CHAPTER 3

INTRODUCTION OF GENETIC ALGORITHM

Genetic Algorithm is based upon theory of evolution, used to solve optimization problems [13,14]. This chapter illustrates the steps of genetic algorithm and its operators to solve such problem. Initial population generation, fitness evaluation, selection, cross over and mutation are major genetic operations which are used in GA to solve optimization problems.

3.1 Basic Concept of Genetic Algorithm

Genetic algorithm is a metaheuristic search technique which is based on natural selection and is used to solve optimization problems in computer science and engineering [14]. GA is based on the principle of survival of the fittest and inspired by Darwin's theory of origin of various specifies. In nature the fittest species survives and unfit species are lost or eliminated. Genetic algorithm works on a number of solutions available and tries to improve the fitness value or quality of these solutions by its genetic operators. GA represents these solutions by chromosomes and calculates the quality of these solutions represented by fitness of the chromosomes. For example, in case of TSP problem the distance travelled by the salesman decides the fitness of the chromosome. Genetic Algorithm is used to solve many NP-Complete and NP-Hard problems [20,24,31,32].

In cross over operation, more fit chromosomes are selected to generate new children chromosomes. These newly generated children chromosome may replace less fit chromosomes of the population (population is the set of chromosomes). This cross over operation and replacement will continue till an optimal solution is found or terminating criteria is not reached. Cross over was optimized by many researchers to solve NPC and NPH problems [27, 28, 39].

In this chapter, an overview of genetic algorithm is given which is based upon the available genetic algorithm operations which include encoding, fitness calculation, selection, cross over and mutation. Genetic algorithm is mostly useful and applied to solve those problems for which general numerical methods or traditional algorithms fail or take a very long time to solve that problem [6,7,8]. Nowadays Genetic Algorithm is applied on a number of application areas that include robotics, machine learning, optimization problems, graph theory, electronics and may more.

Genetic Algorithm is mainly used to solve optimization problems, mainly constraint problems and may also be used to solve unconstraint problems [31-35]. It is based upon natural selection and evolution.

Genetic algorithm applies genetic operators again and again to solve a problem. It starts its process by encoding the problem in genetic form and then calculates the fitness of that population. Then in every step of selection, cross over and mutation it improves the fitness of the population. In every step the population converges towards the optimal solution [32,78]. Genetic algorithm can also be used to solve problems which are not suited for standard optimization algorithms. The theory of Genetic Algorithm is based upon the Darwin theory of origin of species. According to Darwin's theory after several generations biological organism evolves on the basis of the survival the fittest principle [12]. In nature the individuals in the population competes to get food and other resources for their survival. In nature the mates compete with each other for reproduction. So the poor fit individuals are having less chances of meeting and thus less chances of reproduction. The most fit individuals participate in meeting and have more chances to participate in reproduction. After a long-long time the share of the most fit individuals in the population is very high and poor fit individual and their chromosomes are almost eliminated from the population.

In 1975, Holland applied the idea of evolution and natural selection to solve problems of science and engineering [13]. Holland wrote his famous book "Adaptation in natural and artificial systems". Holland described how to apply procedures of natural selection

and evolution to solve scientific problems. Genetic Algorithm describes these procedures in a systematic way to solve problems using evolution and natural selection [14]. GA is very powerful tool nowadays to solve many optimization problems.

3.2 Historical Background of GA

Holland have a great contribution in the development of Genetic Algorithm [13]. But several other scientists, mathematicians and computer programmer also have their contribution in the development of GA [5,6,7]. In 1975 Ken Delong, student of Holland explores Genetic Algorithm to solve optimization problems. Many other students of Holland explore Genetic Algorithm and its capabilities to solve problems of science and engineering. Another student of Holland, David Goldberg apply Genetic Algorithm to solve gas pipeline optimization problem and win award for his research and thesis. In 1989 David Goldberg published a book titled - Genetic Algorithm in Search Optimization and Machine Learning. It was a milestone in the popularity of Genetic Algorithm in the field of Artificial Intelligence and Machine Learning. After this achievement, the application and utilities of Genetic Algorithm grows rapidly.

Genetic Algorithm is most suitable for optimization problems. The popularity of GA in metaheuristics is mainly in solving optimization and search problems. These problems have many application areas in science and engineering. Later on Genetic Algorithm was applied on multi objective optimization problems which have many objectives and constraint. These problems were very difficult to solve using traditional algorithms. GA was also used to solve many NP-Complete problems such as Travelling Salesman Problem [31-35], Multiple Knapsack Problem [71,72], N-Queen Problem [20,25] and many more.

GA solves the optimization problems by the process similar to biological evolution [13,14]. It first represents the problem into genetic form and generates a set of chromosomes called initial population. It then applies genetic operators such as selection, cross over and mutation. These operators work on the principal of survival

of the fittest. The selection operation selects some chromosomes from the population which will participate in cross over.

The selection operation selects most fit chromosomes to perform cross over. Then the cross over operation generates new children chromosomes. The new children which is more fit replace the chromosomes from the population which are less fit. Then the mutation operation mutates some of the chromosome of the population. It produces some accidental and random changes in the population. These steps will generate a new population with better fitness in every iteration and thus converges to an optimal solution after some iteration.

3.3 Basic Genetic Algorithm

The genetic algorithm is generally implemented and simulated by a computer program written in programming languages such as C, C++, JAVA, Dot Net, MATLAB etc. This section illustrates the steps of basic genetic algorithm [14].

Algorithm: Basic Genetic Algorithm

1. *[Initial Population Generation] - Generate an initial population of n chromosomes.*
2. *[Fitness calculation] – Calculate fitness value of every chromosome of the initial population.*
3. *[New Population Generation] – Generate a new population of chromosomes by following steps 3 to 9.*
4. *[Selection] – Select some chromosomes from the population (On the basis of their fitness value). These selected chromosomes will participate in cross over operation. Better is the fitness of a chromosome better will be its chances to get selected for cross over.*
5. *[Cross Over] – This operation generates new children chromosomes by combining parent chromosomes selected in selection operation. The rate of*

cross over will decide how many new children chromosomes will be generated using cross over.

6. *[Mutation] – This operation mutate/change some chromosomes of the population. The number of chromosomes mutated is specified by mutation probability.*
7. *[Accepting] – This operation accepts newly generated chromosomes to add these chromosomes in the population.*
8. *[Replace] - This operation replaces some chromosomes of the population by new children generated chromosomes. The less fit chromosomes are replaced by more fit chromosomes. So the overall fitness value of the population increases after every iteration.*
9. *[Test] – This step tests the termination condition. If satisfied, then the best solution from the population is returned as solution and go to step 11.*
10. *[Loop] – Go to step 3.*
11. *[Stop]*

Genetic algorithm starts from the generation of initial population of chromosomes. Chromosomes are basically strings appeared in DNA. These chromosomes are the encoded solution of the problems. The chromosome undergoes a sequence of steps to find the solution of the encoded problem.

The selection operation selects some chromosomes for the cross over. The cross over operation generates new children. The mutation operator changes some chromosomes in the population. The replacement operator replaces some less fit chromosomes by some more fit chromosomes in the population. The terminating criteria are checked whether a terminating state is reached. If reached, then the best fit chromosome from the population is returned as the solution. Otherwise the steps from selection to mutation are repeated many times until a terminating criterion is not reached. Figure 3.1 is showing flowchart of standard genetic algorithm. The next section illustrates the steps of genetic algorithm in more detail for the purpose of its implementation in any programming language.

3.4 Implementation of Genetic Algorithm

The major steps to implement genetic algorithm are as follows:

1. Initialization
2. Encoding
3. Fitness Function
4. Selection
5. Cross Over
6. Mutation

3.4.1 Initialization

To start Genetic Algorithm to solve a problem the first step is initialization. In this step the problem is formulated into genetic form, i.e. the form in which genetic operations can be applied on it. The initial population plays an important role in the performance of genetic algorithm.

O. Yugay et al. [32] optimized the performance of genetic algorithm by sorting the size of initial population whereas S. Zou et al. [78] applied multiple populations in GA to optimize the GA performance. The two major steps which are problem dependent in genetic algorithm are (a) Encoding (b) Fitness Function.

In the step of initialization, representation of the problem into genetic form shall be appropriate because it affects the end result. In this step, a complete layout of the problem into genetic form is represented and requires clear understanding of the problem. Initialization process in GA includes the decision about the encoding scheme to be used (for example binary encoding or permutation encoding), fitness function to be applied, the type of selection (such as random selection, roulette wheel selection, rank selection etc.), cross over operation used (one point, two point, multipoint etc.), mutation operator used and other genetic parameters such as cross over rate, mutation rate, population size etc.

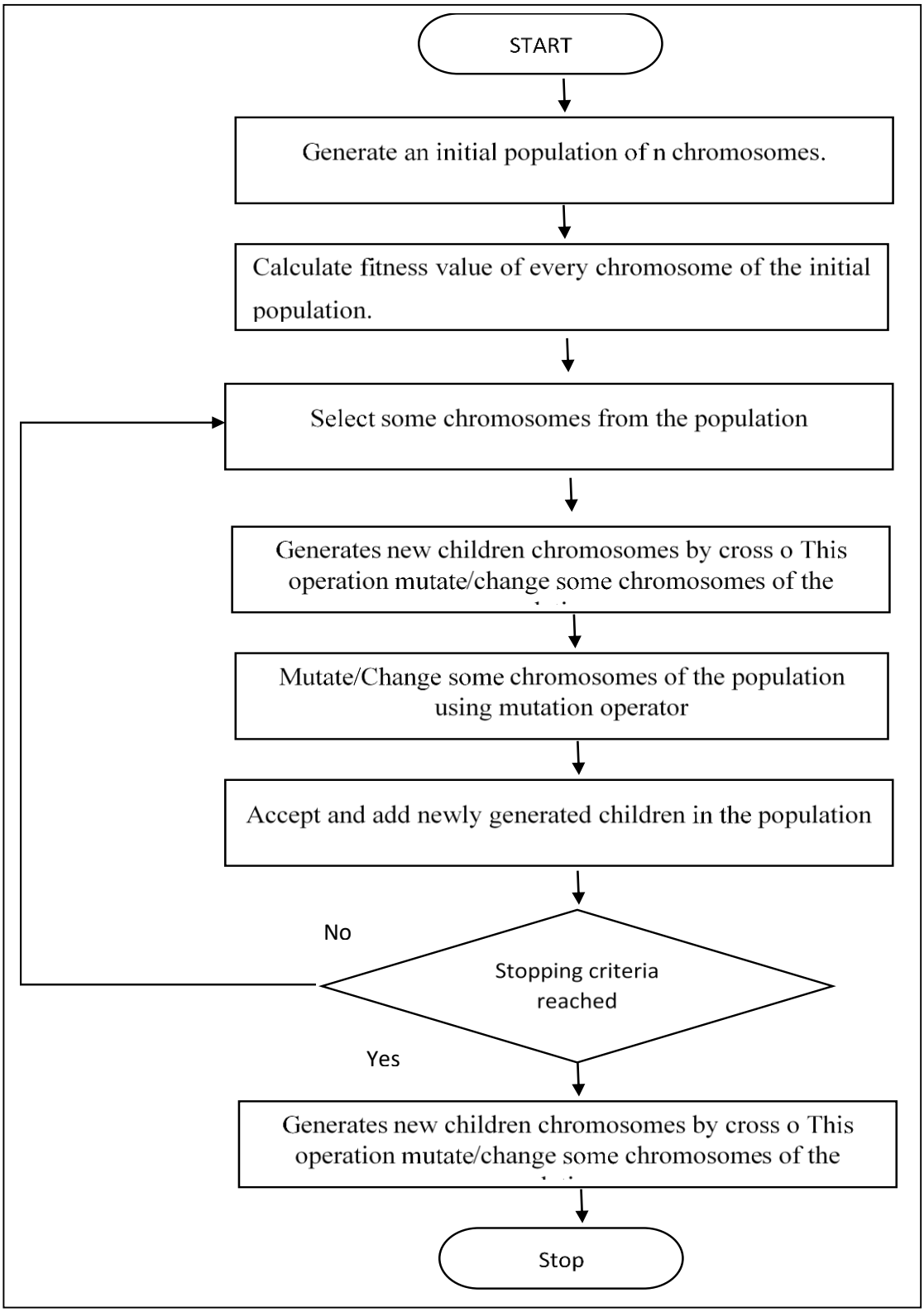

Figure 3.1: Flowchart of Genetic Algorithm

3.4.2 Encoding

This step encodes the problem into genetic form and generates chromosomes for a given problem. The Genetic Algorithm starts from encoding the problem. The encoding scheme converts the problem into a representation suitable for Genetic Algorithm. Encoding scheme generates chromosome which are the random candidate solution of the problem. The quality of these solutions is evaluated by fitness function. Many encoding schemes are available and the type of encoding scheme used is dependent upon the type of problem.

Some encoding scheme represents the chromosome in a string of zeros and ones, while other represents the chromosome into a sequence of integer numbers called permutation encoding and several other encoding schemes are also available. For a given problem, the encoding scheme suitable for that problem can be wisely opted. If none of the encoding scheme is suitable then either a new encoding scheme is to be developed otherwise the problem cannot be solved using Genetic Algorithm because it leads to erroneous result. The most commonly used encoding schemes are as follows:

Binary Encoding

The binary encoding scheme represents the chromosome in the form of a string of zeros and ones. This binary encoding scheme is the most basic type of the encoding scheme. A chromosome is a collection of basic components called alleles. One allele is represented by a binary zero or binary one in binary encoding.

The zero in the chromosome for an allele indicates that this allele of the chromosome is not selected and one in the chromosome for an allele indicates that this allele of the chromosome is selected. This binary encoding is useful for solving the problem like Knapsack problem. The chromosomes using binary encoding for knapsack problem for 10 knapsacks is represented in Figure 3.2.

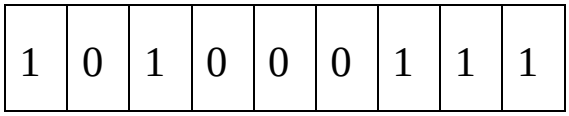

Figure 3.2: Representation of Chromosome in Binary Encoding

The chromosome shown in figure 3.2 is an instance of chromosome for knapsack problem of having 10 knapsacks having a string of 10 binary bits. “Zero” in the chromosome indicates that the item is not selected in the solution while “One” in the chromosome indicate that the item is selected.

Permutation Encoding

Using permutation encoding, the chromosome is represented in the form of a sequence of numbers. The number is the value of an allele of the chromosome. This type of encoding scheme is suitable for many problems such as Travelling Salesman Problem, N Queen problem and many more problems. An instance of chromosomes for the TSP problem of 10 cities is shown in Figure 3.3.

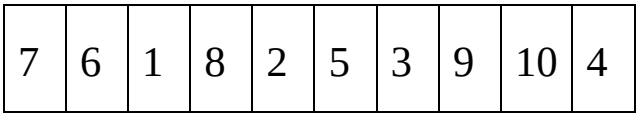

Figure 3.3: Representation of Chromosome in Permutation Encoding

The chromosome indicates that there 10 cities in the TSP problem. City 7 is the starting city and thus this city appeared in the first place and at the last place. So the first city visited by the salesman in the chromosome is the city 7 which is followed by city 6, city 1 and so on.

Value Encoding

In value encoding the chromosome is represented in the form of a string containing values. Values may be integers, characters or real numbers. Each allele in the chromosomes contains a value. The use of binary digits may not be suitable in some problems. If value encoding is used in chromosomes, then normal cross over and

mutation operators may not work. Some new cross over and mutation operators should be developed for value encoding chromosomes. The cross over and mutation operations are challenging in case of value encoding. Figure 3.4 is showing a chromosome in value encoding.

AB	ER	JU	LK	OP	ER	DS	XV	HJ	LM

Figure 3.4: Representation of Chromosome in Value Encoding

3.4.3 Fitness evaluation

Fitness function is a very important function used in Genetic Algorithm. This function is used to calculate fitness of a chromosome. The fitness of the chromosome represents quality of the solution for the problem. Higher is the value of the fitness better is the solution represented by the chromosome for the given problem. For some given NP-Complete problems the fitness function determines different values. For Travelling Salesman Problem, the fitness function calculates the distance travelled by the salesman and it is a minimization function because the distance travelled should be minimized. For N-Queen problem, the fitness function calculates the number of attacks between the queens. It is also a minimization function for N-Queen problem because attacks are to be minimized. For Knapsack problem, the fitness function calculates the profit earned by placing different items in the knapsacks. It is a maximization function in case of knapsack problem because the profit is to be maximized in knapsack problem.

3.5 Genetic Operators

Genetic Algorithm applies generic operators to find solution of a problem. After initialization and encoding a sample population of chromosome is created. The fitness function evaluates the fitness value of every chromosome of the population. After calculating the fitness of every chromosomes of the population, genetic operators such as selection, cross over and mutation are applied on the population. These operators

improve the fitness value of the population. These operators may be applied many times till the terminating criteria are not satisfied. For obtaining the favorable solution of optimization problems genetic operators can be modified [36,37,38] and also to optimize the performance of GA. Hybrid GA operators are used by many researchers to optimize the performance of GA [27,28,78].

3.5.1 Selection

Selection operation is used to select some chromosomes from the population to perform cross over. This operation is very important as it select the chromosomes which will participate in cross over and will generate chromosomes of future generations. Commonly those chromosomes are selected from the populations which have high values of the fitness.

The basic reason to select chromosomes is to follow the principle "survival of the fittest". Many authors applied different types of selection operators. The most commonly used selection operators are: Random Selection, Roulette Wheel Selection, Rank Selection, Tournament Selection, Boltzmann Selection, Steady State Selection and many more.

3.5.2 Crossover

Cross over operation is used to combine two or more chromosomes to generate new children chromosome [14]. In recent years cross over operator is modified by many researches to apply GA on many problems efficiently [27, 28, 39, 40, 43, 57,120,121]. Many types of cross over operators are used [39] depending upon the type of problem.

Some cross over operators are very easy to apply, whereas other cross over operators needs special care so that constraint defined on the chromosome (if any) should not be violated. Commonly used cross over operators are One Point Cross Over, Two Point Cross Over, Multi Point Cross Over, Uniform Cross Over, Partially Mapped Cross

Over PMX, Ordered Cross Over OX and many more [39,40,43,57,120,121]. The working procedure of some of the cross over operators is discussed here.

One Point Crossover

In one-point cross over at first cross over point is selected. This cross over point may be a fixed point or may be selected randomly [14]. The two parent chromosomes strings are exchanged or swapped before and after the cross over point. To generate a child chromosome, the values from the parent-1 are copied into child -1 from staring point to cross over point. After cross over point values from the parent -2 are copied into child-1. To generate child-2, the values form parent-2 are copied into child-2 from starting point to cross over point. After cross over point the values from parent-1 are copied into child-2. Thus two parent chromosomes are crossed with each other at cross over point and this cross over operation usually generate two child chromosomes from two parents. Figure 3.5 shows the process of one-point cross over. This operator is used by many researchers in GA [31-38].

Two Point Crossover

The two-point cross over is almost equal to one-point cross over but it is having two cut points [13,14]. The chromosome string from parent-1 and parent -2 are copied into child-1 and child -2 which are exchanged at cross over points. To generate child-1 the values from starting point to first cross over point are copied from parent-1. From first cross over point to second cross over point, values are copied from parent-2 and from cross over point 2 to end values are copied again from parent-1. The same procedure is applied to generate child-2 except that parent-2 is selected first, then parent-1 and then again parent-2 is selected. Figure 3.6 shows the process of two-point cross over.

Multi-Point Crossover (N-Point crossover)

In multi-point cross over there are more than two cross over points [13,14]. The different segments of two parents are selected alternatively between the cross over

points to generate child. For example, to generate child -1 the genes are selected from parent -1, from the starting point to first cross over point. Between first cross over pointy and second cross over point genes are selected from parent-2 into child-1. Then from second cross over point to third cross over point genes are selected from parent-1 and so on until the complete child is generated. The alternative portions of the two parents are selected to generate child-2.

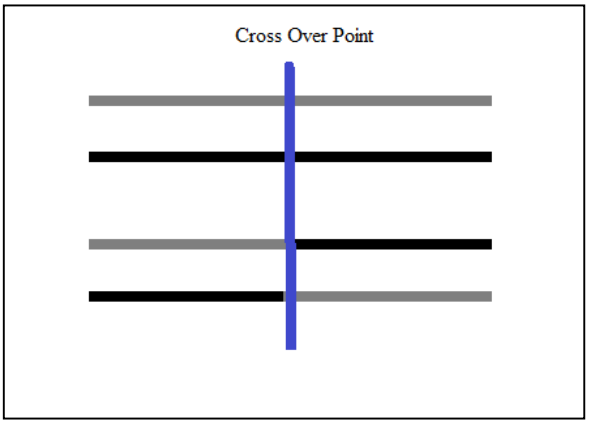

Figure 3.5: One Point Cross Over

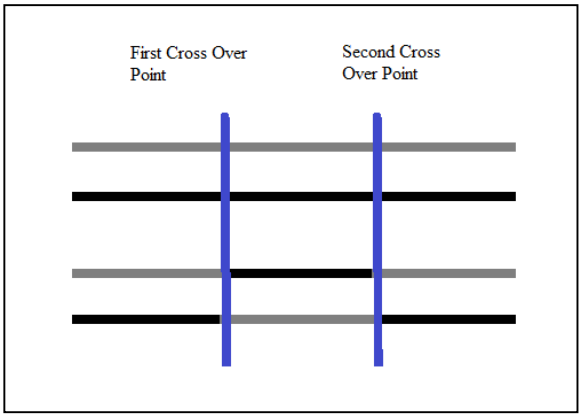

Figure 3.6: Two Point Cross Over

Uniform Crossover

In uniform cross over there are more than two cross over points [14]. It is almost like the N-Point cross over except that the selection of parent-1 and parent-2 is not fixed but either of the parent is selected randomly by a mask. This mask is string of 1 and 0 and its length is equal to the length of the chromosome. The occurrence of 1 in the mask indicates that genes are selected from parent-1. The occurrences of 0 in the mask indicate that genes are selected from parent-2. Therefore, the child is a random mixture of genes from both of the parents.

3.5.3 Mutation

Mutation is a very important operation in Genetic Algorithm. Mutation operator changes some genes of the population randomly [5,6,13,14,31]. The genetic parameter mutation probability decides the rate of the mutation. Generally, the rate of mutation is kept very low i.e. very small portion of the population is changed. This operation is performed after cross over and while generating the population for the next generation. Mutation operation is performed in many ways. The type of mutation may also be dependent on the type of problem and its encoding scheme. Most commonly used mutation operators are as follows:

Flipping

This type of mutation operation is suitable for the problems encoded in binary encoding scheme [1,14]. In flipping mutation operation, the genes are selected from the population and value of the gene is flipped i.e. if it is already zero then it is changed to one, if it is already one then it is changed to zero. This process of changing zero into one and one into zero is called flipping.

Interchanging

In this type of mutation, two genes are randomly selected and their values is exchanged or swapped [1,14]. This type of mutation operation is suitable for the problems encoded

using permutation encoding scheme. In Travelling Salesman Problem this mutation operation is used.

3.6 Termination

The termination step decides when to stop executing generations of the Genetic Algorithm [14,31,32]. This criterion is set before starting iterations of the Genetic Algorithm to solve any problem using GA. Some commonly used termination criteria are as follows:

Maximum Generations

In this type of termination process, given number of iterations of the genetic algorithm are executed. When these numbers of iterations are executed, the Genetic Algorithm stops and the best chromosome of the current population is returned as the solution.

Elapsed Time

In this type of termination, the iterations of the Genetic Algorithm are executed for a given time period. After that time period the Genetic Algorithm terminates.

No Change in Fitness

This type of termination criteria stops the iterations of the Genetic Algorithm if the fitness of the population does not change and remain same as previous generation.

3.7 Parameters of Genetic Algorithm

The working of the Genetic Algorithm is controlled by some parameters. The most important parameters of Genetic Algorithm are as follows:

Crossover probability

The cross over probability will determine what fraction of chromosomes of the current population will participate in cross over [14] If it is 100% then all chromosomes of the

current population will participate in cross over. If it is 0% then cross over will not take place and new children will not be generated. The rate of cross over is selected in the start before starting generations of the Genetic Algorithm. Normally cross over rate from 20%-40% is better but may vary from 0% to 100%. Many researchers select different value of cross over rate to optimize the performance of GA for solving different problems [27,28,39].

Mutation Probability

This parameter decides what fraction of the population will be mutated or changed by mutation operator. Normally mutation probability is kept very low in the range of 0%-5%. The exact value of the mutation probability is decided before starting generations of the Genetic Algorithm.

Population Size

This parameter decides the number of chromosomes in the population. The higher value of population size may improve the convergence of the Genetic Algorithm but it may be time consuming. The population size of 20-50 is normally quite enough but in some cases it may be in hundreds. The population size varies and affect the performance of GA to solve a particular problem [32,78].

Chapter 3 discussed the genetic operators and their optimization to solve NPH and NPC problems. In this work, GA is used to solve NPH and NPC problems and applied on TSP, MKP and N-Queen problems. Greedy approach is used and hybridized greedy genetic operators are proposed to solve NPH and NPC problems. Chapter 4 discusses the methodology used to solve NPH and NPC problems using Greedy Genetic Algorithm GGA.