8-2016

Application of Genetic Algorithm in Multi-objective Optimization of an Indeterminate Structure with Discontinuous Space for Support Locations

Rahat Sultana
Grand Valley State University

Application of Genetic Algorithm in Multi-objective Optimization of an Indeterminate Structure with Discontinuous Space for Support Locations

Rahat Sultana

A Thesis Submitted to the Graduate Faculty of

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Engineering

Product Design and Manufacturing Engineering

August 2016

Dedication

To my parents, Engr. Md. Robiul Islam and Mrs. Mahamuda Islam, and my husband M Shahriar Mazid Khan, for their selfless and relentless support to my pursuit of higher studies.

Acknowledgement

I would first like to convey my sincere thanks to Dr. Shabbir A Choudhuri, my thesis advisor, for supervising my work and for his guidance, expertise, patience and involvement to this work. This work would not have been completed without his instructions, appreciations, and inspiration. I cannot thank him enough for enriching me with first-hand research experience, and helping to be habituated with apt brainstorming, planning research works and accomplishing them within deadline. I owe countless thanks to Dr. Wendy Reffeor for her continuous involvement with the work, technical guidance, insight, and valuable time. My heartfelt thanks go to Dr. Lindsay Corneal for her persistent support and encouragement throughout my research work. I would also like to thank Graduate School, Grand Valley State University for supporting the research by funding this project. M Shahriar Khan, my husband, has kept my courage high during the hard times. Without his support, it was not possible for me to endure the struggle of graduate studies. And finally, I would like to thank my parents, Md Robiul Islam and Mrs. Mahamuda Islam for their continuous support and faith in me.

Abstract

In this thesis, an indeterminate structure was developed with multiple competing objectives including the equalization of the load distribution among the supports while maximizing the stability of the structure. Two different coding algorithms named "Continuous Method" and "Discretized Method" were used to solve the optimal support locations using Genetic Algorithms (GAs). In continuous method, a continuous solution space was considered to find optimal support locations. The failure of this method to stick to the acceptable optimal solution led towards the development of the second method. The latter approach divided the solution space into rectangular grids, and GAs acted on the index number of the nodal points to converge to the optimality. The average value of the objective function in the discretized method was found to be 0.147 which was almost one-third of that obtained by the continuous method. The comparison based on individual components of the objective function also proved that the proposed method outperformed the continuous method. The discretized method also showed faster convergence to the optima. Three circular discontinuities were added to the structure to make it more realistic and three different penalty functions named flat, linear and non-linear penalty were used to handle the constraints. The performance of the two methods was observed with the penalty functions while increasing the radius of the circles by 25% and 50% which showed no significant difference. Later, the discretized method was coded to eliminate the discontinuous area from the solution space which made the application of the penalty functions redundant. A paired t-test (α=5%) showed no statistical difference between these two methods. Finally, to make the proposed method compatible with irregular shaped discontinuous areas, "FEA Integrated Coded Discretized Method (FEAICDM)" was developed. The manual elimination of the infeasible areas from the candidate surface was

replaced by the nodal points of the mesh generated by Solid Works. A paired t-test (α=5%) showed no statistical difference between these two methods. Though FEAICDM was applied only to a class of problem, it can be concluded that FEAICDM is more robust and efficient than the continuous method for a class of constrained optimization problem.

Table of Content

List of Illustrations

List of Tables

1. Introduction:

Genetic algorithms (GAs), members of the large class of "Evolutionary Algorithms", are metaheuristics approach for solving various optimization problems. GAs are inspired by the natural selection process. GAs operate on a set of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. Based on the fitness level of the individual solution in each generation, a new pool of parents is selected for breeding the next generation using various operators adopted from natural evolution. Thus at each generation, GAs try to generate offspring exhibiting better fitness level which are better suited to their environment than the population they are generated from [1]. In various multi-objective optimization problems, the applicability of GAs has been proven through numerous research works. Also, there are competing optimization methods available to solve structural design problems. But certain characteristics of this class of problems have made GAs popular in this research field. GAs are suitable for continuous problems as well as for discrete and non-differentiable problems. Additionally, GAs are very efficient for searching global optimal solutions.

However, there are some interesting areas related to the application of GAs to the structural optimization problems which are not yet fully explored. The following areas require further investigation

- In constrained optimization problems, application of penalty function is very common. GAs are successfully used with penalty function applications. However, there is no systematic approach to understand the influence of the magnitude and trend of penalty function on the convergence towards the global optima.

- GAs have been used in numerous optimization problems having a discontinuity in the solution space. It is important to observe the impact of the size of the discontinuity of the solution space on the performance of GAs.
- Application of GAs in constrained optimization is a complex process. Proper selection of different parameters, which is a prerequisite for getting a better optimal result, makes the process more convoluted. Thus, the development of a method for a specific class of problems where parameter selections are not required could be very beneficial for applying GAs in discrete solution space [2].

1.1 Objective:

In this work, GAs have been used to solve for the support locations of a multi-objective indeterminate structural model. The objectives of this research work are given below:

- Development of a test case with competing objective functions
- Development of a Genetic Algorithm based approach to solve the support locations
- Executing comparative analysis of the performance of flat, linear and non-linear penalty functions in handling constraint in GAs
- Determination of the effect of the size of the discontinuous solution space on the performance of GAs
- Development of a coding algorithm to handle constraint in GAs

1.2 Scope:

In this work, GAs have been applied for a specific class of optimization problem with unconstrained and constrained conditions. Though there are various penalty functions to make GAs applicable in constrained optimization, only flat, linear and non-linear penalties have been considered for this thesis.

1.3 Organization of the Thesis:

The thesis is organized in the following five chapters:

- **Chapter 1:** In Chapter 1, the introduction, objectives of the thesis and limitation are discussed.
- **Chapter 2:** Relevant theoretical background is described in this chapter which covers the basic concept of optimization, single and multi-objective optimization, classical and evolutionary methods of solving optimization problems, fundamentals of GAs, application of GAs in constrained optimization and structural optimization problems etc. This chapter provides a brief idea about the contextual reasoning of the thesis.
- **Chapter 3:** The methodology to carry on the work is discussed in this chapter.
- **Chapter 4:** The results obtained from the observation are illustrated and analyzed here.
- **Chapter 5:** Based on the findings, the conclusion of the work will be abridged and future scope of the work will be discussed.

2. Literature Review:

2.1 Optimization:

Optimization is simply defined as a process to find a better solution. In technical terms, optimization is a selection process to determine the best course of action for a decision problem from some set of available alternatives under the restriction of limited resources. Here, a function, which is called an objective function, cost function or fitness value, is minimized or maximized relative to some set (which represents a range of alternatives available in a certain situation) and by computing this function different choices are compared to determine which one is the "best". Thus, some inputs or variables are tweaked to find the maximum or minimum objective function. Optimization is also referred as "Mathematical Optimization" as the generalization of the optimization theory and techniques requires knowledge of a large area of applied mathematics.

Most of the optimization related research works have considered a single objective whereas most of the real life optimization problems contain multiple objectives to satisfy. Some trade-off optimal solutions are searched for multi-objective optimization problem as it is not possible to get one single optimum solution to multiple conflicting components of the objective function. In this thesis, the focus will be on multi-objective optimization.

2.1.1 Approaches to Solve Optimization:

Optimization problems can be solved using two major approaches which are Classical Methods and Evolutionary Algorithms.

2.1.1.1 Classical Methods:

The classical methods update a single solution at each iteration by following a deterministic approach to reach to the optimality [1]. The steps followed by most classical methods start with the

guess of a random solution. Then a search direction is intimated using a pre-specified deterministic approach followed by a unidirectional search along a suggested direction. The same iteration is repeated until the stopping criteria is met. Classical optimization methods are classified into two groups which are direct methods and gradient-based methods [3]. The basic difference between these two methods is that the direct methods use the objective function and constraint values to direct the search method whereas the gradient methods use the derivatives of the objective function and constraints for convergence. This difference has made direct methods slower than gradient methods as they require the evaluation of many functions to guide the search process, but at the same time, they can be applicable to a number of problems without making major changes in the algorithm. Another drawback of the classical methods is that the selection of the initial solution plays a vital role in the convergence to the optimal solution. These methods are inefficient for optimization problems having a discrete solution space. Most of the cases, these methods have a tendency to get stuck to a local solution. There is also a lacking of an appropriate general algorithm for various classes of optimization problems [1].

2.1.1.2 Evolutionary Algorithms (EAs):

The concept of EAs was developed based on the processes of Darwinian evolution. EAs are computer programs, and their components are developed in a suitably coded representation. Some simulated creatures, known as individuals (fixed length vectors or strings), compete with each other over the search space of a problem to find out better optimal areas in the search space. Each and every individual has a possibility to be a solution to the given problem. At the very beginning, initial individuals are generated randomly using random number generator. Then they are evaluated based on how well they can satisfy the objective of the problem. Based on their performances, every individual is assigned a score. The individuals with larger scores represent the better solutions to the problem. A pre-determined number of better individuals are selected from this phase and undergo

other evolutionary operators named crossover and mutation to breed children. Based on the performance of these children, a set of the population is selected from them and used as the current population. Then, the same iteration is repeated until a stop criterion is met. Thus, in EAs, individuals with better fitness score are selected, and less fit individuals are removed gradually.

2.1.2 Classification of EAs:

There are mainly three dialects of evolutionary algorithms [4, 5, 6], Genetic Algorithms (GAs), Evolution Strategies (ESs), and Evolutionary Programming (EP), which follow the general outline mentioned above. Each of these three algorithms has been proved capable of yielding approximate optimal solutions for given complex, multimodal, non-differential, and discontinuous search spaces. Another mentioned evolutionary algorithm is Genetic Programming (GP) [1]. These evolutionary algorithms are explained below:

2.1.2.1 Genetic Algorithms (GAs):

The concept of GAs was developed by Holland [7] and first applied by Goldberg in his work [8]. GAs have become the most popular among all EAs. According to the established concept of GAs, a population of random individuals are generated and based on their fitness score, participants to breed next generations are selected using "Roulette wheel parent selection". Recombination and mutation operators perform to evolve the next generations and again more suited individuals are selected to replace the parents in the population set. Same steps are followed until the termination conditions are met. As GAs were used in this thesis work, they will be explained in more details later.

2.1.2.2 Evolutionary Strategies(ESs):

ESs, developed by Rechenberg and Schwefel [9, 10], use real-valued vector representation to encode individuals, and these strings of real numbers are called objective variables of the optimization

problem. Some strategy parameters (variances and covariance of individuals) are used to direct the actions of the mutation operator. Mutation operator acts on the strategy parameters first and then the object variables are mutated using the probability distribution generated from the mutated strategy parameters. With these self-adapted strategy parameters and deterministic selection process, ESs evolve to optimal solution and stop when any stopping criteria are met. Though only one application of ESs has mentioned in computational chemistry [11], they have the potential to be an alternative to GAs, especially in parameter optimization.

2.1.2.3 Evolutionary Programming (EP):

In EP, limited symbolic alphabets are used to represent the finite state machines. It was first developed by Fogel et al. [12] and later modified by D. B. Fogel to represent real numbers [6]. Though it deals with a string of real numbers similar to ESs, the main difference between them is that EP does not use any recombination operator. Thus, the convergence to better solution depends only on mutation operator by using Gaussian probability distribution. EP is suitable for parameter optimization and has been applied in some other areas too [13, 14].

2.1.2.4 Genetic Programming (GP):

Individuals are embodied as computer programs in GP. Based on a given problem, GP generates computer program automatically to solve that problem. Here, a computer program is encoded as chromosome and evaluated to measure its fitness to meet predefined objectives or goals. It is also considered as an application of GAs for problems having computer programs as the individuals. In 1985, Cramer [15] first developed the modern "Tree-based" GP where programming languages are organized in tree-based arrangements and modified using various GA operators. Koza [16] showed its application in various complex optimization problems along with in modeling DNA expression.

2.1.3 Differences between Evolutionary Algorithm and Classical Methods:

EAs are different from classical methods in several ways, which are mentioned below [8],

- EAs normally do not use any derivatives of the objective function and constraints in its search process.
- EAs follow a population approach to search for an optimal solution which implies that instead of working with a single solution in every iteration, they work with a set of initial solutions. But, most of the classical methods start their searching process with one initial solution (point approach). Evolutionary methods become computationally quick because of its parallel processing of a set of solutions and are more suitable for multi-objective optimization problems. Another advantage is, these algorithms can normalize decision variables along with objective and constraint functions within a population by using the information of the best and worst performed individuals of that population.
- EAs use stochastic operators instead of the deterministic approach used in classical optimization. Thus, classical methods use a fixed transition rule to guide the search direction. On the other hand, the operators in EAs reach towards the desired outcome by applying higher probabilities which provide them with the capability to deal with multiple optima and other complexities in a better way than classical methods.

2.2 Multi-Objective Optimization Problem (MOOP):

A MOOP has more than one objective function. In the real world, most of the optimization problems are multi-objective, for example, machine learning (accuracy vs. speed vs. complexity), finance (expected return vs. standard deviation) etc. In most of the engineering problems, many decisions involve multiple objectives which may conflict each other, such as minimize cost, maximize efficiency or performance, maximize reliability, etc. For these type of problems, one

optimal solution of one objective may not provide the best solution of other objectives. As one extreme solution would not provide the optimal solution for all objectives, a set of solutions which compromises between different objectives is required to present optimal solutions of all objective functions [1].

In the past, due to the lacking of a proper algorithm, MOOPs were modified as a single objective problem. As the working principles of the single and MOOP are different, a single optimal solution can satisfy the single objective problem, but a MOOP requires an optimal solution for each objective.

2.2.1 Formulation of Multi-Objective Optimization Problem:

An MOOP has more than one objective function. In this thesis, two objectives have been considered in the unconstrained problem, and three objectives have been used for the constrained problem. The problem has been formulated as a minimization problem. These type of problems have a number of constraints including inequality, equality and/or variable bounds which determine the feasibility of any solution. The general form of MOOPs is given below,

$$\text{Minimize/ Maximize} \quad f_m(x), \qquad m = 1,2, \dots \dots, M;$$

$$\text{Subject to} \quad g_j(x) \leq 0, \qquad j = 1,2, \dots \dots, J;$$

$$h_k(x) = 0, \qquad k = 1,2, \dots \dots, K;$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1,2, \dots \dots, n;$$

Here, x is the vector of n design variables, $x_i^{(L)}$ and $x_i^{(U)}$ are the lower and upper boundaries of the design variables respectively, the number of inequality and equality constraints are j and k

respectively, $g_j(x)$and $h_k(x)$are the constraint functions, and $f_m(x)$ is the objective function to be optimized.

2.2.2 Basic Concepts and Terminology:

Some basic concepts are required to understand the multi-objective optimization algorithm in more depth, which are given below:

- **Decision Variable Space:**

The space generated by the lower and upper limit of each decision variable is called decision variable space. The variable bounds restrict each variable within its boundary limit.

- **Objective Space:**

In a MOOP, values of objective functions generate a multidimensional space which is called objective space. For each solution in the decision variable space, there is a point in the objective space.

- **Feasible and Infeasible Solution:**

A feasible solution satisfies all constraints (linear and non-linear, equality and inequality) and variable bounds. The solution having the opposite characteristic of the feasible solution is called the infeasible solution.

- **Linear and Non-linear MOOP:**

A linear MOOP having linear objectives and constraint functions is called Multi-objective Linear Problem (MOLP). On the other hand, if any constraint and/or objective functions are non-linear, it is called a non-linear MOOP [1].

- **Convex and Nonconvex MOOP:**

A MOOP is convex if all the objective functions and feasible region are convex. For a convex function: $R^n \rightarrow R$, any two pair of solutions $x_1, x_2 \in R^n$ will satisfy the following conditions:

$$g(\alpha x_1 + (1-\alpha)x_2) \leq \alpha g(x_1) + (1-\alpha)g(x_2) \ldots\ldots\ldots \text{ where } 0 \leq \alpha \leq 1$$

Both spaces (decision and objective function spaces) of a MOOP problem should be evaluated to test their convexity. Even one of them can be non-convex while another one is convex. A MOLP has been defined as a convex problem [1].

- **Ideal Objective Vector:**

The ideal objective vector consists of an array with the lower bound of all objective functions of a MOOP resulting in non-conflicting objective functions. It can only be possible for a feasible solution when the minimum of all objective functions are identical. Otherwise, it does not exist. If $x^{*(i)}$is a solution vector of variables that minimize or maximize the ith objective in a MOOP having M conflicting objectives,

$$\exists x^{*(i)} \in \Omega, x^{*(i)} = \left[x_1^{*(i)}, x_2^{*(i)}, \ldots\ldots x_M^{*(i)}\right]^T : f_i\left(x^{*(i)}\right) = OPT\ f_i(x)$$

Thus the ideal vector is defined as following,

$$z^* = (f^*) = (f_1^*, f_2^*, \ldots\ldots f_M^*)^T$$

where f_M^* is the optimum value of Mth objective function and the point in decision variable space which determines this vector is the ideal solution.

- **Utopian Objective Vector:**

The objective vector having components slightly less than that of an ideal objective vector for the minimization of a MOOP problem is called the utopian objective vector. It is used for algorithms

requiring a better solution than any other solution in the search space strictly. The utopian objective vector, z^{**} is expressed as following:

$$\forall\, i = 1,2,3,\ldots\ldots, M \quad : \quad z^{**} = z^{*} - \epsilon_i \;\; , \;\; \epsilon_i > 0.$$

- **Nadir Objective Vector:**

The nadir objective vector is expressed as $z^{nad}$and this vector contains an array of the upper bounds of each objective function in the Pareto-optimal set. It does not consider the entire solution space. So, the m[th] component of the nadir objective vector z^{nad} is the constrained maximum of the following problem:

max $\quad f_m(x)$

subject to $x \in P$

where P is the Pareto-optimal set. The objective functions can be normalized by using ideal and nadir objective vectors by using the following equation:

$$f_i^{norm} = \frac{f_i - z_i^*}{z_i^{nad} - z_i^*}$$

- **Dominance Relation:**

For multi-objective optimization, one optimal solution for one objective function might not necessarily be optimal for other objective functions. In MOOP, $\lhd$ is used to show the dominance of one solution over others. In general, if two feasible solutions of a MOOP having M conflicting objectives are $x_1$and x_2 and x_1 is defined to dominate x_2, then the following statements must be true:

1. The solution $x_1$is no worse than $x_2$in all objectives, or mathematically, $f_i(x_1) \not\succeq f_i(x_2)$ for all i=1,2,3,….,M

2. The solution x_1 is strictly better than x_2 in at least one objective, or mathematically, $f_j(x_1) \lhd f_j(x_2)$ for at least one $j \in 1,2,3,\ldots,M$

The dom inance relation does not have reflexive property as no solution dominates itself. It also does not exhibit symmetric characteristic as the dominance of one solution x over another solution y does not mean the dominance of y over x. If x dominates y and y dominates z (a third solution), then x dominates z which shows the transitive property of the dominance relation.

- **Pareto-Optimal Set (Non-dominated set):**

A set is said to be a non-dominated set or Pareto-optimal set if it is not dominated by any other solution that belongs to the solution set. The Pareto-optimal set is the best optimal solution for all objective functions and cannot be improved with respect to one objective by worsening another one. Mathematically, if P is a set of solutions, the non-dominated set of solutions P^* comprises those solutions which are not dominated by any member of the set P. The non-dominated set of solutions can be generated by comparing all possible pairs of a given solution set and determining which solutions dominate which, and which are not dominated by each other. The Pareto-optimal set, P^* can be written as:

$$P^* = \{x \in \Omega \mid \neg\exists\, x' \in \Omega\, F(x') \preccurlyeq F(x)\}$$

Pareto-optimal sets are called global when the set of solutions, P, is the entire search space. If for every member x in a set P, there exist no solutions y in the neighborhood of x, then $\|y - x\| \leq \varepsilon$, dominating any member of the set, then P establishes a locally Pareto-optimal set.

- **Pareto-front:**

The Pareto-front contains the values of objective functions for all solutions in the Pareto-optimal set in the objective space. If the Pareto-front is PF^* for a given MOOP having objective function F(x), then mathematically:

$$PF^* := \{u = F(x) | x \in P^*\}$$

- **Methods for solving the non-dominated set:**

In each iteration of a MOOP, the non-dominated set is needed to be determined. Thus, a computationally efficient approach is required to perform the determination step. Three different approaches were mentioned which are (i) Naïve and slow, (ii) Continuously updated and (iii) Kung et al.'s efficient method [1]. All of these methods use the concept of domination to determine the non-dominant set with respect to different objective functions. But the most efficient [17] and least computationally complex method is the third one.

2.2.3 Approaches to solve Multi-objective Optimization:

Extensive studies have been conducted in multi-objective optimization algorithms. But most of the research work has avoided the complexity by transforming the problem into single- objective optimization with the use of some user-defined parameters. Deb [1] classifies the approaches towards solving multi-objective optimization in two groups.

- Ideal multi-objective optimization, where a set of solutions in the form of a trade-off curve is obtained, and the desired solution is selected based on some higher level information of the problem.
- Preference based multi-objective optimization, where using the higher level information a preference vector transforms the multi-objective problem to a single-objective optimization. The optimal solution is obtained by solving the single-objective problem.

The ideal approach is less subjective than the preference-based approach. It requires analysis of non-technical, qualitative and experimental information to find the preference vector. In the absence of higher level information in an optimization problem within the ideal approach, none of the Pareto-optimal solutions is preferred over others. Therefore, in the ideal approach, the main objective is to

converge to a set of the solution as close as possible to the true Pareto-optimal set, which is the common objective of all optimization tasks. However, diversity in the obtained Pareto-optimal set is the second objective specific to multi-objective problems. With a more diverse set of solutions that covers all parts of the Pareto-front in the objective space, the decision-making process at the next level using the higher level information is easier. Since two spaces are involved in MOOP, diversity of solutions in both decision and objective space is defined. Solutions with a large Euclidean distance in variable and objective space are referred to as a diverse set of solutions in variable and objective space, respectively. The diversity in the two spaces is often Symmetric, however in complex and non-linear problems this property may not be true. Hence, Deb [1] assumes that there are two goals in multi-objective optimization:

a. To find a set of non-dominated solutions with the least distance to the Pareto-optimal set.
b. To have maximum diversity in the non-dominated set of solutions.

2.2.4 Classification of Multi-Objective Optimization Algorithm (MOOA):

In section 2.1.1 it was mentioned that optimization solving methods are classified into the classical and the evolutionary methods. That classification is also valid for multi-objective optimization problems.

2.2.4.1 Classical Method for MOOA:

In the classical method, objectives are transformed into one objective function using different techniques. The classical methods for MOOA will not be discussed in detail as this research work is based on the EAs.

2.2.4.2 Evolutionary Algorithms (EAs) for MOOA:

The characteristic of EAs of using a population of solutions that evolve in each generation is well suited for multi-objective optimization problems. Since one of the main goals of MOOP solvers is

to find a set of non-dominated solutions with the minimum distance to Pareto-front, evolutionary algorithms can generate a set of non-dominated solutions in each generation.

The requirement of little prior knowledge about the problem, less vulnerability to shape and continuity of Pareto-front, easy implementation, robustness and the ability to be carried out in parallel are some of the advantages of evolutionary algorithms listed in Goldberg's study [18].

The first goal in multi-objective optimization is achieved by a proper fitness assignment strategy and a careful reproduction operator. Diversity in the Pareto-set can be obtained by designing a suitable selection operator. Preserving the elitism during generations to directly carry over the elite solutions to the next generation shall be carefully considered in evolutionary algorithms [1].

Coello [19] presents the basic concepts and approaches of multi-objective optimization evolutionary algorithms. The book further explores some hybrid methods and introduces the test functions and their analysis. Various applications of multi-objective evolutionary algorithms (MOEA) are also discussed in the book. Deb's book [1] is another comprehensive source of different MOEAs. The book divides the evolutionary algorithms into non-elitist and elitist algorithms.

2.3 Genetic Algorithm:

Genetic Algorithms, one of the popular optimization techniques, are stochastic global search procedures which impersonate the "Natural Theory of Evolution" developed by Charles Darwin via three basic operations: selection, recombination, and mutation [20]. They deal with a population of prospective solutions concurrently following the evolution theory "Survival of the fittest" and produce better approximations to the solution for the next generation based on the fitness of the objective function. At each generation a new collection of individuals is generated based on the level of fitness using natural operators such as crossover, mutation, etc. and thus those features that make an individual more suited are preserved and better competent individuals survive until a satisfactory result is obtained. GAs are the most widely known evolutionary algorithms [7, 21, 22]. In the areas

of management science, operations research, and industrial and system engineering, the application of GAs is increasing day by day because of the advantages of GAs over the conventional methods [8, 20, 22]. They have also been successfully applied in different real-world scenarios, for example in aeronautics, electrical engineering, scheduling and signal processing, etc. The concept of using GAs was first introduced by Holland [7]. Then it was developed theoretically [7] and applied in various fields [8]. Simpson et al. [23] used simple GAs and Dandy et al. [24] experimented with the fitness function, mutation operator, and gray codes. Abdel-Gawad [25] showed and explained the effect of different selection, crossover and mutation schemes of the GAs on the network optimization.

2.3.1. GA Operators:

2.3.1.1. Initialization:

In GAs, decision variables or parameters are encoded and a set of initial solutions, called the population, is generated. GAs operate on the population simultaneously. A random generator is used to generate the required number of initial individuals or population within the desired range. Bramlette [26] suggested an approach of generating individuals where for each individual a number of individuals are generated, and the best-performed one is selected for the initial population. Another approach, which is only applicable to well-known problems, is to initialize the population with some individuals from the vicinity of global optimal results [27, 28]. The binary string representation is the most popular representation where each variable is encoded in the binary string and concatenated to form a complete chromosome. In traditional binary representation, one problem is that the hamming distances between adjacent values are constant which affect the search space by deceiving it while searching global optima [29]. Gray coding is used to improve the standard system.

There are some other approaches which can replace the binary string representation such as real-valued representation, integer representation, etc. Sometimes, application of integer representations is more suitable and convenient for some classes of problems such as subset selection, route selection problems, etc. as binary representation might obfuscate the nature of the problem [26].

Application of real-valued representation has some advantages over binary representation such as increased efficiency, the requirement of less computational time and less memory, no loss of precision which happen while discretizing to binary or other values and a wide range of operators to be used.

2.3.1.2. Objective and Fitness Functions:

The decision to select an individual for the next generation is made based on the objective function. The objective or fitness function measures the performances of individuals. If the problem is a maximization problem, the best-performed individual will have the maximum numerical value of the objective function. The fitness function is used to transform the objective function to a relative fitness, which can be expressed as:

$$F(x) = g(f(x))$$

where, x is the decision variable, f is the objective function, g is the function to transform the objective function, and F is the resulting relative fitness. In proportional fitness assignment, the ratio of the raw performance of each individual and the performance of all individuals of the population is used to transform the objective function, thus:

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{Nind} f(x_i)}$$

Here, x_i is the value of individual i and $Nind$ is the size of the population. Before this fitness assignment, the objective function goes through a linear transformation using the following equation:

$$F(x) = af(x) + b$$

Here, the sign of a, the scaling factor, depends on the optimization type, if it is maximization problem, a will be positive, and vice versa. The combination of this scaling and fitness assignment ensures rapid convergence to the optimal results. Another approach to transforming the objective function is power law scaling, mathematically:

$$F(x) = f(x)^k$$

Here, k is the problem dependent variable and can be changed to control the range of fitness measures if required. Baker [27] suggested a rank based approach to prevent premature convergence, where instead of using raw performance, the rank of individuals in the population is used to measure relative fitness.

2.3.1.3. Selection:

Selection is the determination process of how many times a particular individual will participate in reproduction. It comprises of two processes. In the first process, the raw fitness values are converted into a real-valued expectation of an individual's probability to be chosen for reproduction. The second process, also known as "sampling", selects an individual probabilistically for reproduction based on its fitness relative to other individuals. The performance of the selection algorithm can be determined by using three measures which are bias, spread, and efficiency. The absolute difference between the actual and expected probability of an individual for getting selected is defined as bias. Spread can be defined as the range of the possible number of times in which the

individual may be selected. And the efficiency depends on the GAs overall time complexity. Thus, a selection algorithm will be appropriate with zero bias, minimum spread and a minimum contribution to the GAs time complexity.

Roulette wheel mechanism is one of the popular approaches used in the selection method. In Stochastic Sampling with Replacement (SSR) method, an interval with a range from 0 to "Sum" is used to map the individuals one to one adjacently where "Sum" can be measured as the summation of either the individuals' raw fitness values over all the individuals in that population or individuals' expected selection probabilities. A random number is generated within the interval [0, Sum] and the individual having that random number in its area is selected. This selection process stops when the required number of individuals are selected.

In Stochastic Sampling with Partial Replacement (SSPR), an individual's segment size is reduced by 1.0 for each time it is selected. Another method, Remainder Sampling method, comprises of integral phase and fraction phase. In the first phase, a deterministic approach based on the integer part of individuals' expected trials is used to select the sample. Then the remainders are selected probabilistically based on the fractional part of their expected trials. In the latter phase, roulette wheel mechanism is used. In Remainder Stochastic Sampling without Replacement (RSSWR) after selecting an individual, its fractional part is reduced to zero. Another widely used algorithm is Stochastic Universal Sampling (SUS) with zero bias. It is a single phase method where instead of one selection pointer N pointers are used which are spaced equally by a distance determined by a random number generated in the range [0, Sum/N]. If the generated number is "a", N pointers are equally spaced by one starting from "a" [28].

2.3.1.4. Crossover:

GAs use "Crossover or Reproduction" operator for producing new offspring from the parent population having some parts of both parents' chromosome. Single point crossover is the most common method for binary chromosome where parents exchange their parts of chromosomes after a pre-specified point. Other common crossover methods are multipoint crossover, uniform crossover, shuffle crossover, surrogate crossover, intermediate recombination, etc.

In the multipoint crossover, multiple crossover points are chosen randomly without duplication and sorted in an ascending manner. Then the bits between successive crossover points are exchanged between the two parents and thus new offspring are generated from their parents though the bits between the first allele position and the first crossover point are not exchanged. It may happen that the parts of a chromosome exercising most impact on the performance of a particular individual might not be located adjacently, which makes the multipoint crossover more suitable for various optimization problems.

In uniform crossover, every locus has a potential to be a crossover point. A crossover mask is generated randomly and based on the value of a particular bit of the mask it is decided from which parent bits will be supplied for offspring for that location. The uniform crossover can be parameterized by applying a probability to the swapping of the bits. It can reduce the biasedness towards the length of the chromosome representation by controlling the disruption during the crossover. Another crossover method, shuffle crossover [29], shuffles the bits before performing the recombination at a single cross point and after recombination the bits are unshuffled. In reduced surrogate operator, recombination occurs at only those points where gene values differ [30]. Intermediate recombination and line recombination are two other types of crossover operators.

2.3.1.5. Mutation:

Mutation is a random process which replaces one allele of a gene by another to produce a new gene. In GA, mutation is performed with a very low probability. A mutation operator is mainly used for two purposes. One is to ensure that the probability of searching any particular solution is never zero. At the same time, it may recover the good candidates if they are lost due to selection and crossover operations.

2.3.2. Genetic Algorithm with Multi-objective Optimization:

In real engineering challenges, most of the optimization problems are multi-objective, for example, minimization of cost, maximization of profit, maximization of utilization, etc. There are two basic approaches to handling multi-objective optimization problems. All individual objective functions are combined into a single function in the first approach where a weighted sum method can be used to determine a single objective function. One inherent problem with this is to determine the weightage value precisely and effectively as it affects the optimal result tremendously if changed even a little amount.

Another approach is to determine an entire Pareto optimal solution set comprising sets of solutions having no dominance over each other. This approach is preferred as decision makers are given a set of optimal solutions allowing them to choose by trading-off various parameters. Being a popular metaheuristic approach, GAs are well suited for multi-objective optimization problems and Jones et al. [31], mentioned that 70% of all metaheuristics approaches use evolutionary algorithms as their fundamental basis. Various regions of the solution space are being searched simultaneously resulting in a diverse set of solutions. In most cases, prioritization, weightage or scaling are not required in multi-objective GA which makes it more useful for solving non-convex, discontinuous and multi-modal solutions spaces [32].

David Schaffer [33] proposed the first multi-objective GA in 1980, named Vector evaluated GA (VEGA), with a limitation of having the search direction parallel to the axes of the objective space. Two approaches were suggested to improve VEGA. Following the work of Schaffer, a good number of multi-objective GAs has been developed and suggested by various researchers with variation in framework and operator [1, 19]. A complete list of these popular multi-objective GA approaches with their advantages and disadvantages have been discussed by Konak et.al [32]. Some of them are mentioned here [33-44]: Vector Evaluated GA (VEGA), Vector Optimized Evolution Strategy (VOES), Weight-Based GA (WBGA), Multiple Objective GA (MOGA), Niched Pareto GA (NPGA, NPGA2), Non-dominated Sorting GA (NSGA,NSGA-II), Distance-Based Pareto GA (DPGA),Thermo-dynamical GA (TDGA), Strength Pareto Evolutionary Algorithm (SPEA, SPEA2), Multi-Objective Messy GA (MOMGA-I, II, III), Pareto Archived Evolution Strategy (PAES), Pareto Enveloped Based Selection Algorithm (PESA, PESA-II), and Micro GA-MOEA (μGA, μGA2).

Among all these methods, determining which one is the best-performed technique has become a very common question in the research field of multi-objective optimization. Several test problems have been designed and developed by scientists and researchers and these techniques have been applied to solve them. However, the most representative, discussed and compared evolutionary algorithms are Strength Pareto Evolutionary Algorithm (SPEA, SPEA2), Pareto Archived Evolution Strategy (PAES), Pareto Enveloped Based Selection Algorithm (PESA, PESA-II), and a Non-dominated Sorting GA (NSGA-II). Several comparison studies and numerical simulations using various test cases exhibit NSGA-II and SPEA2 as better MOEA technique than other methods. Even for multi-objective optimization having more than two objectives, SPEA2 seems more advantageous over NSGA-II.

Multi-objective genetic algorithms have been used in a variety of fields including a bi-criteria transportation problem [45], electric power dispatch problem [46], vehicle routing problem with time windows [47], structural design problems [48-55], etc.

2.3.3. Genetic Algorithm in Structural Design Problem:

Being a simple and easily applicable method, GAs have gained popularity in the research field of structural design optimization because of their proficiency to search for a global optimal solution. There are numerous optimization methods available which can be applied to solve structural design problems. But certain characteristics of this kind of optimization have made GAs popular in this research field. GAs are suitable for continuous problems as well as for discrete and non-differentiable problems. Additionally, these methods are very efficient for searching global optimal solutions. Though GAs are competent with optimization problems having continuous and discrete variables [48, 49], in most of the cases the simple GA has been used to solve structural design optimization problems having a discrete design space [50-54]. The modification of the simple GA has been performed to improve the reliability of the performance of the continuous GA where incremental design variables along with a Novel Genetic Algorithm were used [55]. The performance of this method was tested for several optimization problems including structural design problems, but none of the cases was multi-objective. Some other examples show the application of GAs by combining them with other approaches.

2.3.4. Genetic Algorithm for Constrained Optimization Problem:

GAs are structured in a way that they cannot be directly used for the problems with constraint. For solving classical optimization problems with constraints, there are basically two methods which are i) Generic Methods and ii) Specific Methods. Penalty function, Lagrange multiplier, and complex search methods are the example of the Generic Methods which do not interrupt the mathematical

structure of the constraint. Specific Methods are the cutting plane method, the reduced gradient method, and the gradient projection method. These methods are only applicable for the special type of constraint. Straightforwardness and ease of execution have put some advantages to Generic Methods over Specific methods [56].

Coello [57] categorized various constraint handling optimization methods into five groups which are, Penalty Functions, Special Representations and Operators, Repair Algorithms, Separations of Objectives and Constraints, and Hybrid Methods.

The evolutionary constraint handling methods have been classified again by Michalewicz [58] and, then Michalewicz and Schoenauer [2] into five categories: Methods based on preserving feasibility of solutions, methods based on penalty functions, methods making a distinction between feasible and infeasible solutions, methods based on decoders, and hybrid methods.

According to Takahama and Sakai [59] optimization problem with constraint can be handled using four methods which are: penalty functions, methods based on the preference of feasible solutions over infeasible solutions, methods that use constraint violations and objective functions separately, and methods based on multi-objective optimization techniques. In the following section, penalty functions are discussed in more details as in this thesis penalty functions have been used to handle constraints.

2.3.5. Penalty Function:

Among all generic methods, the penalty function method is the most popular one to apply to GAs as genetic algorithms follow generic search methods [7, 55, 60]. Penalty factors, being highly problem oriented, need to be tweaked to manipulate the severity of penalties for the different level of infeasibility [61].

In 1940, Courant [62] first had the idea of a penalty function to penalize each infeasible solution based on their amount of infeasibility ranging from completely rejecting the individual to decreasing its fitness based on the degree of violation. Afterward, the algorithm of the penalty function was enlarged by Carroll [63] and, Fiacco and McCormick [64]. In classical optimization, there are two types of penalty functions which are the interior penalty function and exterior penalty function. The interior penalty function performs better for a single constraint as for multiple constraints execution of the interior penalty function is more complex. It penalizes feasible solutions so that an optimal solution is obtained between the boundary of feasible and infeasible solutions. On the other hand, by penalizing infeasible solutions, the exterior penalty function starts with an infeasible solution and moves towards the feasible region. This penalty function has three degrees which are (1) barrier methods considering only the feasible solution (2) partial penalty functions where the penalty is applied to the solutions near to the feasibility boundary, and (3) global penalty functions effective to the entire infeasible region [65, 66]. The general formulation of the exterior penalty function can be expressed as follows:

$$\varphi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{n} r_i \times G_i + \sum_{j=1}^{p} c_j \times L_j\right]$$

where $\varphi(\vec{x})$ is the new objective function, $f(\vec{x})$ is the main objective function to be optimized, G_i and L_j are functions of the constraints $g_i(\vec{x})$ and $h_j(\vec{x})$, respectively and r_i and c_j are penalty factors. G_i and L_j are normally formed as follows:

$$G_i = max[0, g_1(\vec{x})]^{\beta}$$

$$L_j = \left|h_j(\vec{x})\right|^{\gamma}$$

where, β and γ are normally 1 or 2.

Schoenauer and Xanthakis [67] showed that the penalty function is the best and easiest solution technique for the larger feasible region and smooth problems. The penalty function is not only used in Genetic Algorithms, but it was also proved by Golalikhani, Ali and Zhuang [68] that static and dynamic penalty function could be used and perform efficiently for solving constrained optimization problems with the Electromagnetism-like method (EM).

Because of its appropriateness in various optimization problems with the constraint, researchers have worked in this particular area for a long time and formulated various penalty functions such as static, multi-level [69], dynamic [70], adaptive, co-evolved, fuzzy-adapted, etc. [71].

2.3.5.1. Static Method:

In a simple static method, a constant penalty is applied to all infeasible solutions. Thus, the objective function is a combination of the un-penalized objective function and penalty for violating feasibility. It remains constant during the entire process. Afterward, instead of using the constant penalty, a function of a number of constraint violations was proposed to use as the penalty function. Thus an individual is evaluated using the following equation:

$$\text{Fitness } (\vec{x}) = f(\vec{x}) + \sum_{i=1}^{m} (R_{k,i} \times max[0, g_i(\vec{x})]^2)$$

where, $R_{k,i}$ are the penalty coefficients, the number of constraints violation is m, $f(\vec{x})$ is the main objective function which is to be penalized, l is the level of constraint violation. Here, not only number of violations is considered, the level of violation is also evaluated.

Later Richardson et al. [60] introduced another idea to penalize infeasible solutions based on the distance from the feasibility. The distance metric is another effective approach to applying the penalty function which can be continuous, discrete, linear or non-linear. Penalties that are functions of the distance from feasibility perform better than those that are merely functions of the number of

violated constraints. If a problem has few constraints and few feasible solutions, penalties which are solely functions of the number of violated constraints are not likely to find solutions. The success of the static penalty method depends on the proper penalty coefficients chosen for constraints

2.3.5.2. Dynamic Method:

In this method, the penalty function increases with time thus severity of the penalty increases with the progression toward the optimum solution. Highly infeasible solutions can be considered at initial generation, but gradually it converges to the feasible solution. In an approach proposed by Joines and Houck [70], individuals are evaluated using the following equation:

$$\text{Fitness } (\vec{x}) = f(\vec{x}) + (C \times t)^{\alpha} \times SVC(\beta, \vec{x})$$

Where C, α and β are constants defined by the user and $SVC(\beta, \vec{x})$ is defined as:

$$SVC(\beta, \vec{x}) = \sum_{i=1}^{n} {D_i}^{\beta}(\vec{x}) + \sum_{j=1}^{p} D_j(\vec{x})$$

$$\text{And } D_i(\vec{x}) = \begin{cases} 0, & g_i(\vec{x}) \leq 0, \\ |g_i(\vec{x})|, & otherwise \end{cases} \quad 1 \leq i \leq n$$

$$D_j(\vec{x}) = \begin{cases} 0, & -\epsilon \leq h_j(\vec{x}) \leq \epsilon, \\ |h_j(\vec{x})|, & otherwise, \end{cases} \quad 1 \leq j \leq p$$

A dynamic penalty function requires fine tuning of many parameters to solve a problem efficiently otherwise it may result in an infeasible solution [72].

2.3.5.3. Death Method:

The most simple penalty function is known as "death penalty" and may be used for the convex solution space. Under this scheme, the infeasible solutions are assigned the worst possible fitness values or are simply eliminated from the optimization process [10, 73]. As the infeasible solutions

are not considered for the selection process for the next generation, if the initial population does not contain any feasible solutions the whole population is rejected and a new generation is generated again [56].

2.3.5.4. Adaptive Method:

An adaptive penalty function changes the value of penalty based on the feedback from the search progress [56]. In this penalty function, each individual is evaluated by using the following equation,

$$\text{Fitness } (\vec{x}) = f(\vec{x}) + \lambda(t)\left[\sum_{i=1}^{n} g_i^2(\vec{x}) + \sum_{j=1}^{p} \left|h_j(\vec{x})\right|\right]$$

where $\lambda(t)$is updated at every generation t using the following way,

$$\lambda(t+1) = \begin{cases} \left(\frac{1}{\beta_1}\right).\lambda(t) & if\ case\ 1 \\ \beta_2.\lambda(t) & if\ case\ 2 \\ \lambda(t) & if\ case\ 3 \end{cases}$$

Here, the best individual in the last k generations is always feasible in case 1 and in case 2 the best individual is never feasible. If there are some feasible and infeasible individuals that stand in the best position in the population, the penalty is not changed.

Later, the severity of the penalty was designed to modify dynamically according to the fitness of the best solution obtained till that progression [56]. Crossley and Williams showed that the best approach for the adaptive penalty function is to apply it based on the corresponding specific problem [74]. Birak Girma [75] tried to solve some drawbacks of adaptive penalty function by proposing a more reliable, free of any parameter tuning and easily implementable method.

Various penalty function methods have various advantages and disadvantages. Michalewicz [57] discussed demerits of each method and compared the performance of these algorithms on a number

of test problems with a conclusion that the static penalty function method (without any sophistication) is a more robust approach than the sophisticated methods.

While using the penalty function, it is very important to define the relationship between infeasible solution and feasible solution area as this is the basis of the value of the penalty factor. An individual solution can be penalized in a different way such as [58]:

1. It can be penalized irrespective of how much infeasible it is. That means if it is infeasible it will be penalized.
2. It can be penalized based on the amount of infeasibility which creates a proportional relationship between severity of penalty and amount of infeasibility.
3. An effort can be made to make that infeasible individual a feasible one.

If the degree of parameters is tuned according to the problem, the obtained result will be more satisfactory.

2.3.5.5. Exact Absolute Value & Augmented Lagrangian Penalty Methods:

Normal penalty functions consider infinitely large penalty values to limit the optimal solution in the feasible region which can cause numerical difficulties and other side effects on the optimization process. To obtain a reasonable finite value for the penalty parameter, "Exact absolute value penalty method" has been introduced. In "Augmented Lagrangian Penalty Methods", the penalty function enjoys the property of being differentiable and is also known as a multiplier penalty function. These approaches were first introduced for problems with equality constraints. Later, their scope was extended for inequality constraints also. As in this approach, the ordinary Lagrangian function is augmented by a quadratic penalty term; it is called "Augmented Lagrangian Penalty Function" [76].

3. Methodology:

3.1. Developing an Indeterminate Structure with an Unconstrained Solution Space:

The objective of the work was to formulate Genetic Algorithms based multi-objective optimization methodology for solving a support location problem of an indeterminate structure. To reach that goal, the first step was to develop a generic and simplified indeterminate structure. The support locations of that structure were determined by satisfying multiple objectives. These multiple objectives were not apparent to solve and posed competing nature. Having contending multiple objectives means the optimal value of one objective might negatively impact the optimality of other objectives. The balancing of all objectives in a proper way was a prerequisite to lead towards the acceptable optimal results. The structure required to be designed in such a way that the heuristic calculation could provide a benchmark to evaluate the optimal objective solution.

3.1.1. Test Case:

An 8x6 meter rectangular shaped solid indeterminate plate was considered as a simple and generic test case in this work. A myriad of similar examples would be found in real life including decorative overhung lights. The target was to overhang this structure, made out of aluminum, with three cables in a way that the load is evenly distributed among the cables while ensuring maximum stability. So, the objectives included the minimization of the tension difference in the supports and the maximization of the area enclosed by the support locations to increase stability. It was assumed that the support locations in the geometric coordinate system were (x_1, x_4), (x_2, x_5) and (x_3, x_6), and the reactive forces acting upon the supports were R_1, R_2 and R_3, respectively. These reactive forces and the force due to the self-weight of the plate (F) were assumed to be acting in the z- direction

(Though in figure 3.1, due to 2D drawing it seems that F is working along y axis). The indeterminate structure is shown in Figure 3.1.

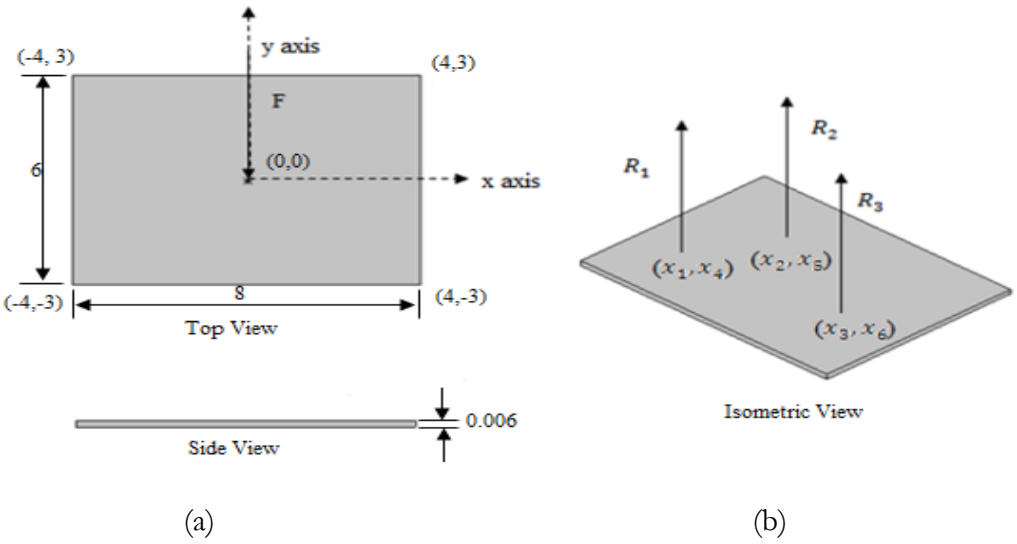
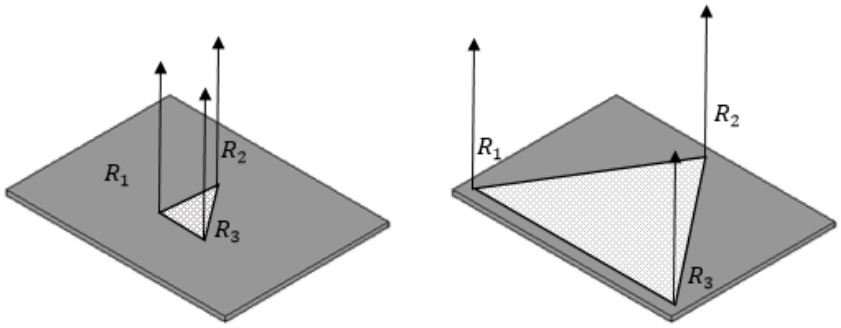


Figure 3.1: (a) Top view and (b) Isometric view of the indeterminate plate

3.1.2. Conflicting Objectives:

The two objectives mentioned in section 3.1.1 were conflicting in a sense that the minimum tension differences in the supports might end up with a much smaller enclosed area (Figure 3.2(a)). Similarly, the enclosed area could have a very good value while the reactive forces were not evenly distributed (Figure 3.2(b)). One of the challenges faced while formulating the problem was to satisfy both objectives in a balanced way to reach optimality. Figure 3.2 shows this contradictory behavior of the objectives.



(a) (b)

Figure 3.2: (a) Balanced reactive forces with smaller enclosed area and (b) Bigger enclosed area with unbalanced reactive forces

3.2. Developing the Genetic Algorithm Based Approach to Solve the Support Locations:

After developing the test case, the next step was to develop the GA based optimization methodology. For solving the optimal support locations of the test case, a code was developed using GA. At that phase, the objective function, GA parameters, and various GA operators were determined. While generating the objective function, the most critical challenge was to develop one single objective function from multiple objectives. If the impact of each component of the objective function on the optimal value was not treated properly, it could put more focus on a single component and thus hamper the optimality of other elements. The primary challenge for this multi-objective optimization was to ensure the simultaneous convergence of all elements. Taking that into consideration, the objective function was developed.

GA operators, such as population initialization, selection method, crossover, and mutation were selected carefully based the requirement of the problem.

GA parameters such as the number of generations, population size, probability of crossover and mutation are very much important to obtain good optimal results. There is no particular standard value for these parameters which can be used for any arbitrary optimization problem. As the GA parameters are problem dependent, several runs were performed varying GA parameters and the combination for which the simple GA converged to the optimal solution with more efficiency and fastness than others was selected as the suitable one.

3.2.1. GA based Methodology:

Genetic algorithm toolbox of Scilab 5.5.0 was used to write code to obtain the optimal support locations for this multi-objective problem.

3.2.1.1. Design Variable:

At the very beginning of the development of a GA-based methodology, design variables were needed to be set. Here, the design variables consisted of the abscissa and the ordinate of each support location. Thus, the chromosome comprised of real-valued genes of abscissa and ordinates of the support locations and it was represented as follows:

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6\} \quad \text{.....(i)}$$

where, first three components (x_1, x_2, x_3) represented the abscissa and rest of them (x_4, x_5, x_6) were for the ordinates.

3.2.1.2. Objective Function:

This structural optimization problem was to be formulated by satisfying all implicit and explicit constraints. The equations of static equilibrium supplied the implicit constraints which were:

$$\sum \vec{F} = 0 \text{ and } \sum \overrightarrow{M_o} = \sum\left(\vec{r} \times \vec{F}\right) = 0 \text{.....(ii)}$$

where, $\vec{F}$ was the force, $\overrightarrow{M_o}$ was the moment with respect to the origin and $\vec{r}$ was the position vector between the origin and the point where the force was acting. In 3D space, these two vector equations contributed six scalar constraint equations.

Now, for the three supports, the reaction forces were represented by a vector, $\vec{R}$ where, $\vec{R}=[\vec{R}_1, \vec{R}_2, \vec{R}_3]$ and the total force acting downward was $\vec{F}$. It was assumed that the only force acting downward was the self-weight of the structure, F. The length and width of the rectangular structure were L and W respectively. As the plate was homogeneous, the co-ordinate of the center of gravity of this plate was (L/2, W/2). In this study, the center of gravity was assumed to be the origin. Supports as well as the weight of the plate were assumed to be acting in the z-direction.

Thus, equations formulated using static equilibrium were as follows:

$$\sum_{i=1}^{3} \vec{R}_i = F \text{ .....(iii)}$$

$$\sum \vec{M}_x = 0, \text{ or } \sum_{i=1}^{3} x_i \times \vec{R}_i = 0 \text{.....(iv)}$$

$$\text{and } \sum \vec{M}_y = 0, \text{ or } \sum_{i=1}^{3} x_{i+3} \times \vec{R}_i = 0 \text{.....(v)}$$

The matrix representation of these three equations was used to solve reaction forces acting on three supports, which is shown below:

$$\vec{R} = A^{-1}\vec{F}, \text{ where, , } A = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \end{bmatrix}, \vec{R} = \begin{bmatrix} \vec{R}_1 & \vec{R}_2 & \vec{R}_3 \end{bmatrix} \text{ and } \vec{F} = \begin{bmatrix} F \\ 0 \\ 0 \end{bmatrix}$$

The objective function of this problem was written as:

$$\min Z = (\, max\, \{ R_i \;\; \forall \; i \in \; \varphi \} - min\, \{ R_i \;\; \forall \; i \in \; \varphi \}) + (\Delta_{max} - \Delta_{current}) \ldots\ldots\ldots\ldots\ldots\ldots \text{(vi)}$$

where, φ was the set of the supports and expressed as {1,2,3}, $\Delta_{current}$ was the area generated by the three support locations and Δ_{max} was the maximum possible area. Different units, huge numerical differences, and disparate requirements of the two components of the objective function made it quite difficult to assign preference weightage to the components of the objective function. Therefore a no-preference strategy [77] defining global criteria by semi-norm mapping [78] of the functions was chosen.

It was assumed that when $R_i = \bar{F} \; \forall \; i \in \varphi \; \exists \; \Delta_{max}$, the upper bound of the enclosed area by any subset of supports and a reasonable value of Δ_{max} can be heuristically determined. If F was equally distributed among the three supports, $R_1 = R_2 = R_3 = \bar{F} = F/3$. Now, using these values in equation (iii), (iv) and (v), it was expressed as:

$$x_1 + x_2 + x_3 = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(vii)}$$

$$\text{and } x_4 + x_5 + x_6 = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(viii)}$$

Using these two conditions, the heuristically calculated maximum possible area (ΔPQR) enclosed by the supports with equal reaction forces was 18 m^2 (Figure 3.3 b), though the maximum possible area was 24 m^2.

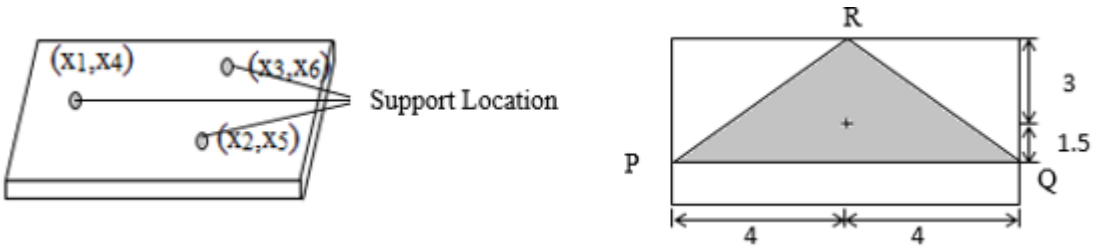


Figure 3.3: (a) Location of three supports which generates an area on the surface (b) heuristically calculated maximum possible area with equally loaded supports

But, it may happen that GAs converge to an optimal solution with unbalanced reaction forces where $\Delta_{current}$ is bigger than 18 m^2 .Then, the overall impact on the objective function gets worst. To avoid that problem, a slightly bigger value than 18 m^2, 20 m^2, was used. Thus, the objective function became,

$$Z = (max\ \{R_i\ \forall\ i \in\ \varphi\} - min\ \{R_i\ \forall\ i \in\ \varphi\}) + (20 - \Delta_{current}) \quad \text{(ix)}$$

3.2.1.3. Continuous Method:

To search for the optimal solution while minimizing the difference between the reactive forces and maximizing the stability of the structure, the "Continuous Method", simple GA was selected. In this method, the GA searched optimal solutions in a continuous solution space considering each and every location in the solution space as a candidate. Here, explicit constraints, the boundary conditions, were the upper and lower limit of the design variables. As the center of gravity was taken as the origin, the boundary constraints for the design variables of the continuous solution space could be expressed as:

$$-L/2 < x_i < L/2 \text{ and } -W/2 < x_{i+n} < W/2 \quad \text{(x)}$$

3.2.1.3.1. GA Operator:

In the "Continuous Method", the initial populations were generated by satisfying the boundary constraints. After generating the initial population, the fitness of each individual was evaluated using the objective function and the required number of individuals were selected for crossover and mutation to breed the next generation.

The stochastic acceptance with elitist selection method was used to select the population for the next generation. In this process, a small portion of the fittest individuals was chosen to pass to the next generation without any crossover and mutation. Sometimes, it might happen that the best candidates are lost due to the crossover and mutation operations which may result in a less fit new generation than the parents. GA may regain those individuals later, but it may take more time to converge. The elitist selection method has overcome this problem, and sometimes it exhibits patent impact on the performance of the GA as it avoids the lost time required to regain the lost good individuals.

To perform the crossover operation, a random number m was generated using a uniform distribution within the range of (0, 1) which could be expressed as, $m \in U(0,1)$. Now, if the individuals selected for crossover were X_i and X_j , where i, j could take any value in the range of [1, number of population], individuals obtained after crossover were:

$$Child_1 = (m \times X_i) + \big((1 - m) \times X_j\big) \quad \text{.....(xi)}$$

and $Child_2 = ((1 - m) \times X_i) + \big(m \times X_j\big)$.....(xii)

The mutation for the continuous variable was performed by changing an individual by a very small amount using a random number, *p*. This random number was generated using uniform distribution within a range of [0, 1]. The mutation was performed on an individual with a defined probability.

After that, the fitness of the generated population was evaluated and, using the elitist selection method, parents were selected to produce next generation. These steps were repeated until the stop criteria were satisfied.

3.2.1.3.2. GA Parameter:

After selecting the operators, the next step was to determine suitable GA parameters. The generated objective function was evaluated to obtain the optimal result using the following control parameters (Table 3. 1):

Table 3.1: GA parameter

Parameter	Value	Parameter	Value
Population size	100	**Crossover probability**	0.7
Generation number	40	**Mutation probability**	0.1

These GA parameters were determined based on some preliminary observations which exhibited that this specific combination provides better optimal values than other combinations.

3.3. Introducing Physical Discontinuity to the Structure and Applying Penalty Functions:

The next step was to introduce physical discontinuity to the structure which converted the multi-objectives unconstrained optimization problem into a constrained one. Several structural design optimization problems can be mentioned where the solution space is not continuous, and thus the problems become more interesting.

In reality, objects, similar to the test case, have lots of discontinuities in the form of fixtures, holes, etc. which makes the solution space discontinuous. That is why, to make the test case more realistic and complex, physical discontinuities were added to it. For simplicity, it was assumed that the shape of the discontinuities was circular, and three circular holes with 1 m, 0.5 m, and 0.75 m radius respectively were considered for the further inspection. Similar to the previous study, the center of mass was considered as the origin.

When the solution space for an optimization problem is not continuous, it becomes a constrained problem. GAs have been developed in a way that they cannot be directly used for constrained problems. As per the discussion made in section 3.3, three different penalty functions were used to make GAs suitable for the modified constrained test case in this work. As the solution space contained three discontinuous areas, it was important to make sure that the optimum results were not in those areas. So, the flat, linear & non-linear penalty functions were used to ensure the feasibility of the results. If the location of any support was in those discontinuous spaces, penalty function would add some penalty value to the objective function. The structure used in this case is shown in the Figure 3.4,

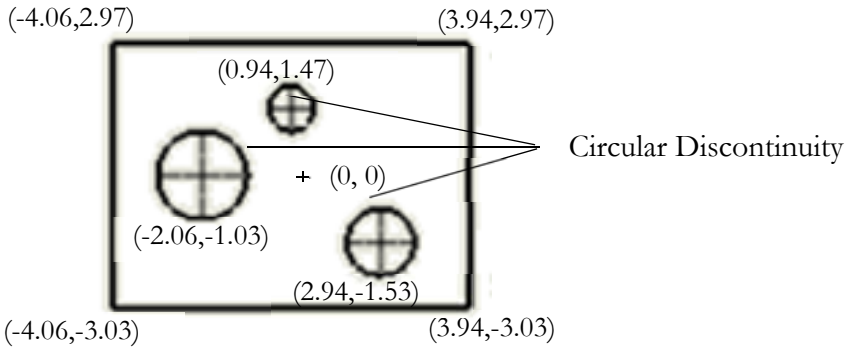


Figure 3.4: Modified test case with circular discontinuities

The application of GA in a constrained optimization problem requires the handling of various parameters in a proper way. In this work, to maintain the feasibility of the subsequent generations, an exterior penalty function was applied to the objective function evaluation routine. This is one of the most commonly used penalty functions in GA based optimization because it has no restriction to start the searching process with an initial feasible solution. The exploration begins with an infeasible solution and gradually moves toward a better feasible solution. In this work, three different types of penalty functions were used which are (i) Flat, (ii) Linear and (iii) Non-Linear penalty functions.

3.3.1. Flat Penalty:

In this method, if there is any infeasible solution, the objective function will be penalized with a fixed amount for each infeasible solution. This type of penalty function only considers the presence of the infeasibility, not the intensity or distance of the infeasible solution from the feasible zone. In the present investigation, as there were q discontinuities, the feasibility of each location was checked on each of these restricted areas. For each support location, a fixed penalty was added if there was any infeasibility. Otherwise, the assigned penalty remained zero. So, the flat penalty function of i^{th} support location for j^{th} discontinuous area can be expressed as follows:

$$P_{i,j} = 0, \text{ if the solution is feasible}$$

$$1, \text{ if the solution is infeasible}$$

Then adding penalties for all support locations, total penalty is obtained which is:

$$p = \sum_{i=1}^{n} \sum_{j=1}^{q} P_{i,j} \quad \text{(xiii)}$$

So, if there are n supports, the total penalty of a population can take any value between 0 and n as in the best situation all of the support locations will be in the feasible area and in the worst case all of them will take place in the infeasible region.

3.3.2. Linear Penalty:

Linear penalty function not only considers the presence of infeasibility but also linearly increases the penalty value with the increase in the distance of the infeasible solution from the boundary between the feasible and the infeasible region.

This penalty function can be expressed as,

$$P_{i,j} = max(0, e_{i,j}) \quad \text{(xiv)}$$

where, $P_{i,j}$ is the penalty of i^{th} support location for j^{th} discontinuous area and $e_{i,j}$ can be expressed as the intensity of the infeasibility of that specific support location. At first, each support location was tested for each discretized area, and the corresponding penalty value was assigned. After that, the total penalty value was calculated using equation (xiii).

3.3.3. Non-Linear Penalty:

The third penalty function used was a non-linear penalty function which penalizes the infeasible solution by maintaining a non-linear relationship with the distance of the infeasible solution from the feasible region boundary. Here, the penalty function can generally be expressed as follow:

$$P_{i,j} = max(0, e_{i,j})^2 \quad \text{(xv)}$$

All the nomenclatures of this equation are the same as for the linear penalty function. The only difference is that the amount of penalty will increase in a non-linear fashion with the intensity of the infeasibility. Similar to the linear penalty, the total penalty value for all the support locations was

calculated using equation (xiii). The performance of all penalty functions was evaluated which provided a clear indication that whether making the penalty function more complicated provided any better result or not. After doing that, the effect of the size of the discontinuous spaces on the performance of these penalty functions was observed.

The three penalty functions were added to objective function separately. The objective function was modified as follows:

$$Z = (max\{R_i \ \forall \ i \in \varphi\} - min\{R_i \ \forall \ i \in \varphi \ \})/2241 + (20 - \Delta_{current}) \ / \ 20 + (\frac{P}{3}) \quad \text{.....(xvi)}$$

Here, P is the total penalty calculated for all support locations. In the worst case, all the supports are in infeasible regions. So the maximum possible penalty value is 3, thus the total penalty was normalized by dividing it by 3. The modified algorithm was run for 30 trials.

3.4. Developing a Coding Algorithm to Handle Constraints in Genetic Algorithms

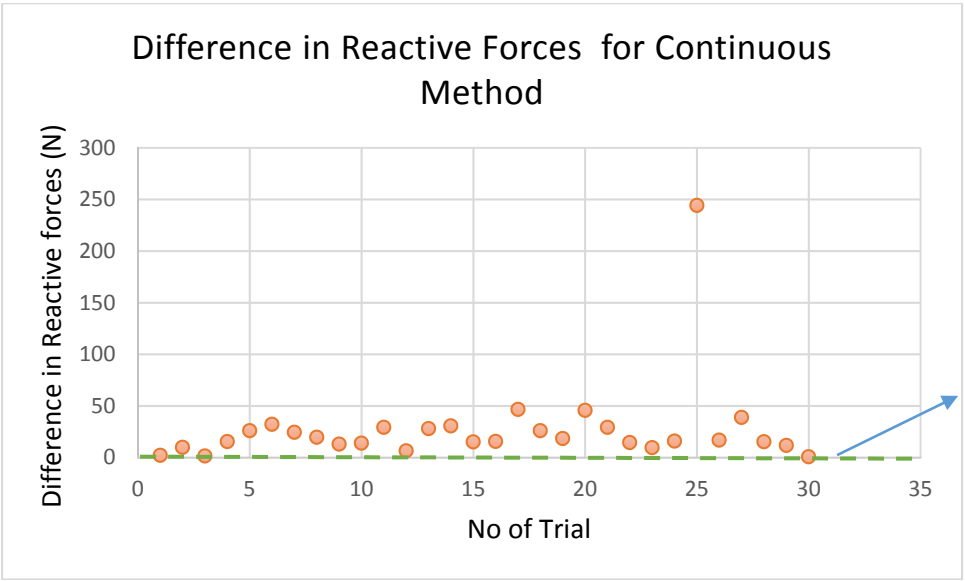
The last step of this research was to develop a coding algorithm to apply GA in the constrained optimization problem. Various methods including penalty functions require the proper selection of relevant parameters which makes the overall process time consuming and complex. A standard, generic and simple methodology was developed which eliminated these time-consuming steps and directly applied GA for a class of structural design optimization problem.

4. Results & Discussion:

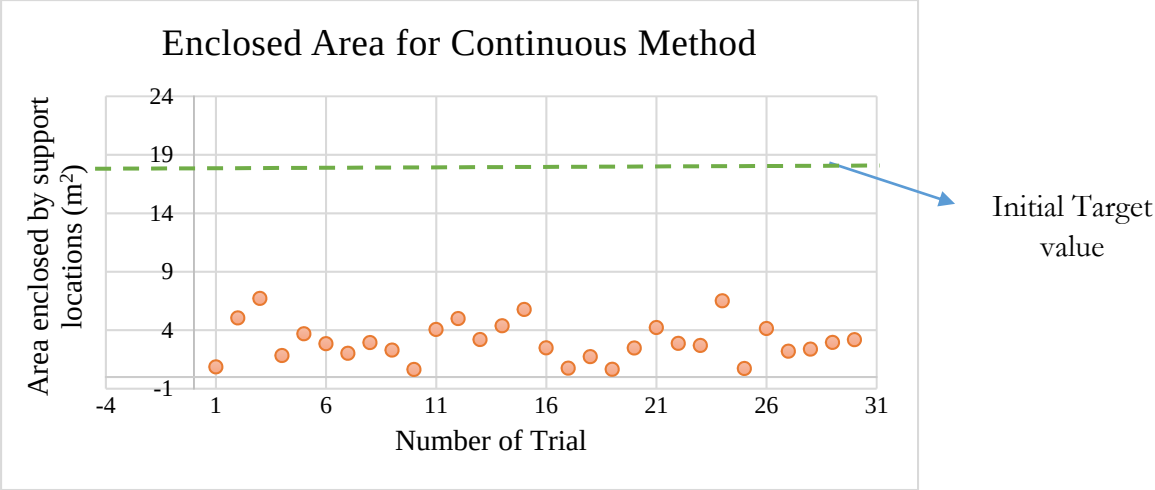
4.1. Unconstrained Problem:

4.1.1. Implementation of the Continuous Method:

After performing the test several times, it was found out that the multi-objectives nature of the objective function was preventing it from satisfying both objectives. The reactive forces had very big numeric value compared to that of the generated area. Thus, the difference among reactive forces had more impact on the overall objective function than the area. So, the GA focused more on to the balance of the reactive forces. GA was designed in a way that it eliminated the negative forces as they increased the value of the objective function. That is why the optimal results obtained using the objective function (ix) provided almost balanced reactive forces, but the area values were poor (Figure 4.1).



(a)



(b)

Figure 4.1: (a) Differences among reactive forces and (b) enclosed area by support locations for the "Continuous Method."

In Figure 4.1 (a), it was observed that the reactive forces were almost equally balanced in most of the cases. But at the same time, the area generated by the support locations were very small, less than half of the target value, 20 m^2.

4.1.2. Normalized Objective Function:

Thus, one of the critical challenges for this multi-objective optimization problem became to handle all components of the objective function in a proper way to guide the GA towards the optimal solution.

To address that problem, a heuristic based normalization technique was used in such a way that the contributions of each component to the objective function remained in the range of 0 to 1. Therefore, $Z \in (0, 2)$. It should be noted that Z can assume a negative value if Δ_{max} is grossly underestimated by the heuristics. On the other hand, the worst case scenario of Z may be lower than 2 if Δ_{max} is overestimated.

Heuristically, it was assumed that the maximum tension difference can be equal to the difference between the average reaction force and zero (it was ensured that no reaction force could take a negative value which results in a minimum reaction equal to zero). The force, F, due to the self-weight of the plate was assumed to be approximately 6726 N. Thus, for normalizing the force component, it was divided by 2242 which is one-third of the total force.

It is already mentioned that the maximum possible area generated by support locations was assumed to be 20 m^2. Thus, the objective function became,

$$Z = (max\{R_i \ \forall \ i \in \varphi\} - min\{R_i \ \forall \ i \in \varphi\})/2241 + (20 - \Delta_{current}) \ / \ 20 \quad \text{.....(xvii)}$$

This normalized equation was used to solve the support locations, and the obtained results showed significant improvement. But, still the results were not consistent. For 30 runs, the optimal values obtained by using continuous GA along with normalized objective function is shown in Figure 4.2,

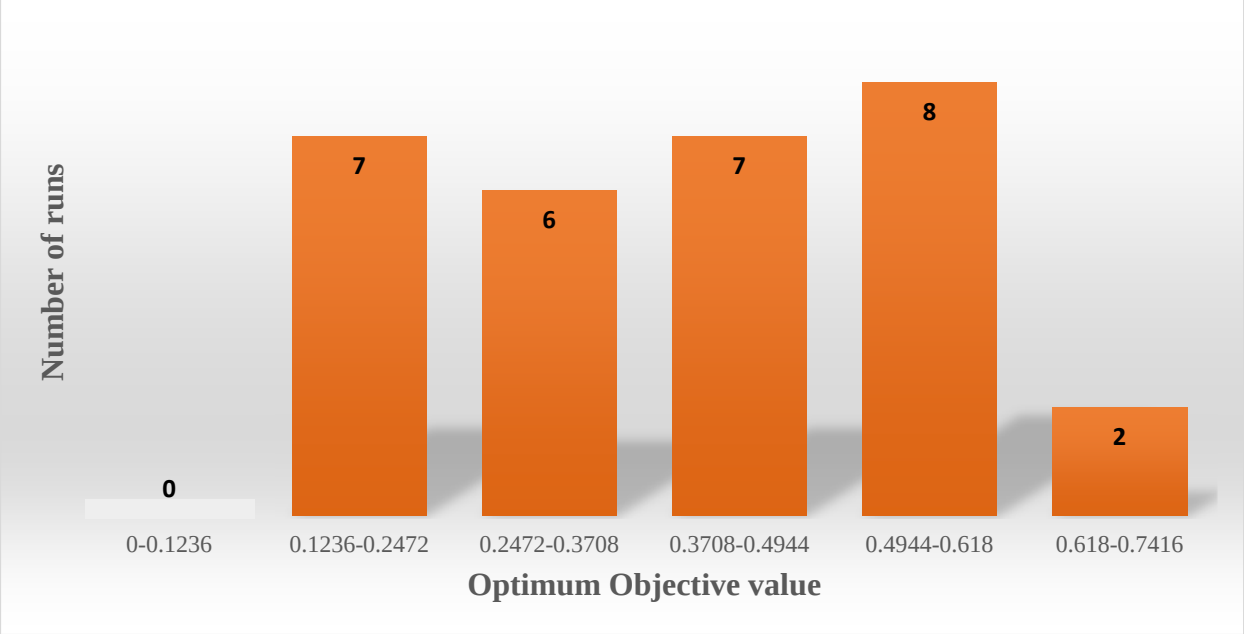


Figure 4.2: Optimum objective value obtained by using "Continuous Method" from 30 run

It is apparent from Figure 4.2 that the optimal results were scattered with a wide range having few good results. In the continuous GA method, each and every possible position can be a candidate for the optimal solution. Thus the size of the search space is huge and sometimes the continuous GA is successfully getting closer to the optimal results and sometimes not. One way to delimit this problem was to reduce the search space while ensuring that the optimality of the results was not impeded. This observation led to the development of the proposed method, named "Discretized Method," which is discussed in the next section.

4.1.3. Discretized Method:

The discretized method is a simple modification of the conventional continuous method where the continuous search space is assumed to be a sum of a number of very small horizontal and vertical strips. The intersections of these strips are called nodal points which are the possible options for the support locations instead of the whole search space. Thus, in the continuous method, support locations can take any place on the structure. But in the discretized method, only the nodal points are the possible values of support location. Figure 4.3 illustrates the conceptual difference between the search space of the continuous method and the discretized method.

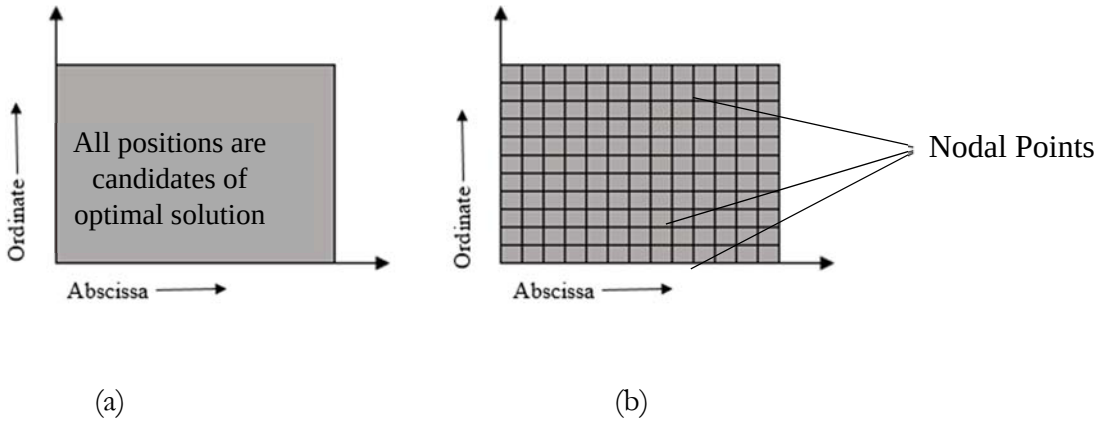


Figure 4.3: Solution space for (a) continuous method and (b) discretized method

The nodal points were generated in a matrix form using an increment of δ between the lower and upper bound of each variable. Thus, each nodal point had a corresponding position number, called the index number, in its matrix. In "Discretized Method", index numbers were used as design variables (genotypes) and, then corresponding nodal locations were determined from the matrix. Here, chromosomes were made of index numbers which indicated the abscissa and the ordinate of the support locations, and thus the GA operated on the coded chromosomes which were decoded later to evaluate the objective function.

The matrices were developed for abscissa and ordinate of three support locations with an increment of δ=0.05 within the range of $(-L/2, L/2)$ and $(-W/2, W/2)$ respectively.

Where,

$$M = \{abscissa\ of\ support\ generated\ by\ the\ increment\ of\ \delta\},$$

$$\mathbb{N}_M = size\ of\ M,$$

$$N = \{ordinate\ of\ support\ generated\ by\ the\ increment\ of\ \delta\}$$

and $\mathbb{N}_N = size\ of\ N$.

Thus, to represent the abscissa and the ordinate of a support location the chromosome comprised two index numbers and in total 2*3= 6 components for the three supports. The chromosome can be represented as follows:

$$X = \{l_1, l_2, l_3, l_4, l_5, l_6\}$$

I_i is the set of index numbers indicating the abscissa and the ordinate of the location of the i^{th} support on the surface of the structure according to the coordinate scheme. So,I_i can be written as:

$$I_i = \{l_i, l_{i+n}\}, \text{ where, } 1 < l_i < \mathrm{N}_M \text{ and } 1 < l_{i+n} < \mathrm{N}_N$$

For the evaluation of the objective function, decoding of the design variables was done by finding the corresponding abscissa and ordinate from the generated matrices. For example, if the abscissa and ordinate of the i^{th} support is x_i and x_{i+n}, then,

$$x_i = L(I_i(1)) \text{ and } x_{i+n} = W(I_i(2))$$

4.1.3.1. Objective Function:

The normalized objective function was used for the "Discretized Method".

4.1.3.2. GA Operators for "Discretized Method":

For the breeding of the initial population, random numbers were generated for each design variables satisfying the boundary conditions and the integers of those generated numbers were considered for the rest of the calculation.

After generating the initial population, the binary conversion was performed using an 8-bit representation for each variable. Thus the chromosome became a string of 48 bits representing 6 design variables.

The fitness of each individual was evaluated using the objective function. Before evaluating the objective function, the support locations were searched from M and N matrix using index numbers generated by GA. Similar to the "Continuous Method", the elitist selection was used to select parents for crossover and mutation operation in the "Discretized Method".

Single point binary crossover was performed on the selected parents to breed a new child. The parents were X_i and X_j where i and j indicate the population number and $Child_1$ and $Child_2$ were offspring generated after crossover. Each parent is split into two parts at a point. X_i was splitted into

H_1 and L_1, which were the head and the tail of the parent. Similarly X_j produced H_2 and L_2. So, the offspring generated from the crossover can be represented as:

$$Child_1 = [H_1 + L_2] \text{ and } Child_2 = [H_2 + L_1]$$

After the crossover was complete, the binary mutation was performed with a predefined probability. If an individual was selected for the mutation, a random number was generated to obtain the position of a bit to be mutated, and that bit value was flipped.

After performing these operations, the fitness of the objective functions was evaluated and based on that the new population was selected to pass for breeding the next generation. The same process was repeated until the stop criteria were met.

4.1.3.3. GA Parameter:

The parameters used in the "Continuous Method" were also selected for the "Discretized Method".

4.1.4. Implementation of the Discretized Method:

Using "Discretized Method," the support locations were solved for 30 trials, and the results showed that the application of the proposed method dramatically improved the performance of the GA which is illustrated in Figure 4.4.

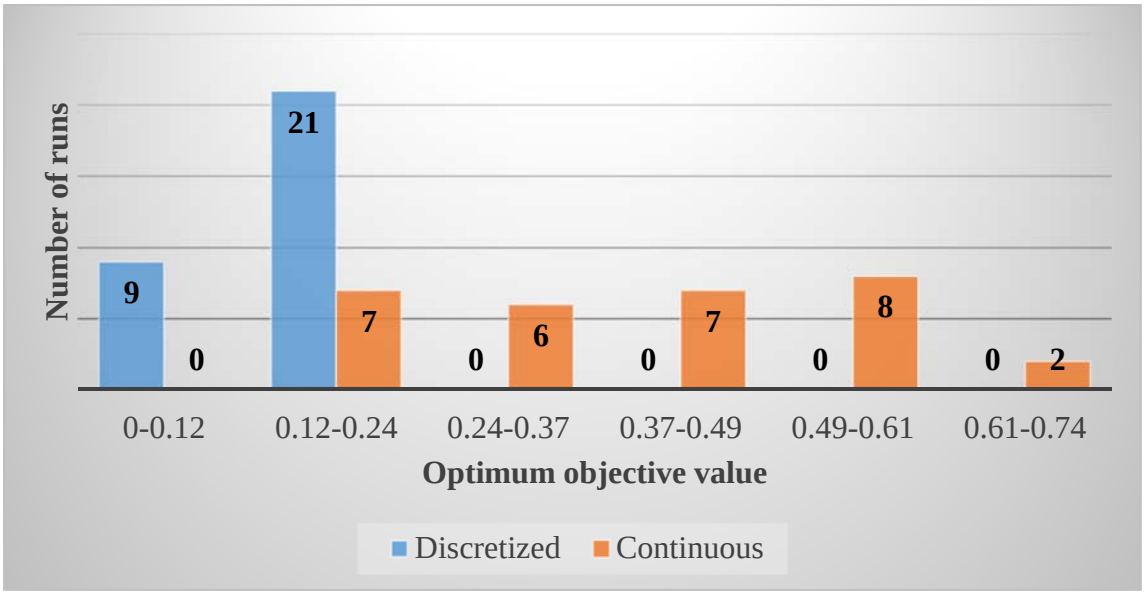


Figure 4.4: Optimum objective value obtained by using two methods

Figure 4.4 shows that discretized method provided optimal results with a more narrowerer range than the continuous method, which might imply that the proposed method was more reliable. The average and standard deviation of the objective function obtained from the 30 trials of the discretized method were significantly smaller than the continuous method, which is tabulated in Table 4.1.

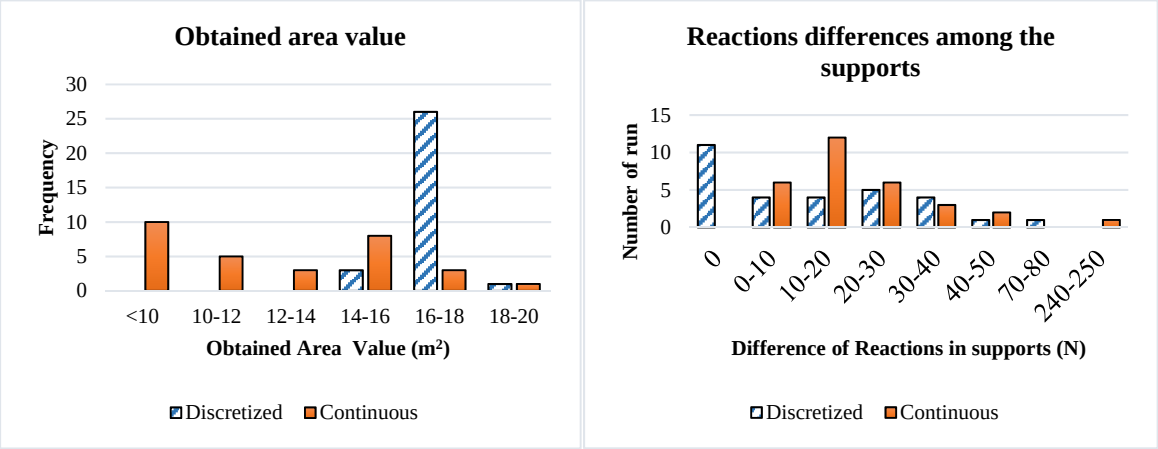
Table 4.1: Objective function obtained using Continuous and Discretized Method

Parameter	Method	Minimum	Average	Maximum	STD. DEV
Objective Function	Continuous	0.17	0.405	0.723	0.162
	Discretized	0.105	0.147	0.218	0.034

These two approaches were also compared based on the two components of the objective function separately. Table 4.2 shows area values, and the difference of maximum and minimum reaction values for both approaches and Figure 4.5 shows the distribution of the objective function components.

Table 4.2: Area and reaction difference obtained from two Methods

Parameter	Method	Minimum	Average	Maximum	ST. DEV
Area (m^2)	Continuous	5.55	12.14	18.41	3.382
	Discretized	15.64	17.19	18.06	0.751
Reaction Difference (N)	Continuous	0.91	27.32	244.29	42.643
	Discretized	0	15.28	77.080	18.023



(a) (b)

Figure 4.5: (a) Distribution of optimal area value and (b) reaction differences by two methods

It is evident from Table 4.2 and Figure 4.5 that the discretized method consistently produced support layouts that enclosed a higher area while equalizing the force distribution among the supports. It was observed that for both the components of the objective function, the standard deviations of the discretized method were significantly smaller than that of the continuous method. It implies that the discretized method is more reliable and stable than the continuous one. It should be noted that, out of 30 samples, 27 samples of the discretized method provided an area value bigger than 16 m^2, whereas 26 samples of the continuous method generated an area value smaller than 16 m^2. Although the continuous method balanced the reaction forces among three supports almost equally, it is worth mentioning that the discretized method provided exactly equal reactions for 11 times out of 30 samples while the continuous method never did. Also, the narrow ranges for both the area value and the reaction difference values bolster the observation that the discretized method worked better with more consistency than the continuous method for this optimization problem.

Generally, it is expected that the progession time of discretized method should be less than the continuous method. To compare the rate of progression towards optimality two randomly selected runs from each method were observed. Minimum (best) objective values of each generation were plotted against the generation number Figure 4.6 shows the graphical representation of the minimum objective value of each generation of the selected samples.

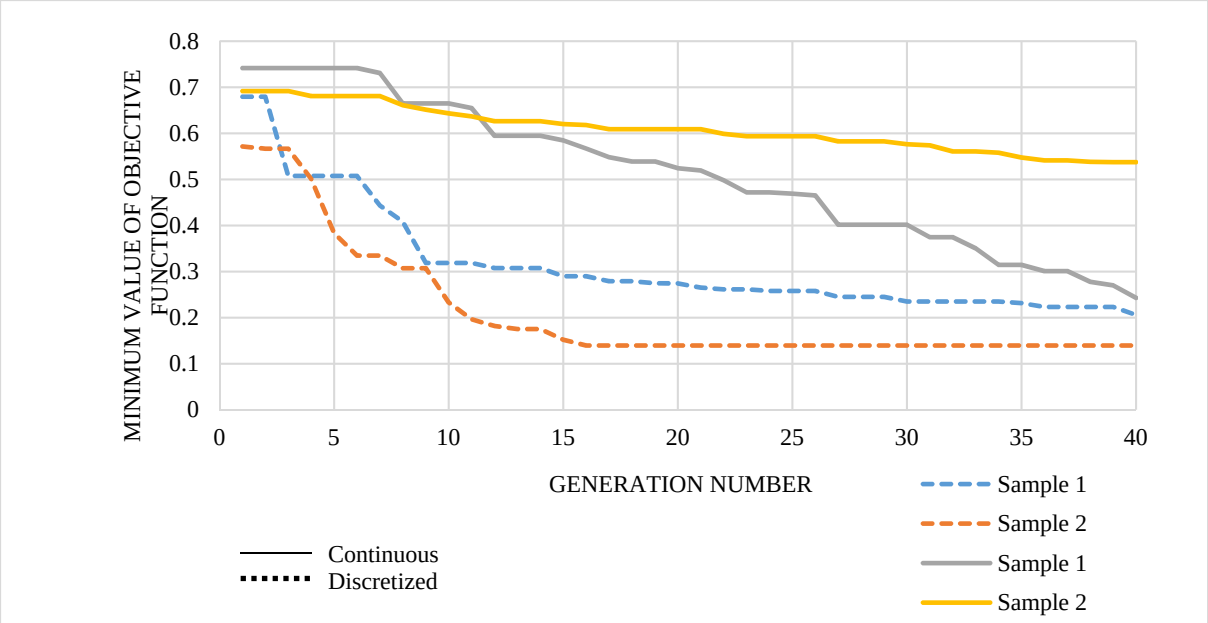


Figure 4.6: Graph of minimum (best) objective value vs. generation number for continuous and discretized method

From the graph, it was evident that the progression of the best objective value for the discretized method was faster than for the continuous method. Its minimum objective value started from a comparatively better position. While for the continuous method, the best result in each generation decreased its value almost linearly, the discretized method showed a sudden fall and then gradually converged to the optimal solution. This graph conspicuously demonstrated that after the 20th generation, the discretized method almost converged to the optimal solution whereas the continuous method reached its optimal value only after 40 generations. So, the hypothesis was proved that the discretized method takes less time to converge.

4.2. Constrained Problem:

4.2.1. Continuous Method:

Results are shown in Table 4.3.

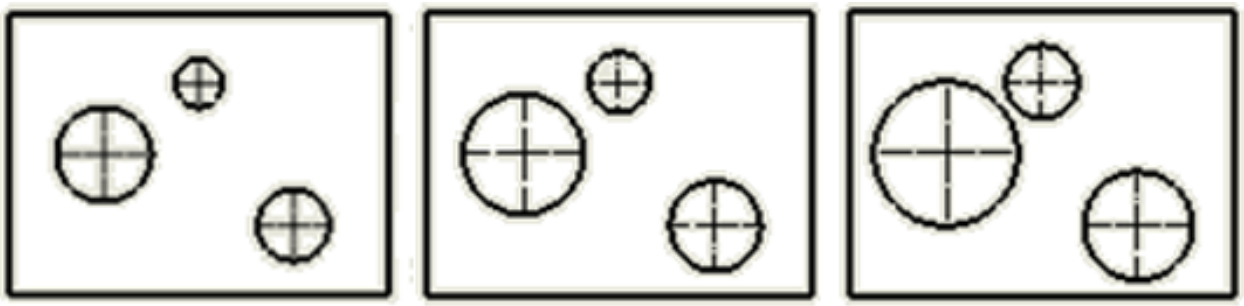
Table 4.3: Average of the optimal results of 30 trials

Penalty Function	Average Objective Function
Flat	0.44
Linear	**0.39**
Non-linear	0.41

Based on the average objective function, it was observed that the linear penalty function provided better results regarding the average value of the objective function than the other two penalty functions. But, no significant difference was observed. The impact of the penalty functions on the enclosed area by the supports and the reaction differences also showed similar results.

Again, the continuous method could not provide better optimal results for the constrained optimization problem as the obtained objective function's value for all penalty functions were comparatively larger. In all cases, the continuous method almost balanced the reaction forces but generated very poor area values.

The next step of this work was to investigate the effect of the size of the discontinuous areas of the solution space on the performance of GA. The area of the discontinuous spaces was increased by 25% and 50% (shown in Figure 4.7) to observe the impact of the increased discontinuous areas on the optimal result.



(a) **(b)** **(c)**

Figure 4.7: Indeterminate structure with (a) previous discontinuous areas (b) 25% increased circular discontinuities and (c) 50% increased circular discontinuities

No significant difference was observed among the performances of the three different penalty functions with three different radius size. Similar results for the average value of the objective function, the area value and the reaction differences were obtained.

4.2.2. Discretized Method:

The "Discretized Method" was also applied to the modified test case using equation (xvii) and Figure 4.8 shows the obtained objective values for the three penalty functions for 30 trials.

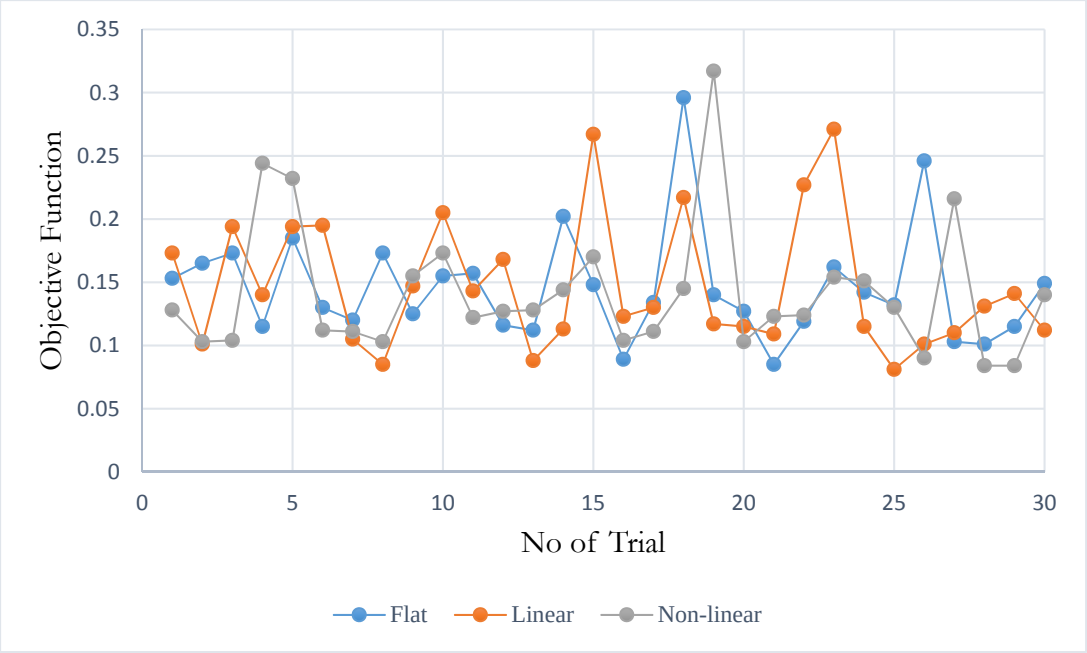
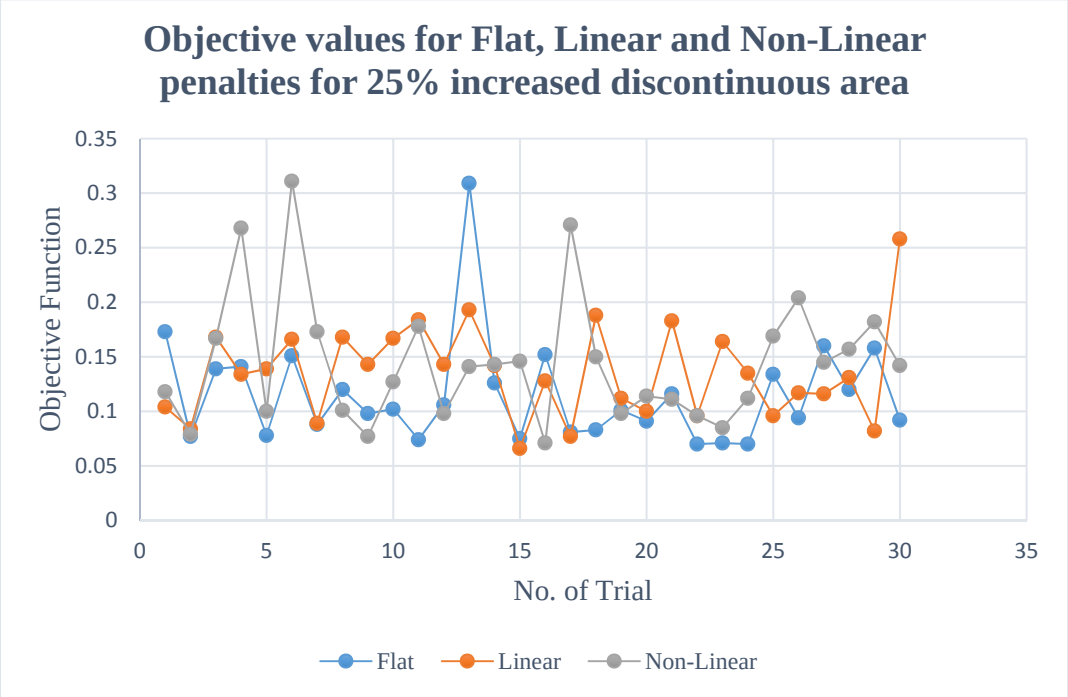


Figure 4.8: Results of applying three different penalty functions- flat, linear and non-linear with discretized method

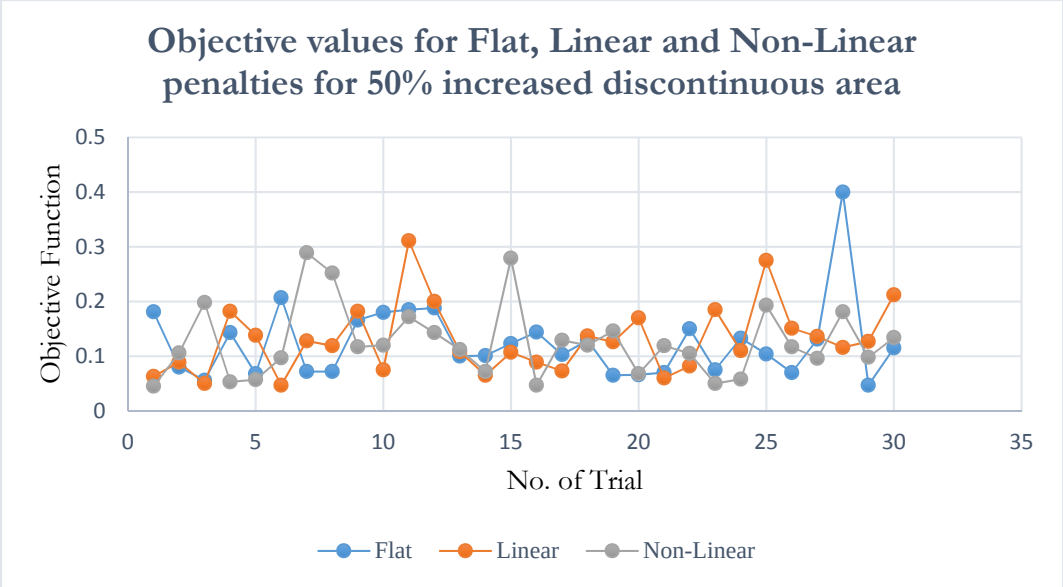
Figure 4.8 shows that there was no significant difference in the performance of the three different penalty functions because they had almost similar numeric optimal results with a sample size of 30. The average value of the objective function of the flat, linear and non-linear penalty functions were 0.1456, 0.1472 and 0.1410 respectively. These penalty functions also performed in a similar fashion for the components of the objective functions separately. Thus for this particular optimization problem, it was observed that the type of penalty function used had no significant effect on the performance of the GA.

This method was again tested for different sizes of the discontinuities in a similar fashion to that described in section 4.2.1. Though the size of the discontinuous spaces was increased, it had no significant impact on the performance of the GA. All three penalty functions successfully provided similar optimum results.

As one of the objectives was to maximize the enclosed area, support locations took place almost at the boundaries of the rectangular plate. The circular holes were not located on the boundaries, so, their increased size might have had some effect on the population generated in the initial generations, but no significant impact on the optimal results. Figure 4.9 represents the same concept by illustrating the results obtained by increasing the size of the circular holes by 25% and 50%.



(a)



(b)

Figure 4.9: Objective functions for the three penalty functions after increasing the discontinuous space by (a) 25% and (b) 50%

In Figure 4.9, it can be observed that all penalty functions acted similarly and provided results with no significant differences. The average optimum objective function for the flat, linear and non-linear penalties were 0.12, 0.14 and 0.14 for 25% increased discontinuity, and 0.12, 0.13 and 0.13 for 50% increased discontinuity. Thus, the "Discretized Method" determined the optimal results successfully irrespective of the type of the penalty function and the size of the discontinuous areas.

4.3. Coded Algorithm for Constrained Optimization:

4.3.1. Coded Discretized Algorithm:

It has been discussed already in section 1 and 2 that GAs are not directly compliant to constrained optimization. Though various methodologies have been developed and implemented to apply GA in these optimization areas, some adjustments are always required based on the type of the methods. In

this thesis work, one of the popular methodologies, penalty functions, was applied. While setting the parameters, it was patent that even a slight change in a single parameter affected the optimal result immensely. So, selection of these parameters was not only critical but also very time-consuming. Thus, the last step of this work was to develop a coded algorithm for GA which can eliminate the requirement of using various methods for a specific class of constrained optimizations.

The proposed method, "Discretized Method," was designed in such a way that it was applicable without any penalty function while ensuring the feasibility of the optimum results. The basic concept of this method was to divide the whole search space into very small horizontal and vertical strips, and the intersection points of those strips comprised the set of the possible solutions instead of the whole solution space. At the beginning of the GA search, that set of possible solutions was generated and later used for the rest of the process. Index numbers of the intersection points were used as design variables.

To eliminate the penalty function, the discretized method was coded and modified to remove the strips located in the discontinuous areas from the set of the possible solutions. Thus, in this method GAs started to work on only the feasible solutions and thus no penalty function or other techniques were required to maintain feasibility. In Figure 4.10, the steps are shown:

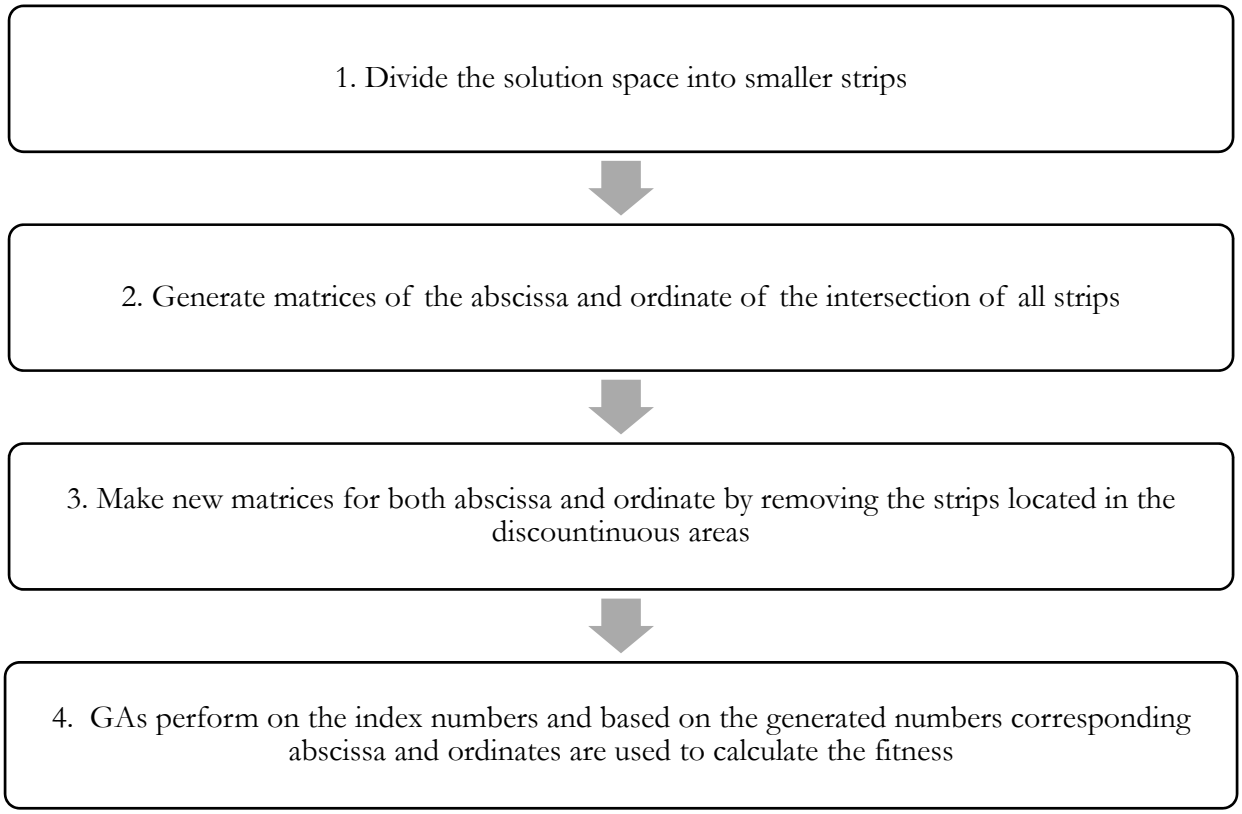


Figure 4.10: Steps followed in coded discretized algorithm

Thus, infeasible locations were not even considered which invalidated the use of penalty functions or other techniques to get feasible solutions. After applying this coded method, the following results (Table 4.4) were obtained,

Table 4.4: Results obtained from Coded discretized method

	Average	**Std Dev**
Normalized Area	0.131	0.957
Normalized Reaction Difference	0.033	0.026
Normalized Objective Function, Z	0.164	0.037

Paired t-test ($\alpha = 5\%$) was conducted to observe any difference between this coded algorithm with the discretized method. No statistically significant difference was observed between the objective values of these two methods. So, the coded algorithm provided the same result without using any penalty functions. Thus the coded algorithm saved unnecessary time required to calculate the feasibility of each solution.

4.3.2. FEA Integrated Coded Discretized Method (FEAICDM):

Though the developed coded algorithm successfully provided a very good optimal result, one limitation of this method is that it is easily applicable to simple and regular shaped discontinuous areas (circular, rectangular, etc.) only. If a problem has irregular shaped discontinuous space, it will be very difficult to eliminate the infeasible locations from the possible solution list. This problem could be solved if the manual generation of the list of possible solutions is replaced by using an FEA (Finite Element Analysis) software to generate the list.

FEA software generates a mesh by dividing the whole object into very small units. The location of those units can be directly used as the list of a possible solution.

Solid Work 2015 x64 Edition was used to draw the test case and then the mesh was generated by setting mesh density at "Fine". After generating the mesh, locations of all nodes on the upper surface of the test case were extracted from the model and saved in an Excel file. Later, these locations were used as the possible solution list for the GA to start the search process. This way, the complexity of the shape of the discontinuous space cannot affect the fastness of the solution approach. This FEAICDM can solve any irregular shape.

The implementation of the FEAICDM provided the results shown in Table 4.5,

Table 4.5: Results obtained from FEA Integrated Coded Discretized method

	Average	Std Dev
Area (m^2)	17.12	0.88
Normalized Area	0.144	0.956
Reaction Difference (N)	53.53	47.97
Normalized Reaction Difference	0.024	0.021
Normalized Objective Function, Z	0.168	0.040

Table 4.5 shows almost similar results to the coded discretized method. The average value of the enclosed area was 17.12 m^2, which was very close to the heuristic value, 20 m^2. The difference among the reaction forces of the supports was minimal. A paired t-test ($\alpha = 5\%$) also showed that these two methods are statistically same which implies that both methods are capable of providing optimal results.

Thus, this method requires no penalty functions or other approaches to make GAs suitable for constrained problem. It only considers the feasible solution and saves the time required to calculate the fitness function for every solution. Moreover, any irregular shaped discontinuous search space can be easily solved by using this method.

5. Conclusion:

In this work, a standard and simplified test case with multiple competing objectives have been developed. The optimal values of these objectives were not so obvious to determine. A Genetic Algorithm based methodology has been developed to minimize the reactive forces acting upon the supports of the test case and to maximize the stability of the structure by maximizing the enclosed area of the supports. The continuous GA has been applied to a predetermined set of GA parameters to obtain the optimal support locations using a no-preference, normalized objective function. The efficiency of the continuous method did not meet a satisfactory level, which has led towards a coding based approach named "Discretized Method". This method considered the entire continuous solution space as a sum of smaller grids. The performance of this method has been compared with the continuous simple GA method. It has been observed that the discretized method provided better optimal results compared to the previous one. The proposed method was able to achieve excellent results at 20^{th} generations. Later, both methods were used for the same test case with discontinuities applied to make the problem more pragmatic. Flat, linear and non-linear penalty functions were used to handle the constrained problem for the GA. Results have shown no mentionable differences among different penalty functions for both methods. The effect of the size of the discontinuous areas on the optimality and the performance of the GAs have also been tested which have also shown insignificant differences. Finally, the discretized method has been coded to eliminate the infeasible areas from the entire search space. Thus feasible optimal results have been obtained without using any penalty functions. This coded method provided very good consistent results. To make this method applicable for any irregular shaped discontinuous search space, a FEA integrated method has been developed. Statistical analysis has shown that this method provides similar results to the coded discretized method. This modification makes discretized method more

robust and less complex by eliminating the necessity of the selection of the suitable penalty function and its parameters. In this work, the developed discretized method has been tested only for a specific class of problem and require more investigation regarding its applicability to other sets of problems too. However, it can be concluded that the proposed FEAICDM has made GA more sturdy and effective for a class of constrained problems by eliminating penalty functions.

Appendices

A. Table 4.7 Results of the indeterminate structure without discontinuity using Continuous Method

Run	Z-value	(Maximum Force-Minimum Force) (N)	Normalized force	Area (m^2)	Normalized Area
1	0.272	2.23	0.001	14.57	0.27
2	0.588	9.91	0.004	8.34	0.58
3	0.593	1.56	0.001	8.15	0.59
4	0.293	15.61	0.007	14.29	0.29
5	0.393	26.08	0.012	12.38	0.38
6	0.57	32.22	0.014	8.9	0.56
7	0.57	24.5	0.011	8.82	0.56
8	0.215	19.71	0.009	15.87	0.21
9	0.291	13.15	0.006	14.29	0.29
10	0.592	13.94	0.006	8.28	0.59
11	0.227	29.33	0.013	15.73	0.21
12	0.52	6.53	0.003	9.65	0.52
13	0.429	28.16	0.013	11.68	0.42
14	0.465	30.77	0.014	10.97	0.45
15	0.17	15.28	0.007	16.74	0.16
16	0.555	15.68	0.007	9.05	0.55
17	0.431	46.67	0.021	11.8	0.41
18	0.292	26.03	0.012	14.39	0.28
19	0.442	18.51	0.008	11.32	0.43
20	0.342	45.83	0.020	13.57	0.32
21	0.178	29.29	0.013	16.7	0.17
22	0.269	14.56	0.006	14.75	0.26
23	0.656	9.67	0.004	6.96	0.65
24	0.45	16.09	0.007	11.14	0.44
25	0.188	244.29	0.109	18.41	0.08
26	0.207	16.9	0.008	16.02	0.20
27	0.595	39.01	0.017	8.44	0.58
28	0.246	15.38	0.007	15.22	0.24
29	0.388	11.84	0.005	12.34	0.38
30	0.723	0.91	0.001	5.55	0.72

B. Table 4.8 Results of the indeterminate structure without discontinuity using Discretized Method

Run	Z-value	(Maximum Force-Minimum Force) (N)	Normalized force	Area (m^2)	Normalized Area
1	0.11	11.27	0.005	17.89	0.11
2	0.17	26.28	0.012	16.84	0.16
3	0.145	0	0.000	17.1	0.15
4	0.217	0	0.000	15.65	0.22
5	0.168	0	0.000	16.64	0.17
6	0.124	0	0.000	17.52	0.12
7	0.112	5.66	0.003	17.81	0.11
8	0.193	0	0.000	16.14	0.19
9	0.155	5.89	0.003	16.95	0.15
10	0.132	0	0.000	17.36	0.13
11	0.107	11.19	0.005	17.97	0.10
12	0.151	33.18	0.015	17.27	0.14
13	0.152	0	0.000	16.97	0.15
14	0.171	8.02	0.004	16.65	0.17
15	0.105	0	0.000	17.9	0.11
16	0.218	0	0.000	15.64	0.22
17	0.117	27.88	0.012	17.91	0.10
18	0.134	43.43	0.019	17.71	0.11
19	0.128	39.03	0.017	17.79	0.11
20	0.114	20.1	0.009	17.9	0.11
21	0.105	16.98	0.008	18.06	0.10
22	0.138	23.13	0.010	17.45	0.13
23	0.216	0	0.000	15.68	0.22
24	0.119	34.32	0.015	17.93	0.10
25	0.191	21.84	0.010	16.38	0.18
26	0.156	17.76	0.008	17.04	0.15
27	0.139	77.08	0.034	17.92	0.10
28	0.132	5.62	0.003	17.41	0.13
29	0.173	0	0.000	16.54	0.17
30	0.117	30.01	0.013	17.92	0.10

C. Paired t-test analysis of Discretized Method and Coded Discretized Method

	Objective Function		
Run	Discretized Method	Coded Discretized Method	Difference
1	0.153	0.197	0.044
2	0.165	0.162	-0.003
3	0.173	0.224	0.051
4	0.115	0.183	0.068
5	0.185	0.148	-0.037
6	0.130	0.223	0.093
7	0.120	0.169	0.049
8	0.173	0.146	-0.027
9	0.125	0.195	0.070
10	0.155	0.140	-0.015
11	0.157	0.155	-0.002
12	0.116	0.121	0.005
13	0.112	0.141	0.029
14	0.202	0.129	-0.073
15	0.148	0.243	0.095
16	0.089	0.126	0.037
17	0.134	0.217	0.083
18	0.296	0.152	-0.144
19	0.140	0.142	0.002
20	0.127	0.183	0.056
21	0.085	0.149	0.064
22	0.119	0.215	0.096
23	0.162	0.124	-0.038
24	0.142	0.124	-0.018
25	0.132	0.131	-0.001
26	0.246	0.152	-0.094
27	0.103	0.212	0.109
28	0.101	0.100	-0.001
29	0.115	0.142	0.027
30	0.149	0.180	0.031
Average	**0.146**	**0.164**	**0.019**
Std Dev	**0.044**	**0.037**	**0.059**
Sample number	**30**	**30**	**30**
t value	**2.045**	**2.045**	**2.045**
alpha, α	**0.050**	**0.050**	**0.050**
95% C. I. Lower Bound	**0.129**	**0.150**	**-0.004**
95% C. I. Upper Bound	**0.162**	**0.178**	**0.041**

D. Paired t-test analysis of Coded Discretized Method and FEAICDM

	Objective Function		
Run	Coded Discretized Method	FEAICDM	Difference
1	0.197	0.135	-0.062
2	0.162	0.153	-0.009
3	0.224	0.184	-0.040
4	0.183	0.192	0.009
5	0.148	0.108	-0.040
6	0.223	0.152	-0.071
7	0.169	0.146	-0.023
8	0.146	0.152	0.006
9	0.195	0.144	-0.051
10	0.140	0.180	0.040
11	0.155	0.111	-0.044
12	0.121	0.101	-0.020
13	0.141	0.207	0.066
14	0.129	0.213	0.084
15	0.243	0.242	-0.001
16	0.126	0.121	-0.005
17	0.217	0.219	0.002
18	0.152	0.132	-0.020
19	0.142	0.269	0.127
20	0.183	0.178	-0.005
21	0.149	0.113	-0.036
22	0.215	0.184	-0.031
23	0.124	0.201	0.077
24	0.124	0.180	0.056
25	0.131	0.159	0.028
26	0.152	0.181	0.029
27	0.212	0.174	-0.038
28	0.100	0.178	0.078
29	0.142	0.134	-0.008
30	0.180	0.190	0.010
Average	**0.164**	**0.168**	**0.004**
Std Dev	**0.037**	**0.040**	**0.048**
Sample number	**30.000**	**30.000**	**30.000**
t value	**2.045**	**2.045**	**2.045**
alpha, α	**0.050**	**0.050**	**0.050**
95% C. I. Lower Bound	**0.150**	**0.153**	**-0.014**
95% C. I. Upper Bound	**0.178**	**0.183**	**0.022**

References

[1] Deb, Kalyanmoy. *Multi-objective optimization using evolutionary algorithms*. Vol. 16. John Wiley & Sons, 2001.

[2] Dasgupta, Dipankar, and Zbigniew Michalewicz, eds. *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013.

[3] Kalyanmoy, D., and K. Amarendra. "Real-coded genetic algorithms with simulated binary crossover: studies on multimodal and multiobjective problems." *Complex Systems* 9.6 (1995): 431-454.

[4] Bäck, Thomas, and Hans-Paul Schwefel. "An overview of evolutionary algorithms for parameter optimization." *Evolutionary computation* 1.1 (1993): 1-23.

[5] Back, Thomas. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[6] Fogel, David B. *Evolutionary computation: toward a new philosophy of machine intelligence*. Vol. 1. John Wiley & Sons, 2006.

[7] John, Holland. "Adaptation in natural and artificial systems." (1992).

[8] Goldberg, David E. "Genetic algorithms in search, optimization and machine learning 'addison-wesley, 1989." *Reading, MA* (1989).

[9] Vent, W. Rechenberg, Ingo. "Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 170 S. mit 36 Abb. Frommann-Holzboog-Verlag. Stuttgart 1973. Broschiert." *Feddes Repertorium* 86.5 (1975): 337-337.

[10] Schwefel, Hans-Paul. *Numerical optimization of computer models*. John Wiley & Sons, Inc., 1981.

[11] Schneider, Gisbert, Johannes Schuchhardt, and Paul Wrede. "Artificial neural networks and simulated molecular evolution are potential tools for sequence-oriented protein design." *Computer applications in the biosciences: CABIOS* 10.6 (1994): 635-645.

[12] Fogel, Lawrence J. *Artificial Intelligence Through Simulated Evolution.[By] Lawrence J. Fogel... Alvin J. Owens... Michael J. Walsh*. John Wiley & Sons, 1966.

[13] Luke, Brian T. "Evolutionary programming applied to the development of quantitative structure-activity relationships and quantitative structure-property relationships." *Journal of Chemical Information and Computer Sciences* 34.6 (1994): 1279-1287.

[14] Gehlhaar, Daniel K., et al. "Molecular recognition of the inhibitor AG-1343 by HIV-1 protease: conformationally flexible docking by evolutionary programming." *Chemistry & biology* 2.5 (1995): 317-324.

[15] Cramer, Nichael Lynn. "A representation for the adaptive generation of simple sequential programs." *Proceedings of the First International Conference on Genetic Algorithms*. 1985.

[16] Koza, John. "What is Genetic Programming (GP)." Genetic Programming Inc., 08 July, 2007. Web 10 May, 2016, http://www.genetic-programming.com/

[17] Kung, Hsiang-Tsung, Fabrizio Luccio, and Franco P. Preparata. "On finding the maxima of a set of vectors." *Journal of the ACM (JACM)* 22.4 (1975): 469-476.

[18] Goldberg, Robert (ed.), A. A. e. J. L. "Evolutionary Multiobjective Optimization: Theoretical Advances and Applications". *Springer London Ltd*, 1st edition. (2005).

[19] Coello, Carlos Coello, Gary B. Lamont, and David A. Van Veldhuizen.*Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media, 2007.

[20] Rawlins, Gregory J. E. *Foundations of Genetic Algorithms*. Vol. 1. San Mateo, CA: M. Kaufmann, 1991.

[21] Gen, Mitsuo, and Runwei Cheng. "A survey of penalty techniques in genetic algorithms." *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. IEEE, 1996.

[22] Venkatraman, Sangameswar, and Gary G. Yen. "A generic framework for constrained optimization using genetic algorithms." *IEEE Transactions on Evolutionary Computation* 9.4 (2005): 424-435.

[23] Simpson, Angus R., Graeme C. Dandy, and Laurence J. Murphy. "Genetic algorithms compared to other techniques for pipe optimization." *Journal of water resources planning and management* 120.4 (1994): 423-443.

[24] Dandy, Graeme C., Angus R. Simpson, and Laurence J. Murphy. "An improved genetic algorithm for pipe network optimization." *Water Resources Research* 32.2 (1996): 449-458.

[25] Abdel-Gawad, HOSSAM AA. "Optimal design of pipe networks by an improved genetic algorithm." *Proceedings of the Sixth International Water Technology Conference IWTC*. 2001.

[26] Bramlette, Mark F. "Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization." *ICGA*. 1991.

[27] Baker, James Edward. "Adaptive selection methods for genetic algorithms."*Proceedings of an International Conference on Genetic Algorithms and their applications*. 1985.

[28] Chipperfield, Andrew, et al. "Genetic algorithm toolbox for use with MATLAB." (1994).

[29] Caruana, Richard A., Larry J. Eshelman, and J. David Schaffer. "Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover." *Proceedings of the 11th international joint conference on Artificial intelligence-Volume 1*. Morgan Kaufmann Publishers Inc., 1989.

[30] Booker, Lashon. "Improving search in genetic algorithms." *Genetic algorithms and simulated annealing* (1987): 61-73.

[31] Jones, Dylan F., S. Keyvan Mirrazavi, and Mehrdad Tamiz. "Multi-objective meta-heuristics: An overview of the current state-of-the-art." *European journal of operational research* 137.1 (2002): 1-9.

[32] Konak, Abdullah, David W. Coit, and Alice E. Smith. "Multi-objective optimization using genetic algorithms: A tutorial." *Reliability Engineering & System Safety* 91.9 (2006): 992-1007.

[33] Schaffer, J. David. "Multiple objective optimization with vector evaluated genetic algorithms." *Proceedings of the 1st international Conference on Genetic Algorithms*. L. Erlbaum Associates Inc., 1985.

[34] Fonseca, C. M., and P. J. Fleming. "Multiobjective genetic algorithms. In: IEE colloquium on Genetic Algorithms for Control Systems Engineering." (1993).

[35] Horn, Jeffrey, Nicholas Nafpliotis, and David E. Goldberg. "A niched Pareto genetic algorithm for multiobjective optimization." *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. Ieee, 1994.

[36] Hajela, Prabhat, and C-Y. Lin. "Genetic search strategies in multicriterion optimal design." *Structural optimization* 4.2 (1992): 99-107.

[37] Srinivas, Nidamarthi, and Kalyanmoy Deb. "Muiltiobjective optimization using nondominated sorting in genetic algorithms." *Evolutionary computation*2.3 (1994): 221-248.

[38] Knowles, Joshua D., and David W. Corne. "Approximating the nondominated front using the Pareto archived evolution strategy." *Evolutionary computation*8.2 (2000): 149-172.

[39] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE transactions on evolutionary computation* 6.2 (2002): 182-197.

[40] Deb, Kalyanmoy, Ashish Anand, and Dhiraj Joshi. "A computationally efficient evolutionary algorithm for real-parameter optimization." *Evolutionary computation* 10.4 (2002): 371-395.

[41] Corne, David W., et al. "PESA-II: Region-based selection in evolutionary multiobjective optimization." *Proceedings of the genetic and evolutionary computation conference (GECCO'2001*. 2001.

[42] Corne, David W., Joshua D. Knowles, and Martin J. Oates. "The Pareto envelope-based selection algorithm for multiobjective optimization."*International Conference on Parallel Problem Solving from Nature*. Springer Berlin Heidelberg, 2000.

[43] Zitzler, Eckart, Marco Laumanns, and Lothar Thiele. "SPEA2: Improving the strength Pareto evolutionary algorithm." *Eurogen*. Vol. 3242. No. 103. 2001.

[44] Zitzler, Eckart, and Lothar Thiele. "An evolutionary algorithm for multiobjective optimization: The strength pareto approach." (1998).

[45] Gen, Mitsua, Kenichi Ida, and Yinzhen Li. "Solving bicriteria solid transportation problem by genetic algorithm." *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on*. Vol. 2. IEEE, 1994.

[46] Tamaki, Hisashi, et al. "Multicriteria optimization by genetic algorithms: a case of scheduling in hot rolling process." *Proceeding of the Third APORS*. 1995.

[47] Abido, Mohammad A. "Multiobjective evolutionary algorithms for electric power dispatch problem." *Computational Intelligence*. Springer Berlin Heidelberg, 2009. 47-82.

[48] Camp, Charles, Shahram Pezeshk, and Guozhong Cao. "Optimized design of two-dimensional structures using a genetic algorithm." *Journal of structural engineering* 124.5 (1998): 551-559..

[49] Nanakorn, Pruettha, and Konlakarn Meesomklin. "An adaptive penalty function in genetic algorithms for structural design optimization." *Computers & Structures* 79.29 (2001): 2527-2539.

[50] Choi, Byung Gun, and Bo Suk Yang. "Optimum shape design of rotor shafts using genetic algorithm." *Journal of Vibration and Control* 6.2 (2000): 207-222.

[51] Dharmadhikari, Sagar R., et al. "Design and Analysis of Composite Drive Shaft using ANSYS and Genetic Algorithm" A Critical Review." *Int. J. Mod. Eng. Res* 3.1 (2013): 490-496.

[52] Ohsaki, M. "Genetic algorithm for topology optimization of trusses."*Computers & Structures* 57.2 (1995): 219-225.

[53] Rajan, S. D. "Sizing, shape, and topology design optimization of trusses using genetic algorithm." *Journal of Structural Engineering* 121.10 (1995): 1480-1487.

[54] Nagendra, Somanath, et al. "Improved genetic algorithm for the design of stiffened composite panels." *Computers & Structures* 58.3 (1996): 543-555.

[55] Al-Shihri, M. A. "Structural Optimization Using a Novel Genetic Algorithm for Rapid Convergence." *International Journal of Civil and Structural Engineering*1.2 (2010): 123.

[56] Deb, Kalyanmoy. "An efficient constraint handling method for genetic algorithms." *Computer methods in applied mechanics and engineering* 186.2 (2000): 311-338.

[57] Coello Coello, Carlos A. "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art."*Computer methods in applied mechanics and engineering* 191.11 (2002): 1245-1287.

[58] Michalewicz, Zbigniew. "Genetic algorithms, numerical optimization, and constraints." *Proceedings of the sixth international conference on genetic algorithms*. Vol. 195. Morgan Kaufmann, San Mateo, CA, 1995.

[59] Takahama, Tetsuyuki, and Setsuko Sakai. "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites." *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006.

[60] Richardson, Jon T., et al. "Some guidelines for genetic algorithms with penalty functions." *Proceedings of the third international conference on Genetic algorithms*. Morgan Kaufmann Publishers Inc., 1989.

[61] Runarsson, Thomas P., and Xin Yao. "Stochastic ranking for constrained evolutionary optimization." *Evolutionary Computation, IEEE Transactions on*4.3 (2000): 284-294.

[62] Courant, Richard. "Variational methods for the solution of problems of equilibrium and vibrations." *Bull. Amer. Math. Soc* 49.1 (1943): 1-23.

[63] Fiacco, Anthony V., and Garth P. McCormick. "Extensions of SUMT for nonlinear programming: equality constraints and extrapolation." *Management Science* 12.11 (1966): 816-828.

[64] Carroll, Charles W. "The created response surface technique for optimizing nonlinear, restrained systems." *Operations Research* 9.2 (1961): 169-184.

[65] Schwefel, Hans-Paul Paul. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993.

[66] Smith, Alice E., and David W. Coit. "Penalty functions." *Handbook on Evolutionary Computation, pages C* 5 (1997): 1-6.

[67] Schoenauer, Marc, and Spyros Xanthakis. "Constrained GA optimization."*ICGA*. 1993.

[68] Ali, M. M., Mohsen Golalikhani, and Jun Zhuang. "A computational study on different penalty approaches for solving constrained global optimization problems with the electromagnetism-like method." *Optimization* 63.3 (2014): 403-419.

[69] Homaifar, Abdollah, Charlene X. Qi, and Steven H. Lai. "Constrained optimization via genetic algorithms." *Simulation* 62.4 (1994): 242-253.

[70] Joines, Jeffrey A., and Christopher R. Houck. "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's." *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. IEEE, 1994..

[71] Mezura-Montes, Efrén, and Carlos A. Coello Coello. "Constraint-handling in nature-inspired numerical optimization: past, present and future." *Swarm and Evolutionary Computation* 1.4 (2011): 173-194.

[72] Hart, Emma, Peter Ross, and Jeremy Nelson. "Solving a real-world problem using an evolving heuristically driven schedule builder." *Evolutionary Computation* 6.1 (1998): 61-80.

[73] B. Dack, Thomas, Frank Hoffmeister, and Hans—Paul Schwefel. "A survey of evolution strategies." *Proceedings of the 4th international conference on genetic algorithms*. 1991.

[74] Crossley, William A., and Edwin A. Williams. "A study of adaptive penalty functions for constrained genetic algorithm based optimization." *AIAA 35th aerospace sciences meeting and exhibit*. Reno, Nevada, 1997.

[75] Tessema, Biruk Girma. "Self-adaptive Genetic Algorithm for Constrained Optimization." (2006).

[76] Bazaraa, Mokhtar S., Hanif D. Sherali, and Chitharanjan M. Shetty.*Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[87] Hwang, C-L., and Abu Syed Md Masud. *Multiple objective decision making—methods and applications: a state-of-the-art survey*. Vol. 164. Springer Science & Business Media, 2012

[78] Cochrane, James L., and Milan Zeleny. *Multiple criteria decision making*. Univ of South Carolina Pr, 1973.