Many real-world problems involve problem difficulty of multiple, conflicting objectives and highly complex search space. The search space can be too large and too complex to be solved by exact methods. Thus, efficient optimization strategies are required to deal with above mentioned difficulties. Evolutionary algorithms possess several characteristics that are desirable for such kind of problems and make EAs preferable, compared to classical optimization methods. Evolutionary algorithms stand for a class of stochastic optimization methods that simulate the process of natural evolution. The origins of EAs can be traced back to the late 1950s. Since 1970s several evolutionary methodologies have been proposed: genetic algorithms, genetic programming, evolutionary programming, and evolution strategies (Back et al. 1997). All of these approaches operate on a set of candidate solutions (Chromosomes in natural evolution). Using strong simplifications, this set is subsequently modified by the two basic principles of evolution: selection and variation. Selection represents the competition for resources among living beings. Some are better than others and more likely to survive and to reproduce their genetic information.

In evolutionary algorithms, natural selection is simulated by a stochastic selection process. Each solution is given a chance to reproduce a certain number of times, dependent on their quality. Thereby, quality is assessed by evaluating the individuals and assigning them scalar fitness values. The other principle, variation, imitates natural capability of creating new living beings by means of recombination and mutation. Although the underlying principles are simple, these algorithms have proven themselves as a general, robust and powerful search mechanism. Moreover, EAs seem to be especially suited to multi-objective optimization because they are able to capture multiple Pareto-optimal solutions in a single simulation run. Some researchers suggested in literature that multi-objective search and optimization might be a problem area where EAs do better than other blind search strategies and heuristics. A heuristic is a part of an
optimization algorithm that uses the information currently gathered by the algorithm to help to
decide which solution candidate should be tested next, or how the next individual can be
produced (Thomas 2007). In fact, various evolutionary approaches have been proposed since
1985 for multi-objective optimization, capable of searching for multiple Pareto optimal solutions
concurrently in a single simulation run. The chromosome, natural selection and variation are
building blocks of natural evolution and evolutionary computation.

- **Chromosome:** A chromosome is a long, complicated thread of DNA (deoxyribonucleic
  acid). Hereditary factors that determine particular traits of an individual are strung along
  the length of these chromosomes, like beads on a necklace. Each trait is coded by some
  combination of DNA.

- **Natural Selection:** In nature, the individual that has better survival traits will survive for a
  longer period of time. This in turn provides it a better chance to produce offspring with
  its genetic material. Therefore, after a long period of time, the entire population will
  consist of lots of genes from the superior individuals and less from the inferior
  individuals. In a sense the fittest survives and the unfit died out. This force of nature is
called natural selection. The existence of competition among individuals of a species was
recognized certainly before Darwin.

To simulate the process of natural selection in a computer, we need to define a representation of
an individual solution/chromosome. At each point during the search process a generation of
individuals is maintained. Each individual is a data structure representing the genetic structure of
a possible solution or hypothesis. Like a chromosome, the genetic structure of an individual is
described using a fixed, finite alphabet. In basic GAs, the alphabet 0, 1 is usually used. This
string is interpreted as a solution to the problem, trying to solve. Depending upon the problem
other specific representations suitable to problem can also be taken. For example, say want to
find the optimal quantity of the three major ingredients in a recipe (say, sugar, lemon, and salt).
The alphabet 1, 2, 3 ..., 9 can be used to denote the number of ounces of each ingredient. Some
possible solutions are 1-1-1, 2-1-4, and 3-3-1. As another example, the traveling salesperson
problem is the problem of finding the optimal path to traverse, say, 10 cities. The salesperson
may start in any city. A solution is a permutation of the 10 cities: 1-4-2-3-6-7-9-8-5-10.
Variation: A change in chromosome occurs during reproduction. The chromosomes from the parent’s exchanges a number of genes randomly by a process called crossover. Therefore, the off-spring exhibits some traits of the father and some traits of the mother. A rarer process called mutation also changes some traits.

Many human inventions are inspired by nature GA is such example. This particular branch of AI was inspired by the way living things evolved into more successful organisms in nature. Genetic algorithms are adaptive heuristic search algorithm (Goldberg 1989), based on computational models inspired by evolution. The most basic concept is that the strong tend to adapt and survive while the weak tend to die out. Genetic algorithm was developed to simulate some of the processes observed in natural evolution, a process that operates on chromosomes. It was developed by John Holland in 1975 (Holland 1975). Genetic algorithms have been found to be capable of finding solutions for a wide variety of problems for which no acceptable algorithmic solutions exist. GA has been used for solving various NP Complete problems (VijayLakshmi and Radhakrishnan 2007). GA attempts to arrive at optimal solutions through a process similar to biological evolution.

GAs searches by simulating evolution, starting from an initial set of solutions or hypotheses, and generating successive generations of solutions. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness and breeding them together using genetic operators inspired by natural genetics. This process leads to the evolution of better populations than the previous populations (Eiben et al. 1999). In broader usage of the term, a genetic algorithm is population-based model that uses selection and recombination operators to generate new sample points in a search space. Genetic algorithms are often described as a global search method that does not use gradient information.

The Genetic Algorithm differ from other search methods in a way that it searches among a population of points, and works with a coding of parameter set, rather than the parameter values. It also uses objective function information without any gradient information. The transition scheme of the genetic algorithm is probabilistic, whereas traditional methods use gradient information. Because of these features of genetic algorithm, they are used as general purpose optimization algorithm. They also provide means to search irregular space and hence are applied
to a variety of function optimization, parameter estimation and machine learning applications. GA is most popular optimization technique among evolutionary algorithms for multi-objective optimization problems. Genetic algorithms are typically implemented using computer simulations in which an optimization problem is specified.

The brief organization of chapter is as follows: Section 2.1 focuses on GA methodology. Section 2.2 describes the major components of GA. Characteristics of GA are introduced in section 2.3. Section 2.4 introduces the GA pseudo-code which can be modified and implemented for various types of problems. Section 2.5, 2.6 and 2.7 introduces the advantages, disadvantages and applications of GA respectively. Summary is given in section 2.8.

2.1 GA Methodology

In order to find the solution of a problem, the principles of Darwin’s theory are followed by Genetic Algorithms As seen in nature; Genetic Algorithms are similar adaptive search techniques which simulate an evolutionary process on the basis of the ideas of selection of the fittest, crossing and mutation. Genetic algorithm is a guided random search technique to solve large scale optimization and combinatorial problems. These operate on a population of potential solutions, applying the principle of survival of the fittest to generate improved estimations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness and breeding them together using genetic operators inspired by natural genetics. This process leads to the evolution of better populations than the previous populations (Eiben et al. 1999). Genetic algorithms are typically implemented using computer simulations in which an optimization problem is specified. The GA consists of an iterative process that evolves a working set of individuals called a population toward an objective function, or fitness function (Goldberg 1989, Wikipedia 2011).

GA produces close to optimal results in a reasonable amount of time and is suitable for parallel processing. It can use a noisy fitness function and simple to develop. GA is blind without the fitness function. The fitness function drives the population toward better solutions and is the most important part of the algorithm. Probability and randomness are essential parts of GA.
The GA methodology is particularly suited for optimization of problems in which one or more very good solutions are needed to search from a solution space consisting of a large number of possible solutions. GA reduce the search space by continually evaluating the current generation of candidate solutions, discarding the ones ranked as poor, and producing a new generation through crossbreeding and mutating those ranked as good. The ranking of candidate solutions is done using some pre-determined measure of goodness or fitness.

The solutions of a specific problem that is to be optimized are encoded on a simple chromosome like data structure. The smallest unit of a genetic algorithm is called a gene, which represents a unit of information in the problem domain (Goldberg 1994). A series of genes, known as a chromosome, represents one possible solution to the problem. Each gene in the chromosome represents one component of the solution pattern. GA creates an initial population of feasible solutions. Each chromosome is given a measure of fitness via a fitness (evaluation or objective) function. The fitness of a chromosome determines its ability to survive and produce offspring. GA use probabilistic rules to evolve a population from one generation to the next. The evolutionary cycle starts with a randomly selected initial population. The changes to the population occur through the processes of selection based on fitness and alteration using crossover and mutation. The application of selection and alteration leads to a population with a higher proportion of better solutions. The evolutionary cycle continues until an acceptable solution is found in the current generation of population, or some condition parameter such as the number of generations is exceeded. The generations of the new solutions are developed by following genetic recombination operators:

- **Reproduction**: Selecting the fittest to reproduce
- **Crossover**: Combining parent chromosomes to produce children chromosomes
- **Mutation**: Flipping some genes in a chromosome

The following steps illustrate the structure of a basic genetic algorithm and the algorithm flow is given in the figure 2.1.

i. Encoding of solutions.

ii. Initialization of Initial set of solutions as population.
iii. Fitness evaluation of all solutions in the population.
iv. Selection of the parent solutions according to their fitness.
v. Crossover between parents to form new solutions.
vi. Mutation of a small proportion of these new solutions.
vii. Fitness evaluation of new solutions.
viii. Replacement of old solutions by new ones keeping some of the (best) old solutions.
ix. Returning to step 3 if stopping criteria are unfulfilled

**Figure 2.1:** Flow Chart of Basic Genetic Algorithm

An implementation of a genetic algorithm begins with a population of typically random chromosomes. The chromosomes are then evaluated and are allocated reproductive opportunities in such a way that those chromosomes which represent a better solution to the target problem are given more chances to reproduce than those chromosomes which are poorer solutions. The goodness of a solution is typically defined with respect to the current population. The evolutionary cycle of GA and generational behavior is depicted in figure 2.2 and 2.3 respectively.
2.2 Components of GA

Generally, the algorithm for solving optimization problem is a sequence of computational steps which asymptotically converge to optimal solution. Most classical optimization methods generate a deterministic sequence of computation based on the gradient or higher-order derivatives of objective function. This point-to-point approach takes the danger of falling in local
optima. Genetic algorithms perform a multiple directional search by maintaining a population of potential solutions. The population-to-population approach attempts to make the search escape from local optima. Population undergoes a simulated evolution: At each generation the relatively good solutions are reproduced while the relatively bad solutions die. Genetic algorithms use probabilistic transition rules to select someone to be reproduced and someone to die so as to guide their search toward regions of the search space with like improvement. To implement GA following components are required:

- Representation (solution definition)
- Evaluation function (or fitness function)
- Population
- Parent selection mechanism
- Variation operators (crossover and mutation)
- Survivor selection mechanism (replacement)
- Termination Condition

### 2.2.1 Solution Representation

Usually there are only two main components of most genetic algorithms that are problem dependent: the problem encoding and the evaluation function. Consider a parameter optimization problem where we want to optimize a set of variables either to maximize some target such as a profit, or to minimize cost. Such a problem views as a black box with a series of control dials representing different parameters, the only output of the black box is a value returned by an evaluation function indicating how well a particular combination of parameter settings solves the optimization problem. The goal is to set the various parameters so as to optimize some output. In more traditional terms, we wish to minimize, or maximize, some function \( F(X_1, X_2, \ldots, X_m) \).

The most common form of representing a solution as a chromosome is a string of binary digits. Each bit in this string is a gene. The process of converting the solution from its original form into the bit string is known as coding. The specific coding scheme used is application dependent. The solution bit strings are decoded to enable their evaluation using a fitness measure. All chromosomes are of the same type but can have different length. The population size remains
constant from generation to generation. GAs performs best when solution vectors are binary. If the problem has more than one variable, a multi-variable coding is constructed by concatenating as many single variables coding as the number of variables in the problem. Chromosomes can be represented as:

- **Bit Strings**: (1011 ... 0100)
- **Real**: (19.3, 45.1, 12.9, ... 6.2)
- **Integers**: (1,4,2,7,5,9,3,6,8) Permutations of 1..n
- **Characters**: (A, G, Q, ... F) Permutations
- **List of rules**: (R1, R2, ... R20)

### 2.2.2 Fitness function

Fitness function is the measure of the quality of an individual. The fitness function should be designed to provide assessment of the performance of an individual in the current population. The selection step is preceded by the fitness assignment which is based on the objective value. This fitness is used for the actual selection process. For example to minimize a simple function of two variables:

\[
\min f(x) = 50 * (x(1)^2 - x(2))^2 + (1 - x(1))^2;
\]

### 2.2.3 Population

The represented solutions together form the population. Population holds the possible solutions. During evolutilional search, the population size remains constant in almost all GA applications.

### 2.2.4 Selection

In biological evolution, only the fittest survive and their gene pool contributes to the creation of the next generation. Selection in GA is also based on a similar process.
- **Parent Selection Mechanism:** The role of parent selection (mating selection) considers the individuals quality, and allow the better individuals to become parents of the next generation. Parent selection is probabilistic. Thus, high quality individuals get a higher chance to become parents than those with low quality. The low quality individuals are often given a small, but positive chance; otherwise the whole search could become too greedy and get stuck in a local optimum.

- **Survivor Selection Mechanism:** The role of survivor selection is to distinguish among individuals based on their quality. In GA, the population size is (almost always) constant, thus a choice has to be made which individuals will be allowed in the next generation. This decision is based on their fitness values, favoring those with higher quality.

The parent selection is stochastic whereas the survivor selection is often deterministic. For instance, ranking the unified multi-set of parents and offspring and selecting the top segment (fitness biased), or selection only from the offspring (age-biased). Following are the selection methods used in genetic algorithms:

- Rank-based fitness assignment
- Roulette wheel selection
- Local selection
- Tournament selection
- Steady-state selection

A decision about the method of selection to be applied is one of the most important decisions to be made. Selection is responsible for the speed of evolution and is often cited as the main reason in cases where premature convergence halts the success of a genetic algorithm. Commonly used selection schemes are Goldberg and Deb. (1991):

a) Ranking selection
b) Tournament selection
c) Genitor (steady state) selection
a) **Rank Selection**: J. E. Baker introduced the notion of ranking selection to genetic algorithm Baker (1985). The idea is straightforward. Sort the population from best to worst, assign the number of copies that each individual should receive according to a non-increasing assignment function, and then perform proportionate selection according to that assignment.

b) **Tournament Selection**: A form of tournament selection attributed to unpublished work by Wetzel was studied in (Brindle 1981) Brindle's dissertation, and other studies using tournament schemes are found in a number of works by Goldberg and others (Suh and Van Gucht 1987), (Muhlenbein 1989), (Goldberg et al. 1990). The idea is simple. Choose some number of individuals randomly from a population (with or without replacement), select the best individual from this group for further genetic processing, and repeat as often as desired (usually until the mating pool is filled). Tournaments are often held between pairs of individuals (tournament size $s = 2$), although larger tournaments can be used and may be analyzed.

c) **Genitor**: It works individual by individual, choosing an offspring for birth according to linear ranking, and choosing the currently worst individual for replacement. Because the scheme works one by one it is difficult to compare to generational schemes.

### 2.2.5 Alteration/Variation Operators

The alteration step creates the new solution from the old ones to produce the next generation of candidate solutions (Kingdon 1997). It is carried out by performing crossover and mutation.

- **Crossover**: This is a version of artificial mating. It takes two individuals with different but desirable features, and produces an offspring which combines both of those features. Individuals with high fitness should have high probability of mating. Crossover represents a way of moving through the space of possible solutions based on the information gained from the existing solutions. The chromosomes from two individuals are combined to create the chromosome for the next generation. This is done by splicing two chromosomes from two different solutions at a crossover point and swapping the spliced parts. The idea is that some genes with good characteristics from one chromosome may as a result combine with some good genes in the other chromosome to
create a better solution represented by the new chromosome. Crossover combines the fittest chromosomes and passes superior genes to the next generation. Examples of crossover are one point, two point, uniform, multipoint crossovers etc. New crossover methods can be developed specific to the problem.

- **Mutation:** Mutation is another recombination technique. It is a random adjustment in the genetic composition. It is useful for introducing new characteristics in a population - something not achieved through crossover alone. Crossover only rearranges existing characteristics to give new combinations. It is used to make sure all the elements in a population are not homogeneous and diversity is maintained. For example, the mutation operator changes the current value of a gene to a different one. For bit string chromosome this change amounts to flipping a 0 bit to a 1 or vice versa.

![Figure 2.4: Crossover and Mutation Operators of Genetic Algorithm](image)

Although useful for introducing new traits in the solution pool, mutations can be counterproductive, and applied only infrequently and randomly. Mutation ensures the entire state-space will be searched, and helps genetic algorithms to escape local optima in the search for the global optima. Mutation rate is usually ranges not higher than 30% (Lima et al. 2005).

### 2.2.6 Termination Condition

GA is stochastic in nature therefore there are no guarantees to reach an optimum. Common termination conditions are given below:

a) The total number of fitness evaluations reaches a given limit

b) The maximally allowed CPU times elapses
c) For a given period of time, the fitness improvement remains under a threshold value
d) The population diversity drops under a given threshold.

2.3 Characteristics of Genetic Algorithm

It is best used when the objective function is, discontinuous, highly nonlinear, stochastic, and has unreliable or undefined derivatives.

- GA does not require derivatives, just an evaluation function (a fitness function)
- It samples the space widely, like an enumerative or random algorithm, but more efficiently
- It can search multiple peaks in parallel, so is less hampered by local extremes than gradient-based methods
- Crossover allows the combination of useful building blocks, or schemata (mutation avoids evolutionary dead-ends)

2.4 Genetic Algorithm Pseudo-Code

The GA works with a coding of the parameter rather than the actual parameter. It works with a population of strings instead of a single point. Application of GA operators causes information from the previous generation to be carried over to the next. The GA uses probabilistic transition rules, not deterministic rules. To implement the above statements the pseudo-code algorithm is given below (Michalewicz 1996):

i. Choose initial population (usually random)
ii. Repeat (until terminated)
iii. Evaluate each individual's fitness on basis of fitness function
iv. Prune population as per strategy
v. Select pairs to mate from best-ranked individuals
vi. Replenish population (using selected pairs)

   a. Apply crossover operator
   b. Apply mutation operator
vii. Check for termination criteria (number of generations, amount of time, minimum fitness threshold satisfied, fitness has reached a plateau etc.)

viii. Loop, if not terminating (Go to step 2)

Population Size, Evaluation Function, Crossover Method and Mutation Rate are very important factors for implementation (Roger 2006). Determining the size of the population is a crucial factor.

- Choosing a population size too small increases the risk of converging prematurely to local minima, since the population does not have enough genetic material to sufficiently cover the problem space.
- A larger population has a greater chance of finding the global optimum at the expense of more CPU time.

Genetic Algorithm processes a number of solutions simultaneously. Hence, in the first step a population having P individuals is generated by pseudo random generators whose individuals represent a feasible solution. This is a representation of solution vector in a solution space and is called initial solution. This ensures the search to be robust and unbiased, as it starts from wide range of points in the solution space. In the next step, individual members of the population are evaluated to find the objective function value. In the third step, the objective function is mapped into a fitness function that computes a fitness value for each member of the population. This is followed by the application of GA operators and checking of termination condition.

### 2.5 Advantages of Genetic Algorithm

Genetic algorithms are advanced and popular optimization technique. Following are the advantages of using the GA:

i. The advantage of the GA approach is the ease with which it can handle arbitrary kinds of constraints and objectives; all such things can be handled as weighted components of the fitness function, making it easy to adapt the GA scheduler to the particular requirements of a very wide range of possible overall objectives.
ii. GA can be used when no algorithms or heuristics are available for solving a problem. A GA based system can be built as long as a solution representation and an evaluation scheme can be worked out. Since it only requires the description of a good solution and not how to achieve it, the need for expert access is minimized.

iii. Optimization problems in which the constraints and objective functions are non-linear and/or discontinuous are not amenable to solution by traditional methods such as linear programming. GA can solve such problems. GA does not guarantee optimal solutions, but produce near optimal solutions which are likely to be very good.

iv. Solution time with GA is highly predictable – it is determined by the size of the population, time taken to decode and evaluate a solution and the number of generations of population.

v. GA use simple operations, but are able to solve problems which are found to be computationally prohibitive by traditional algorithmic and numerical techniques. One example is the TSP problem.

2.6 Drawbacks of Genetic Algorithm

The algorithm implementation has following drawbacks also:

i. GA themselves are blind to the optimization process, as they only look at the fitness value of each chromosome rather than knowing what the fitness value actually means. As a result, their capability to explain why a particular solution was arrived at is practically very poor or nil.

ii. Although GA are moderately scalable - an increased number of variables can be accommodated by increasing the length of the chromosome - a longer chromosome also makes finding the solution more time consuming. The longer the chromosome, the larger the population needs to be since there are more potential combinations of genes. This result in more time required for decoding and fitness evaluation.

iii. In general, GA does not require extensive access to data. But some applications may require access and process data from the organization’s databases to be able to evaluate the fitness of solutions. For these applications, the quality and quantity of data is important.
2.7 Applications of Genetic Algorithm

Roger specified various areas where GA has application Roger (2006). Today GA is being used as an optimization and search tool by researchers in every field. Following are the main application areas:

- **Scheduling:** manufacturing, facility, resource allocation, Production, Job, and Transportation Scheduling
- **Design:** Circuit board layout, Communication Network design, keyboard layout, Parametric design in aircraft, semiconductor layout
- **Control:** Missile evasion, Gas pipeline control, Pole balancing
- **Machine Learning:** Designing Neural Networks, Classifier Systems, Learning rules
- **Robotics:** Trajectory Planning, Path planning
- **Combinatorial Optimization:** TSP, Set Covering, Graph Bisection, Routing
- **Signal Processing:** Filter Design
- **Image Processing:** Pattern recognition
- **Business:** Economic Forecasting; evaluating credit risks, Detecting stolen credit cards before customer reports it is stolen
- **Medical:** Studying health risks for a population exposed to toxins
- **Models of social systems:** GAs has been used to study evolutionary aspects of social systems, such as the evolution of cooperation, the evolution of communication, and trail-following behavior in ants.
- **Economic models:** GAs has been used to model processes of innovation, the development of bidding strategies, and the emergence of economic markets.
- **Immune system models:** GAs has been used to model various aspects of the natural immune system, including somatic mutation during an individual’s lifetime and the discovery of multi-gene families during evolutionary time.
- **Ecological models:** GAs have been used to model ecological phenomena such as biological arms races, host-parasite co-evolutions, symbiosis and resource flow in ecologies.
- **Automatic Programming:** GAs has been used to evolve computer programs for
specific tasks, and to design other computational structures, for example, cellular automata and sorting networks.

- **Optimization:** GAs have been used in a wide variety of optimization tasks, including numerical optimization, and combinatorial optimization problems such as traveling salesman problem, circuit design, job shop scheduling and video & sound quality optimization.

### 2.8 Summary

Evolutionary algorithms stand for a class of stochastic optimization methods that simulate the process of natural evolution. Since 1970s several evolutionary methodologies have been proposed: genetic algorithms, genetic programming, evolutionary programming, and evolution strategies. All these approaches operate on a set of candidate solutions. The chromosome, natural selection and variation are building blocks of natural evolution and evolutionary computation.

In order to find the solution of a problem, the principles of Darwin’s theory are followed by Genetic Algorithms As seen in nature; Genetic Algorithms are adaptive search techniques which simulate an evolutionary process on the basis of the ideas of selection of the fittest, crossing and mutation. Genetic algorithm is a guided random search technique to solve large scale optimization and combinatorial problems. These operate on a population of potential solutions, applying the principle of survival of the fittest to generate improved estimations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness and breeding them together using genetic operators inspired by natural genetics. This process leads to the evolution of better populations than the previous populations.

To simulate the process of natural selection in a computer, we need to define a representation of an individual solution/chromosome. At each point during the search process a generation of individuals is maintained. Each individual is a data structure representing the genetic structure of a possible solution or hypothesis. Like a chromosome, the genetic structure of an individual is described using a fixed, finite alphabet. In basic GAs, the alphabet 0, 1 is usually used. This
string is interpreted as a solution to the problem, trying to solve. Depending upon the problem other specific representations suitable to problem can also be taken.

To implement GA following components are required:

- **Representation (solution definition)**
- **Evaluation function (or fitness function)**
- **Population**
- **Parent selection mechanism**
- **Variation operators (crossover and mutation)**
- **Survivor selection mechanism (replacement)**
- **Termination Condition**

GA operates on a population of potential solutions, applying the principle of survival of the fittest to generate improved estimations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness and breeding them together using genetic operators inspired by natural genetics. This process leads to the evolution of better populations than the previous populations until some termination condition meets. If only mutation is used, the algorithm is very slow. Crossover makes the algorithm significantly faster.