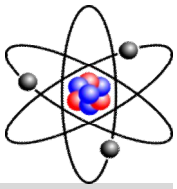
International Archive of Applied Sciences and Technology
Volume 3 [2] June 2012: 47 - 57
ISSN: 0976-4828
©Society of Education, India
Website: www.soeagra.com/iaast/iaast.htm



Original Article

Efficient Genetic Algorithm on Linear Programming Problem for Fittest Chromosomes

Subrata Datta, Chayanika Garai and Chandrani Das
Department of Information Technology
Neotia Institute of Technology, Management and Science
datta.nitmas@gmail.com
Department of Information Technology
Neotia Institute of Technology, Management and Science
chayanikagarai@gmail.com
Department of Information Technology
Neotia Institute of Technology, Management and Science
moon.queen12@gmail.com

ABSTRACT
Genetic Algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form such an algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (Strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. Linear Programming (LP) is the most commonly applied form of constrained optimization. In this paper we proposed an efficient genetic algorithm applied on linear programming problem for find out the Fittest Chromosomes. The experimental performances find out the fittest chromosomes regarding to constraint linear programming problem, so that it gives a better result.
Keywords: *Chromosome, Genetic algorithm, Fitness, LPP.*

INTRODUCTION

Computer simulation of evolution by biologists became more common in the early 1960s, and the methods were described in books by Fraser and Burnell (1970) and Crosby (1973). Fraser simulations included all of the essential elements of modern genetic algorithm. In addition Hans-Joachin Bremermann published a series of papers in the 1960s that also adopted a population of solution to optimization problems, undergoing recombination, mutation and selection. Bremermann's research also included the elements of modern genetic algorithms. Genetic algorithms in particular became popular through the work of John Holland in the early 1970s and particularly his book "Adaptation in Nature and Artificial Systems" (1975). His work originated with the study of cellular automata. Holland introduced a formalized framework for predicting the quality of next generation, known as Holland's Schema Theorem.

Genetic algorithm basically stands on the platform of evolutionary algorithm. Genetic algorithm helps to find out the potential solution of a specific problem through a simple chromosome like data structure. Genetic algorithm helps to generate those chromosomes which give a better solution of the target problem. While randomized, Genetic Algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance [1- 4].

In Linear Programming (LP), all of the mathematical expressions for the objective functions and the constraints are linear. The programming in linear programming is an archaic use of the word "Programming" to mean "Planning". We can think linear programming as "Planning with linear models". The main motive of this paper is to find out the fittest chromosomes regarding to constraint linear programming problem.

BACKGROUND TECHNIQUES

Genetic Algorithm (GA)

In the computer science field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful

solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.
→ GAs are inspired by Darwin's Theory about Evolution "Survival Of Fittest".
→ GAs are adaptive heuristic search based on the evolutionary ideas of natural selection and genetics.
→ GAs are intelligent exploitation of random search used in optimization problem.
→ GAs, although randomized, exploit historical information to direct the search into the region of better performance within the search space
→Genetic algorithms find application in bioinformatics, phylogenetics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields.

Definitions
Chromosome: A chromosome (also sometimes called a genome) is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve. The chromosome is often represented as a simple string, although a wide variety of other data structures are also used. We have to redefine the Chromosome representation for each particular problem, along with its fitness, mutate and reproduce methods.
Gene: A Gene is a part of chromosome. A gene contains a part of solution. For example if 162759 is a chromosome then 1, 6, 2, 7, 5 and 9 are its genes.
Fitness: Fitness (often denoted ω in population genetics models) is a central idea in evolutionary theory. It can be defined either with respect to a genotype or to a phenotype in a given environment. In either case, it describes the ability to both survive and reproduce, and is equal to the average contribution to the gene pool of the next generation that is made by an average individual of the specified genotype or phenotype. If differences between alleles at a given gene affect fitness, then the frequencies of the alleles will change over generations; the alleles with higher fitness become more common [5-7].
The basic steps involved in Genetic Algorithm are as follows:
➔ ***Initialization:***
Initialization involves setting the parameters for the algorithm, creating the scores for the simulation, and creating the first generation of chromosomes. In this benchmark, seven parameters are set:
→ The genes value is the number of variable slots on a chromosome;
→ The codes value is the number of possible values for each gene;
→ The population size is the number of chromosomes in each generation;
→ Crossover probability is the probability that a pair of chromosomes will be crossed;
→ Mutation probability is the probability that a gene on a chromosome will be mutated randomly;
→The maximum number of generations is a termination criterion which sets the maximum number of chromosome populations that will be generated before the top scoring chromosome will be returned as the search answer; and
→The generations with no change in highest-scoring (elite) chromosome is the second termination criterion which is the number of generations that may pass with no change in the elite chromosome before that elite chromosome will be returned as the search answer.
→The attempted optimization is to find the code for each gene in the solution chromosome that maximizes the average score for the chromosome. Finally, the first generation of chromosomes are generated randomly.
➔ ***Selection:***
Selection is the process of choosing two parents from the population for crossing. It is the first operator applied on population. From the population, the chromosomes are selected to be parents to crossover and produce off- spring. After deciding on an encoding, the next step is to decide how to perform selection that is how to choose individuals in the population that will create off-spring for the next generation and how many off-Spring each will create Reproduction selects good strings in a population and forms a mating pool. This is one of the reasons for the reproduction operation to be sometimes known as the selection operator.

- ***How to select these chromosomes***

According to Darwin's evolution theory "Survival of the Fittest" –the best ones should survive and create new off-spring. Selection means extract a subset of genes from an existing population, according to any definition of quality. Every gene has a meaning, so one can derive from the gene a kind of quality measurement called fitness function. Following this quality (fitness value), selection can be performed. The higher the fitness function, the better chance that an individual will be selected. The selection pressure is defined as the degree to which the better individuals are favored. The higher the selection pressured, the more the better individuals are favored. Fitness function quantifies the optimality of a solution (chromosome) so that a particular solution will be ranked against all the other solutions. The function depicts the closeness of a given 'solution' to the desired result.

Typically we can distinguish two types of selection schemes:

Proportionate Based Selection: It picks out individuals based upon their fitness values relative to the fitness of the other individuals in the population.

Ordinal Based Selection: This scheme selects individuals not upon their raw fitness, but upon their rank within the population. This requires that the selection pressure is independent of the fitness distribution of the population.

The most commonly used methods of selecting chromosomes for parents to crossover are:

1. Roulette Wheel Selection
2. Boltzmann Selection
3. Tournament Selection
4. Rank Selection
5. Random Selection

➔ Crossover

Crossover is the process of taking two parent solutions and producing from them a child. After the Selection process, the population is enriched with better individuals. Reproduction makes clones of good string but does not create new ones. Crossover operators applied to the mating pool with the hope that it creates a better offspring. Crossover is the Recombination operator which can be described as follows:

→In crossover operation, recombination process creates different individuals in the successive generations by combining materials from two individuals of the previous generation.

→The two strings participating in the crossover operation are known as Parent Strings and the resulting strings are known as Children String. Crossover selects gene from parent chromosomes and creates a new offspring.

→The idea behind Crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents.

→It is intuitive from this construction that good sub-strings from parent strings can be combined to form a better child string, if an appropriate site is chosen.

→With a random site, the children strings produced may or may not have a combination of good sub-strings from parent strings, depending on whether or not the crossing site falls in the appropriate place.

→The effect of cross over may be detrimental or beneficial.

→In order to preserve some of the good strings that are already present in the mating pool, all strings in the mating pool are not used in crossover. When a crossover probability, defined here as pc is used, only 100pc per cent strings in the population are used in the crossover operation and 100(1-pc) per cent of the population remains as they are in the current population.

Crossover is a recombination operator that proceeds in three steps:

1. The reproduction operator selects at random a pair of two individual strings for the mating.
2. A cross site is selected at random along the string length.
3. Finally, the position values are swapped between the two strings following the cross site.

Many crossover operators exist in the GA literature. The operators are selected based on the way chromosomes are encoded. The various crossover techniques are as follows:

→ Single-Point Crossover

→ Two-Point Crossover

→ Multi-Point Crossover/N-Point Crossover

→ Uniform Crossover
→Three-Parent Crossover
→ Arithmetic Crossover
→ Heuristic Crossover
→ Ordered Crossover
→ Partially Matched Crossover

➔ Mutation

After the Crossover, the strings are subjected to mutation. Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. Mutation may cause the chromosomes of individuals to be different from those of their parent individuals. Mutation adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima. The Mutation is the operator which can be described as follows:

→It is an operator that introduces diversity in the population whenever the population tends to become homogeneous due to repeated use of reproduction and crossover operators.

→ Mutation alters all or one gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With the new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible.

→ Mutation is an important part of the genetic search, helps to prevent the population from stagnating at any local optima. Mutation is intended to prevent the search falling into a local optimum of the state space.

→Mutation operates at the bit level; when the bits are being copied from the current string to the new string, there is probability that each bit may become mutated. This probability is usually a quite small value, called as mutation probability pm.

→The need for mutation is to create a point in the neighborhood of the current point, thereby achieving a local search around the current solution.

→Mutation plays the role of recovering the lost genetic material as well as for randomly distributing genetic information. It is an insurance policy against the irreversible loss of genetic material.

→ Mutation helps escape from local minima's trapped and maintains diversity in the population.

→ Mutation keeps the gene pool well-stocked, thus ensuring periodicity.

There are many different forms of mutation for the different kinds of representation. For binary representation, a simple mutation consists in inverting value of each gene with a small probability. The probability is usually taken about 1/L, where L is the length of the chromosome. It is also possible to implement kind of hill climbing mutation operator that do mutation only if it improves the quality of the solution. Such an operator can accelerate the search; however, care should be taken, because it might also reduce the diversity in the population and make the algorithm converge toward some local optima. The mutation operators are of many types:

→ Flipping
→ Interchanging
→ Reversing
→ Boundary
→ Uniform
→ Non-uniform
→ Gaussian

➔ Termination

The loop of chromosome generations is terminated when certain conditions are met. When the termination criteria are met, the elite chromosome is returned as the best solution found so far. The common terminating conditions are:

A solution is found that satisfies minimum criteria

→ Fixed number of generations reached:

It is a termination method that stops the evolution when the user-specified maximum numbers of evolutions have been run. This termination method is always active.

Evolution Time:

It is a termination method that stops the evolution when the elapsed evolution time exceeds the user-specified max evolution time. By default, the evolution is not stopped until the evolution of the current generation has completed, but this behaviour can be changed so that the evolution can be stopped within a generation.

Fitness Threshold:

It is a termination method that stops the evolution when the best fitness in the current population becomes less than the user-specified fitness threshold and the objective is set to minimize the fitness. This termination method also stops the evolution when the best fitness in the current population becomes greater than the user-specified fitness threshold when the objective is to maximize the fitness.

Fitness Convergence:

It is a termination method that stops the evolution when the fitness is deemed as converged. Two filters of different lengths are used to smooth the best fitness across the generations. When the smoothed best fitness from the long filter is less than a user-specified percentage away from the smoothed best fitness from the short filter, the fitness is deemed as converged and the evolution terminates.

Population Convergence:

It is a termination method that stops the evolution when the population is deemed as converged. The population is deemed as converged when the average fitness across the current population is less than a user-specified percentage away from the best fitness of the current population [1-5] and [8-9].

Different techniques of linear programming: ***Simplex Method:-*** The simplex method is an iterative procedure by which a new basic feasible solutions can be obtained from a given basic feasible solution which improves the value of the objective function.

Dual Simplex Method:- In the simplex method or primal simplex method we begin with and maintain at each iteration a basic feasible solution and we change bases until the optimality criteria is satisfied. In the dual simplex method we maintain continues satisfaction of the optimality criterion and change bases until we get the

→Feasibility of the basic solution:

Thus the dual simplex method is a kind of mirror image of primal simplex method as regards feasibility.

The general nature of the primal simplex method consists in selecting a vector to enter to leave the bases; the entering or leaving vectors corresponds to adjacent extreme points of the convex set of feasible solution. The sequence of steps in selecting the entering or leaving vectors are different in different algorithms but they ensure both primal or dual feasibility and a monotone increase or decrease in the objective function.

Areas of application of linear programming: The potentiality of linear programming as a tool for solving problems is substantial. It is used to solve problems of procurement of raw materials in changing situations, production planning, assembly line balancing and many other problems of operation management [10].

PROPOSED TECHNIQUES

The proposed Genetic Algorithm is blind optimizers which do not use any auxiliary information such as derivatives or other specific knowledge about the special structure of the objective function. If there is such knowledge, however, it is unwise and inefficient not to make use of it. Several investigations have shown that a lot of synergism lies in the combination of genetic algorithm and conventional methods. The basic idea is to divide the optimization task into two complementary parts. The GA does the course, global optimization while local refinement is done by the conventional method. A number of variants are reasonable:

1. The GA performs the coarse search first. After the GA is completed local refinement is done.
2. The local method is integrated in the GA. For instance, every k generations, the population is doped with a locally optimal individual.

3. Both methods run in parallel: all individuals are continuously used as initial values for the local method. The locally optimized individuals are re-implanted into the current generation.

Pseudo code of Genetic Algorithm:

```
Begin
Input initial population
Select n number of random population
 Do
Select best ranking individuals to reproduce;
Apply crossover operator;
Apply mutation operator;
Determine functional value ;
Evaluate each individual's fitness;
End
```

In Constrained Optimization we have to find the best solution (The optimum point) with respect to various constraints.

In Linear Programming (LP), all of the mathematical expressions for the objective functions and the constraints are linear. The programming in linear programming is an archaic use of the word "Programming" to mean "Planning". We can think linear programming as "Planning with linear models".

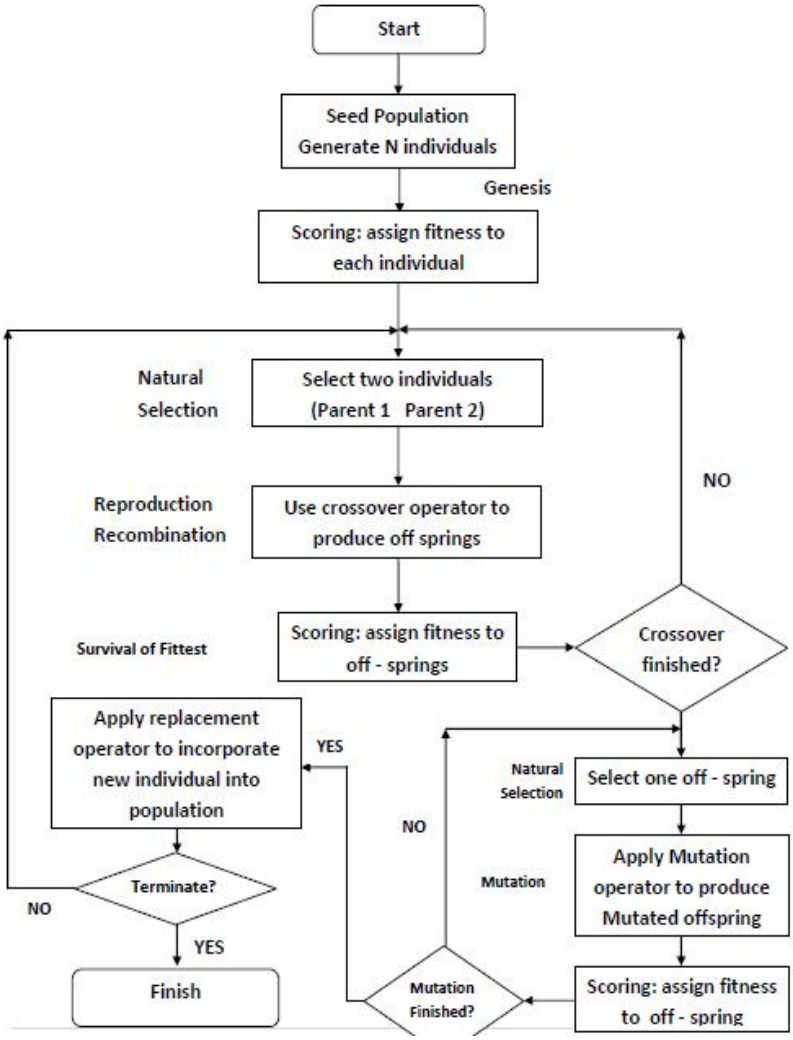


Figure 1: Flowchart of Genetic Algorithm

Here we are applying genetic algorithm to a specific optimization problem step by step. Consider the problem of maximizing the function $f(x) = x^2$, where x is permitted to vary between 0 and 31.

Step 1:- To use a genetic algorithm we must first code the decision variables of our problem as some finite- length string. So we will code the variable x as a binary unsigned integer of length 5. As with a five-bit (binary digit) unsigned integer we can obtain numbers between 0(00000) and 31 (11111).

Step 2:- Select an initial population at random. The initial population is obtained in the form of binary strings. These strings are then converted to their corresponding decimal values. And then calculate the corresponding fitness function value for each x.

String No	**Initial Population (Randomly Generated)**	***x* Value (Unsigned Integer)**	***f(x)* = x^2**
1	01101	*13*	*169*
2	11000	*24*	*576*
3	01000	*8*	*64*
4	10011	*19*	*361*

Step 3:- A generation of the genetic algorithm begins with reproduction. We select the mating pool of the next generation by spinning the weighted roulette wheel four times. Actual simulation of this process using coin tosses has resulted in string 1 and string 4 receiving one copy in the mating pool, string 2 receiving two copies, and string 3 receiving no copies, as shown in the following table. Comparing this with the expected number of copies (n.pselecti) we have obtained what we should expect: the best get more copies, the average stay even, and the worst die off.

Step 4:- With an active pool of strings looking for mates, simple crossover proceeds in two steps:-

i) Strings are mated randomly

ii) Mated string couples cross over.

Referring to the following table, random choice of mates has selected second string in the mating pool to be mated with the first. With a crossing site of 4, the two strings 01101 and 11000 cross and yield two new strings 01100 and 11001. The remaining two strings in the mating pool are crossed at site 2; the resulting strings may be checked in the following table. Here the crossover probability is assumed to be unity pc =1.0.

Step 5:- The last operator, mutation, is performed on a bit-by-bit basis. We assume that the probability of mutation in this test is 0.001. With 20 transferred bit positions we should expect 20*0.001=0.02 bits to undergo mutation during a given generation. Simulation of this process indicates that no bits undergo mutation for this probability value. As a result, no bit positions are changed from 0 to 1 or vice-versa during this generation.

RESULTS ANALYSIS

We have applied Genetic algorithm and Simplex method on the same LPP and find out the difference in the result. Our tested LPP is as follows

$$\text{Max } F(x1, x2) = 4x1 + 3x2$$
$$2x1+3x2 <= 6$$
$$-3x1+2x2 <=3$$
$$2x1+x2 <=4; \quad 0<=x<=2$$

When we have applied the Simplex method then it generates only one set of solution where x1 = 1.5 and x2 =1 and thus F(x1, x2) = 9. But when we have applied the GA we get a series of feasible solutions and at a certain generation we have achieved the greater value of first chromosome i. E x1=1.6 in comparison with previous one. So this is a big achievement for us. Beside this we have a set of feasible solutions which can be applicable where it is needed according to the requirements. This is a remarkable benefit of using Genetic Algorithm. The best fitness, the best individual, best selection graphs determinate with our proposed genetic algorithm in the range 0-1.0 to 0-1.5.

Best Fitness Graphs using Genetic Algorithm

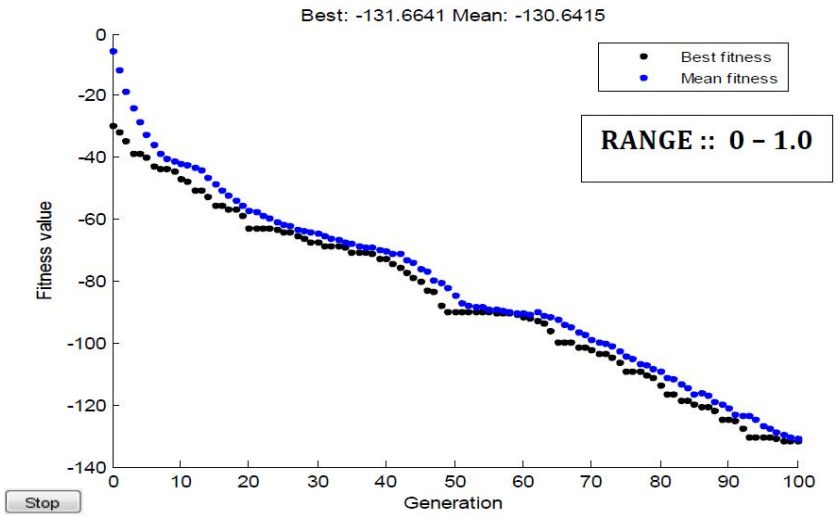


Figure 2: Best Fitness Graph Using Genetic Algorithm (Range 0-1.5)

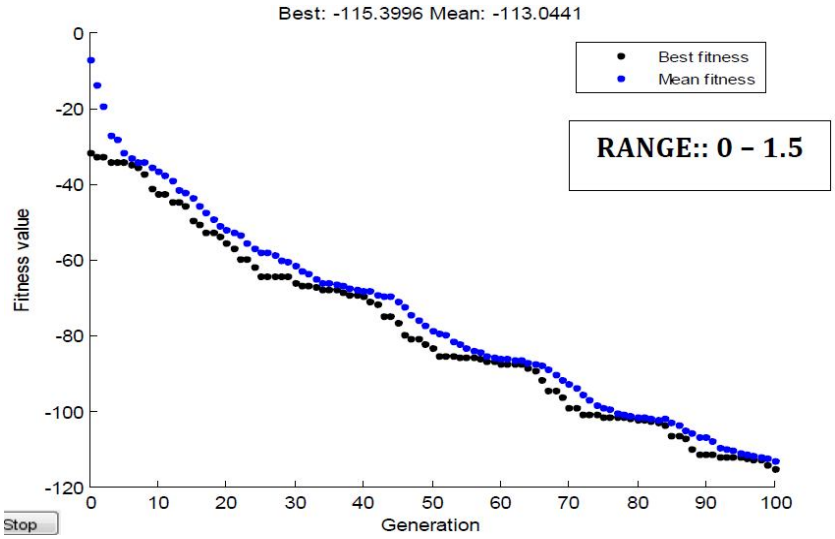


Figure 3: Best Fitness Graph Using Genetic Algorithm (Range 0-1.5)

Best Individual Graphs using Genetic Algorithm

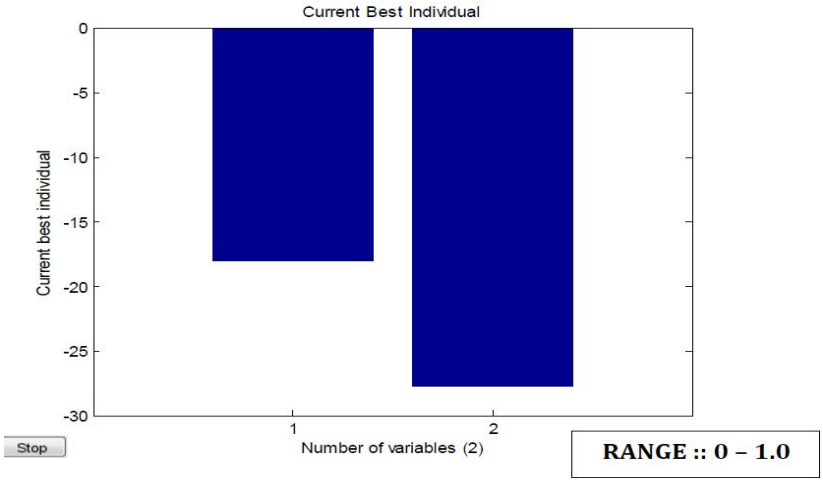


Figure 4: Best Individual Graphs using Genetic Algorithm with Range 0-1.0

Best Selection Graphs using Genetic Algorithm

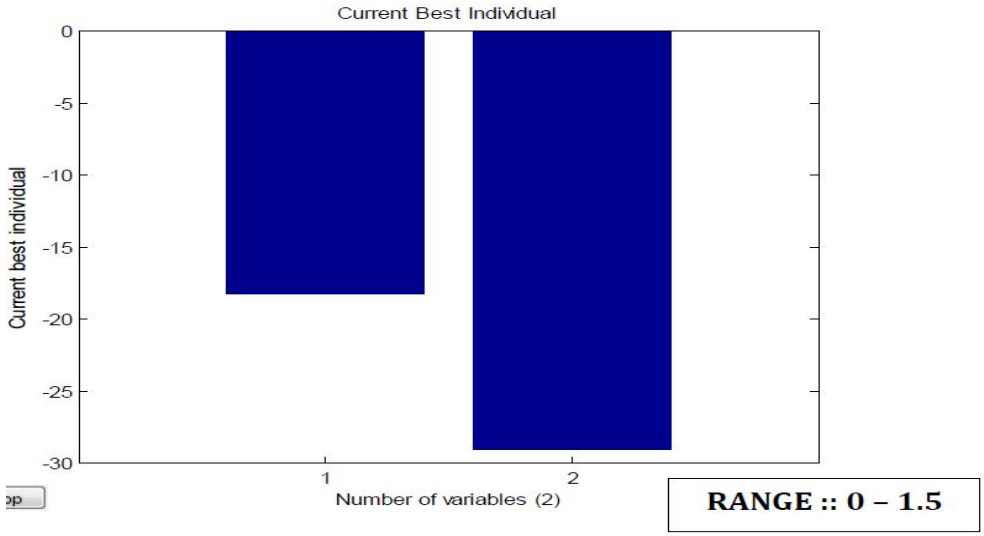


Figure 5: Best Individual Graphs using Genetic Algorithm with Range 0-1.5

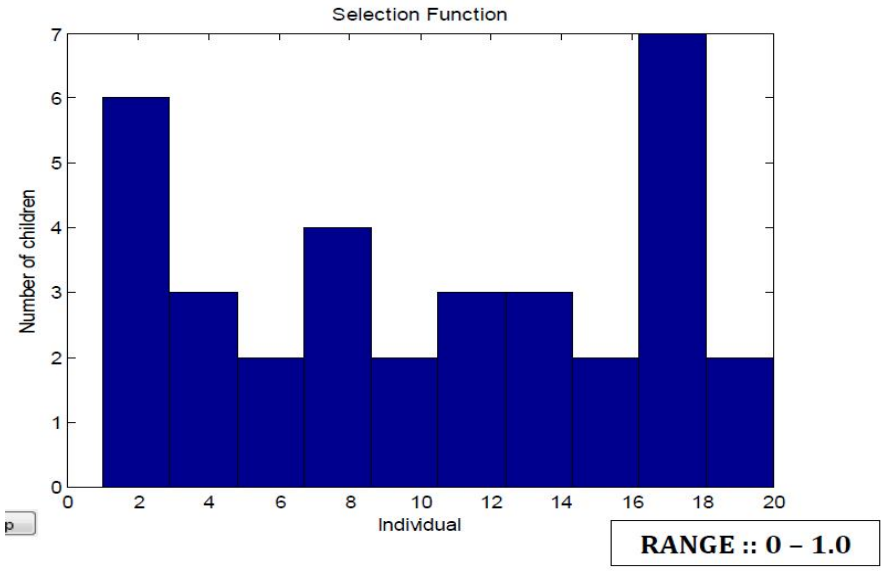


Figure 6: Best Individual Graph for range (0-1.0)

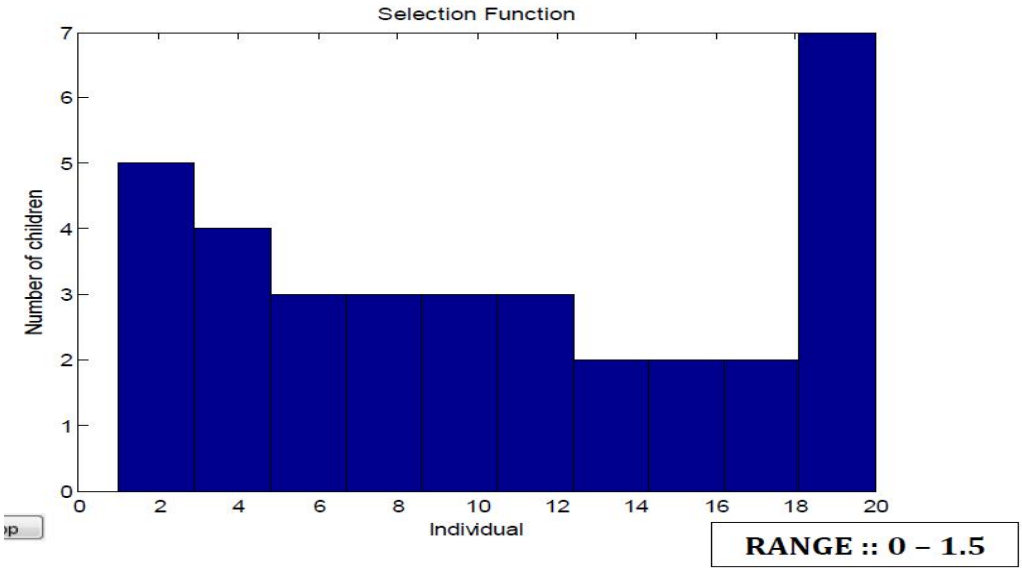


Figure 7: Best Individual Graph for range (0-1.5)

Fitness Chart of Our Experiment:

Generation	*f-count (Best)*	*f(x) (Mean)*	*f(x) (Stall)*	*Generations Range (0-1)*
1	40	2.761	5.648	0
2	60	-0.02282	4.37	0
3	80	-0.2393	2.883	0
4	100	-3.239	1.95	0
5	120	-3.239	0.4281	1
6	140	-4.739	-1.021	0
7	160	-6.239	-1.626	0
8	180	-6.239	-2.667	1
9	200	-6.239	-4.346	2
10	220	-8.052	-5.667	0
11	240	-8.052	-5.927	1
12	260	-8.989	-6.643	0
13	280	-8.989	-7.193	1
14	300	-8.989	-8.177	2
15	320	-8.239	-8.693	0
16	340	-8.989	-8.952	0
17	360	-11.24	-8.927	0
18	380	-11.24	-8.483	1
19	400	-12.24	-9.564	0
20	420	-12.24	-9.596	0

CONCLUSION AND FUTURE WORKS

Genetic algorithm (GA) is an intelligent search technique. GA combines the good information hidden in a solution with good information from another solution to produce new solutions with good information inherited from both parents, inevitably leading towards optimality. The results can be very good on some problems, and rather poor on others. If only mutation is used, the algorithm is very slow. Crossover makes the algorithm significantly faster. GA is a kind of hill-climbing search; more specifically it is very similar to a randomized beam search. As with all hill-climbing algorithms, there is a problem of local maxima. Local maxima in a genetic problem are those individuals that get stuck with a pretty good, but not optimal, fitness measure. Any small mutation gives worse fitness. Fortunately, crossover can help them get out of a local maximum. Also, mutation is a random process, so it is possible that we may have a sudden large mutation to get these individuals out of this situation. If the conception of a computer algorithm being based on the evolutionary of organism is surprising, the extensiveness with which this algorithm is applied in so many areas is no less than astonishing. These applications, be they commercial, educational and scientific, are increasingly dependent on this algorithms, the Genetic Algorithms. Its usefulness and gracefulness of solving problems has made it a more favorite choice among the traditional methods, namely gradient search, random search and others. GAs is very helpful when the developer does not have precise domain expertise, because GAs possesses the ability to explore and learn from their domain. In this thesis we have applied GA on linear programming problem (LPP), but this can also be applicable in non-linear programming problem. We can apply GA in those cases where a number of solutions may come but optimal solution is not known. The most vital situation is crossover point and mutation step. Selection of crossover point is basically done in random wise. So in future, a technique should be developed for choosing the crossover point, so that it helps Genetic algorithm to become more robust and fast optimization procedure. In future, we would witness some developments of variants of GAs to tailor for some very specific tasks. This might defy the very principle of GAs that it is ignorant of the problem domain when used to solve problem. But we would realize that this practice could make GAs even more powerful.

REFERENCES

1. GA in search, optimization and machine learning' by David E. Goldberg
2. Principles of soft computing' by S. N. Sivanandan, PSG college and S. N. Deepa, Anna University
3. Artificial Intelligence' by Amit Konar, Jadavpur University
4. Computational Intelligence' by Amit Konar, Jadavpur University

5. R.Sivaraj and Dr.T.Ravichandran, "An Efficient Grouping Genetic Algorithm", International Journal of Computer Applications (0975 – 8887) Volume 21– No.7, May 2011.
6. Ehsan Heidari and Ali Movaghar, "AN EFFICIENT METHOD BASED ON GENETIC ALGORITHMS TO SOLVE SENSOR NETWORK OPTIMIZATION PROBLEM", International journal on applications of graph theory in wireless ad hoc networks and sensor networks (GRAPH-HOC) Vol.3, No.1, March 2011.
7. 'Genetic Algorithm' by Tom V. Mathew, IIT Bombay
8. Chambers, L. (1995), Practical handbook of genetic algorithms: Applications, Vol.I, CRC Press, Boca Raton, Florida.
9. Gen, M. and Cheng, R. (2000), Genetic algorithms and engineering opimization, John Wiley, New York.
10. 'Linear programming and game theory' by J.G. Chakravorthy, University of Calcutta and P.R.Ghosh, University of Calcutta.