Optimization of Benchmark Functions Using Genetic Algorithm

Vinod Goyal
GJUS&T, Hisar

Sakshi Dhingra
GJUS&T, Hisar

Jyoti Goyat
GJUS&T, Hisar

Dr Sanjay Singla
IET Bhaddal Technical Campus , Ropar, Punjab

Abstrat** **This paper presents the optimization of various benchmark functions using Genetic Algorithm. The comparative study is performed using benchmark functions. This main purpose of this paper is to find a comparative study of various encoding schemes, selection methods, scaling mechanism, crossover and mutation operators. This includes recording the performance of GA with various combinations of parameters.

1. INTRODUCTION

The GA is a stochastic global search method that mimics the metaphor of natural biological evolution. GAs operates on a population of potential solutions applying the principle of survival of the fittest to produce (hopefully) better and better approximations 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 in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

Individuals, or current approximations, are encoded as strings, *chromosomes,* composed over some alphabet(s), so that the *genotypes* (chromosome values) are uniquely mapped onto the decision variable (*phenotypic*) domain. The most commonly used representation in GAs is the binary alphabet {0, 1} although other representations can be used, e.g. ternary, integer, real-valued etc.

Genetic Algorithms (GAs) [1] are search algorithms that simulate the process of natural selection. GAs attempt to find a good solution to some problem (e.g., finding the maximum of a function) by randomly generating a population of potential solutions to the problem and then manipulating those solutions using genetic operators. Through selection, mutation, and crossover operations, better solutions are hopefully generated out of the current set of potential solutions. This process continues until an acceptable solution is found. GA's have many advantages over other search techniques in complex domains. They tend to avoid being trapped in local sub optima and can handle different types of optimization variables (discrete, continuous, and mixed).A large amount of literature exists about the application of GAs to optimization problems [2].

Optimization is an activity that aims at finding the best (i.e., optimal) solution to a problem. For optimization to be meaningful there must be an OBJECTIVE FUNCTION to be optimized and there must exist more than one feasible solution, i.e., a solution which does not violate the constraints. The term optimization does not apply, usually, when the number of solutions permits the best to be chosen by inspection, using an appropriate criterion.

The optimal solution (or "solution to the optimization problem") is values of decision variables xl, x2,..,xn that satisfy the constraints and for which the objective function attains a maximum (or a minimum, in a minimization problem). Very few optimization problems can be solved analytically, that is, by means of explicit formulae. In most practical cases appropriate computational techniques of optimization (numerical procedures of optimization) must be used. Optimization is the process of maximizing or minimizing a desired objective function while satisfying the prevailing constraints. The optimization problems have two major divisions. One is linear programming problem and other is non-linear programming problem. But the modern game theory, dynamic programming problem, integer programming problem also part of the Optimization theory having wide range of application in modern science, economics and management.

The goal of an optimization problem can be stated as follows: find the combination of parameters (independent variables) which optimize a given quantity, possibly subject to some restrictions on the allowed parameter ranges. The quantity to be optimized (maximized or minimized) is termed the objective function; the parameters which may be changed in the quest for the optimum are called control or decision variables; the restrictions on allowed parameter values are known as constraints.

2. GENETIC ALGORITHM:

Idea of evolutionary computing was introduced in the 1960s by I. Rechenberg in his work "*Evolution strategies*". His idea was then developed by other researchers. Genetic Algorithms (GAs) were invented by John Holland and developed by him and his students and colleagues. In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method "genetic programming" (GP). LISP programs were used, because programs in this language can express in the form of a "parse tree", which is the object the GA works on. GA's are stochastic search techniques based on natural a phenomenon which simulates survival of the fittest and genetic inheritance. In genetic algorithm a chromosome consists of gene. Each gene encodes a particular value. Possible values for genes are called alleles. Each gene has its own position in the chromosome. This position is called locus. Complete set of genetic material (all chromosomes) is called genome. Particular set of genes in genome is called genotype. The genotype is with later development after birth base for the organisms. Phenotype is physical and mental characteristics of the individual. Genetic Algorithm has been applied widely in the domain of data mining. The main motivation behind using GA for rule mining is due to their ability to perform a global search. Also, they tend to cope better with attribute interaction than the other greedy rule induction algorithm. Design of genetic algorithm for rule mining is shown in figure2.1.

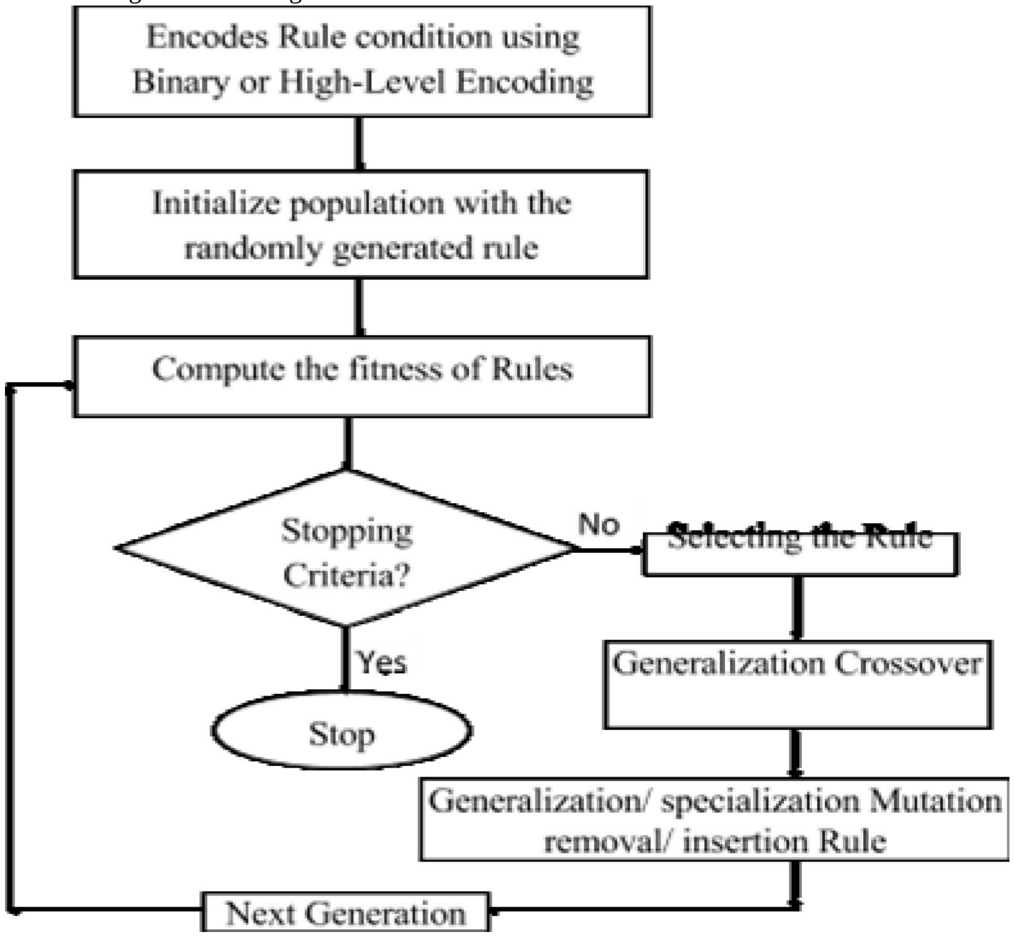
Encodes Rule condition using Binary or High-Level Encoding
Initialize population with the randomly generated rule
Compute the fitness of Rules
Stopping Criteria?
No
Selecting the Rule
Yes
Stop
Generalization Crossover
Generalization/ specialization Mutation removal/ insertion Rule
Next Generation

Fig2.1: Genetic Algorithm for rule mining

2.1 *BASIC PRINCIPAL:*

The principal behind working of Genetic Algorithm includes:

2.1.1 INITIAL POPULATION:

Population**:**

Population field specify options for the population of the genetic algorithm. Some parameter of population

- **Population type** specifies the type of the input to the fitness function. You can set Population type to be Double vector, or Bit string,.
- **Population size** specifies how many individuals there are in each generation.
- **Creation function** specifies the function that creates the initial population. The default creation function Uniform creates a random initial population with a uniform distribution.
- **Initial population** enables you to specify an initial population for the genetic algorithm.
- **Initial scores** enable you to specify scores for initial population. If you do not specify Initial scores, the algorithm computes the scores using the fitness function.
- **Initial range** specifies lower and upper bounds for the entries of the vectors in the initial population.

2.1.2 FITNESS EVALUATION

- **Fitness scaling** : The scaling function converts raw fitness scores returned by the fitness function to values in a range that is suitable for the selection function. Scaling function specifies the function that performs the scaling. Following are fitness functions type:

- **Rank scales** the raw scores based on the rank of each individual, rather than its score. The rank of an individual is its position in the sorted scores. The rank of the fittest individual is 1, the next fittest is 2, and so on.
- **Proportional** makes the expectation proportional to the raw fitness score. This strategy has weaknesses when raw scores are not in a "good" range.
- **Top scales** the individuals with the highest fitness values equally. If you select this option, you can specify Quantity, the number of fittest individuals that produce offspring. Quantity must be an integer between 1 and Population Size or a fraction between 0 and 1 specifying a fraction of the population size.
- **Shift linear** scales the raw scores so that the expectation of the fittest individual is equal to a constant, which you can specify as Maximum survival rate, multiplied by the average score.

2.1.3 SELECTION METHOD

The selection function chooses parents for the next generation based on their scaled values from the fitness scaling function. Following are the function that specify and performs the selection in the Selection field:

- **Stochastic uniform** lays out a line in which each parent corresponds to a section of the line of length proportional to its expectation. The algorithm moves along the line in steps of equal size, one step for each parent. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size.
- **Remainder** assigns parents deterministically from the integer part of each individual's scaled value and then uses roulette selection on the remaining fractional part.
- **Uniform** select parents at random from a uniform distribution using the expectations and number of parents. This results in an undirected search. Uniform selection is not a useful search strategy, but you can use it to test the genetic algorithm.
- **Shift linear** scales the raw scores so that the expectation of the fittest individual is equal to a constant,

which you can specify as Maximum survival rate, multiplied by the average score.

- **Roulette** simulates a roulette wheel with the area of each segment proportional to its expectation. The algorithm then uses a random number to select one of the sections with a probability equal to its area.
- **Tournament s**elects each parent by choosing individuals at random, the number of which you can specify by Tournament size, and then choosing the best individual out of that set to be a parent.

2.1.4 GENETIC OPERATOR

There have been several proposals of genetic operators designed particularly for rule discovery. Genetic operators are used to balance between the exploration and exploitation. Exploration means to cover the search space and on the other hand, exploitation considers GA convergence property. Crossover used for exploitation, and mutation is used for exploration.

Crossover

Crossover combines two individuals, or parents, to form a new individual, or child, for the next generation. Following function specifies and performs the crossover in the Crossover field. List of crossover parameter given below:

- **Scattered** creates a random binary vector. It then selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, and combines the genes to form the child. For example,
  p1 = [a b c d e f g h]
  p2 = [1 2 3 4 5 6 7 8]
  Random crossover vector = [1 1 0 0 1 0 0 0] Child = [a b 3 4 e 6 7 8]
- **Single point** chooses a random integer n between 1 and Number of variables, and selects the vector entries numbered less than or equal to n from the first parent, selects genes numbered greater than n from the second parent, and concatenates these entries to form the child. For example,
  p1 = [a b c d e f g h]
  p2 = [1 2 3 4 5 6 7 8]
  Random crossover point = 3
  Child = [a b c 4 5 6 7 8]
- **Two point:** Selects two random integers m and n between 1 and Number of variables. The algorithm selects genes numbered less than or equal to m from the first parent, selects genes numbered from m+1 to n from the second parent, and selects genes numbered greater than n from the first parent. The algorithm then concatenates these genes to form a single gene. For example,

  p1 = [a b c d e f g h]
  p2 = [1 2 3 4 5 6 7 8]
  Random crossover points = 3,6
  Child = [a b c 4 5 6 g h]
- **Intermediate c**reates children by a weighted average of the parents. Intermediate crossover is controlled by a single parameter Ratio:
  child1 = parent1+... rand*Ratio*(parent2 - parent1)

  If Ratio is in the range [0, 1] then the children produced are within the hypercube defined by the parents locations at opposite vertices. If Ratio is in a larger range, say 1.1 then children can be generated outside the hypercube. Ratio can be a scalar or a vector of length Number of variables. If Ratio is a scalar, then all of the children will lie on the line between the parents. If Ratio is a vector then children can be any point within the hypercube.
- **Heuristic** creates children that lie on the line containing the two parents, a small distance away from the parent with the better fitness value in the direction away from the parent with the worse fitness value.
- **Arithmetic** creates children that are the weighted arithmetic mean of two parents. Children are feasible with respect to linear constraints and bounds.

Mutation

Mutation functions make small random changes in the individuals in the population, which provide genetic diversity and enable the Genetic Algorithm to search a broader space. Following are the function that performs mutation in the Mutation field. List of the following mutation parameter given below:

- **Gaussian** adds a random number to each vector entry of an individual. This random number is taken from a Gaussian distribution centred on zero. The variance of this distribution can be controlled with two parameters. The Scale parameter determines the variance at the first generation. The Shrink parameter controls how variance shrinks as generations go by. If the Shrink parameter is 0, the variance is constant. If the Shrink parameter is 1, the variance shrinks to 0 linearly as the last generation is reached.
- **Uniform** is a two-step process. First, the algorithm selects a fraction of the vector entries of an individual for mutation, where each entry has the same probability as the mutation rate of being mutated. In the second step, the algorithm replaces each selected entry by a random number selected uniformly from the range for that entry.
- **Adaptive feasible** randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation. A step length is chosen along each direction so that linear constraints and bounds are satisfied.

2.1.5 TERMINATION CRITERIA

The following are the termination criteria

- A solution is found that satisfies minimum criteria.
- Fixed number of generations is reached.
- Allocated budget (computation time/money) is reached.
- The highest-ranking solution's fitness is reached or has reached a plateau such that successive iteration no longer produces better result.
- Manual setting of inspection criteria.

Combination of two or more criteria can also be used.

3. BENCHMARK FUNCTIONS

Benchmarking has been used to compare the performance of a variety of technologies, including computer systems, information retrieval systems, and database management systems. In these and other research areas, benchmarking has caused the discipline to make great strides. A benchmark is defined as a standardised test or set of tests used for comparing alternatives. A benchmark has three components, a Motivating Comparison, a Task Sample, and Performance Measures. This definition was developed by looking at existing definitions and case histories of benchmarks. A general definition is used to ensure wide applicability of the theory, but within this paper the discussion is restricted to benchmarks for computer science research.

According to [3], a good benchmark must possess three characteristics: accuracy, scalability, and representativeness. First, benchmarks must accurately reflect real-world applications. Additionally, they can accurately measures system performance in a target application. Second, benchmarks must be able to test different data sets without requiring major changes. In addition, they must be comparable across system implementations. Third, benchmarks must accurately represent a system-under-test, cover most functionalities of one domain, and provide results that correlate to real-world performance.

Six Bench Mark functions are used in this paper namely:

Test Functions: We use six popular benchmark functions for studying the performance of GAs and GA operators. They are Rastrigin's function, Rosenbrock Function, Sphere Function, Ackley Function, Generalized Rastrigin and Branin Function.These functions and the fitness functions are described in the following,

3.1 *Rastrigin's Function:-*

Rastrigin's function is based on the function of De Jong with the addition of cosine modulation in order to produce frequent local minima. Thus, the test function is highly multimodal. However, the location of the minima is regularly distributed.

Function has the following definition:

$F(x) = 10n +_i^2 - 10 \cos (2\pi x_i)]$.

Test area is usually restricted to hypercube $-5.12 <= x_i <= 5.12,\ i=1 \ldots n$.

Global minimum $f(x) = 0$ is obtainable for $x_i = 0,\ i= 1 \ldots n$.

3.2 *Rosenbrock's Valley:-*

Rosenbrock's valley is a classic optimization problem, also known as banana function or the second function of De Jong. The global optimum lies inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult and hence this problem has been frequently used to test the performance of optimization algorithms. Function has the following definition

$F(x) = 100(x_i+1-x^2_i)^2 + (1-x_i)^2]$.

Test area is usually restricted to hypercube ***-2.48<=x_i2.48, i=1… n.***

Its global minimum equal $f(x) = 0$ is obtainable for x_i, i=1… n.

3.3 *Sphere Function:-*

The Sphere function is defined as follows:

$F(x) = {}^2_i$

Where D is the dimension and x = (*x*1, *x*2… *xD*) is a D-dimensional row vector (i.e., a $1\times D$ matrix). The Sphere function is very simple and is mainly used for demonstration. In this test suite this function serves as separable part when using a naturally nonseparable function to form some partially nonseparable functions.

3.4 *Ackley's function:-*

Ackley's is a widely used multimodal test function. It has the following definition

$F(x) = -a.\exp(-b.{}^2_i) - \exp(1/n(cx_i)) + a + \exp(1)$

It is recommended to set $a = 20$, $b = 0.2$, $c = 2\pi$

Test area is usually restricted to hyphercube **-32.768 <=x_i<= 32.768, i= 1… *n*.**

Its global minimum $f(x) = 0$ is obtainable for $x_i = 0$, i= 1… *n*.

3.5 *Generalized Rastrigin Function:-*

The Generalized Rastrigin Function (Equation 1) is a typical example of non-linear multimodal function. It was first proposed by Rastrigin as a 2-dimensional function and has been generalized by Miihlenbein et al in. This function is a fairly difficult problem due to its large search space and its large number of local minima.

$F(x) = A.\ n + 2i - A.\cos(\omega.\ x_i)$

$A=10$, $\omega=2.\pi$, x_i □ $[-5.12, 5.12]$

The Rastrigin function has a complexity of O (n1n (n)), where n is the dimension of the problem. The surface of the function is determined by the external variables A and ω, which control the amplitude and frequency modulation respectively.

3.6 *Branins's function:-*

The Branin's function is a global optimization test function having only two variables. The function has three equal-sized global optima, and has the following definition:

$F(x1, x2) = a(x_2 - bx^2_1 + cx_1 + d)^2 + e(1-f)\cos(x_1) + e.$

It is recommended to set the following values of parameters: $a = 1$, $b = 5.1/4\pi^2$,

$c = 5/\pi$, $d = 6$, $e = 10$, $f=1/8\ \pi$. Three global optima equal $f(x_1, x_2) = 0.397887$ are located as follows: $(x_1, x_2) = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$.

4. RESULT AND CONCLUSION

Number of experiments has been conducted to study the performance of GA on several benchmarks functions.

Table 1 shows the various types of encoding schemes, selection mechanisms, scaling mechanisms and various GA operators.

Table1: Various Parameter of encoding, Crossover, Mutation, Selection and Scaling.

Encoding Scheme	Crossover	Mutation	Selection	Scaling
1. Binary 2. Real	● Arithmetic ● Heuristic ● Intermediate ● Two Points ● Single Point ● Scattered	● Adaptive Feasible ● Uniform ● Gaussian	● Tournament ● Roulette Wheel ● Uniform ● Remainder ● Stochastic	● Rank ● Proportion ● Top ● Shift Linear

Tables2, Table3, Table4 shows fitness with 2 variables, 10 variables and 20 variables when Population Type is changing while other options remain default which are shown as:-
Fitness scaling function=Rank, Selection function=Stochastic uniform,
Mutation function=Gaussian, Crossover function=Scattered

Table2: Fitness with 2 variables when population type changes

Functions	Population Type			
	Double Vector	Bit String		
	Best Fitness	*Mean Fitness*	*Best Fitness*	*Mean Fitness*
Rastrigin’s function	0.04066	3.78650	0	0
Rosenbrock Function	0.01149	18.8125	0	5.0000
Sphere Function	0.00275	0.34290	0	0
Ackley Function	0.01972	1.17570	8.8818e-16	8.8818e-16
Generalized Rastrigin	0.02989	8.42150	0	0
Branin Function	0.39799	0.95491	0	0

Conclusion: Table2shows the results for fitness when we vary the population type. The table shows the result for

two variables. The results show that when we change the population type from Double Vector to Bit String, there is change in fitness (best fitness, mean fitness) and the fitness is decrease. Remember that we are minimizing the benchmarks functions, so lower the fitness value, better is the performance. The table shows when we choose Bit String as population type, the performance of genetic algorithm is much better than double Vector
Table3: Fitness with 10 variables when population type changes

Functions	Population Type			
	Double Vector	Bit String		
	Best Fitness	*Mean Fitness*	*Best Fitness*	*Mean Fitness*
Rastrigin’s function	16.8745	55.0668	0	0.05
Rosenbrock Function	39.4568	465.619	9	9
Sphere Function	0.19307	1.22630	0	0.05
Ackley Function	0.76575	1.94870	-8.8818e-16	0.06128
Generalized Rastrigin	15.1837	52.0701	2	2
Branin Function	0.39813	0.97380	0	0.05

Conclusion: Table3 shows the result for best fitness and mean fitness value when we varies the population type and use the different benchmarks functions with ten variables. The fitness value is increased as compared to fitness value when we use the function with two variables. As a comparative study when we compare the two of population type using function with ten variables the result shows that the performance of bit string population type is better than double vector.

Table4: Fitness with 20 variables when population type changes

Functions	Population Type			
	Double Vector	Bit String		
	Best Fitness	*Mean Fitness*	*Best Fitness*	*Mean Fitness*
Rastrigin’s function	27.4594	98.786	1	1.15
Rosenbrock Function	145.9792	1014.52	105	105
Sphere Function	1.37	3.6269	1	1.15
Ackley Function	1.8748	2.6855	1.2257	1.2257
Generalized Rastrigin	42.7229	114.3473	1	1.15
Branin Function	9.718	17.4735	4.2813	4.2813

Conclusion: Table4 shows the results for different benchmarks functions with twenty variables when we change the population type. The results confirm the conclusion of table1 and table2. Like table1 and table2, the result of table3 shows that performance of genetic algorithm is much better when we use the bit string as population type as compare to double vector.

Graphs for fitness for 2 variables according to population type of different functions:-

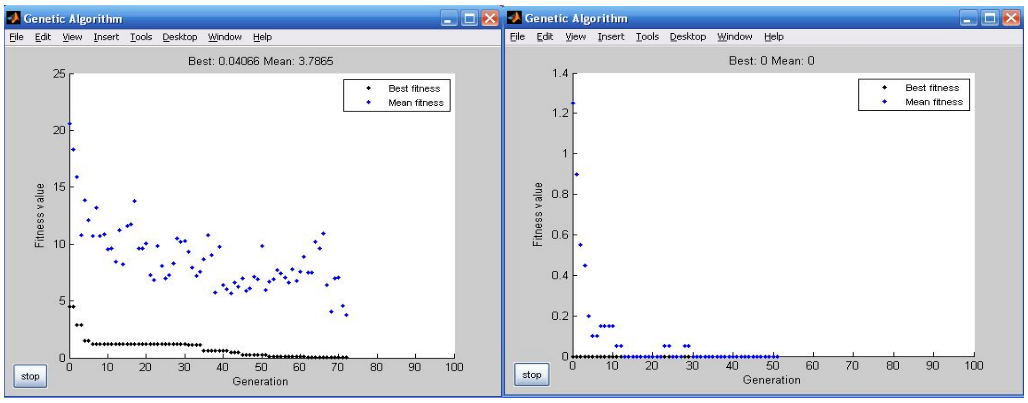

Fig 4.1: Rastrigin's function

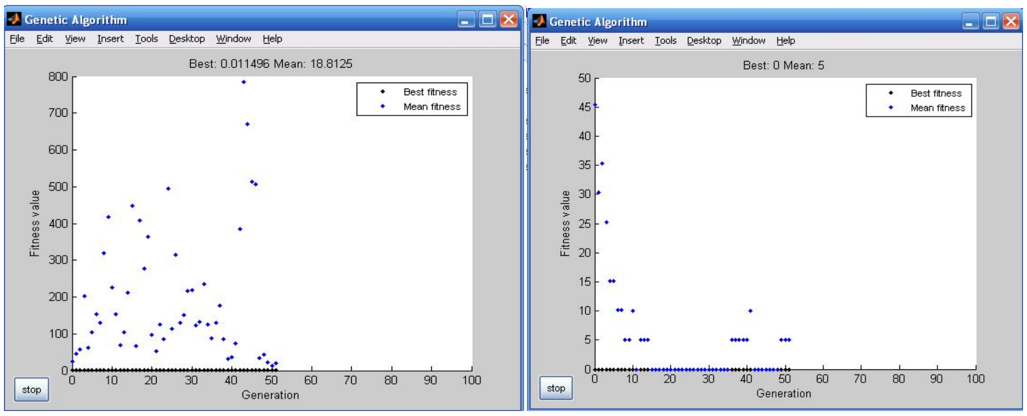

Fig 4.2: Rosenbrock Function

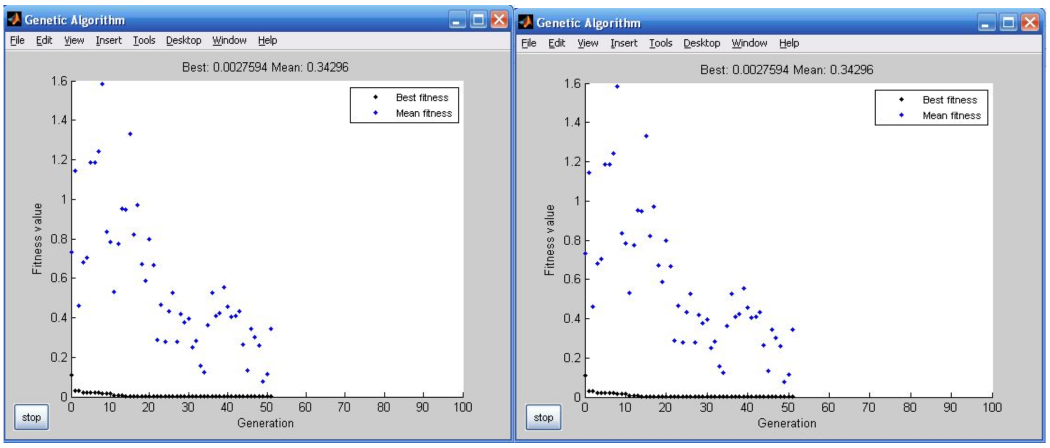

Fig 4.3: Sphere Function

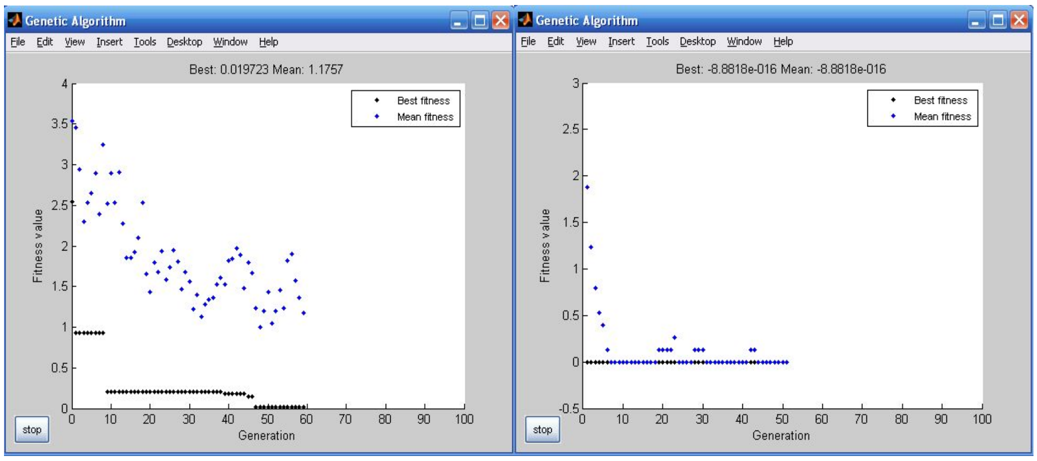

Fig 4.4: Ackley Function

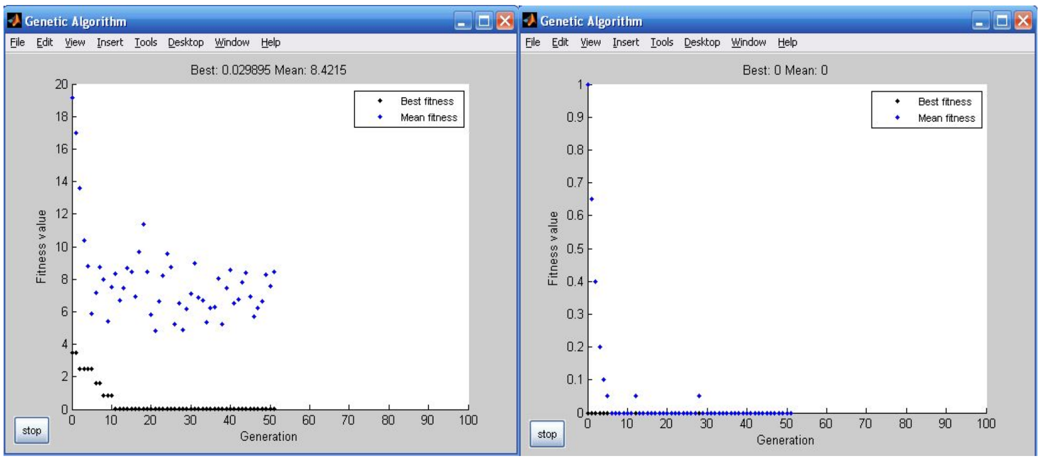

Fig 4.5: Generalized Rastrigin function

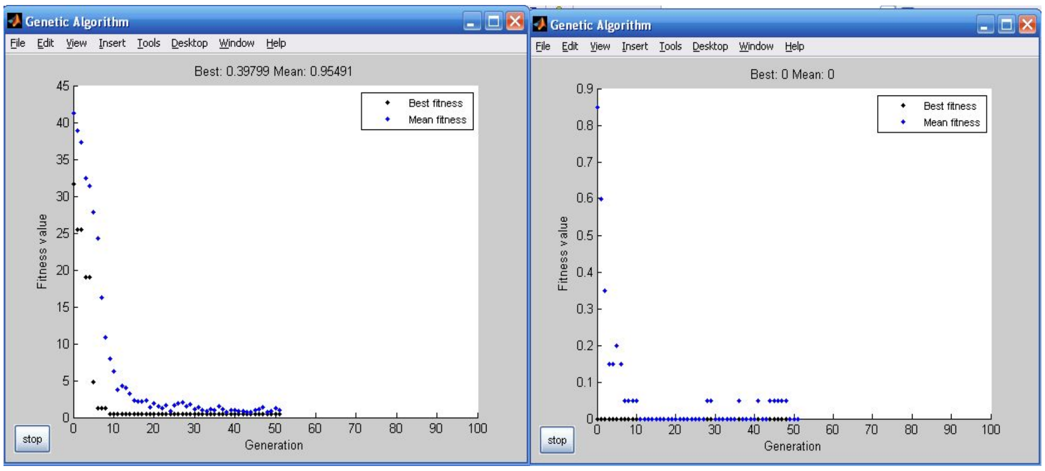

Fig 4.6: Branin Function

Table for fitness when Scaling Function is changing while other options remain default which is shown as:-

Population type=Double vector, Selection function=Stochastic uniform,
Mutation function=Gaussian, Crossover function=Scattered

Table5: Fitness when scaling function changes

Function	Scaling Function							
	Rank	Proportion	Top	Shift Linear				
	Best	*Mean*	*Best*	*Mean*	*Best*	*Mean*	*Best*	*Mean*
Rastrigin’s function	0.0351	6.3410	0.0171	6.3242	0.2162	4.0440	0.1479	1.5511
Rosenbrock Function	0.0151	42.474	0.0066	25.532	0.3895	9.4210	0.0355	30.445
Sphere Function	0.0006	0.3179	0.0003	0.2083	0.0018	0.1147	0.0010	0.2313
Ackley Function	0.0040	1.3805	0.0942	1.0969	0.9159	1.3704	0.09123	0.8008
Generalized Rastrigin	0.0666	5.3002	0.0974	5.1373	0.0513	3.3444	0.0220	4.3985
Branin Function	0.3984	0.6950	0.3983	0.9330	0.4085	0.7495	0.4000	1.5352

Conclusion: Table5 shows the performance of six benchmarks function in the form of mean fitness and best fitness. The fitness shown is evaluated using benchmarks function with four variables. The performance shows the comparison between all four types of scaling function (i.e. Rank, Proportion, Top and Shift Linear. The results show

that performance of *proportion* using scaling function is much better than performance of *Rank* scaling function followed by *top* scaling function.
Table for fitness when Selection function is changing while other options remain default which is shown as:
Population type=Double vector, Scaling function=Rank,
Mutation function=Gaussian, Crossover function=Scattered

Table6: Fitness when selection function changes

Function	Selection Function									
	Stochastic	Remainder	Uniform	Roulette Wheel	Tournament					
	Best	*Mean*	*Best*	*Mean*	*Best*	*Mean*	*Best*	*Mean*	*Best*	*Mean*
Rastrigin's function	0.0122	6.5048	0.0105	5.2067	0.0825	14.195	0.0008	8.9790	1.0245	6.2878
Rosenbrock Function	0.0586	68.258	0.0061	24.265	0.0849	143.70	0.5238	25.273	0.4904	32.758
Sphere Function	0.0002	0.2378	0.0003	0.2564	0.0014	0.6107	0.0002	0.3090	0.0110	0.0567
Ackley Function	0.0311	1.5726	0.0766	1.7593	0.2349	2.9226	0.1223	1.0676	0.2763	0.0799
Generalized Rastrigin	1.0062	10.139	0.0097	11.752	0.1039	13.816	1.0073	11.743	1.3382	5.0775
Branin Function	0.3979	2.3684	0.3994	1.2960	0.4044	2.3745	0.3991	0.8085	0.4149	1.2595

Conclusion: The table 6 shows the results when selection function is changing (i.e. stochastic, Remainder, Uniform, Roulette Wheel, Tournament). The table shows the comparison between performances of genetic algorithm by using different types of scaling function in the genetic algorithm.
The table shows that when is used *Remainder* as a scaling function the best fitness and mean fitness is comes better than *Roulette wheel* when it is used as a scaling function followed by the performance of *Stochastic Function*.

Table for fitness when Mutation function is changing while other options remain default as:-
Population type=Double vector, Scaling function=Rank, Selection function= Stochastic uniform, Crossover function=Scattered
Table7: Fitness when mutation function changes

Function	Mutation Function					
	Gaussian	Uniform	Adaptive Feasible			
	Best	*Mean*	*Best*	*Mean*	*Best*	*Mean*
Rastrigin's function	0.2285	9.6641	0.4066	0.4066	0	0
Rosenbrock Function	0.0060	69.693	0.1577	0.1577	0.0060	5.4151
Sphere Function	0.0011	0.2050	0.0018	0.0146	1.7946e-015	1.1542e-013
Ackley Function	0.0612	1.1228	0.2212	0.2212	6.5582e-008	3.6219e-007
Generalized Rastrigin	0.1090	5.4176	1.7482	1.7482	5.2047e-013	3.4807e-012
Branin Function	0.4042	0.9490	30.646	30.646	0.4058	0.4442

Conclusion: Table7 shows the comparisons in the performance of genetic algorithm when we change the mutation function. The mutation function is change due to comparative study of performance evaluation by using the different benchmarks function with four variables. The results from table6 show that when we used *Gaussian* as mutation function the performance of genetic algorithm is improved than other two functions. The performance of *Gaussian* is superior than performance of *Adaptive Feasible* followed by *Uniform* mutation function.

Table for fitness when Crossover function is changing while other options remain default which is shown as:

Population type=Double vector, Scaling function=Rank, Selection function= Stochastic uniform, Mutation function=Gaussian
Table8: Fitness when crossover function changes

Function	Crossover Function										
	Scattered	Single Point	Two Points	Intermediate	Heuristic	Arithmetic					
	Best	*Mean*	*Best*	*Mean*	*Best*	*Mean*	*Best*	*Mean*	*Best*	*Mean*	*Best*
Rastrigin's Function	0.1121	6.9208	0.0028	5.6924	0.0766	6.4230	0.9950	7.5540	0.9949	5.8189	0.99
Rosenbrock Function	0.0103	21.646	0.1497	47.808	0.0023	24.716	0.0058	42.109	0.0005	5.8141	0.00
Sphere Function	0.0005	0.0848	0.0003	0.2586	0.0001	0.1360	5.5e-06	0.1425	6.4e-10	0.1731	4.9e 08
Ackley Function	0.0324	1.2293	0.0361	1.7253	0.0305	1.1409	0.0002	1.3888	8.2e-05	0.8012	0.00
Generalized Rastrigin	0.0977	8.9253	0.1455	5.8766	0.2650	5.3562	0.9949	4.9667	0.9949	5.0541	0.00
Branin Function	0.3997	1.3886	0.3991	0.7614	0.3991	2.2225	0.3980	1.0070	0.3978	0.7386	0.39

ConclusionTable8 shows the best fitness and mean fitness of the all six benchmarks functions. The results shows that when we use *two points* crossover function the results are always

superior to other types of crossover function. After *two points* crossover function the performance of *single point* crossover is much better followed by *scattered* crossover function.

5. CONCLUSION

In this paper we have studied various encoding schemes of Genetic Algorithm in the domain of function optimization. Two encoding schemes (Binary & Real), five selections were tested on several benchmarks mathematical functions. We have reached the following conclusion.In this comparative study:

- Between two encoding schemes (Binary and real); binary encoding is better.
- Between six crossover schemes (Arithmetic, Heuristic, Intermediate, Two Points, Single Point, Scattered); Two Points is best.
- Between Three mutation schemes (Adaptive Feasible, Uniform, Gaussian); Gaussian is best.
- Between five selection schemes (Tournament, Roulette Wheel, Uniform, Remainder); Remainder is best.
- Between four scaling schemes (Rank, Proportion, Top, Shift Linear); Proportion is best.

Table9: Show the Best fitness and mean fitness for the best combination from the options (crossover= Two Points, mutation= Gaussian, selection= Remainder, scaling= Proportion)

Table9: Fitness for best Combination

Functions	Fitness	
	Best fitness	Mean fitness
Rastrigin’s function	0.0129	3.4407
Rosenbrock Function	0.0019	29.578
Sphere Function	0.0020	0.2226
Ackley Function	0.0223	1.0807
Generalized Rastrigin	0.0096	4.2160
Branin Function	0.3979	0.5883

Graphs:

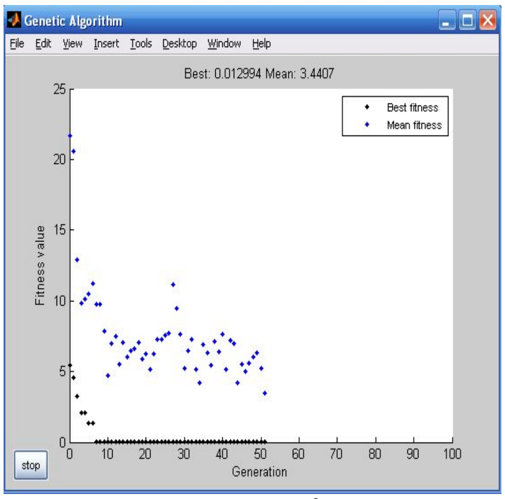

Fig 5.1 Rastrigin's function

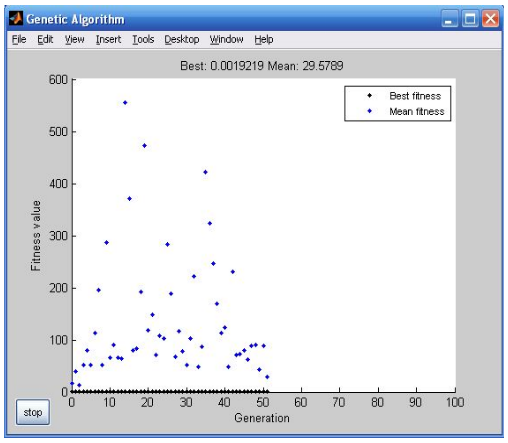

Fig 5.2 Rosenbrock Function

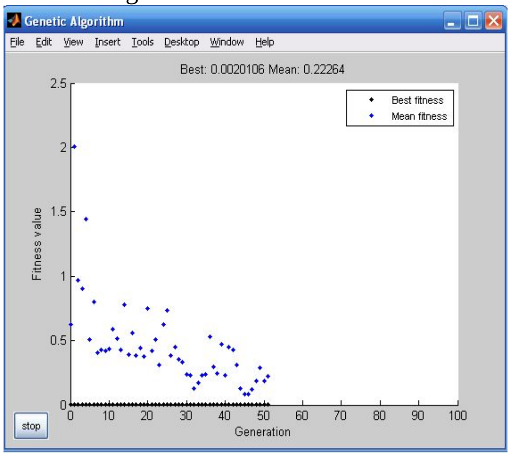

Fig 5.3 Sphere Function

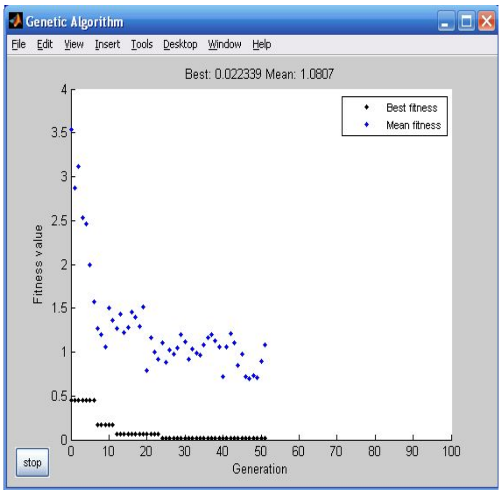

Fig 5.4 Ackley Function

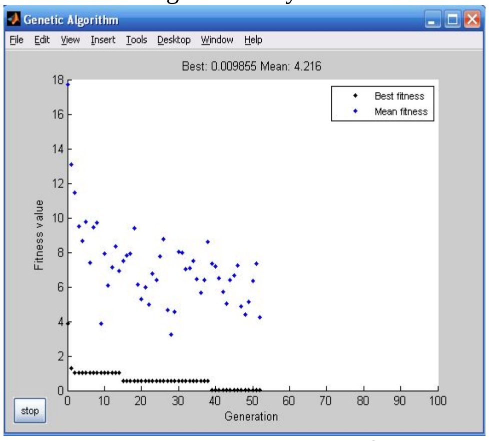

Fig 5.5 Generalized Rastrigin’s function

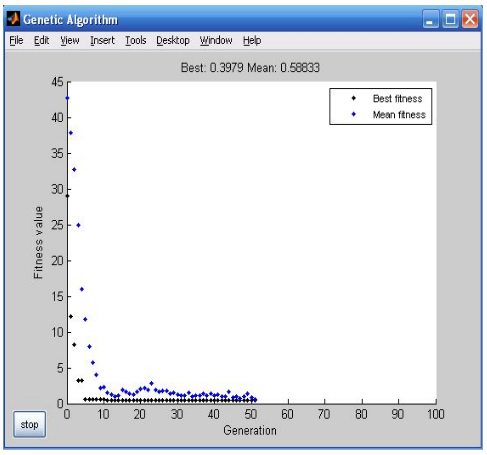

Fig 5.6 Branin Function

6. REFRENCES

[1.] Goldberg, D. E. (1989). ‘‘Genetic algorithms in search, optimization and machine learning’’, New York: Addison-Wesley, pp. 40–45.

[2.] Michalewicz, Z. (1996). *Genetic algorithms+data structures=evolution programs.* Springer-Verlag, New York.

[3.] Walter F. Tichy (1998). “Should Computer Scientists Experiment More?,” *IEEE Computer*, pp.32-40.

[4.] Patton, A. L. , Dexter, T., Goodman, E. D. and Punch, W. F. (1998). ‘‘On the Application of Cohort-Driven Operators to Continuous Optimization Problems using Evolutionary Computation’’, in Lecture Notes in Computer Science, vol 1447, pp. 671–682, Springer - Verlag Inc.

[5.] Baeck, T. (1991). ‘‘Optimization by Means of Genetic Algorithms’’. In Internationales Wissenschaftliches Kolloquium (pp. 1–8). Germany: Technische University’’.

[6.] Capers Jones (2000). “*Software Assessments, Benchmarks, and Best Practices”*. Reading, Massachusetts: Addison-Wesley.

[7.] More, J. J., Garbow, B. S., and Hillstrom, K. E. (1981). ‘‘Testing Unconstrained Optimization Software, {ACM} Transactions on Mathematical Software, vol 7, pp. 17–41.

[8.] Salomon. R. (1995). ‘‘Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions’’, BioSystems vol. 39, pp. 263–278, Elsevier Science.

[9.] Wright, A. (1990). “Genetic algorithms for real parameter optimization”. The First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, pp. 205–218. Morgan Kaufmann, Los Altos, California.