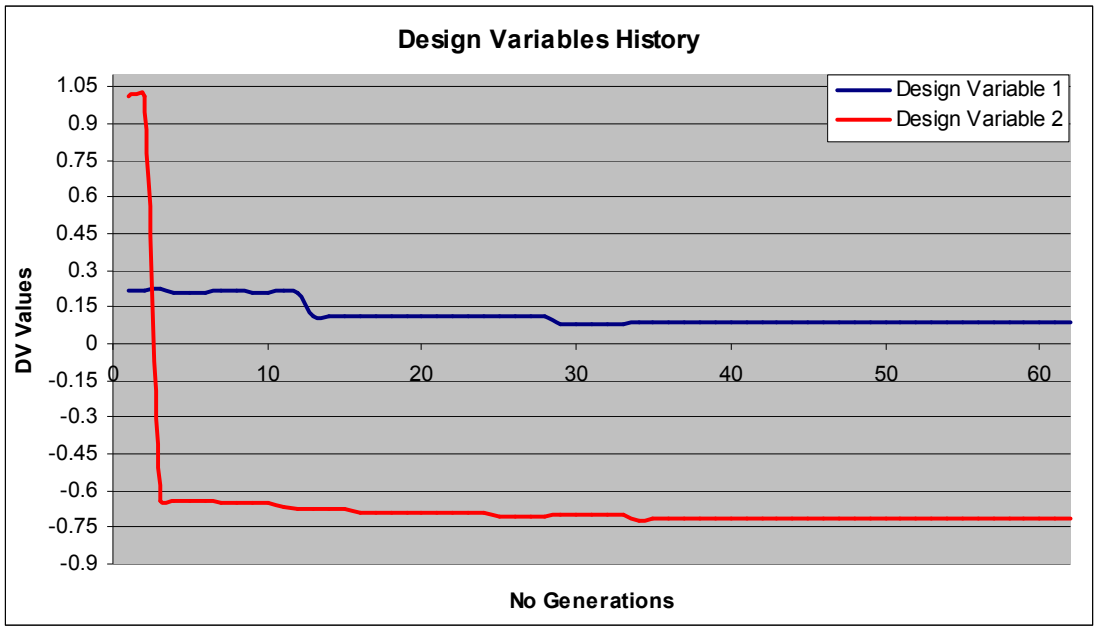


Figure – 5.15 Design Variables History for Test Problem 4

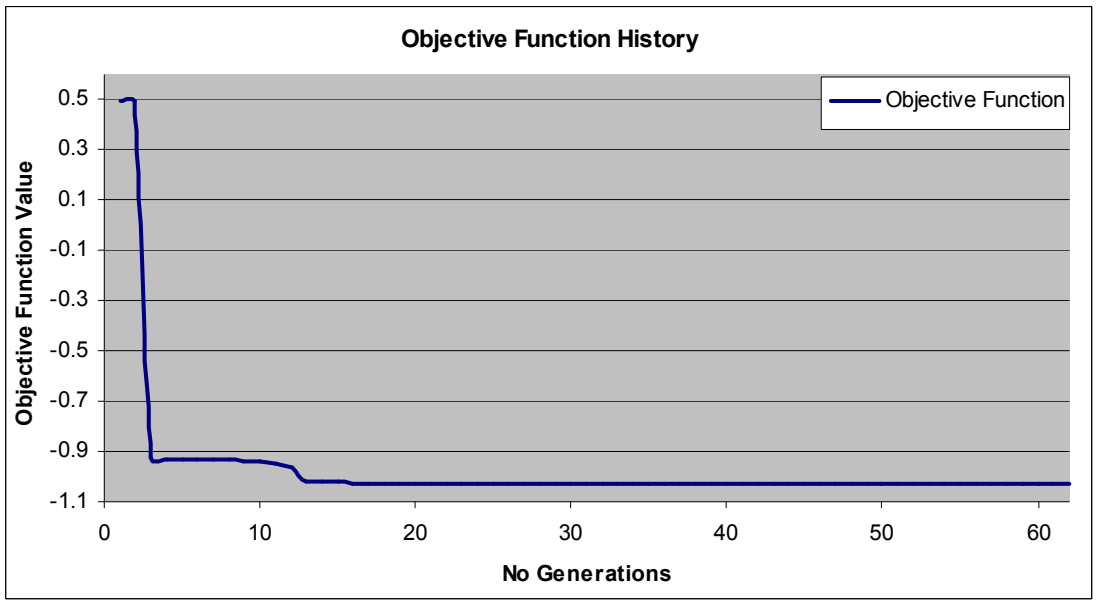


Figure – 5.16 Objective Function History for Test Problem 4

5.2 Structural Optimization Problems

In this section, 9 structural optimization problems are demonstrated. Structural optimization problems are solved and compared with the previous results in literature or with the results obtained by using conventional optimization programs (i.e. GENESIS)

5.2.1 6-bar truss

The first problem to be considered is the 6-bar truss as shown in Figure 5.17. The design variables are six sectional areas of the six members of the truss. There are two types of constraint in this problem, i.e., stress and displacement constraints.

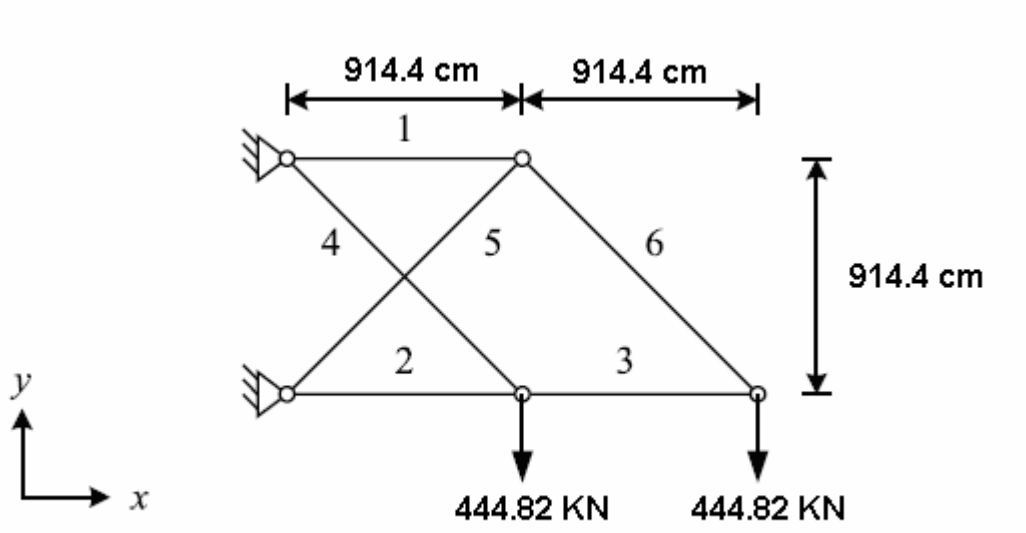


Figure – 5.17 6-bar truss

Design parameters are:

Modulus of Elasticity : 68.95 GPa

Density : 2767.99 kg/m^3

Allowable tensile stress : 172.37 MPa

Allowable compressive stress : 172.37 MPa

Maximum y-displacement : 5.08 cm

The optimization parameters used in the calculation are as follows:

Number of design variables : 6

Number of genes : 12

Initial population : 100

Selection type : Tournament Selection

Crossover type : Uniform crossover

Crossover probability: 1.00

Mutation probability : 0.08

Elitist selection : 1

The convergence is obtained at generation number 93. The search space is 5^{12}. The convergence is obtained by only considering less than $\frac{100 \times 93}{5^{12}} \cdot 100 < 0.01\%$ of the search space.

Table 5.1 – Comparison of results with Ref [8]

Size of member (cm^2)	***Ref [8]***	***Present***
A_1	193.55	180.64
A_2	128.39	154.84
A_3	100.00	90.32
A_4	46.58	38.71
A_5	141.94	148.39
A_6	141.94	135.48
Total Weight (N)	**22072.52**	**21892.37**

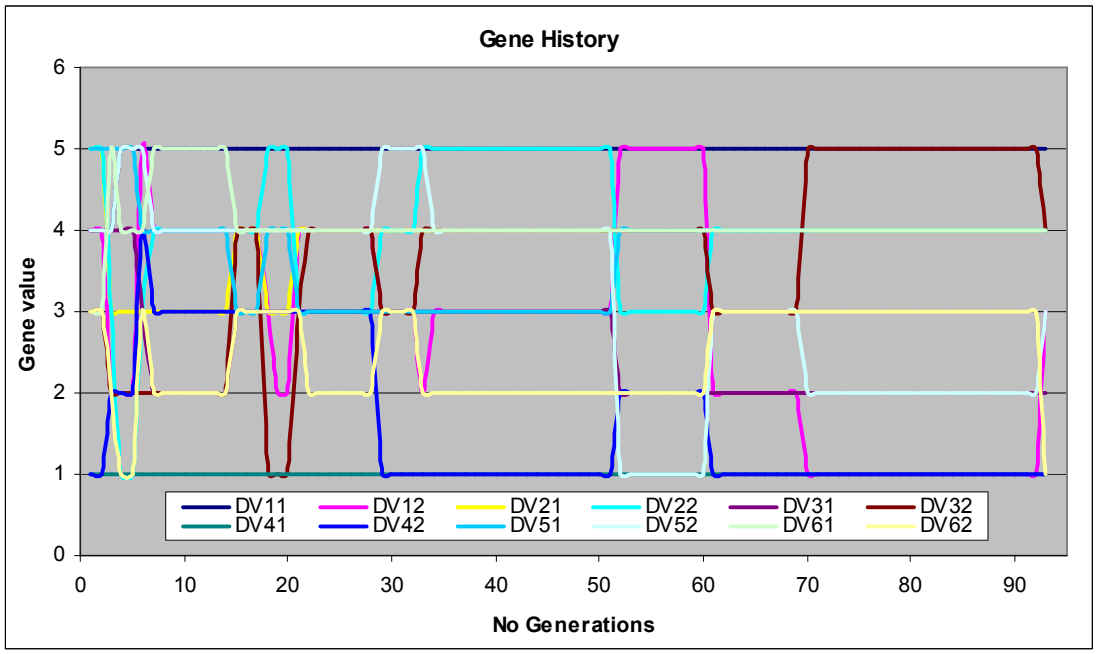


Figure – 5.18 Genes History for 6-bar truss

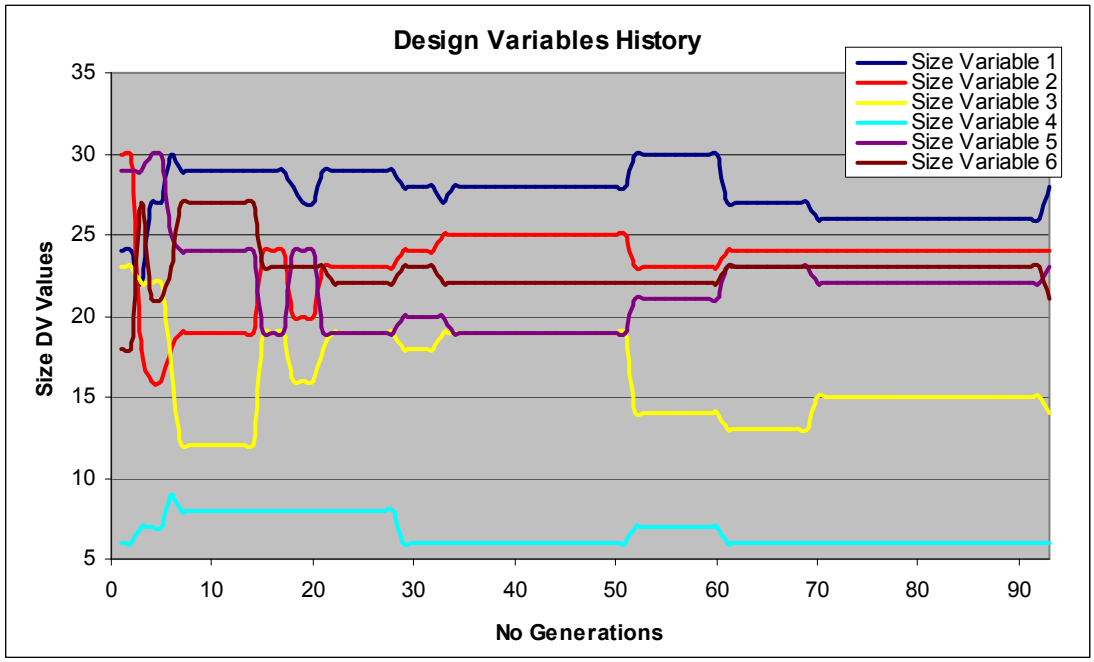


Figure – 5.19 Design Variables History for 6-bar truss

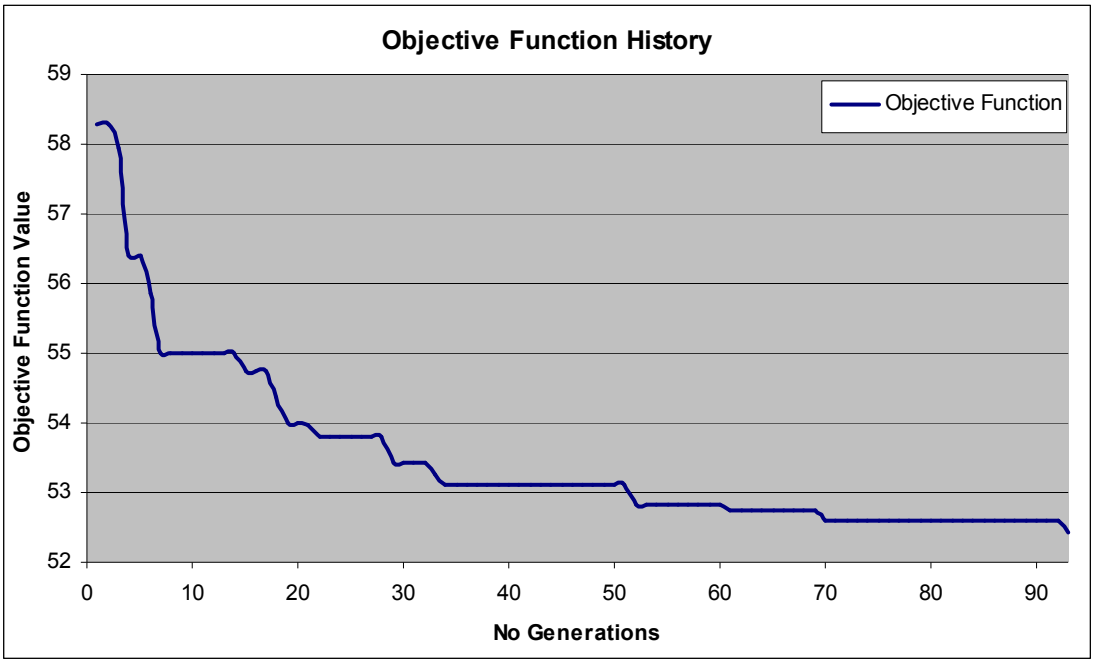


Figure – 5.20 Objective Function History for 6-bar truss

5.2.2 25-bar space truss

The 25-bar space truss problem is solved by considering weight minimization as the objective and cross sectional areas of the truss members as design variables. Due to symmetry of the 25-bar truss, only eight different member sizes are allowed, and hence eight independent design variables are selected by linking various member sizes.

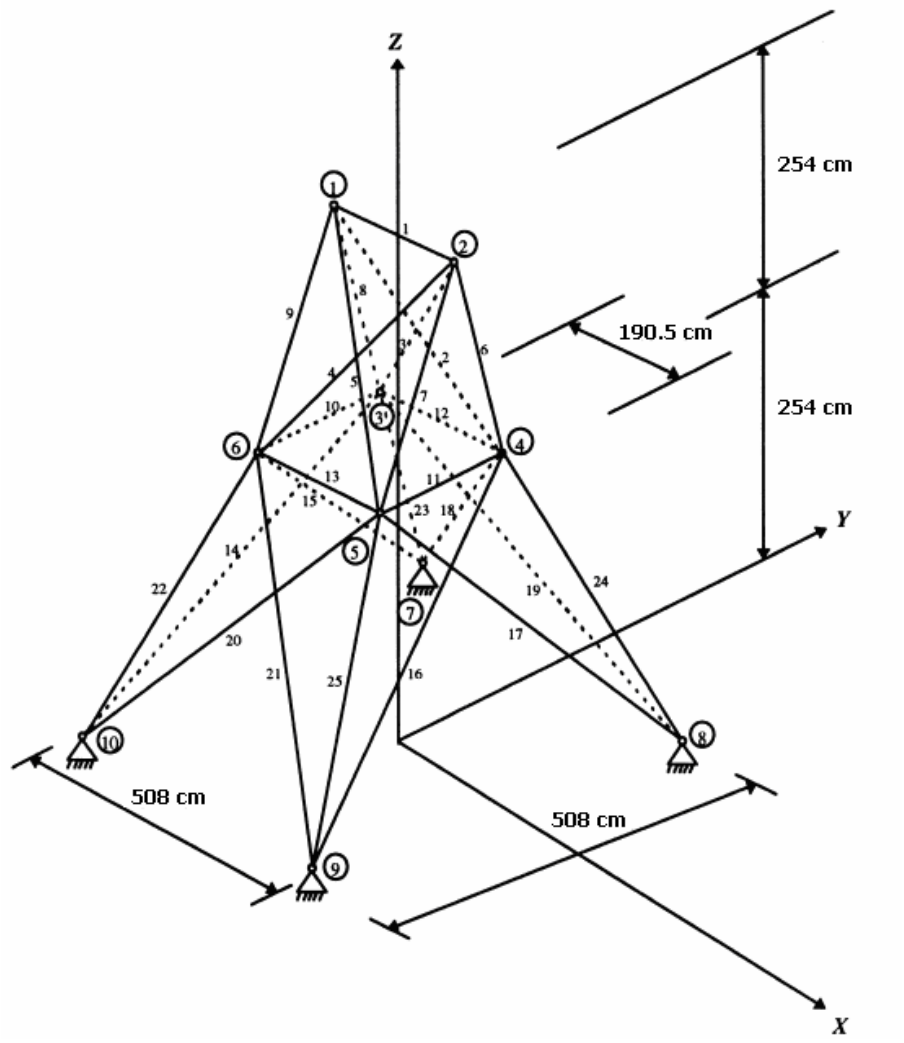


Figure – 5.21 25-bar space truss

Design parameters are:

Modulus of Elasticity :	68.95 GPa
Density :	2767.99 kg/m^3
Allowable tensile stress :	275.79 MPa
Allowable compressive stress :	275.79 MPa
Maximum (x, y) displacements in joints 1&2 :	0.89 cm

The optimization parameters used in the calculation are as follows:

Number of design variables : 8
Number of genes : 16
Initial population : 50
Selection type : Tournament Selection
Crossover type : One-point crossover
Crossover probability: 0.95
Mutation probability : 0.05
Elitist selection : 1

Loading inputs :

Table 5.2 – Loading inputs for 25-bar space truss

Case number	Node number	x(N)	y(N)	z(N)
1	**1**	4448	-44480	-44480
	2	0	-44480	-44480
	3	2224	0	0
	6	2668.8	0	0

The truss members are divided into eight groups, according to Table 5.3.

Table 5.3 – Group membership for 25-bar space truss

Group Number	Members
1	**A_1**
2	**A_2 - A_5**
3	**A_6 - A_9**
4	**A_{10} , A_{11}**
5	**A_{12} , A_{13}**
6	**A_{14} - A_{17}**
7	**A_{18} - A_{21}**
8	**A_{22} - A_{25}**

The convergence is obtained at generation number 221. The search space is 6^{16}. The convergence is obtained by only considering less than $\frac{221\times 50}{6^{16}}\cdot 100 < 0.01\%$ of the search space.

Table 5.4 – Comparison of results with Ref [12]

Size of group (cm^2)	***Ref[12]***	***Present***
G_1	0.65	0.65
G_2	11.61	3.23
G_3	14.84	21.94
G_4	1.29	0.65
G_5	0.65	12.26
G_6	5.16	6.45
G_7	11.61	2.58
G_8	19.35	21.94
Total Weight (N)	**2428.77**	**2157.61**

The results obtained give less weight than Ref [12] results, approximately 11.2%. If the design variables upper limit is extended from 21.94 cm^2 to 23.23 cm^2, the results obtained give less weight than Ref [12] results, approximately 13.3% listed in the table 5.5.

Table 5.5 – Comparison of results with Ref [12] (2)

Size of group (cm^2)	***Ref [12]***	***Present***
G_1	0.65	0.65
G_2	11.61	1.94
G_3	14.84	23.23
G_4	1.29	0.65
G_5	0.65	12.90
G_6	5.16	5.81
G_7	11.61	1.29
G_8	19.35	23.23
Total Weight (N)	**2428.77**	**2106.63**

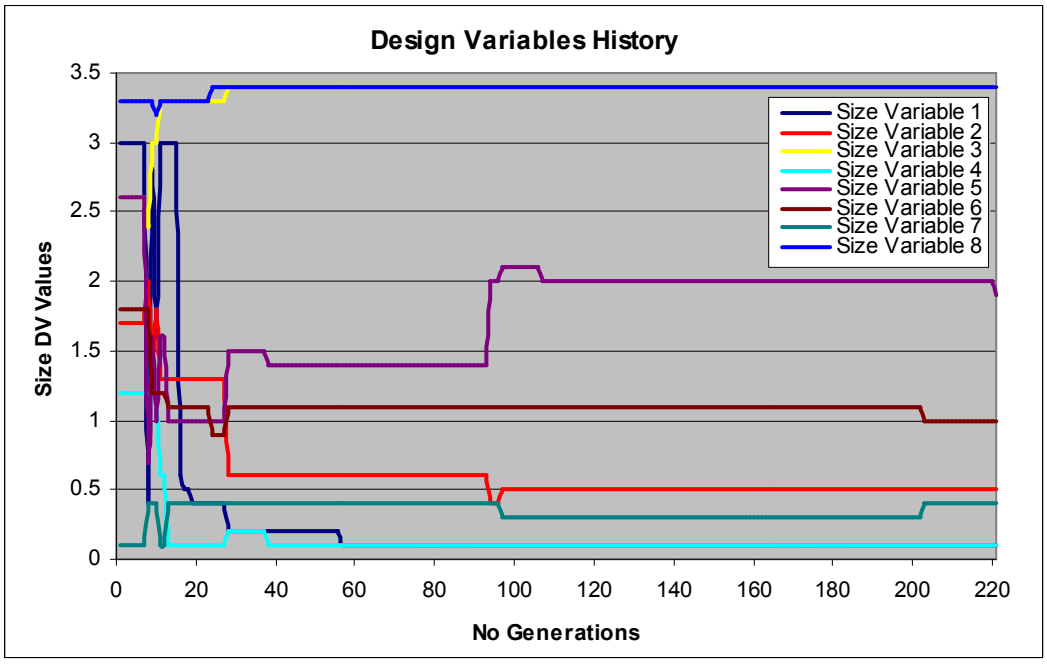


Figure – 5.22 Design Variables History for 25-bar space truss

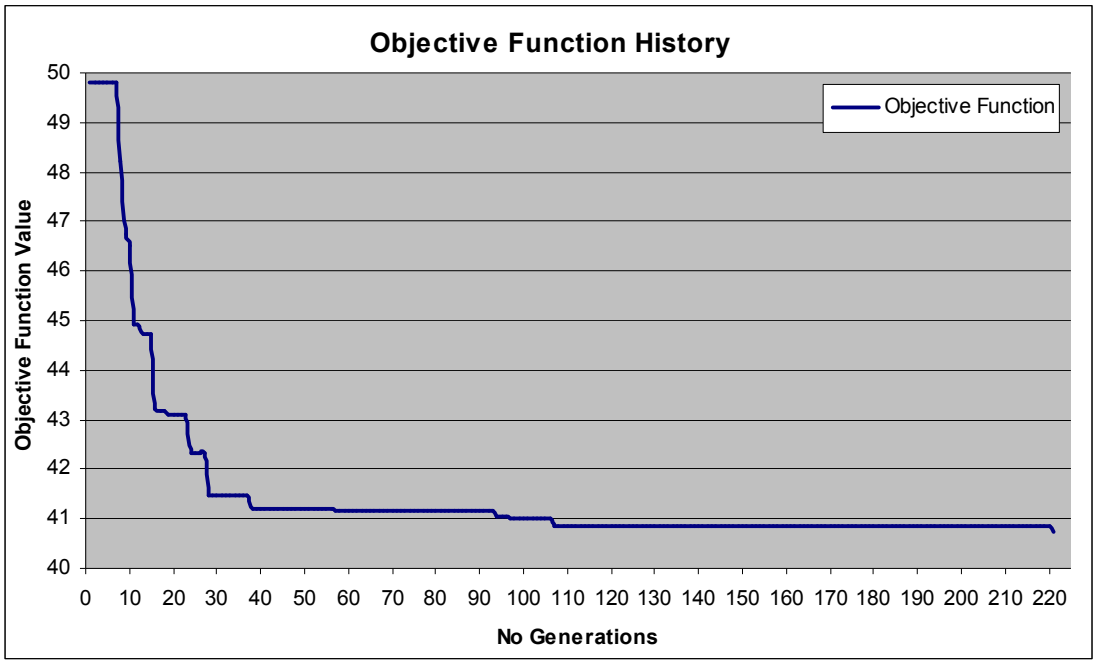


Figure – 5.23 Objective Function History for 25-bar space truss

5.2.3 72-bar space truss

The 72-bar space truss structure, shown in Figure 5.24, is a relatively large size problem. Weight minimization is the objective while the cross sectional areas of the members are design variables. The number of the design variables is 72 and the number of constraints is found to be 152. However, by linking the design variables into 16 groups, the number of design variables becomes 16 and the associated number of the constraints reduces to 40. The structure is subjected to two different loading inputs.

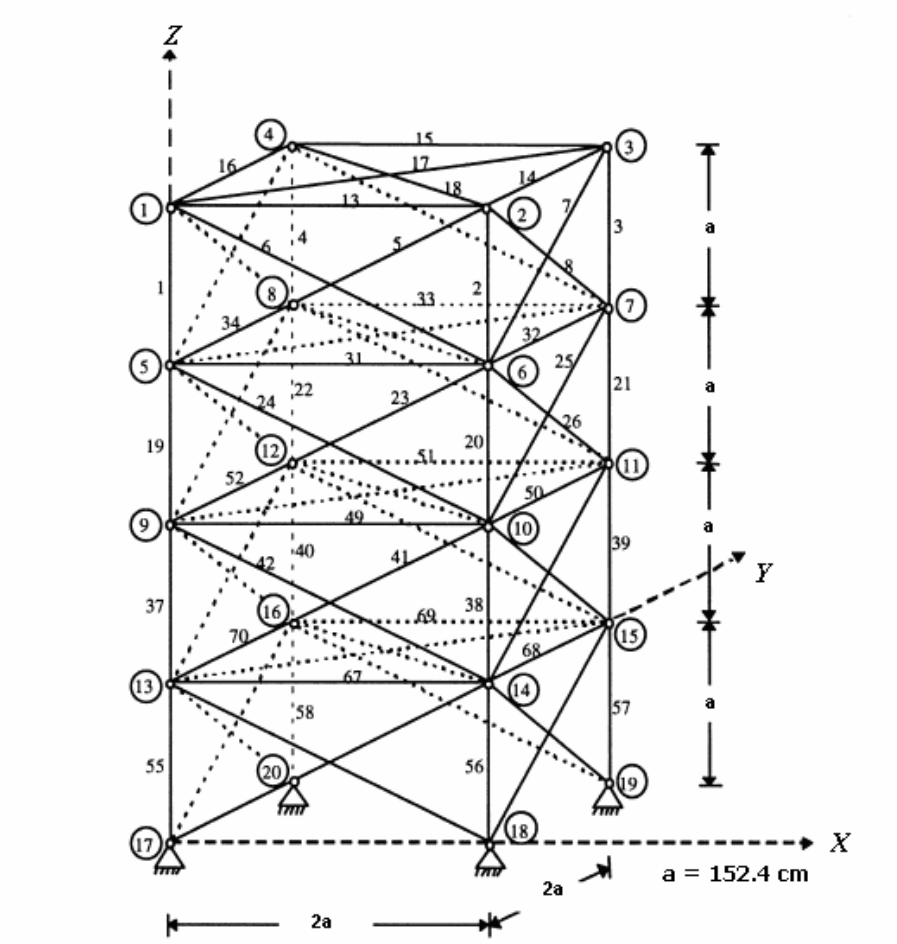


Figure – 5.24 72-bar space truss

Design parameters are:

Modulus of Elasticity : 68.95 GPa

Density : 2767.99 kg/m^3

Allowable tensile stress : 172.37 MPa

Allowable compressive stress : 172.37 MPa

Maximum (x, y) displacements in nodes 1&4 : 0.64 cm

Loading inputs :

Table 5.6 – Loading inputs for 72-bar space truss

Case number	Node number	x(N)	y(N)	z(N)
1	**1**	22240	22240	-22240
2	**1**	0	0	-22240
	2	0	0	-22240
	3	0	0	-22240
	4	0	0	-22240

The optimization parameters used in the calculation are as follows:

Number of design variables : 16

Number of genes : 16

Initial population : 80

Selection type : Tournament Selection

Crossover type : One-point crossover

Crossover probability: 0.95

Mutation probability : 0.07

Elitist selection : 1

The truss members are divided into eight groups, according to Table 5.7.

Table 5.7 – Group membership for 72-bar space truss

Group Number	Members
1	A_1 - A_4
2	A_5 - A_{12}
3	A_{13} - A_{16}
4	A_{17} , A_{18}
5	A_{19} - A_{22}
6	A_{23} - A_{30}
7	A_{31} - A_{34}
8	A_{35} , A_{36}
9	A_{37} - A_{40}
10	A_{41} - A_{48}
11	A_{49} - A_{52}
12	A_{53} , A_{54}
13	A_{55} - A_{58}
14	A_{59} - A_{66}
15	A_{67} - A_{70}
16	A_{71} , A_{72}

The convergence is obtained at generation number 269. The search space is 25^{16}. The convergence is obtained by only considering less than $\frac{269 \times 80}{25^{16}} \cdot 100 < 0.01\%$ of the search space.

Table 5.8 – Comparison of results with Ref [13]

Size of group (cm^2)	***Ref [13]***	***Present***
G_1	1.01	0.99
G_2	3.46	3.55
G_3	2.65	2.65
G_4	3.68	3.42
G_5	3.28	3.42
G_6	3.37	3.29
G_7	0.65	0.65
G_8	0.65	1.00
G_9	8.30	8.13
G_{10}	3.33	3.35
G_{11}	0.65	0.65
G_{12}	0.65	0.65
G_{13}	12.29	12.19
G_{14}	3.34	3.29
G_{15}	0.65	0.65
G_{16}	0.65	0.65
Total Weight (N)	**1694.061**	**1692.37**

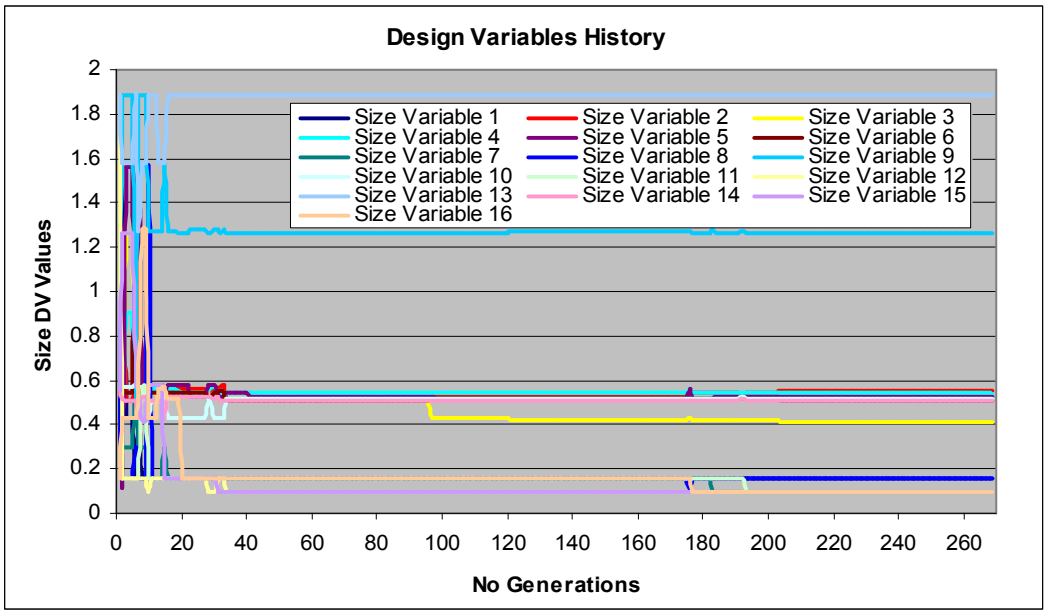


Figure – 5.25 Design Variables History for 72-bar space truss

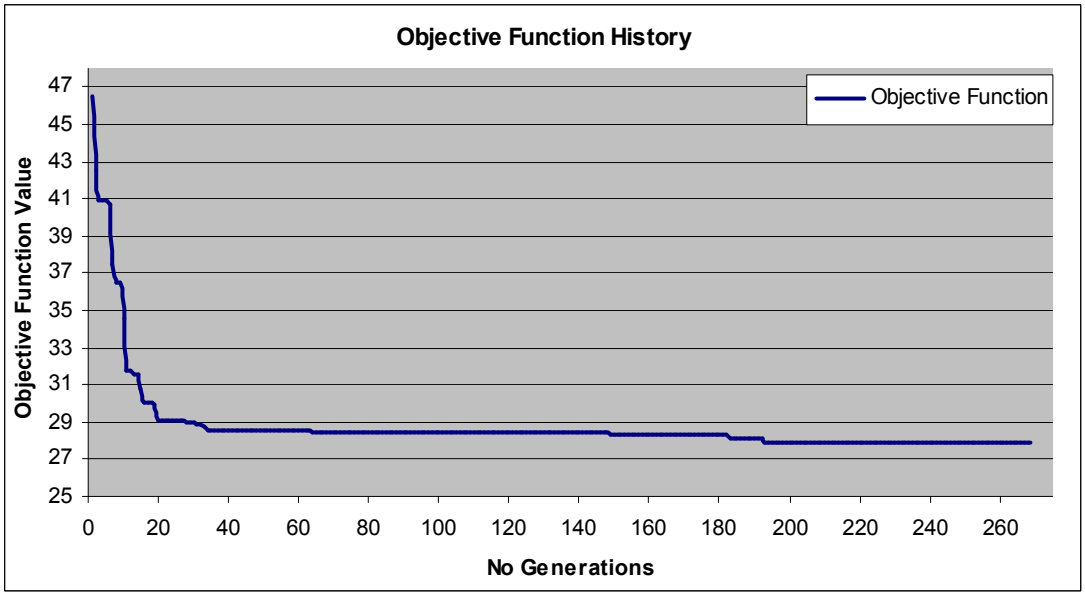


Figure – 5.26 Objective Function History for 72-bar space truss

5.2.4 18-bar truss

In the 18 bar truss structure, the design variables are the cross sectional areas of the truss elements and the Y components of the nodes 3, 5, 7 and 9. Thus, this problem is a size and shape optimization problem.

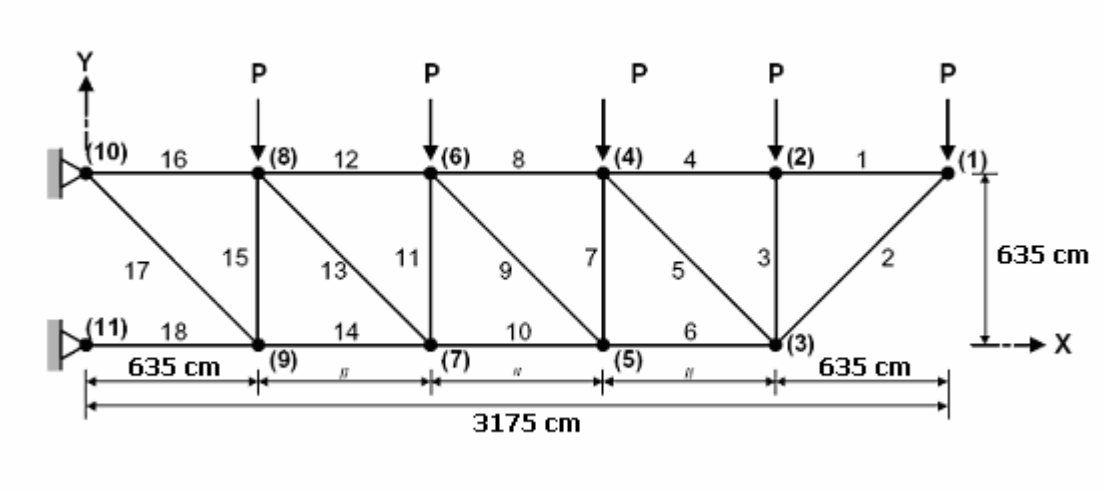


Figure – 5.27 18-bar space truss

Design parameters are:

Modulus of Elasticity :	68.95 GPa
Density :	2767.99 kg/m^3
Allowable tensile stress :	137.90 MPa
Allowable compressive stress :	137.90 MPa

Loading inputs :

P = 88964.43 N acting on the upper nodal points of the truss, as illustrated in Figure 5.27

The optimization parameters used in the calculation are as follows:

Number of design variables : 8
Number of genes : 8
Initial population : 100
Selection type : Tournament Selection
Crossover type : Uniform crossover
Crossover probability: 0.95
Mutation probability : 0.08
Elitist selection : 1

The groups for the sizing variables are listed in table

Table 5.9 – Group membership for 18-bar truss

Group Number	Members
1	**A_1, A_4, A_8, A_{12}, A_{16}**
2	**A_2, A_6, A_{10}, A_{14}, A_{18}**
3	**A_3, A_7, A_{11}, A_{15}**
4	**A_5, A_9, A_{13}, A_{17}**

The shape variables are the Y components of the nodes 3, 5, 7 and 9.
The convergence is obtained at generation number 93. The search space is 15^8. The convergence is obtained by only considering less than $\frac{93 \times 100}{15^8} \cdot 100 < 0.01\%$ of the search space.

Table 5.10 – Comparison of results with Ref [14]

(cm^2) or (cm)	***Ref [14]***	***Present***
G_1	83.87	83.87
G_2	100.00	100.00
G_3	9.68	9.68
G_4	19.35	19.35
Y_3	584.2	584.2
Y_5	482.6	482.6
Y_7	342.9	342.9
Y_9	152.4	152.4
Total Weight (N)	**1776.18**	**1776.18**

The results obtained are exactly the same with Ref [14] results.

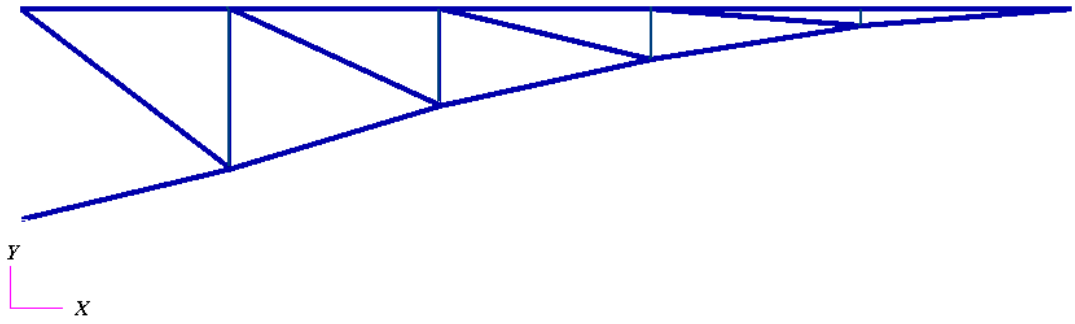


Figure – 5.28 Final shape of the 18-bar space truss

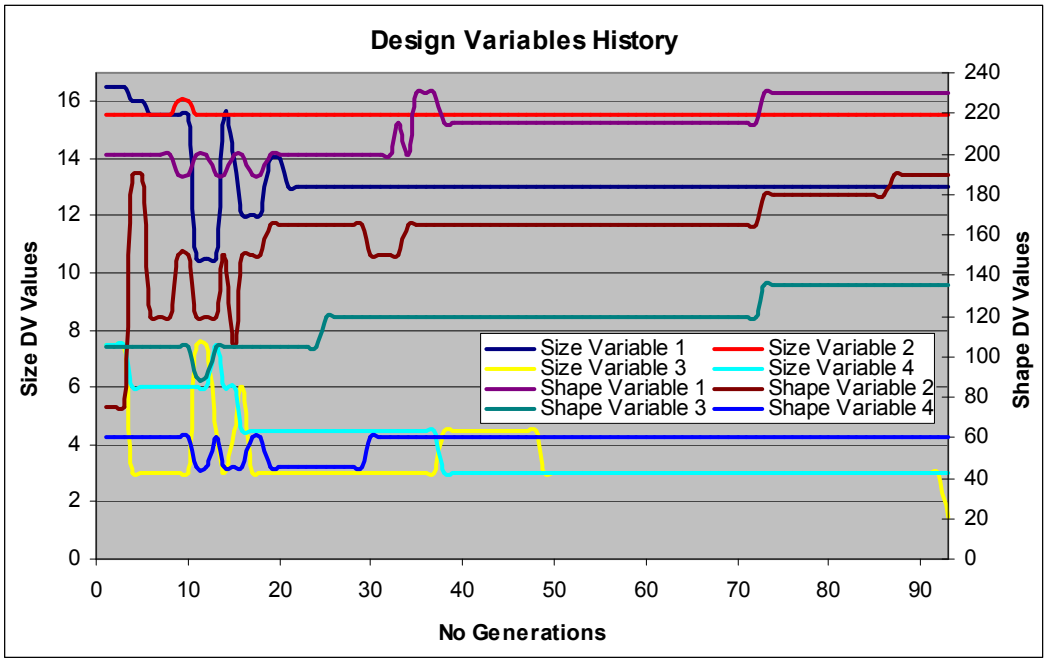


Figure – 5.29 Design Variables History for 18-bar truss

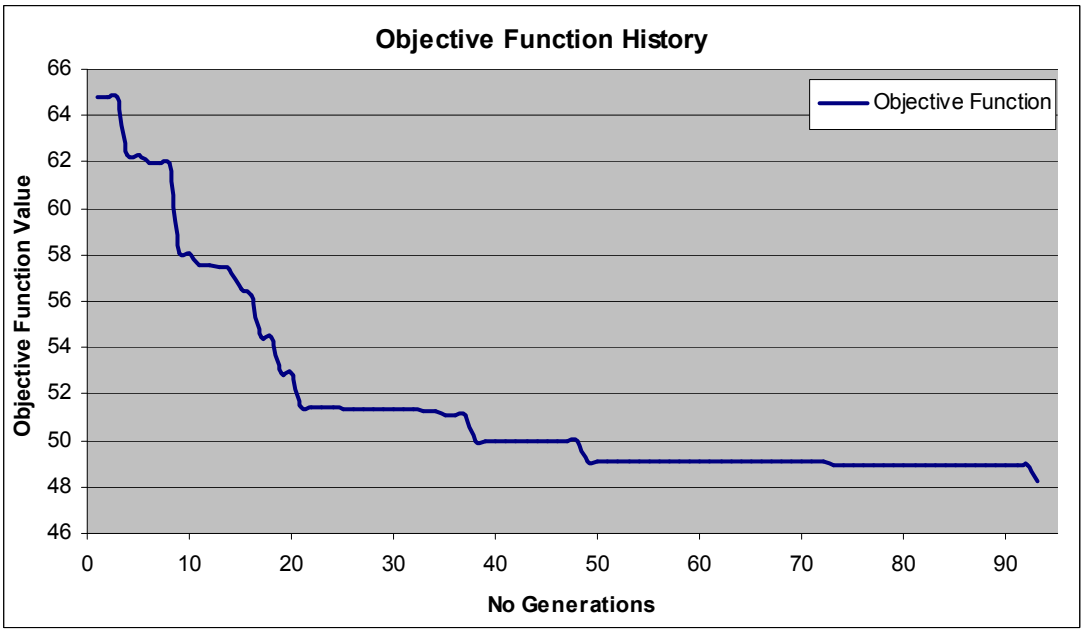


Figure – 5.30 Objective Function History for 18-bar truss

5.2.5 Simply supported bridge shape optimization

In this problem, the objective is to minimize the weight of the simply supported bridge structure by considering the vertical coordinates of the upper nodes as design variables.

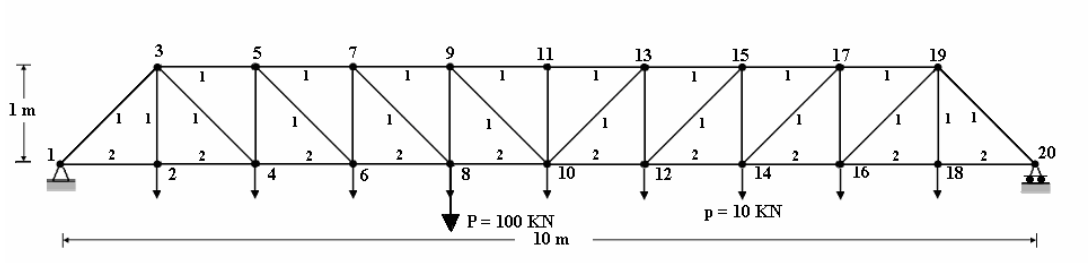


Figure – 5.31 Simply supported bridge structure

Design parameters are:

Modulus of Elasticity : 2.1 10^7 Pa

Density : 7800 kg/m^3

Poisson's ratio : 0.3

Maximum y displacements in nodes 8&10 : 1 cm

Rod elements (marked by 1 on the figure) with cross sectional area of 5 cm^2

Beam elements (marked by 2 on the figure) with rectangular cross section having dimensions b = 8 cm and h = 5 cm

Loading inputs :

There are two load cases considered in this problem.

Table 5.11 – Loading inputs for Bridge structure

Case number	Node number	x(N)	y(N)	z(N)
1	**2,4,6,8,1012,14,16,18**	**0**	**-10000**	**0**
2	**8**	**0**	**-100000**	**0**

The optimization parameters used in the calculation are as follows:

Number of design variables : 5
Number of genes : 15
Initial population : 100
Selection type : Tournament Selection
Crossover type : Uniform crossover
Crossover probability: 0.95
Mutation probability : 0.08
Elitist selection : 1

Both of the load cases are considered in the optimization process. Since the loading (in case 1) and geometry of the bridge structure are symmetric, the number of design variables is taken as 5. During the optimization process, the constraints are evaluated by both considering loadcase-1 and loadcase-2.

The convergence is obtained at generation number 67. The search space is 10^{15} . The convergence is obtained by only considering less than $\frac{67 \times 100}{10^{15}} \cdot 100 < 0.01\%$ of the search space.

Table 5.12 – Comparison of results with Ref [15]

(m)	***Ref [15]***	***Present***
Y_{11}	2.544	2.530
Y_{13} (Y_9)	2.449	2.400
Y_{15} (Y_7)	2.172	2.070
Y_{17} (Y_5)	1.646	1.610
Y_{19} (Y_3)	1.042	1.000
Total Mass(kg)	**489.60**	**485.95**

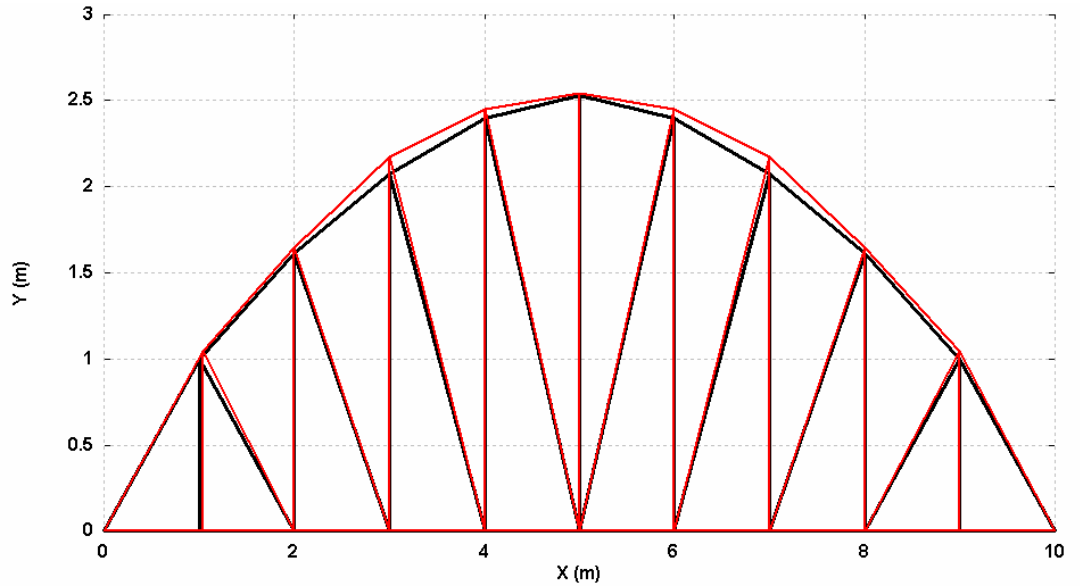


Figure – 5.32 Final shape of the bridge structure

The shape of the bridge structure in Ref [15] is shown in red.

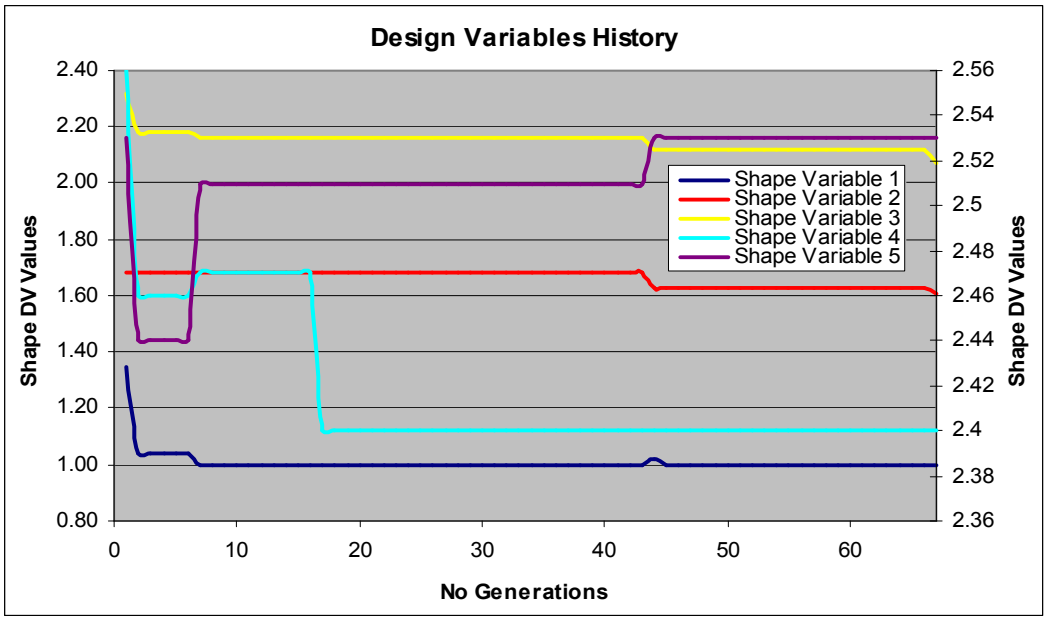


Figure – 5.33 Design Variables History for Bridge Structure

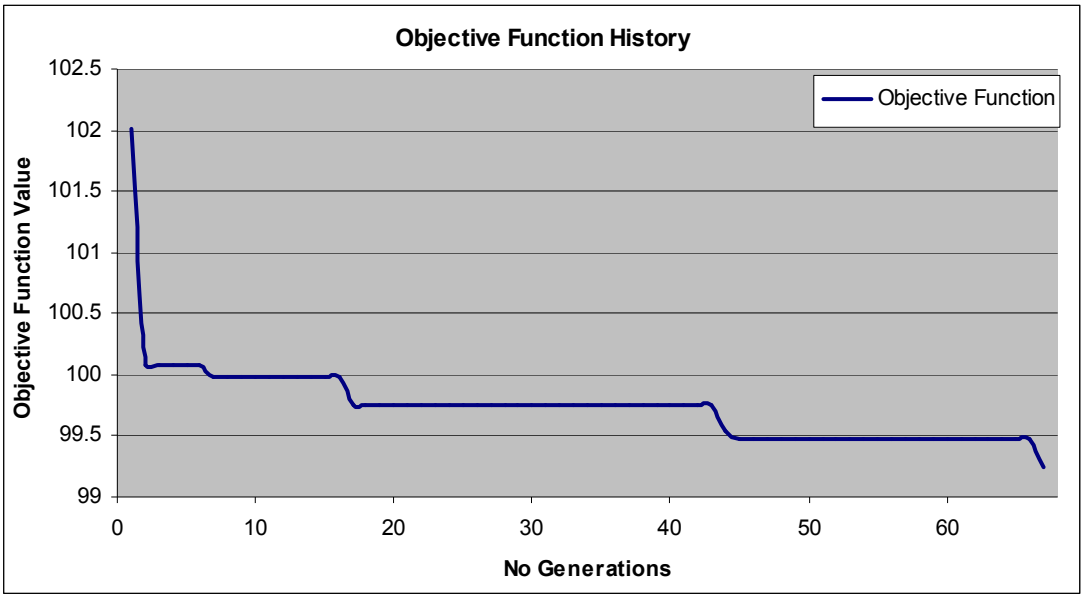


Figure – 5.34 Objective Function History for Bridge Structure

5.2.6 Stepped Cantilever Beam

In the stepped cantilever beam, the height and width of the beam, in all five steps of the cantilever beam, are taken as design variables. The volume of the beam is to be minimized, subject to the bending stress constraints in all five steps of the beam, to be less than allowable stress, the displacement constraint on the tip deflection is to be less than the allowable deflection and a specified aspect ratio has to be maintained between the height and width of beam cross sections.

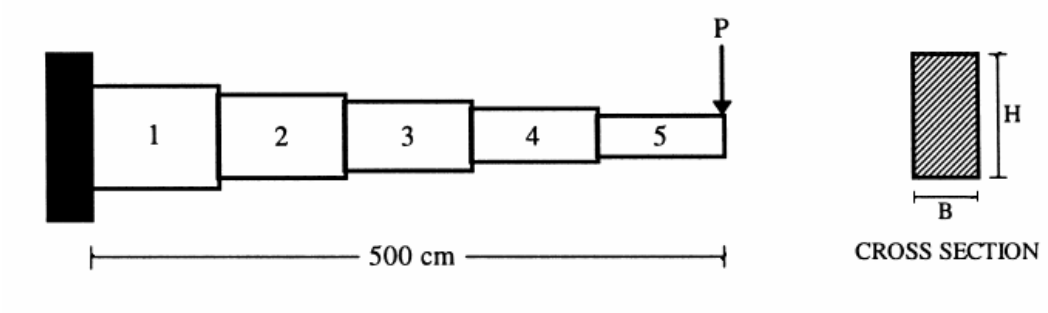


Figure – 5.35 Stepped Cantilever Beam

Design parameters are:

Modulus of Elasticity :	200 GPa
Allowable tensile stress :	140 MPa
Allowable compressive stress :	140 MPa
Maximum displacement at the tip :	2.7 cm
Aspect ratio H/B :	<20

Loading inputs :

P = 50 KN

The optimization parameters used in the calculation are as follows:

Number of design variables : 10
Number of genes : 10
Initial population : 40
Selection type : Tournament Selection
Crossover type : One-point crossover
Crossover probability: 1.00
Mutation probability : 0.06
Elitist selection : 1

The convergence is obtained at generation number 42. The search space is 10^{10}. The convergence is obtained by only considering less than $\frac{42 \times 100}{10^{10}} \cdot 100 < 0.01\%$ of the search space.

Table 5.13 – Comparison of results with Ref [16]

(cm)	***Ref [16]***	***Present***
B_1	3	3.2
B_2	3	2.9
B_3	3	2.6
B_4	3	2.3
B_5	2	2
H_1	60	60
H_2	57	56
H_3	49	52
H_4	38	44
H_5	33	36
Total Volume (cm^3)	**67.80**	**66.28**

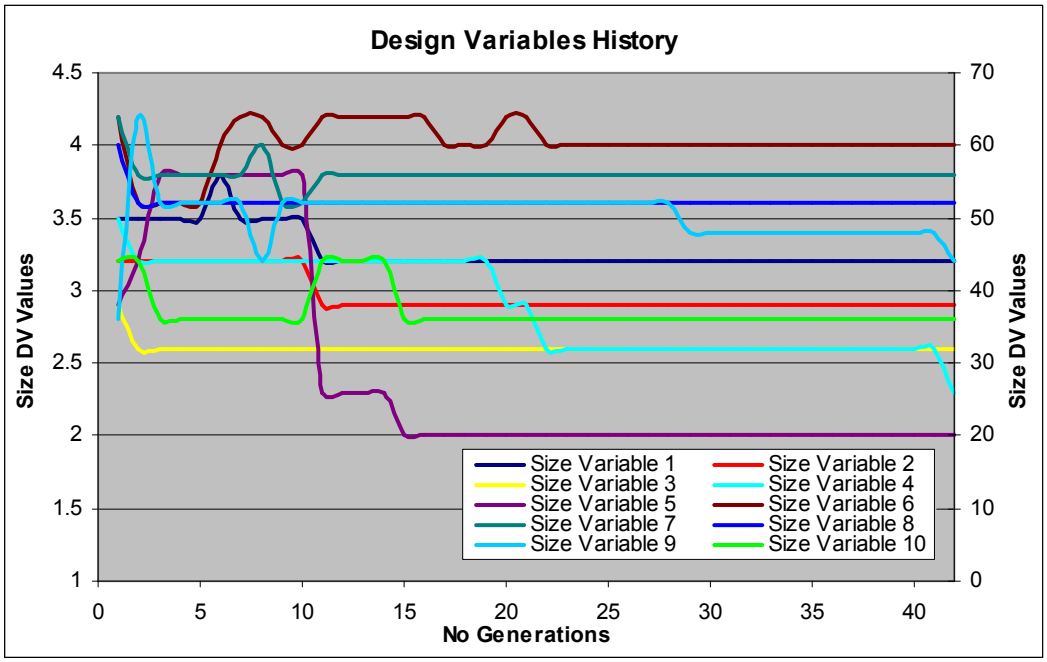


Figure – 5.36 Design Variables History for Cantilever Beam

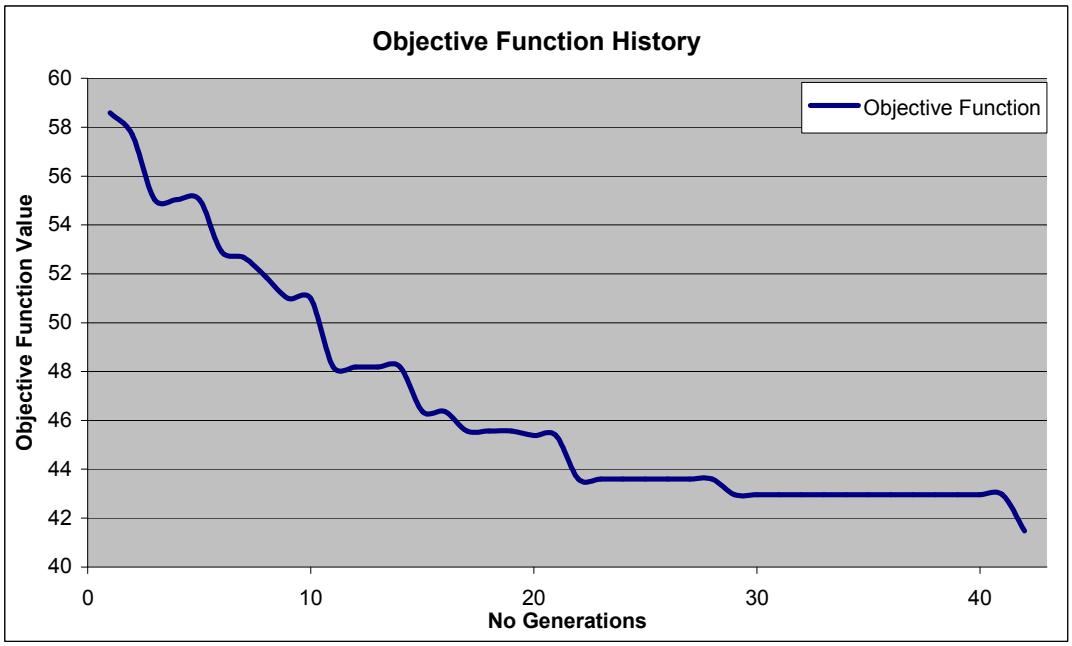


Figure – 5.37 Objective Function History for Cantilever Beam

5.2.7 Portal Frame

The volume of a three-bar portal frame is minimized in this problem. There are stress and displacement constraints in the problem. The design variables are the geometric dimensions of the cross sections of the three bars.

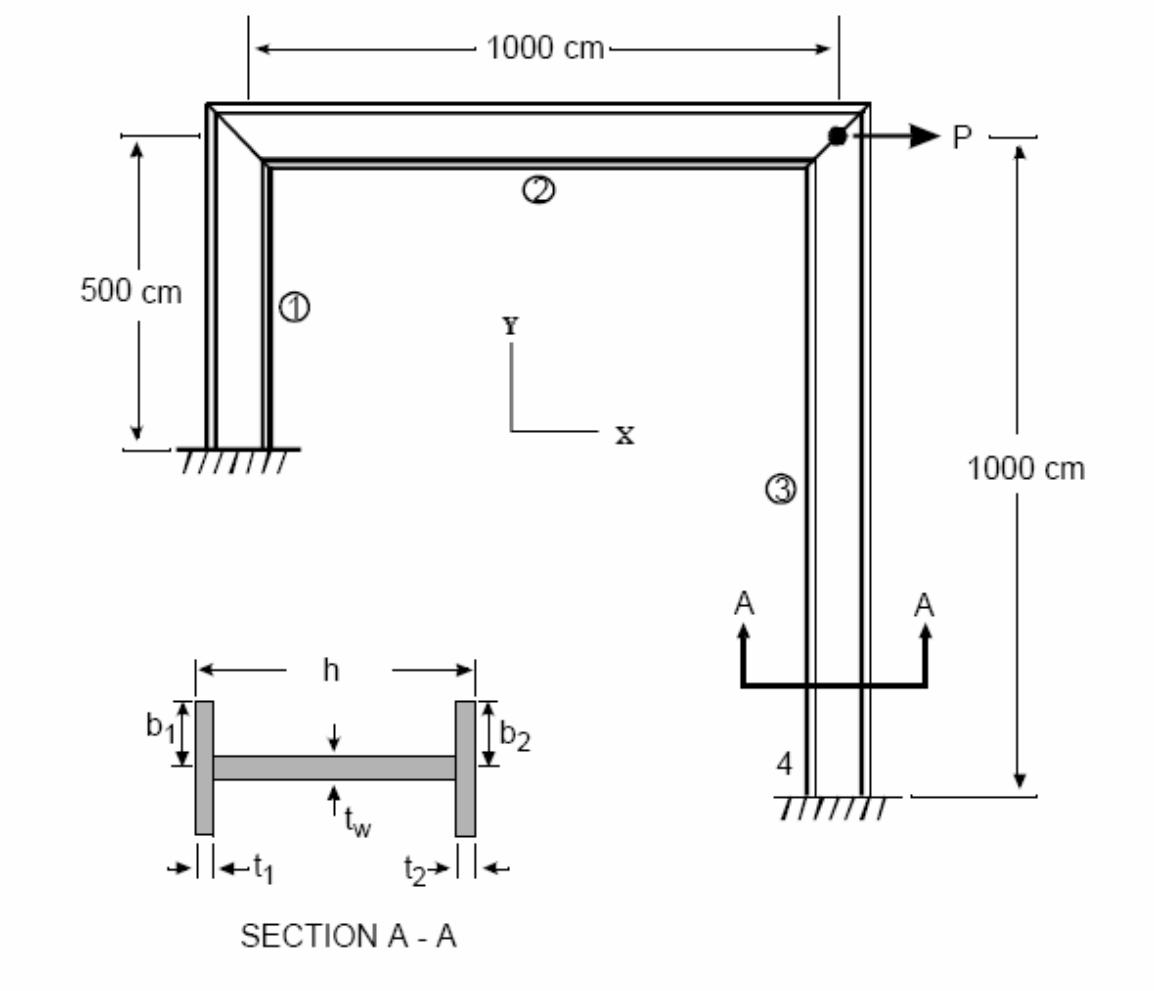


Figure – 5.38 Portal Frame

Design parameters are:

Modulus of Elasticity : 7.06 MPa

Allowable Axial stress : 20,000 Pa

Maximum displacement x direction : 4.8 cm

Loading inputs :

P = 50 KN

The optimization parameters used in the calculation are as follows:

Number of design variables : 18

Number of genes : 18

Initial population : 80

Selection type : Tournament Selection

Crossover type : One-point crossover

Crossover probability: 0.95

Mutation probability : 0.07

Elitist selection : 1

The convergence is obtained at generation number 112. The search space is 10^{18}. The convergence is obtained by only considering less than $\frac{112 \times 80}{10^{18}} \cdot 100 < 0.01\%$ of the search space.

Table 5.14 – Comparison of results with GENESIS results

Bar No	***(cm)***	***GENESIS***	***Present***
1	Height	100.885	100
1	b1	6.86	6.3334
1	b2	6.9	7
1	tw	0.1	0.2
1	t1	0.438	0.46664

1	t2	0.447	0.43331
2	Height	100.616	100
2	b1	6.08	5.2224
2	b2	6.264	5.4446
2	tw	0.1	0.1
2	t1	0.294	0.23333
2	t2	0.322	0.2
3	Height	96.5679	80.00002
3	b1	5.56	5.2224
3	b2	6.05	5.2224
3	tw	0.1	0.1
3	t1	0.2645	0.2
3	t2	0.3034	0.2
	Total Volume (cm^3)	**45073.1**	**42781.4**

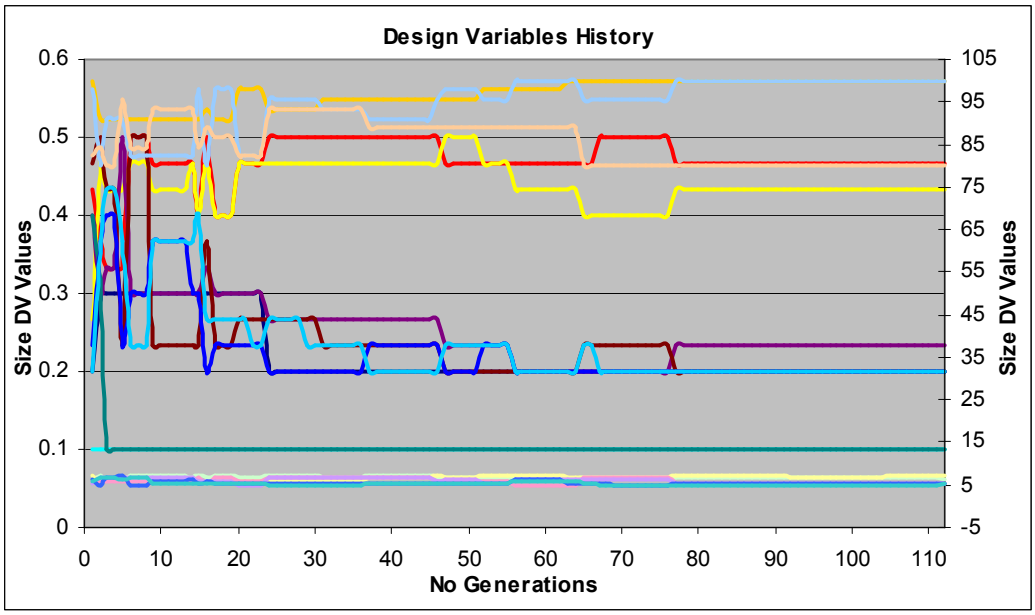


Figure – 5.39 Design Variables History for Portal Frame

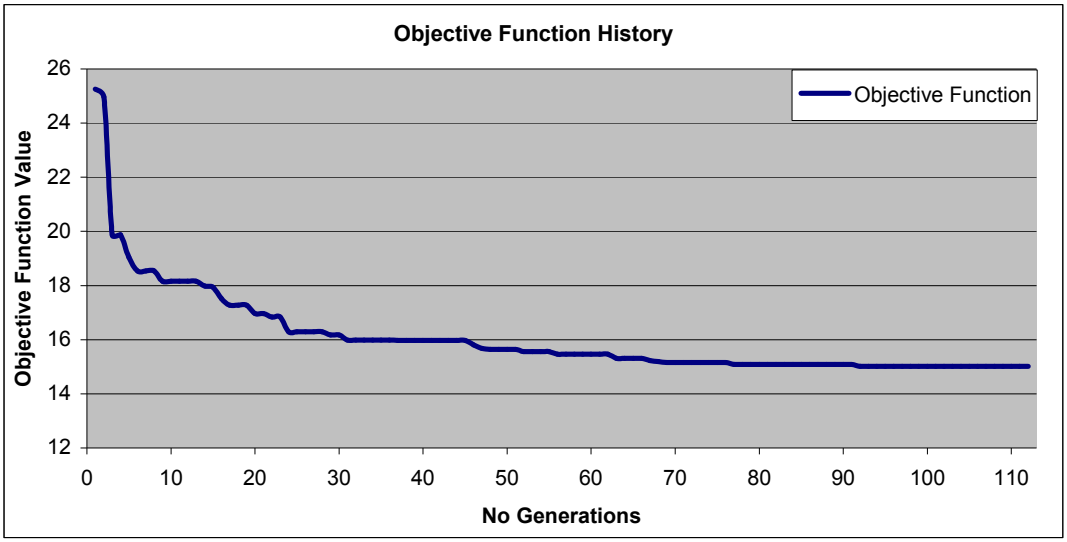


Figure – 5.40 Objective Function History

5.2.8 Bridge size, shape and topology optimization

This example involves a ground structure that includes the cost of placing a pier in the middle of the span of a bridge. The objective is to minimize the weight of bridge structure by considering size, shape and topology variables.

The ground structure for this example is shown in figure below.

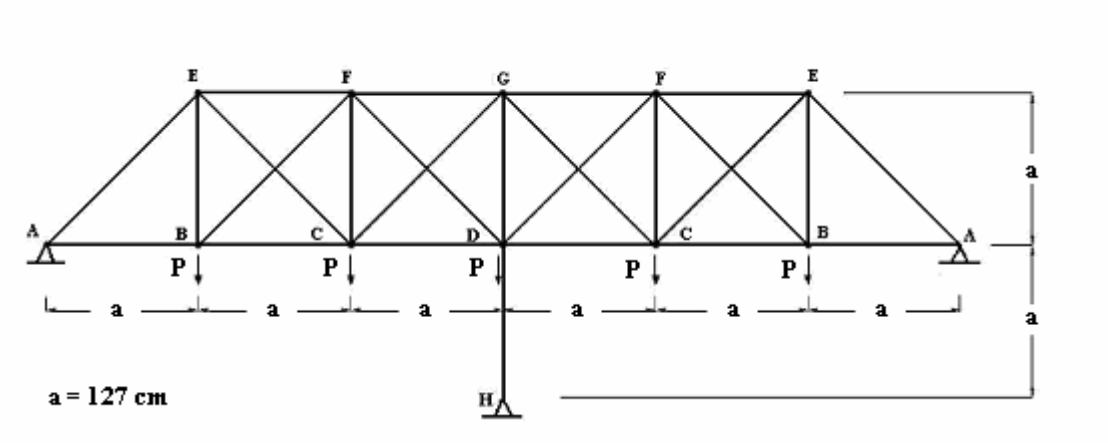


Figure – 5.41 Bridge ground structure

Because the structure is considered symmetric, joints A, B, C, E, and F are the same on both sides of member DG. Member DH simulates the cost of placing a pier in the middle of the span.

Design parameters are:

Modulus of Elasticity :	199.95 GPa
Weight Density :	7750.37 kg/m^3
Maximum x and y displacements :	25.40 cm
Normal allowable stress :	413.69 Mpa

The point loads at joints B, C, and D are 22240 N downward. This load is the dead load of the roadway as well as the live loads the bridge will need to carry.

The two shape variables will be linked to joints E and F. Their vertical coordinates are allowed to vary with a minimum value of 0.24 cm and a maximum value of 120 cm.

The sizes available for this example are the AISC Standard Weight Steel Pipes. Below table summarizes their properties.

Table 5.15 – Available Discrete Sizes for AISC Standard Weight Steel Pipe

Diameter	Area(cm^2)	Weight(N/cm)
½	1.61	0.12
¾	2.15	0.16
1	3.19	0.25
1 ¼	4.32	0.33
1 ½	5.15	0.40
2	6.90	0.53
2 ½	10.97	0.85
3	14.39	1.11
3 ½	17.29	1.33
4	20.45	1.58
5	27.74	2.13
6	36.00	2.77
8	54.19	4.17
10	76.77	5.91
12	94.19	7.24

This example uses size variable linking. Table 5.16 indicates which members are linked to the 4 size variables. Each member is linked to its own topology variable except members AB, BC, and CD. These members are linked to the same topology variable.

Table 5.16 – Group membership for Bridge structure

Group Number	Members
1	**AB, BC, CD**
2	**AE, EF, FG**
3	**BE, CF, BF, CE, CG, DF**
4	**DG, DH**

The optimization parameters used in the calculation are as follows:

Number of design variables : 11
Number of genes : 11
Initial population : 120
Selection type : Tournament Selection
Crossover type : Uniform crossover
Crossover probability: 0.95
Mutation probability : 0.08
Elitist selection : 1

The convergence is obtained at generation number 29. The search space is $16^6 \cdot 2^5$. The convergence is obtained by only considering less than $\frac{29 \times 120}{16^6 \cdot 2^5} \cdot 100 < 0.23\,\%$ of the search space.

Table 5.17 – Comparison of results with Ref [17]

cm/ cm²	***Ref [17]***	***Present***
Y_E	0.30988	43.18
Y_F	42.60596	1.27
A_1	-	-
A_2	2.15	1.61
A_3	1.61	1.61
A_4	5.15	2.15
Total Mass(N)	**247.77**	**227.08**

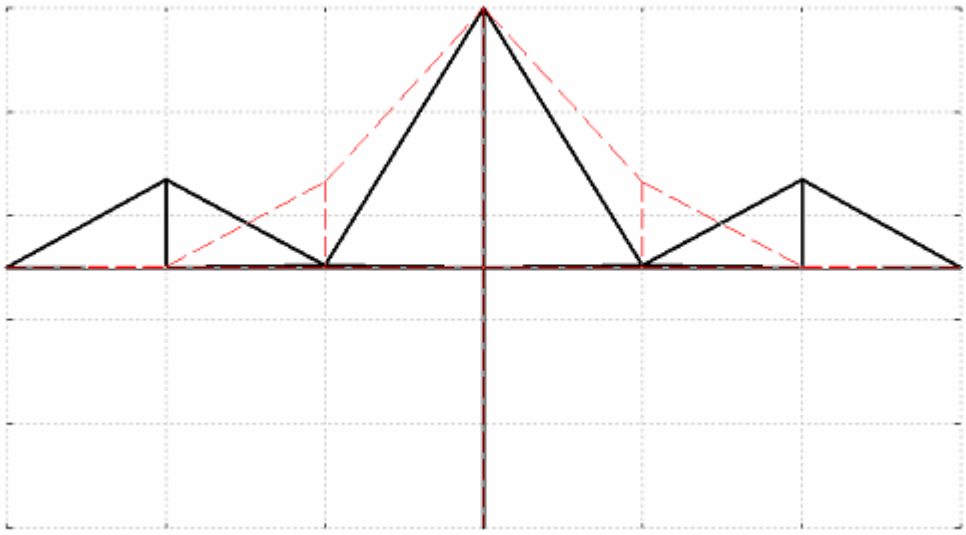


Figure – 5.42 Final shape of the bridge structure

The shape of bridge structure in Ref [17] is shown with dash-lines in the above figure.

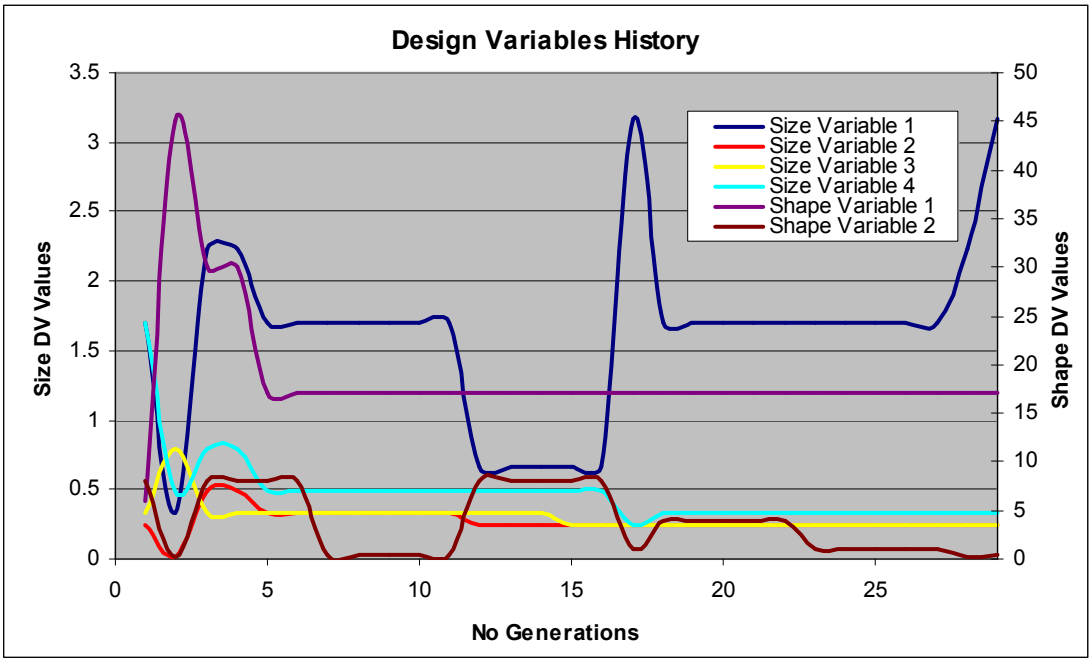


Figure – 5.43 Design Variables History for Bridge Structure-1

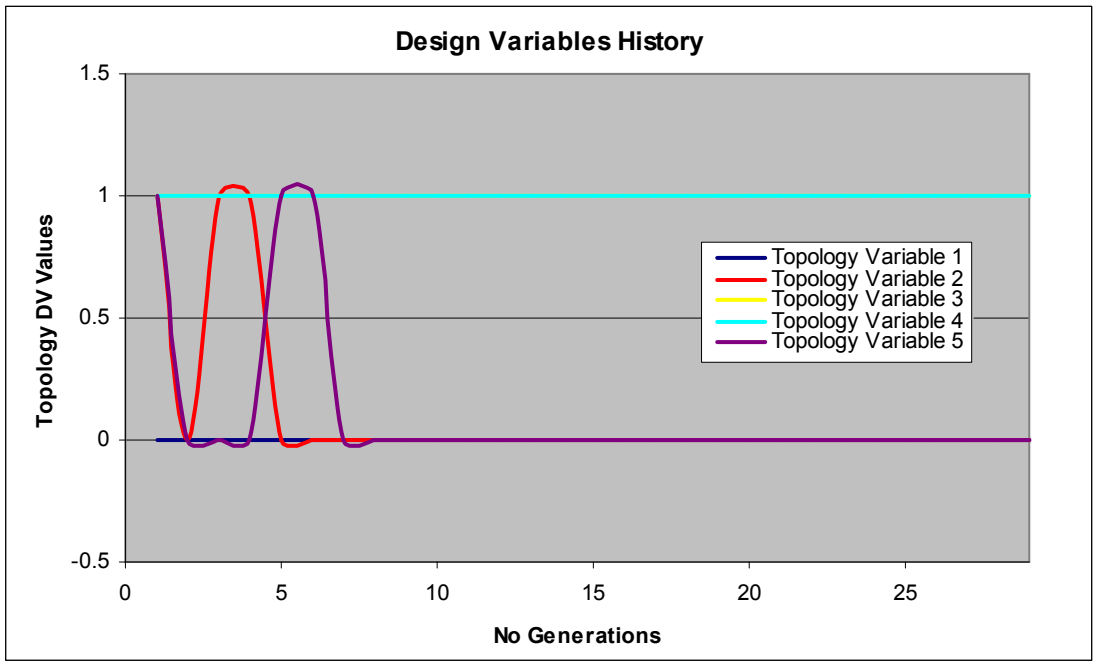


Figure – 5.44 Design Variables History for Bridge Structure-2

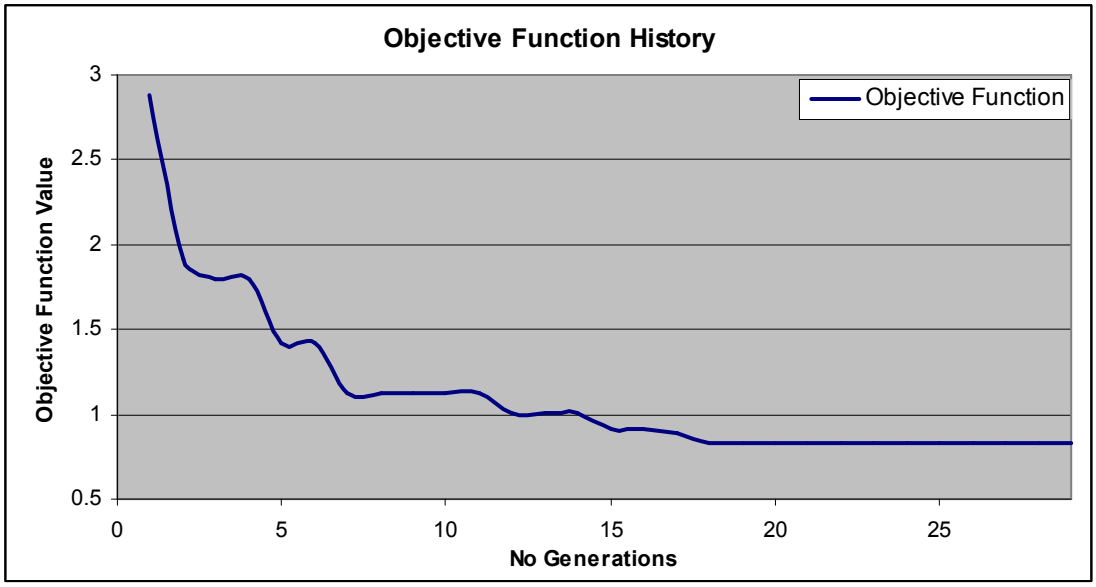


Figure – 5.45 Objective Function History for Bridge Structure

5.2.9 Rajan's Truss

This example involves shape, size and topology design variables. Rajan optimized a 14-node truss with shape, size and topology as the design variables [18].

The ground structure for this example is shown in figure below.

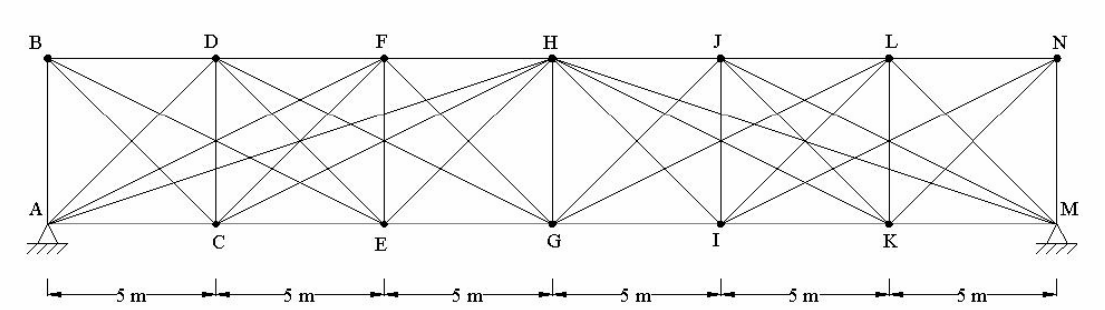


Figure – 5.46 Rajan's truss ground structure

Design parameters are:

Modulus of Elasticity :	210 GPa
Maximum y displacement :	1 cm
Allowable tensile stress :	104 Mpa
Allowable compressive stress :	130 Mpa

Loading conditions are shown in the table below.

Table 5.18 – Loads for Rajan's Truss Example

Load case	Joint no	Direction	Load (N)
1	4	y	-3000000
2	6	y	-3000000
3	8	y	-3000000
4	10	y	-3000000
5	12	y	-3000000

There are 4 shape variables. Vertical coordinates for the top joints (B, D, F, H, J, L, and N) can vary between 2 m and 8 m. The structure is symmetric about middle member GH. Therefore joints B and N will have the same vertical coordinate, joints D and L will have the same vertical coordinate, and joints F and J will have the same vertical coordinate. Joint H is allowed to vary as well.

The available cross-sectional areas are listed in the table below.

Table 5.19 – Available Discrete Areas

No	Area(m^2)
1	0.01
2	0.016
3	0.022
4	0.028
5	0.034
6	0.04
7	0.046
8	0.052
9	0.058
10	0.064
11	0.07
12	0.076
13	0.082
14	0.088
15	0.094
16	0.1

This example uses size variable linking. Table 5.20 indicates which members are linked to the 4 size variables.

Table 5.20 – Group membership for Rajan's truss structure

Group Number	Members
1	**BD, DF, FH, HJ, JL, LN**
2	**AC, CE, EG, GI, IK, KM**
3	**BC, AD, DE, CF, FG, EH, HI, GJ, JK, IL, LM, KN**
4	**AB, CD, EF, GH, IJ, KL, MN, AF, CH, BE, DG, AH, GL, IN, JM, HK, HM**

The optimization parameters used in the calculation are as follows:

Number of design variables : 19
Number of genes : 19
Initial population : 100
Selection type : Tournament Selection
Crossover type : Single-point crossover
Crossover probability: 0.95
Mutation probability : 0.08
Elitist selection : 1

The convergence is obtained at generation number 213. The search space is $16^8 \cdot 2^{11}$. The convergence is obtained by only considering less than $\frac{213 \times 100}{16^8 \cdot 2^{11}} \cdot 100 < 0.01\%$ of the search space.

Table 5.21 – Comparison of results with Ref [18]

m/ m²	***Ref [18]***	***Present***
Y_B	2.33	2
Y_D	4.31	4.8
Y_F	6.68	7.2
Y_H	7.66	8
A_1	0.053	0.04
A_2	0.016	0.022
A_3	0.022	0.016
A_4	0.034	0.010
Total Volume (m^3)	**5.1**	**4.42**

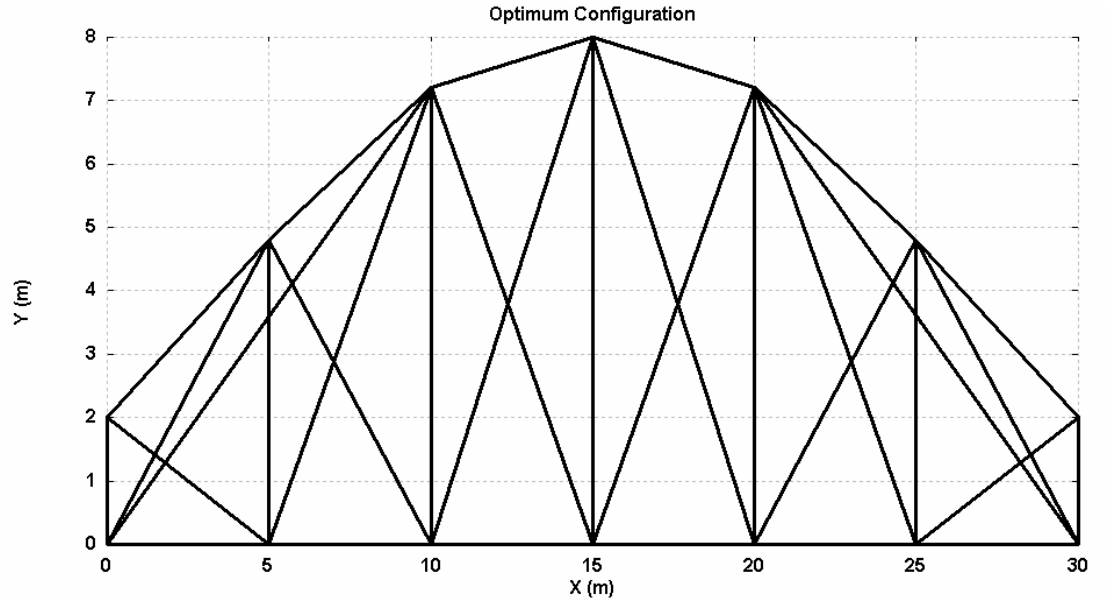


Figure – 5.47 Final shape of the bridge structure

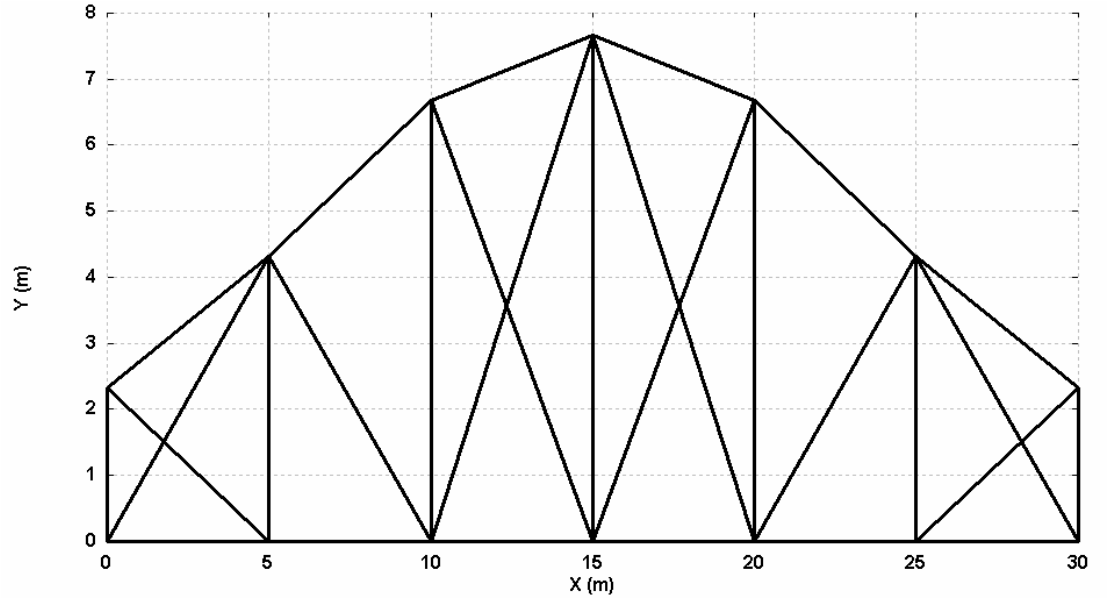


Figure – 5.48 Shape of the bridge structure in Ref [18]

To investigate the stability of the algorithm, 10 consecutive runs have been performed using the same optimization parameters. The results obtained are listed in the table below.

Table 5.22 – Results obtained at consecutive runs

Run	Volume(m^3)
1	4.42
2	4.42
3	4.47
4	4.44
5	4.43
6	4.42
7	4.42
8	4.53
9	4.42
10	4.53

The topologies obtained in these runs are shown below.

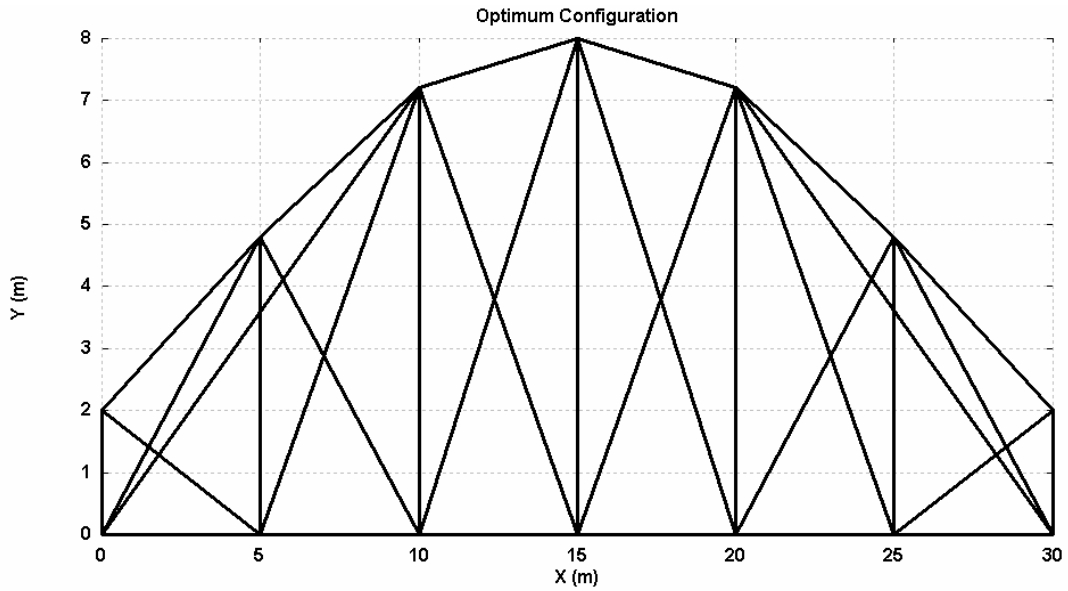


Figure – 5.49 Shape of the bridge structure for runs 1, 2, 6, 7 and 9

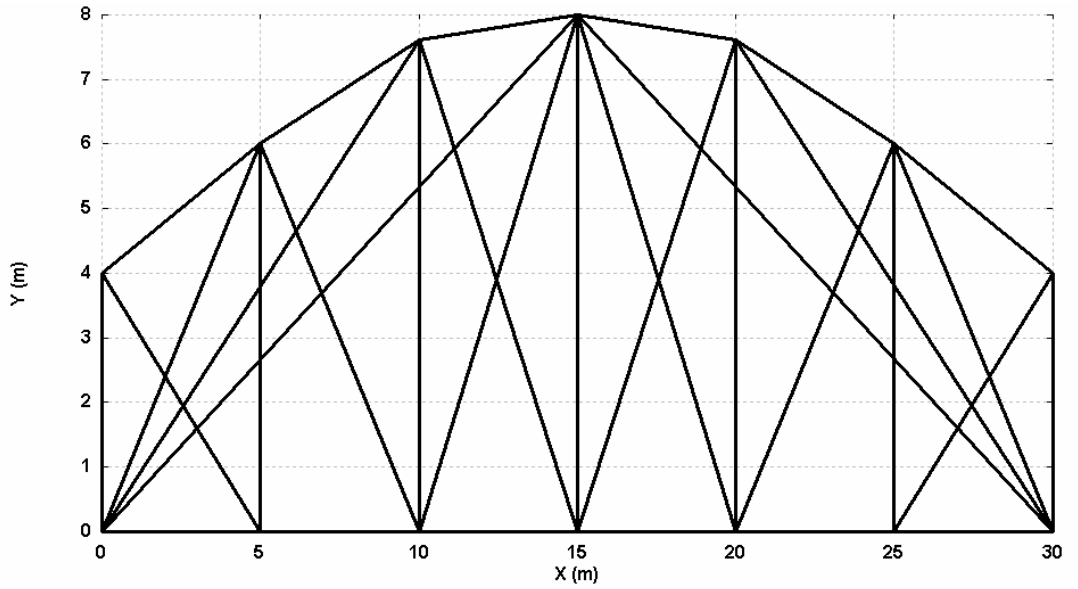


Figure – 5.50 Shape of the bridge structure for run 3

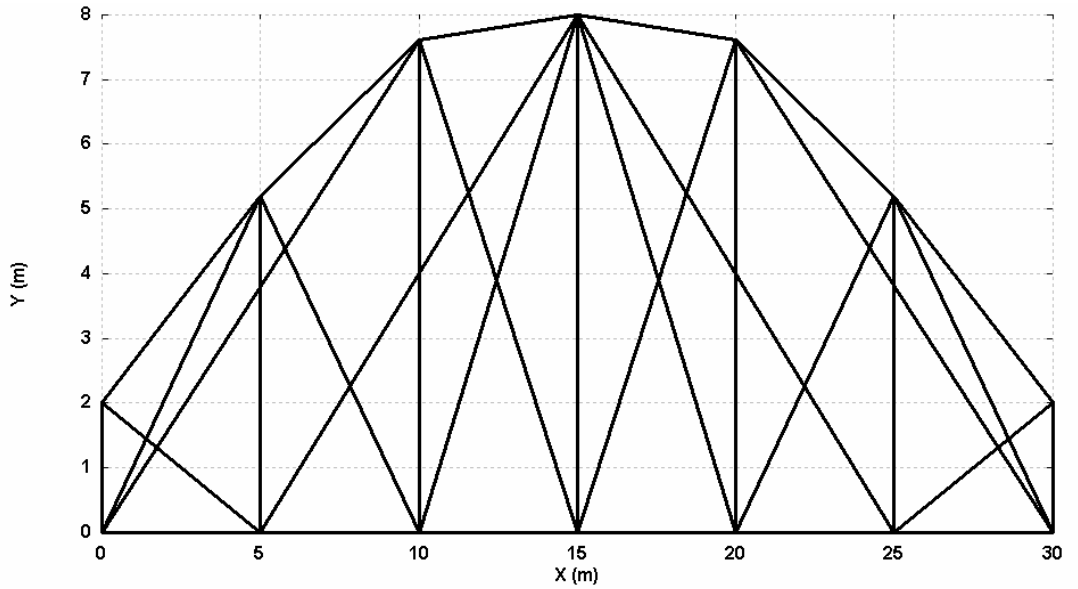


Figure – 5.51 Shape of the bridge structure for run 4

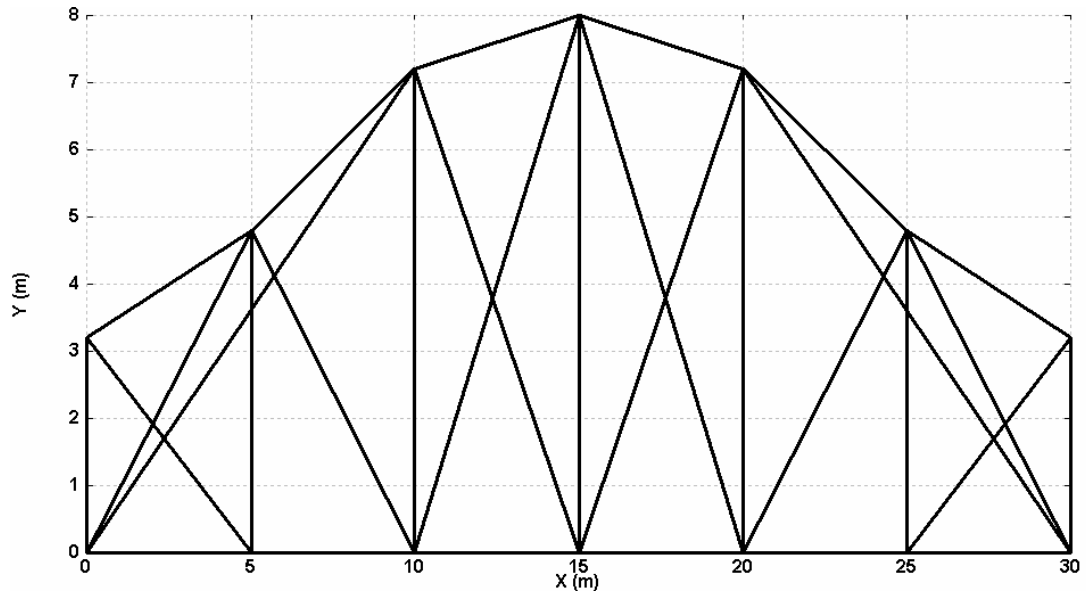


Figure – 5.52 Shape of the bridge structure for run 5

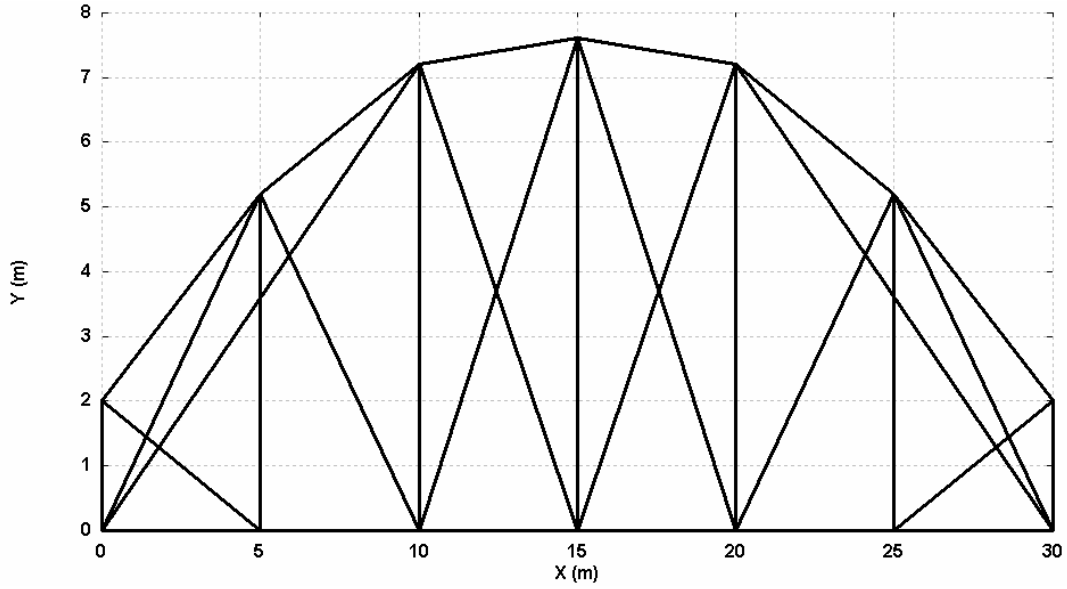


Figure – 5.53 Shape of the bridge structure for runs 8 and 10

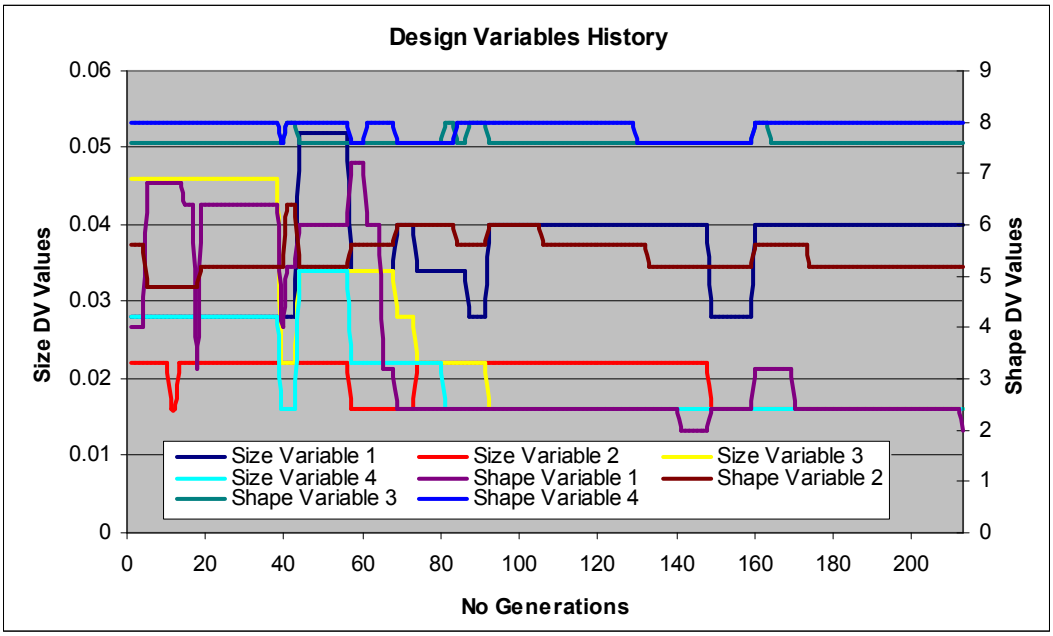


Figure – 5.54 Design Variables History for Rajan's truss-1

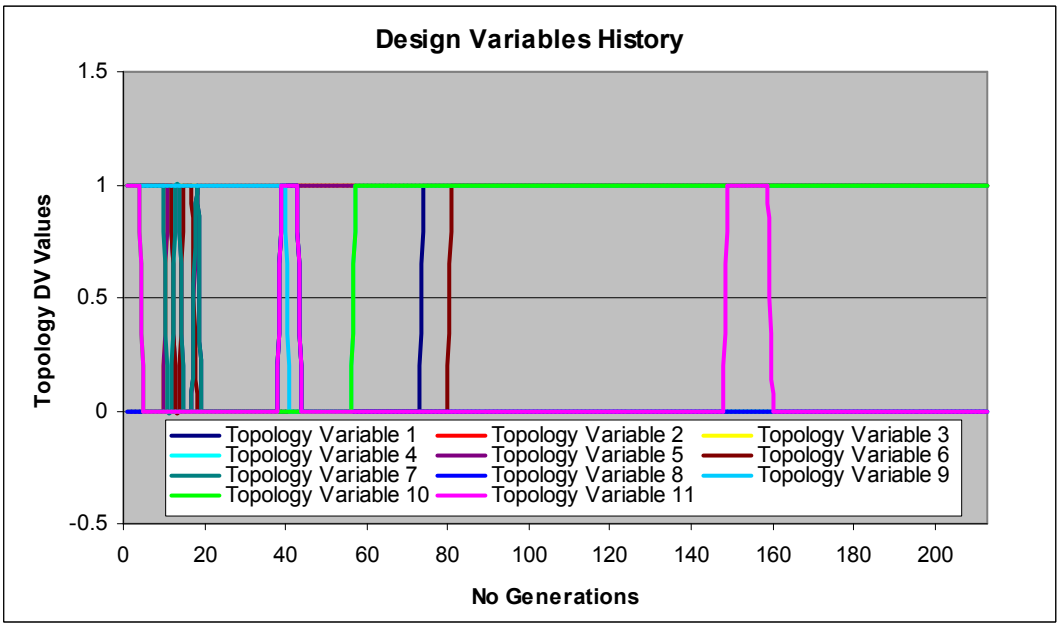


Figure – 5.55 Design Variables History for Rajan's truss-2

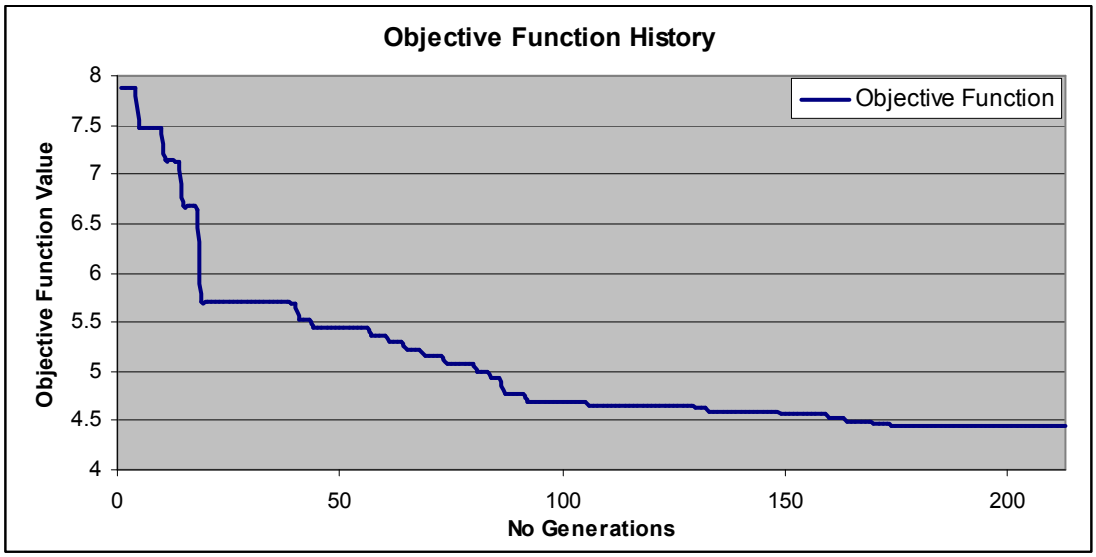


Figure – 5.56 Objective Function History for Rajan's truss

CHAPTER 6

DISCUSSION AND CONCLUSION

6.1 Discussion

The aim of this study is to get into Genetic algorithms and investigate the effectiveness of the algorithm by experimenting it on several test and structural optimization problems. A FORTRAN code, *GABSO,* has been developed for this purpose.

In order to test the algorithm, first a set of numerical test problems has been solved. These problems are difficult optimization problems in which there are more than one local optima. In the first test problem, the global maximum is surrounded by concentric rings of second, third and fourth level maximums. For a gradient-based algorithm, the probability of a randomly chosen starting point to give the global maximum is 1% [5]. There are two local maxima in the second test problem and the global maximum covering only 1% of the parameter space. The global maximum is located in a very small region when compared with the secondary maximum. This is a very difficult optimization problem when the algorithm used is gradient based. A constrained multimodal function was used as a third test problem. The function has multiple relative minima. The linear constraint for the problem only excludes one relative minimum from the feasible space. And the last test problem is the six-hump camel back function. Within the bounded region six local minima are located; two of them are global minima.

The test problems show the capability of the algorithm to avoid the local optima points. And also these examples show that Genetic algorithm can also be used not only for discrete optimization but also for continuous optimization.

In the structural optimization problems presented in this study, the objective is to minimize the weight of the structure. And mainly this is achieved by considering the cross sectional properties of the bars or beams. For the shape optimization problems, the nodal coordinates are also used as design variables. All the problems considered are constrained optimization problems in which there are stress, displacement and geometrical constraints.

Optimization problems start with 6-bar truss structure. It is followed by 25-bar truss and 72-bar truss structures. In these problems, the objective is to minimize the weight of the structure by using the cross sectional areas of the bars as design variables. In 25 and 72 bar space structures, number of design variables is reduced by linking various member sizes. There are two types of constraints used in theses problem, stress and displacement constraints.

In the 18-bar truss structure, the nodal coordinates are also taken as design variables as well as the cross sectional areas of the bars. This problem is a combined optimization problem, size and shape optimization. Only the nodal coordinates are taken as design variables in the shape optimization of bridge structure. Both of the optimization problems are chosen in order to show the shape optimization capability of the algorithm.

Next, the stepped cantilever beam and portal frame design problems are solved. In these problems, the design variables are the geometric dimensions of the cross sections and the objective is weight minimization. Finally, the Pier-cost and Rajan's truss examples are solved. These problems also include topology variables as well as size and shape design variables. The topology variables are used to determine the presence or absence of truss members. By using these variables, the element connectivities are updated in each run.

In all truss structures except 72-bar space structure, TRUSS.f90 is used as finite element solver. For the rest of the design problems, MSC/NASTRAN® is used. All the structural and test problems are solved by using GABSO.

6.2 Conclusion

During this study, mainly two major characteristic of Genetic algorithm have been observed. First of all, the Genetic algorithm is a stochastic search algorithm. This means that there is randomness in the optimization process. So each run of the optimization process is unique. Secondly, by making random moves in the design space, a GA can avoid local optima points easier than the gradient based optimization techniques.

In all the problems, the optimum had been obtained by only considering a very small percentage of the design space. Mainly, Tournament selection is used in the optimization process and Elitisizm is activated. The Tournament selection used in these problems creates a high variation in the generations. And the Elitisizm results in a fast convergence. But using the Elitisizm can also cause the algorithm to stuck at local optimums. So it is logical to use Elitisizm with tools that creates high variations in the populations like Tournament selection, uniform crossover and mutation with high probability. Creep mutation is also an important tool that can be used for jumping the Hamming walls

When Rank selection and Elitisizm are used in the optimization process, the algorithm may converge to a local optimum very quickly, and stucks at that point. So in order to achieve the necessary variation in the population to jump from local optimum, the mutation probability is increased. After obtaining the variation, the mutation probability is set to its predefined value. Mutation probability variation is a characteristic feature of the GA used in this study.

Selecting improper set of optimization parameters may result in immature solutions at the end of the optimization process. These results can be improved by searching the neighborhood of the optimum, obtained by the algorithm. A scatter search algorithm can be activated to check the possibility to have a better solution in the neighborhood. This is a unique feature used in the current algorithm.

It is also observed that, the initial population is crucial in the optimization process. Since, the individuals in the population lead the process. Increasing the size of the population creates high probability of convergence but it also cause the number of function calls to be increased, so an increase in running time.

The GA parameters and also the weight terms in fitness function used should be chosen carefully according to the physics of the problem. These parameters directly affect the optimization process. Wrong choices for these parameters can lead to meaningless results. In the fitness function, the constraint violation terms should be treated very carefully, since some infeasible designs can create very fit individuals due to the nature of crossover and mutation processes. But also it must be kept in mind that, penalizing the constraint violations by ineffective penalties can lead to infeasible designs at the end of the optimization process. So in the design of the fitness function, a balance must be obtained.

REFERENCES

[1] Belegundu A.D., Chandrupatla T.R. "Optimization Concepts and Applications in Engineering", Prentice-Hall Inc., 1999

[2] Kamat M.P., **"**Structural Optimization: Status and Promise", Progress in Astronautics and Aeronautics, Volume 150, 1993

[3] Haftka R.T., Gürdal Z., "Elements of Structural Optimization", Kluwer Academic Publishers, 1990

[4] Dianati M., Song I., Treiber M., **"**An Introduction to Genetic Algorithms and Evolution Strategies", University of Waterloo, July 2002

[5] Charbonneau P., "An Introduction to Genetic Algorithms for Numerical Optimization", NCAR Technical Note, March 2002

[6] Said Y.H., **"**On Genetic Algorithms and their Applications", Handbook of Statistics, Vol. 24 ISSN: 0169-7161, 2005

[7] Moore G.J.**, "**MSC/NASTRAN® Design Sensitivity and Optimization", MSC.Corporation, 1994

[8] Nanakorn P., Meesomklin K., **"**An Adaptive Penalty Function in Genetic Algorithms for Structural Design Optimization", Computers and Structures 79 2527-2539, 2001

[9] Pedersen P., "Optimal Designs- Structures and Materials - Problems and Tools", Technical University of Denmark, Department of Mechanical Engineering, May 2003

[10] "GENESIS User Manual Volume 4 Optimization Demonstration Problems", VMA Engineering, 1998

[11] Gantovnik V.B., Anderson-Cook C.M., Gürdal Z. and Watson L.T., "A Genetic Algorithm with Memory for Mixed Discrete–Continuous Design Optimization", Computers and Structures, 81 2003-2009, 2003

[12] Rajeev S, Krishnamoorty C.S., "Discrete Optimization of Structures using Genetic Algorithms", Journal of Structural Engineering ASCE 118(5): 1233±50, 1992

[13] Xicheng W, Guixu M., "A Parallel Iterative Algorithm for Structural Optimization", Computer Methods in Applied Mechanics and Engineering 96:25±32, 1992

[14] Yu X., Johnson E.H., Zhang S., **"**Discrete Optimization in MSC/NASTRAN®", MSC.Software Corporation, 2001

[15] Wang D., Zhang W.H., Jiang J.S., "Truss Shape Optimization with Multiple Displacement Constraints", Comput. Methods Appl. Mech. Eng., 191 3597-3612, 2002

[16] Thanedar P.B., Vanderplaats G.N., "Survey of Discrete Variable Optimization For Structural Design", Journal of Structural Engineering, Vol. 121. No: 6341, 1995

[17] Gillman K.M., "Optimization of Shape, Size and Topology Design Variables In Trusses with a Genetic Algorithm", Brigham Young University, April 2005

[18] Rajan S.D., "Sizing, Shape and Topology Design Optimization of Trusses using Genetic Algorithm", Journal of Structural Engineering 121 1480-1487, 1995

[19] McCALL J., "Genetic Algorithms for Modelling and Optimization", Journal of Computational and Applied Mathematics, 184 205-222, 2005