Chapter 2
GENETIC ALGORITHM AND ITS ADAPTIVENESS

2.1 INTRODUCTION

The recent times have witnessed rapid developments in the research field of algorithms to solve optimization problems. These achievements have been made possible because of the progress in computer technology and development of user friendly software like MATLAB. The different optimization techniques include Gradient search algorithms, Evolutionary algorithms, Stochastic techniques, Simulated annealing, Ant colony optimization, Taboo search etc. [12]

2.2 GENETIC ALGORITHM (GA)

Genetic algorithms are probabilistic search algorithms which are inspired on the principle of survival of the fittest, derived from the theory of evolution described by Charles Darwin in *The Origin of Species* [38]. Genetic algorithms maintain a collection of potential solutions, which evolve according to a measure reflecting the quality of solutions. The evolution process of a genetic algorithm works on an encoding of the search space, represented by a chromosome.

Genetic algorithms are search methods that employ processes found in natural biological evolution. These algorithms search or operate on a given population of potential solutions to find the solution that
approach some specification or criteria. During the searching process the algorithm applies the principle of survival of the fittest to find the best possible solutions. At each generation, a new population set is created by the process of selecting individual potential solutions (individuals) according to their level of fitness in the problem domain. The solution is reached by breeding them together using operators borrowed from natural genetics. Just as in the case of natural evolution, this process leads to the generation of new population of individuals which are better suited to their environment than the individuals that they were created from [9].

Selection, Crossover and Mutation are the three fundamental genetic operations employed in genetic algorithms. The chosen solutions are modified through these operations and the most appropriate offspring is selected to be passed on to succeeding generations. Genetic algorithms simultaneously consider many points in the search space. They have been found to provide a rapid convergence to a near optimum solution in many types of problems. It is seen that they usually exhibit a reduced chance of converging to local optimum. Genetic algorithms were first introduced by Holland in 1975 [18] and they have been applied in different types of optimization problems [33].
2.3 BASICS OF GENETIC ALGORITHMS

The standard Genetic algorithms [21] can be represented as shown below

1. Choose coding to represent problem parameters, Select the criteria for reproduction, Crossover and Mutation
2. Input the initial population size, probabilities of cross over and mutation, search domain of the variables, termination criteria or maximum number of iteration as Tmax
3. Set T = 0, Generate initial population from the search domains randomly
4. Evaluate each string of the population for fitness
5. If Termination Criteria is satisfied, or $T > T_{max}$, then STOP
6. Perform reproduction on the population
7. Perform crossover on the population
8. Perform mutation on the population
9. Evaluate the strings of the new population
10. $T = T + 1$, go to step 4

A Genetic algorithm has the ability to create an initial population of feasible solutions and to randomly initialize them at the beginning of a computation. This initial population is then compared against the specifications or some fitness value. The individuals with the highest fitness factor are then recombined to form the mating pool for the next generation. This is the selection process.
Each feasible solution is encoded as a chromosome (string) also called a genotype and each chromosome is given a measure of fitness (fitness factor) through a fitness (evaluation or objective) function. The fitness of a chromosome determines its ability to survive and produce offspring. A finite fixed population of chromosomes is maintained. If the optimization criteria are not met, then the creation of a new generation starts. Individuals are selected (parents) according to their fitness for the production of offspring. Parent chromosomes are combined to produce superior offspring chromosomes through crossover at some crossover point. All offspring will be mutated (altering some genes in a chromosome) with a certain probability. The fitness of the offspring is then computed. The offspring are inserted into the population replacing the parents, producing a new generation. This cycle is performed until the optimization criteria are reached. In some cases, where the parent already has a high fitness factor, it is better not to allow this parent to be discarded when forming a new generation, but to be carried over. Mutation ensures the search of the entire space and thus it is an effective way of leading the population out of a local minima trap.

A Genetic Algorithm operates through a simple cycle of stages:

i) Creation of a “population” of strings,

ii) Evaluation of each string,

iii) Selection of best strings and

iv) Genetic manipulation to create new population of strings.
The cycle of a Genetic Algorithms is presented below

Before a Genetic algorithm can be run, a suitable coding (or representation) for the problem must be devised. We also require a fitness function, which assigns a figure of merit to each coded solution.
During the run, parents must be selected for reproduction, and recombined to generate offspring. The aspects involved in the process are described below.

- Encoding
- Fitness function

2.4 ENCODING

For any Genetic algorithm a chromosome representation is required to describe each individual in the population of interest. The representation scheme determines how the problem is structured in the Genetic algorithm and also determines what genetic operators are used. Each individual or chromosome is made up of a sequence of genes from a certain alphabet. This alphabet could consist of binary digits (0 and 1), floating point numbers, integers, symbols (i.e., A, B, C, D), matrices, etc. Each element of the string represents a particular feature in the chromosome. The first thing that must be done in any new problem is to generate a code for this problem.

2.5 FITNESS FUNCTION

A fitness function must be devised for each problem to be solved. The fitness function and the coding scheme are the most crucial aspects of any Genetic algorithm. They are its core and determine its performance. The fitness function must be maximized. In most forms of evolutionary computation, the fitness function returns an individual's assessed fitness as a single real-valued parameter that reflects its success at solving the problem at hand. That is, it is a measure of fitness or a
figure-of-merit that is proportional to the “utility” or “ability” of that individual represented by that chromosome. This is an entirely user-determined value. The general rule to follow when constructing a fitness function is that it should reflect the value of the chromosome in some “real” way. To prevent premature convergence (the population converging onto a local minimum rather than a global minimum), the population fitness is required to be scaled properly. As the average evaluation of the strings in the population increases, the variance in fitness decreases in the population. There may be little difference between the best and the worst individual in the population after several generations and the selective pressure based on fitness is correspondingly reduced.

2.6 OPERATORS OF GENETIC ALGORITHM

A basic genetic algorithm comprises of three genetic operators namely, Selection, Crossover and Mutation [9]. Starting from an initial population of strings (representing possible solutions), the Genetic algorithm uses these operators to calculate successive generations. The pairs of individuals of the current population are selected to mate with each other to form the offspring, which then form the next generation.
2.7 SELECTION

This operator selects the chromosome in the population for reproduction. The more fit the chromosome, the higher its probability of being selected for reproduction. Thus, selection is based on the survival-of-the-fittest strategy, but the key idea is to select the better individuals of the population, as in tournament selection, where the participants compete with each other to remain in the population. The most commonly used strategy to select pairs of individuals is the method of roulette-wheel selection, in which every string is assigned a slot in a simulated wheel sized in proportion to the string’s relative fitness. This ensures that highly fit strings have a greater probability to be selected to form the next generation through crossover and mutation. After selection of the pairs of parent strings, the crossover operator is applied to each of these pairs.

2.8 CROSSOVER

A crossover operator recombines two parent strings to produce better offspring strings. It involves the swapping of genetic material (bit-values) between the two parent strings. In practice, all parents in the mating pool are not selected for crossover operation so that some of the good strings may be preserved. This is achieved by selecting a fixed percentage of parents from the mating pool and it known as the crossover probability. Many crossover operators are available in GA literature. The most commonly used crossovers are the following
2.8.1 Single Point Crossover

This operator randomly chooses a locus (a bit position along the two chromosomes) and exchanges the sub-sequences before and after that locus between two chromosomes to create two offspring i.e., one crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent and the rest is copied from the second parent as shown in figure 2.2

Parent 1  10100110  offspring 1  10100100
Parent 2  11010100  offspring 2  11010110

Figure 2.2
Single Point Crossover

2.8.2 Two Point Crossover

Here two crossover points are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent as shown in figure 2.3

Parent 1  10100110  offspring 1  10110110
Parent 2  11010100  offspring 2  11000100

Figure 2.3
Two Point Crossover
2.9 MUTATION

The two individuals (children) resulting from each crossover operation will now be subjected to the mutation operator in the final step to form the new generation. This operator randomly flips or alters one or more bit values at randomly selected locations in a chromosome. For example, the string 1000 0001 0011 might be mutated in its second position to yield 1100 0001 0011. Mutation can occur at each bit position in a string with some probability and in accordance with its biological equivalent; usually this is very small, for example, 0.001. If 100% mutation occurs, then all of the bits in the chromosome have been inverted.

The mutation operator enhances the ability of the Genetic algorithm to find a near optimal solution to a given problem by maintaining a sufficient level of genetic variety in the population, which is needed to make sure that the entire solution space is used in the search for the best solution. In a sense, it serves as an insurance policy; it helps prevent the loss of genetic material. A single mutation process is given in the following figure.

<table>
<thead>
<tr>
<th>Mutation point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offspring</td>
</tr>
<tr>
<td>Mutated Offspring</td>
</tr>
</tbody>
</table>

**Figure 2.4**

Mutation Operator
2.10 CONVERGENCE

With a correctly designed and implemented Genetic Algorithm, the population will evolve over successive generations so that the fitness of the best and the average individual in each generation increases towards the global optimum. Convergence is the progression towards increasing uniformity. A gene is said to have converged when 95% of the population share the same value. The population is said to have converged when all of the genes have converged.

At the start of a run, the values for each gene for different members of the population are randomly distributed giving a wide spread of individual fitness. As the run progresses some gene values begin to predominate. As the population converges the range of fitness in the population reduces. This reduced range often leads to premature convergence and a slow finish.

2.11 PROPERTIES OF GENETIC ALGORITHMS

- generally good at finding acceptable solutions to a problem reasonably quickly
- free of mathematical derivatives
- no gradient information is required
- free of restrictions on the structure of the evaluation function
- fairly simple to develop
- do not require complex mathematics to execute
• able to vary not only the values, but also the structure of the solution
• get a good set of answers, as opposed to a single optimal answer
• make no assumptions about the problem space
• blind without the fitness function. The fitness function drives the population toward better solutions and is the most important part of the algorithm.
• not guaranteed to find the global optimum solutions probability and randomness are essential parts of Genetic Algorithms
• can be hybridized with conventional optimization methods
• potential for executing many potential solutions in parallel
• The solution time is very predictable, and is not radically affected as the problem gets larger.
• Handles non-linear and discontinuous functions equally as well as linear and continuous.
• You need only to be able to describe a good solution; you do not need to know how to build it. Thus, it does not require heavy use of expert knowledge.
• Can produce novel results among a set of good solutions. “I would have never thought of that one!”
• Tend to be compact, containing only the fitness function and a little code to handle the Genetic Algorithms functions
• Can usually be embedded easily, and are easy to hybridize
2.12 ADAPTIVE GA

The primary work of the simple genetic algorithm is performed in three routines namely selection, crossover and mutation. As already explained selection is done using stochastic methods like roulette wheel selection. Crossover and mutation perform in such a way that their action is coordinated by a procedure called generation. It produces a new population at each successive generation. The efficiency of a GA can be improved by modifying or adapting some techniques in the basic GA operators. In literature, so many adaptive techniques are available and in this section we review some of them.

Jinn-Moon Yang et al described an algorithm [55] known as the adaptive mutations genetic algorithm (AMGA) for training an Artificial Neural Networks. AMGA incorporates three mutation operators, decreasing-based Cauchy mutation (Mdc), self adaptive Cauchy mutation (Mc), and self-adaptive Gaussian mutation (Mg). The core philosophy of AMGA is to design mutation operators by using the adaptive rules and family competition for cooperating each other.

Kim et al proposed adaptive genetic algorithms for multi-resource constrained project scheduling problem with multiple modes [22]. The paper proposes the adaptive genetic algorithm (AGA) for solving the mcPSP- mM problems. They firstly design priority-based encoding for activity priority and multistage-based encoding for activity mode for GA encoding. Secondly they use order-based crossover operator for activity priority and local search-based mutation
operator for activity mode. Thirdly they propose an iterative hill-climbing method to carry out local searches around a convergence solution in GA loop, and finally it use auto-turning for the rates of crossover and mutation operators.

Wang Lei, and Shen Tingzhi proposed an Improved Adaptive Genetic Algorithm and its application to image segmentation [26]. The proposed algorithm introduces three parameters fitmax, fitmin and fitave and to measure how close the individuals are, so as to improve the Adaptive Genetic Algorithm (AGA) proposed by M. Srinivas [45]. At the same time, the elitist strategy is employed to protect the best individual of each generation, and Remainder Stochastic Sampling with Replacement (RSSR) is employed in the proposed IAGA to improve the basic reproduction operator.

Adimurthy et al proposed a GA with Adaptive Bounds (GAAB) [1] and this algorithm is implemented successfully in obtaining precise lunar gravity assist transfers to Geostationary orbits [39]. In this approach the parameter bounds of GA are modified during the search process. The initial bounds on the input parameters are redefined within the existing bounds after a certain number of generations around the current best solution values of the parameters. Present work also deals with the modification of the limits in an adaptive way, but the approach is different and the present one shrinks as well as elongates even outside the initial limits and in turn the range of the limit increases or decreases accordingly.
West J M and Antonio J K introduced a GA approach to scheduling communications for a class of parallel space-time adaptive processing algorithms [51]. It focuses on off-line optimization of message schedules for a class of radar signal processing techniques known as space-time adaptive processing on a parallel embedded system.

Mak K L et al [29] proposed an adaptive genetic approach which is an effective means of providing the optimal solution to the manufacturing cell formation problem in the design of cellular manufacturing systems.

Sherif M R et al [43] have uses Gas to solve an optimization problem occurred in wireless ATM-based networks in which admission control is required to reserve resources in advance for calls requiring guaranteed services. In the case of a multimedia call, each of its sub streams has its own distinct quality of service requirements and the network attempts to deliver it by allocating an appropriate amount of resources. They have developed and analysed an adaptive allocation of resources algorithm based on GA.

Wu B et al [52] presented a fast GA namely Generalized Self-Adaptive GA (GSAGA) to solve the problem between searching performance and convergence of GAs and they have verified that searching performance and global convergences are greatly improved compared with many existing GAs.
Martin-Bautista M J et al [30] developed an approach to a Genetic Information Retrieval Agent Filter (GIRAF) for documents from the internet using a GA with fuzzy set genes to learn the user’s information needs. The population of chromosomes with fixed length represents such user’s preferences. Each chromosome is associated with a fitness that may be considered the system’s belief in the hypothesis that the chromosome, as a query, represents the user’s information needs. They have developed a prototype of GIRAF and tested.

Oyama A et al [36] have developed an adaptive technique namely Real-coded Adaptive Range Gas (ARGAs) to find a solution to an aerodynamic airfoil shape optimization problem. The results confirm that the real-coded ARGAs consistently find better solutions than the conventional real-coded GAs.

Magyar G et al [28] proposed a hybrid GA with an adaptive application of genetic operators for solving the three-matching problem (3MP) which is an NP-complete graph problem. The three MP is to find the partition of appoint set into triplets of minimal total cost where the cost of a triplet is the Euclidian length of the minimal spanning tree of the 3 points. They introduced several general/heuristic crossover and local hill climbing operators for the 3MP and applied adaptation at the level of choosing among the operators. The GA combined these operators to form an effective problem solver.
Herrera F and Lozano M [17] have developed a Two-loop Real-coded GA with Adaptive Control of Mutation Step Sizes (TRAMSS). A problem in the use of GA is premature convergence; a premature stagnation of the search caused by the lack of population diversity. The mutation operator is the one responsible for the generation of diversity and therefore may be considered to be an important element in solving this problem. TRAMSS adjusts the step size of a mutation operator applied during the inner loop, for producing efficient local turning. It also controls the step size of a mutation operator used by a restart operator performed in the outer loop, for re-initializing the population in order to ensure that different promising search zeroes are focused by the inner loop throughout the run.

Jiang T Z and Evans D J [20] have proposed a novel efficient method for image restoration. Image restoration is an essential preprocessing step for many image analysis applications. The main idea in the new method is to combine the hybrid GA with adaptive preconditioning and they have shown that this method has remarkable advantage over the existing techniques available.

Deb K and Beyer H G [10] have developed a self-adaptive GA. Self-adaptation is an essential feature of natural evolution. In this paper they demonstrated the self-adaptive feature of real parameter GAs using a simulated binary crossover operator and without any mutation operator.
Ezziane Z [14] have used GA to solve the 0-1 knapsack problem which is an NP hard problem. In the proposed adaptive GA special consideration is given to the penalty function where constant and self-adaptive penalty functions are adopted.

Wu Z Y and Simpson A R [53] have proposed a new approach called the self-adaptive boundary search strategy for selection of penalty factor within GA optimization. The approach co-evolves and self-adapts the penalty factor such that the GA search is guided towards and preserved around constraint boundaries and it reduces the amount of simulation computations within the GA search. It also enhances the efficacy at reaching the optimal or near optimal solution. Its effectiveness is demonstrated by a case study of the optimization of a water distribution system.

Espinoza F P etal [13] have examined the effects of local search on hybrid GA performance and population sizing. It compared the performance of a Self-Adaptive Hybrid GA (SAHGA) to a Non-Adaptive Hybrid GA (NAHGA) and the Simple GA (SGA) on eight different test functions including unimodal, multimodal and constraint optimization problems. The adaptive nature of SAHGA reduces population sizes and it allows for faster solution of the test problems without sacrificing solution quality.

Barbosa H J C and Lemonge A C C [3] have proposed a parameter–less adaptive penalty scheme for steady-state GAs applied to constrained optimization problems. For each constraint, a penalty
parameter is adaptively computed along the run according to information extracted from the current population such as the existence of feasible individuals and the level of violation of each constraint. Using real coding, rank-based selection and operators available in the literature, very good results are obtained.

Yang S X [54] has developed a Statistics based Adaptive Non Uniform Mutation (SANUM) for genetic algorithms, within which the probability that each gene will subject to mutation is learnt adaptively over time and over the loci. SANUM uses the statistics of the allele distribution in each locus to adaptively adjust the mutation probability of that locus. The experimental results show that it performs persistently well over a range of typical test problems while the performance of traditional mutation operators with fixed rates greatly depends on the problems.

Martikainen J and Ovaska S J [31] have proposed a Multiplicative General Parameter (MGP) approach to finite impulse response (FIR) filtering to realize cost effective adaptive filters in compact very large scale integrated circuit (VLSI) implementations used for example in mobile devices. MGP filter structure comprises of additions and only a small number of multiplications, thus making the structure very simple.

Mattes M and Mosig J R [32] have proposed a new adaptive sampling to accelerate frequency-domain calculations which use rational functions to approximate the frequency response. The
sampling algorithm is derivative free and well-adapted to devices with rapidly varying frequency responses like microwave filters. The criteria for convergence checking and to determine the location of additional sampling points are easy and fast to evaluate. They provide an estimation of the approximation error and can be used to determine whether the algorithm has problems to reach convergence.

Tang M L [48] presented a new GA for the solution of the Minimal Switching Graph (MSG) problem which is NP complete. Different from the original GA, this new GA has a self adaptive encoding mechanism that can adapt the permutation of the genes in the encoding scheme. Experimental results show that this adaptive GA outperforms the original GA.

Sang H et al [42] have proposed an Adaptive Hybrid Immune GA (AHIGA) to find solution for the maximum cut problem. The goal of the maximum cut problem is to partition the vertex set of an undirected graph into two parts in order to maximize the cardinality of the set of edges cut by the partition. AHIGA includes key techniques such as vaccine abstraction, vaccination and affinity-based selection. A large number of instances have been simulated and the results show that the proposed algorithm is superior to the existing algorithms.

Szeto K Y and Zhang J [47] introduced a new AGP using mutation matrix and implemented in a single computer using the quasi-parallel time sharing algorithm for the solution of the 0-1 knapsack problem. The mutation matrix is constructed using the locus statistics
and the fitness distribution in a population with N rows and L columns where N is the size of the population and L is the length of the encoded chromosomes. The mutation matrix is parameter free and adaptive as it is time dependent and captures the accumulated information in the past generation. Two strategies of evaluation, mutation by row (chromosome) and mutation by column (locus) are discussed. Based on the investment frontier of time allocation, the optimal configuration for solving the knapsack problem is found.

Moon C et al [34] proposed an advanced planning model to decide process plans and schedules for the manufacturing supply chain (MSC). A main function for supporting global objectives in a manufacturing supply chain is planning and scheduling. The model is formulated with mixed integer programming which considered alternative resources and sequences, a sequence dependent set up and transportation times. The objective of the model is to analyse alternative resources and sequences to determine the schedules and operation sequences that minimize makespan (time). A new AGA approach is developed to solve the model. Numerical experiments are carried out to demonstrate the efficiency of the developed approach.

Tomioaka S et al [49] proposed an adaptive domain method (ADM) using real-coded GAs to solve non-linear problems. In conventional least square regressions for non-linear problems, it is not easy to obtain analytical derivatives with respect to target parameters that comprise a set of normal equations. Even if the derivatives can be
obtained analytically or numerically, one must take care to choose the correct initial values for the iterative procedure of solving an equation, because some undesired locally optimized solutions may also satisfy the normal equation. In the application GAs for non-linear least square it is not necessary to use normal equations and a GA is also capable of avoiding localized optima. It is to be noted that the convergence of population and reliability of solutions depends on the initial domain of parameters, similarly to the choice of initial values in the above mentioned method using normal equations. This disadvantage of applying GAs for non linear least square can be avoided by ADM. They have demonstrated the effectiveness of the new method by citing an example problem.

Dai Y S et al [6] proposed an adaptive immune-genetic algorithm (AIGA) to avoid premature convergence for global optimization to multi variable functions. Rapid immune response, adaptive mutation and density operators in the AIGA are emphatically designed to improve the searching ability, converging speed and to decrease locating the local maxima due to premature convergence. They have verified the efficiency by the simulation results obtained from the global optimization to four multi variable and multi extreme functions.

Bingul Z [4] presented an adaptive genetic algorithm with dynamic fitness function for multi objective problems in a dynamic environment. The developed method is used to find an optimal force
allocation for a combat simulation and also to control the cross over and the mutation rates based on statistics of the aggregate fitness.

Srinivasa K.G. et al [45] introduced a self adaptive migration GA (SAMGA) model for data mining. Data mining involves non trivial process of extracting knowledge or patterns from large data bases. GAs are efficient and robust searching and optimization methods that are used in data mining. In SAMGA the parameters of population size, the number of points of cross over and mutation rate for each population are adaptively fixed. The effective performance of the algorithm is verified using standard test bed functions and a set of actual classification data mining problems.

Salah SA et al [41] proposed a new cross over technique for GA. The technique named as probabilistic adaptive cross over denoted by PAX includes the estimation of the probability distribution of the population the proposed methodology is compared with some of the existing cross over techniques and verified the efficiency over other methods using test problems.

Many other types of adaptations are also available in GA literature. Some new adaptive techniques which were discovered during the research have been introduced in the following chapters.