Chapter 2

Overview of Genetic Algorithm

2.1 Introduction

A Genetic Algorithm (GA) is a computerized search and optimization algorithm based on the mechanism of natural genetics and natural evolution. Its inspiration comes from Darwinian Theory of evolution by means of natural selection. Even though this view of natural selection is simplified, it serves as an introduction to artificial evolution of GA. The concept of GA was first conceived by Prof. John Holland of the University of Michigan in the year 1960. Thereafter, a number of his students and other researchers have contributed much to developing this field. Most of the initial research work can be found in various International conference Proceedings. A more complete discussion of genetic algorithms, including extensions and related topics, can be found in the books on GA (Goldberg, 1989; Gen and Cheng, 1997; Holland, 1975; Michalewicz, 1992; Mitchell, 1996; Vose, 1999; Davis, 1991).

GA is fundamentally different from classical optimization algorithms. It creates the considerable attraction to the researchers in a number of fields to use as a tool for optimization. GA repeatedly operates on a population composed of many solutions to a problem. These solutions are coded using finite length strings with either binary or real number representation. Because of the biological inspiration, each solution of a population is often termed as individual. The population is manipulated using a cycle of selection of good individuals followed by the generation of new individuals by operators such as crossover and mutation.
Elitism is often applied to preserve the potential solution/solutions of the population. Every iteration of this cycle is called a generation. The genetic process starts with the initial population of individuals/chromosomes whose components, called genes, are generated randomly from the prescribed bounds. Every individual/chromosome is evaluated by a fitness function and the individuals which are known to be good or promising (have a high fitness score) are then selected as parents to produce new individuals which are referred to as offspring for the next generation. Fitter solutions in the population are selected by the selection operator and allowed to produce offspring by crossover and mutation operators. Fitter solutions are selected in such a way that the offspring inherit the good characteristics from the parents so that the average quality of the solution becomes better than that of the previous generation. The new individuals are inserted in the population if they are acceptable, while inferior individuals are discarded. By repeated selection and mating the quality of the solutions in the population progressively gets better with subsequent generation. A major reason for the success of GA in attaining global or near to global solution is its ability to exploit the information accumulated about an initial unknown search space in order to bias subsequent searches into useful subspace. This is their key feature, particularly in large, complex and poorly understood search spaces, where classical search tools are inappropriate, offering a valid approach to problems requiring efficient and effective search techniques. The basic procedure of GA consists of the following steps:

**ALGORITHM 2.1**

*Begin*

Set $t \leftarrow 0$ \([t \text{ represents the current generation}].\)

Initialize $P(t)$, population (set of chromosomes) at $t$-th generation.

Evaluate the fitness function of $P(t)$.

*Repeat*

(i) Increase $t$ by unity for next generation.

(ii) Select $P(t)$ from $P(t-1)$.

(iii) With probability $p_c(t)$ perform crossover operation and generate offspring.

(iv) With probability $p_m(t)$ perform mutation operation and generate offspring.

(v) Evaluate the fitness function of $P(t)$.

*Until* termination condition is not satisfied

*Return* the best found chromosome together with the corresponding fitness value.

*End*
In applying GA to solve the particular optimization problems, further detailed considerations concerning the values for various parameters of GA, a genetic representation for potential solutions, a way to create an initial population, an evaluation process in terms of their fitness, genetic operators and termination conditions are required. These are discussed below:

### 2.2 Parameters of the Genetic Algorithm

The key parameters which significantly influence the GA convergence are as follows:

(i) **Population size \( (p_{\text{size}}) \):** The population size is clearly problem dependent and will need to increase with the dimensions of the problem. However, population size is restricted by both computing time and space limitations. As might be expected, GA tends to be ineffective if very small populations are used and so the choice of population size is always a compromise.

(ii) **Crossover probability \( (p_c) \):** The crossover probability ranges from 0.60 to 0.95 to produce better offspring.

(iii) **Mutation probability \( (p_m) \):** Like crossover probability the mutation probability also varies from 0.05 to 0.20 to produce better offspring.

(iv) **Maximum number of generations \( (m_{\text{gen}}) \):** The maximum number of generations is prescribed as stopping criteria to make sure that the solution has converged and therefore the algorithm must be terminated.

However, in this thesis we have considered variable rates of crossover and mutation probabilities. The detailed have been discussed in chapter 3.

### 2.3 Representation of chromosomes

Representation (encoding) of chromosomes plays a major role in the development of GA. Having a good representation scheme, which can describe the problem-specific characteristic well, is crucial because it significantly influences all the subsequent steps of GA. The chromosome (a sequence of genes) is in the form of a string/ array of numbers representing the complete set of all solutions. The genes can be binary or real numbers. In binary representation, each chromosome is encoded as binary string of fixed length with either ‘0’ or ‘1’ bit. An
example of a binary encoded chromosome that has $N_{var}$ variables, each encoded with 10 bits is as follows:

$$\text{Chromosome} = [11110010010011011111 \ldots 0000101001]$$

In real number representation, each chromosome is encoded in the form of an array of real numbers. If the chromosome has $N_{var}$ variables (an $N$ dimensional optimization problem) given by $V_1, V_2, ..., V_{N_{var}}$ then the chromosome is represented as an array of $1 \times N_{var}$ elements as shown below:

$$\text{Chromosome} = [V_1, V_2, ..., V_{N_{var}}]$$

In this case, the variable values are represented as floating-point numbers. In traditional genetic algorithms, chromosomes are represented by binary strings. However, this representation scheme has some drawbacks which are discussed as follows:

(i) The precession of the real coded GA is generally much better than that of the binary coded GA. Although the precession of the binary representation can be extended by introducing more bits, this considerably slows down the algorithm.

(ii) Real coding representation is capable of representing quite large domains. On the other hand, binary representation must sacrifice precession with an increase in domain size, given fixed binary length.

(iii) Real coding representation is simple and easy to implement since the chromosome is directly used to evaluate the fitness of the function. A binary chromosome will need to be decoded from its binary string to real string and then the decoded chromosome is used to evaluate the fitness of the function. Thus, much computation time is wasted in the encoding and decoding processes.

(iv) When solving a problem having a continuous search space using a binary-coded GA, hamming cliff occurs with certain strings (such as string 01111 and 10000) from which a transaction to a neighboring solution (in real space) requires the alternation of many bits. Hamming cliff present in a binary coding causes artificial hindrance to a gradual search in the continuous search space.

(v) Traditional binary coding for function optimization is known to have a weakness due to the large change of a real parameter value arising from changing a single bit.
in the binary string of the parameter. For example, the binary strings 011111 and 111111 are equal to decimal numbers of 63 and 31, respectively, while they are only different in one bit.

With these observations in mind, in this thesis in order to solve the optimization problems we have used real coding representation of chromosomes.

2.4 Initial population

To begin the GA, we define an initial population of $p_{size}$ chromosomes. A matrix represents the population with each row in the matrix being a $1 \times Nvar$ array (chromosome) of continuous values. For a given initial population, the full matrix of $p_{size} \times Nvar$, random values are generated from the respective bounds of the variables. In case of binary representation each element of a matrix is a substring represented by binary sequences having a determined length. In case of real coding representation each element of the matrix is a real number generated randomly using Gaussian or Uniform distribution function. Here we shall discuss the process for generation of random numbers using uniform distribution function. A random number $x$ is uniformly distributed over the interval $[a, b]$ if and only if its probability density function is given by

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{if } a \leq x \leq b \\ 0, & \text{otherwise} \end{cases}$$

A uniformly distributed random number on an interval $[a, b]$ can be generated by $x = a + r(b-a)$ where $r$ lies between 0 and 1.

2.5 Fitness function

After obtaining a population of potential solutions to the problem, we have to calculate the fitness value of each chromosome. Fitness function plays the same role in GA as that which the environment plays in natural evolution. The fitness function is chosen in such a way that the highly fitted chromosomes (possible solution) have high fitness values. It is the only index used to select a chromosome to reproduce the next generation. We have chosen the objective function of the optimization problem as the fitness function. Generally, the fitness function
EVAL(X) for the chromosome X is equivalent to the objective function f(X). The value of the objective function determines the environment in which the solutions live. In real coding representation the genes can be directly placed into the objective function, whereas binary representation will need to decode binary chromosomes into real values and then the decoded chromosome is put into the objective function.

2.6 Operators of GA

Genetic Algorithm operators perform actions that simulate natural genetic operations and manipulate structures of individuals during evolution. Evolution from generation to generation is simulated by preserving, redistributing or altering genetic material contained in the chromosome strings of the individuals. There are three principal genetic operators and they are selection/reproduction, crossover and mutation.

2.6.1 Selection/reproduction operator

Selection operator follows the role of the nature's survival of the fittest mechanism. Here fitter solutions survive while weaker ones are abolished. In the selection procedure, some selection criteria are applied to select a certain number of individuals/chromosomes, namely parents from the population according to their fitness values. Chromosomes with higher fitness values have more opportunities to be selected for reproduction in next step. During the reproduction phase the fitness value of each chromosome is assessed. This value is used in the selection process to provide bias towards fitter individuals. Just like in natural evolution, a fit chromosome has a higher probability of being selected for reproduction. The commonly used selection operators are as follows:

(i) Roulette Wheel Selection (Goldberg, 1989; Holland, 1975): Each individual has a selection probability proportional to its fitness.

(ii) Tournament Selection (Goldberg, Korb and Deb, 1989; Sastry and Goldberg, 2001): A group of individuals is chosen from the population and the most fit in the group is selected. The size of the group chosen is called the tournament size. A tournament size of 2 is a binary tournament.
(iii) Ranking Selection: The population is sorted by fitness and assigned a rank from best to worst. The selection probabilities are assigned to the individuals according to their rank.

Apart from the above, there are other selection operators which are as follows:

(i) Stochastic Universal Sampling.
(ii) Truncation selection.

In the thesis to solve the optimization problems we have used Ranking Selection and Tournament Selection.

2.6.2 Crossover

The crossover operator is the main genetic operator. The frequency of occurrence of this genetic operation is controlled by certain predefined probability called probability of crossover. The operator recombines genetic material from selected individuals called parents to form one or more offspring/children. The offspring produced share some of the characteristics of the parents and in that way the characteristics are passed on to the future generations. For different chromosome representations different techniques are used to perform the crossover operation. One-point, two-point, or uniform crossovers are commonly used in binary coded chromosomes. For the real coded chromosomes the commonly used crossover operators are simple crossover, single arithmetic crossover, whole arithmetic crossover, heuristic crossover etc. The general algorithm of crossover operation is given below:

**ALGORITHM 2.2**

Step 1: Calculate the integral value of $p_c \times p_{\text{size}}$ and store it in $N$.
Step 2: Select two or more chromosomes randomly from the current population.
Step 3: Perform crossover operation using an appropriate crossover technique.
Step 5: Repeat Step 2 and Step 3 for approximate times.

2.6.3 Mutation

After crossover, the next genetic operation is mutation. In this operation the frequency of its occurrence is controlled by certain predefined probability, called probability of mutation. This operator introduces random variations in the population and is applied to a single chromosome. Mutation is treated as supporting operator for the purpose of restoring lost genetic material. It
Chapter 2. Overview of Genetic Algorithm

helps to maintain the genetic diversity in the solution space and prevents the convergence to local optima. With binary representations this usually involved flipping bits of a chromosome. Real valued chromosome representations implement mutation differently. Some of the interesting mutation operator’s suitable for real coded GA are non-uniform mutation, boundary mutation, Gaussian mutation etc.

The general algorithm of mutation is given below:

**ALGORITHM 2.3**

Step 1: Calculate the integral value of \( p \times p\_size \) and store it in \( N \).

Step 2: Select an individual randomly from the current population.

Step 3: Perform mutation operation using an appropriate mutation technique.

Step 5: Repeat Step 2 and Step 3 for \( N \) times.

### 2.7 Termination criteria

The GA moves from generation to generation selecting and reproducing parents until a termination criterion is met. Different types of termination criteria are as follows:

(i) When the number of generation reaches a prescribed maximum number of generations (\( m\_gen \)).

(ii) There is no improvement in the best solution found so far for a certain number of generations.

(iii) Convergence of the population, which is often based on a large percentage of the population being identical and therefore indicates that the search has terminated.

In this thesis, we have used the criterion (i) as the termination criteria.

### 2.8 Advantages and disadvantages of Genetic Algorithm

The main advantages of Genetic Algorithm can be formulated as follows:

(i) It can easily be implement.

(ii) Unlike gradient methods it is capable of dealing with large set of decision/ design variables.
DEVELOPMENT OF SOME HYBRID GENETIC ALGORITHMS AND APPLICATIONS

(iii) It is ideally suited for problems with solution spaces that are too large to be extensively searched.

(iv) Unlike classical methods, GAs are not gradient based, i.e., they do not require the objective function to be continuous, neither do they need information about the derivatives of the objective functions, therefore they can handle problems with discrete solution space.

(v) The search mechanism is stochastic in nature, which makes them capable of searching the entire solution space with more likelihood of finding the global optima.

(vi) GAs is able to solve problems with non-convex solution space, where classical procedures usually fail.

(vii) The GAs explores the entire search space to search for the optimum solution(s) from a population of solutions to another population of solutions, rather than from one solution to another.

(viii) It also performs well with problems where the objective function is non-linear, discontinuous or has many local optima.

All these advantages make the GAs superior over the classical optimization techniques in some real world applications, particularly for very complex engineering problems.

Though there are several advantages of GA in solving different types of optimization problems, there are some disadvantages also. These are as follows:

(i) It is often found that a genetic algorithm gets caught in a local optimum, and that all or most of the population concentrates on a small part of the search space located around the local optimum. This is usually termed premature convergence.

(ii) The most difficult and time consuming issue in the successful application of GAs is to determine the approximate settings of GA parameters. The parameters of the optimization algorithm need to be tuned for efficiency. However, Michalewicz (1992) mentioned that the determination of proper values of these genetic parameters is an art and the quality of this tuning greatly depends on the users experience as well as their knowledge of the problem.

(iii) Computational efficiency can be lower than in other methods.