Genetic Algorithm and its application to Big Data Analysis

Munawar Hasan

Abstract – In this paper, I have described Genetic Algorithm for combinatorial data leading to establishment of mathematical modeling for Information Theory. The paper describes GA (Genetic Algorithm) in light of information theory and then derives Mathematical Framework covering but not limited to Big Data. This framework is a boon to any domain that has to harness data explosion like neurobiology, statistical inference, quantum computation, entropy management, encoding information theory (cryptography) etc. The efficiency calculated here describes incremental probabilistic approach to reach a solution which is highly stable, evolutionary and self adjusting henceforth invents a extremely new approach for computational analytics.

Index Terms – Genetic Algorithm, Big Data, Evolution, Combinatorial Computation, RB Tree, Sigmoid, Convergence

1. INTRODUCTION

They are the class of algorithm which can leverage evolution based heuristic techniques to solve a problem. During past 20 years, there has been great research and development over the adaptive nature of the Genetic Algorithms (GA). They are represented by chromosome like data structure which uses recursive recombination or search techniques. Let u be a set of sample points, u = {u_1,u_2,u_3,….,u_n}. Then from a genetic algorithm we can obtain optimal set of recombination and selection on basis of some semantic analytics. So from above sample points we could get a set {u_1,u_2,….,u_i} in one set having similar properties and {u_j,u_k,……u_n} as another set.
In broader sense, GA is formulated on five basic principles:

1) Initialization:
The set containing the whole sample points (the entire population) marks the initialization of the GA. Sample points or population may consist of the database tables or direct input of real life scenarios. The later principles changes accordingly. In case of database tables, the semantic analytics are type casted into statistical theorem using random number generators creating density functions. In case of real time data, the whole process shapes as natural language parsing, then statistics is deployed.
2) Selection :
The idea of Selection pivots around the Darwin's theory of survival of the fittest. A subset is created of the set obtained from previous step. This subset is just a way to categorize data. They may contain data that seems to be logically related at a particular instant of time or otherwise. They may contain multiple sets, each having specific domain data like, data relating to customer shopping trends, data relating to customer grievances etc.
3) Cross over/ recombination
The principle of cross over is the line of demarcation of the general theory of GA. The gene, an individual member of a particular set, is crossed over with a gene of another set. This cross over results in exchange of behavior and trends over different sets creating logical relation between set and reducing degree of randomness among sets. At any instant cross over is done only between two respective genes, leaving all other unaltered.
4) Mutation:
The prior idea of mutation is to generate genetic diversity. In cross over, the properties and trends of other genes are taken which may result in suppression of one's own properties (alleles), thus mutation is important way to maintain individuality. Mutation may also result in generation of important alleles via combination of different genes.
5) Acceptance:
After having done with mutation we generate new offspring, i.e. candidate for next level of iteration. But not all candidates are healthy to be taken to next step. This is where the elimination occurs. During this elimination round we calculate the percentage of acceptable features for a single gene. We put a cut off mark, similar to the activation mark in neural network. Those genes that cross that cut off mark are the next level population. This whole process continues till change between two successive level genes is negligibly small.

2. Why and Where GA?

The GA is applied over the problem domain where the outcome is very unpredictable and the process of outcome contains complex inter related modules. Also GA is very apt for such class of problem where problem specification is very difficult to formulate. During the last few decades, computer science has seen huge advancements in demands and its implementation. As per now heavy cross demands are in fire and hence implementation and analytics is becoming more and more chaos. The situation is very apt for applying genetics and getting optimal results. There is also class of problem which is related to idea of super polynomial explosion or combinatorial explosion. In this kind of problem we are interested in suitable solution as per scenario rather than actual solution. GA is also becoming very popular in such type of problematic domains. In health care domain, there are excessively increase in demand for different kind of pharmaceuticals relating to different disease. For example in AIDS, the human immunodeficiency virus becomes resistant to antibiotic after a definite span of time. From there on the patient is feed with completely new kind of antibiotic. This new antibiotic is prepared by analyzing the pattern of the human immunodeficiency virus and its resilient nature. As the time passes by, finding the required pattern becomes very complex and hence inaccuracy emerges. The GA with its evolutionary based theory is a boon in this field. The genes defined by the algo, generates an evolution based antibiotic for the required patient. One such remarkable mention is the IBM's EuResist genomics project in Africa, where they have deployed their powerful genomics architecture with the neural networks for the HIV treatment.
Any Genetic Algorithm includes following steps:

1) Generation of random variable to random expectation for populate chromosomes.

2) Compute fitness function f(x) | ᵛ (x) | x€ chromosome
3) Generate New Population:
   a) Selection : Choose the parent chromosome
   b) Crossover : Create offspring
   c) Mutation : Generate and check the probability keeping each bit of chromosome as locus
   d) Acceptance : Acquire the new generation
4) Replacement : Choose the new offspring
5) Goto Step 1.

2.1 The Chromosome:

The whole process of GA is based on idea of natural evolution and selection. This knowledge of evolution and selection in encoded in chromosome and can be analyzed with conditional or random probability. As evolution passes by, there are some features that become obsolete with time. Henceforth, the idea is to only capture those features which are useful. In layman's term chromosome is the basic unit in running a GA, right from problem formulation to solution presentation. The place where chromosome recedes in a set becomes a disputed topic in field of GA. I have tried to stay away from this dispute when I show GA in some applications by using the random variable generators. This makes the position of chromosome irrelevant.

2.2 Representation of the Chromosome:

The representation of any GA chromosome is dependent on problem its is formulated on. There are basically two broad categories to represent a chromosome. When the problem contains distinguishable components like the Travelling Salesman Problem (TSP), then binary representation may be done. In such representation, each gene is represented by 0 or 1. The alleles (group of genes) so formed becomes the set of binary values. In such cases, the different GA steps like cross over etc are performed using the binary operators. The other form of representation is used when problem has not got decisive components to begin with. Example of such problem is the data mining, health care etc. Here rather than sticking to elementary notion of binaries, the random probability with complex statistical and applied mathematical concept is used. The representation makes it clear that GA may contain complex computing components and with these components comes their *time space trade off* (the complexity analysis). We will discuss this later in this paper under combinatorial explosion.

2.3 The Idea of Random Population:

In applied mathematics, the random variable denotes the expectancy of an event in a sample space. Here sample space means the problem definition and the expectancy is the variation in its (a sample point in sample space) occurrences due to some reason (in GA reasons results from semantics). The general theory of probability made random variables untouched, it was applied mathematics or advance probability where there arouse need to create random number and hence population. In now days, these random numbers are very common in cryptography in generating public and private key (TGS [Ticket Granting System]) and in implementing Kerberos. The probability theory, defines the random probability as the mathematics of non-deterministic science (or theory of chaos).

3. Paradox of Exponential Degradation or Combinatorial Explosion:

They refer to the class of problems where the solution set runs in exponential time. They are primarily known as solution sets that run in super polynomial time. The term super polynomial means that the complexity of the algorithm is defined by O(2^n), where n is the number of inputs, in our case the sample space. Such problems contains huge set of false solution, hence finding and sorting such false solution itself falls into another big solution. The errors in analysis in any one even part deteriorate the solution convergence, a term referred as degradation in computational science. Let take a hypothetical example from health care. Let there be 20 rules to create an antidote. Now due to constantly using this antidote, there harmful bacteria have become resistant to basic formula. So, some 20 additional rules are generated to counter such bacteria. Now each individual 20 rules must be matched with additional 20 rules, there by resulting total combination of magnitude 20^{20} i.e. 104857600000000000000000000. Now let us try to solve this with sun ultra sparc processor having simultaneous 20 processor running at any instant of time. Let all 20^{20} be run independently (which is not practically feasible as different patterns are dependent on each other). It will take about 15 days to solve this kind of combination. This analysis shows how we actually deviate from results in big solution problems. The concept of amortized analysis is introduced to tackle such problems. There are plenty of approaches developed with time and experience that saves us from exponential degradation. IBM uses great amount of amortized analysis in its EuResist project.

Below I have shown the GA application in big data analysis and in optimization of problem. I have developed the methodology to implement them and their approach is entirely new in nature and distinct from all available in market, making the whole suite completely new of its kind.

4. Big Data Analysis using the Genetic Algorithm:

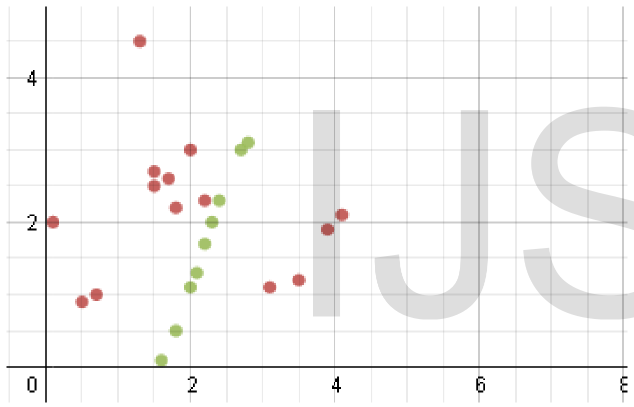
The field of Information Theory refers big data as datasets whose rate of increase is exponentially high and in small span of time; it becomes very painful to analyze them using typical data mining tools. Such data sets results from daily capture of stock exchange, any credit card user's timely usage trends, insurance cross line capture, health care services etc. In real time these data sets go on increasing and with passage of time create complex scenarios. Thus the typical data mining tools needs to be empowered by computationally efficient and adaptive technique to increase degree of efficiency by using adaptive techniques.

Using GA over data mining creates great robust, computationally efficient and adaptive systems. In past there have been several researches on data mining using statistical techniques. The statistics that have heavily contributed are the ANOVA, ANCOVA, Poisson's Distribution, and Random Indicator Variables etc. The biggest drawback of any statistical tactics lies in its tuning. With exponential explosion of data, this tuning goes on taking more time and inversely affects the through put. Also due to their static nature, often complex hidden patterns are left out. The idea here is to use genes to mine out data with great efficiency. Also I will show how this mined data can be effectively used for different purposes.

Rather than sticking to general notion of probabilities, I have here used the concept of Expectations, and have modified the theory of Expectations to achieve the desired results. Any data categorizes of three main components, the constants, the variables and the variants. The constant comprises of data that practically remains unaltered in a given span of time. The variables are changing with time while in case of variants; it is not clear whether they will behave as constant or variables. So, taking this as the first step, we have three set each containing respective data as stated. Now we will calculate the expectancy of each datum inside the data set.

4.1 Calculation of Expectancy:

The expected value of any random variable is consistent when the summation is finite and convergent. The idea of convergence of random variable here used is of tensor (a point in sample space which has a definite magnitude and direction but whose direction is dependent on other point's directions in the sample space). So from this tensor definition, it is clear that there is a degree of interrelation between the random variables, which has a tendency to align degree of randomness towards a particular direction. This tendency of alignment is very apt and exposable using GA.



The graph above shows the plot of all the points in sample space. From the sample space, such points are picked up which creates a tensor. The green points shows the plot of a pattern which leads in some direction. When we are iterating in beginning we are not sure that the direction shown here is leading to a solution. But as we go on towards inner steps, at each step our possibility increases of reaching a solution in direction of plot. From the graph, a notion comes that we may have used some other points. This is where the tensor comes into play. The tensor described here is of recursive nature. Every next point is decided by the position of previous point maintaining continuity. Hence at any instant of there would be exactly one and one solution for plot.

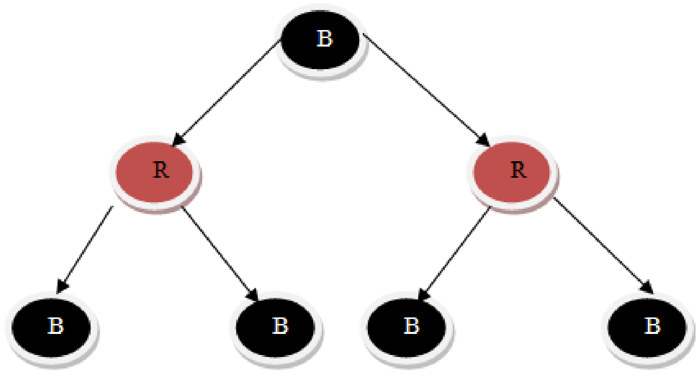
This set with the expected value marks the beginning of the GA. Each individual value inside each set represents a stateless chromosome. The word stateless is used because of the random origin of chromosome. Each chromosome is a potential candidate for the next level. Thus each set contains a set of population. Henceforth, we have created our first condition towards the GA. Now we go ahead and create condition for crossover. I have done the genetic crossover using the concept of RB-Tree. This is very unique concept and hasn't been much explored until now. RB-Tree is heavily used in algorithm design to make heavy optimization and create efficient bucket for data storage. In fact the Linux kernel is also getting into heavy usage of RB-Tree in subsequent releases. Below is the pseudo code for constructing a RB-Tree, I have customized the algo to meet our needs of GA.

```
RB-INSERT(T,z)
        y<-nil[T]
        x<-root[T]
    while x != nil[T]
        do  y<-x
          if key[z]<key[x]
            then x<-left[x]
          else x<-right[x]
          p[z]<-y
        if y=nil[T]
      then root[T]<-z
    else if key[z]<key[y]
          then left[y]<-z
          else right[y]<-z
      left[z]<-nil[T]
      right[z]<-nil[T]
      color[z]<-RED
  RB-INSERT-FIXUP(T,z)
```

```
RB-INSERT-FIXUP(T,z)
while color[p[z]]=RED
do if p[z]=left[p[p[z]]]
   then y<-right[p[p[z]]]
        if color[y]=RED
      then color[p[z]]<-BLACK
           color[y]<-BLACK
           color[p[p[z]]]<-RED
           z<-p[p[z]]
      else if z=right[p[z]]
           then z<-p[z]
           LEFT-ROTATE(T,z)
      else color[p[z]]<-BLACK
           color[p[p[z]]]<-RED
           RIGHT-ROTATE(T,p[p[z]])
   else y<-left[p[p[z]]]
        if color[y]=RED
        then color[p[z]]<-BLACK
             color[y]<-BLACK
             color[p[p[z]]]<-RED
             z<-p[p[z]]
        else if z=left[p[z]]
             then z<-p[z]
             RIGHT-ROTATE(T,z)
        else color[p[z]]<-BLACK
             color[p[p[z]]]<-RED
             LEFT-ROTATE(T,p[p[z]])
color[root[T]]<-BLACK
```

After constructing the required tree, copy all the nodes just before the leaf to leaf creating duplicate values of leaf and its predecessor. This customization will provide stability to solution. The rotate functions proved respective shift to one place right or left.

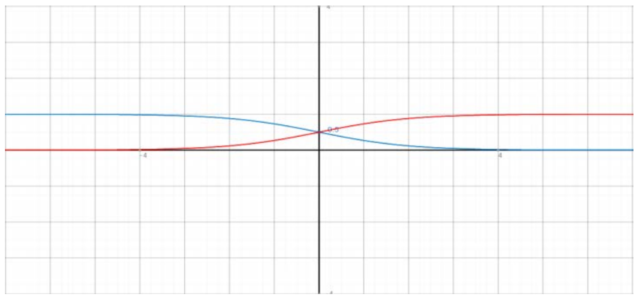
By running above algo, we will have following kind of RB-Tree:



Now we will proceed towards the second step of the GA, i.e. the cross over. For cross over I have used an ambitious approach of tree cross over. As per now we have three RB-Trees, now by cross over we will get hidden relationship between the disjoint sets. I have realized there is a definite way of crossing of tree. Here I define a parameter µ, called the scaling parameter. The black nodes when crossed over with the red nodes, or red nodes crossed over with black nodes, the final value of cross over must be multiplied by 10µ. In other words, black-red or red-black cross over is one tenth of black-black cross over. The red-red cross over must be multiplied by 25 to 30 times the black-black cross over. This is actually weighted GA concept to create uniform probability for all data.
For the cross over function, we define inverse sigmoid as the criteria, given by:

$$f(x) = {}^{1}/_{1+e^{x}} + {}^{1}/_{1+e^{-x}}$$

The function f(x) is hyperbolic in nature. We are only interested in modulus of this evaluation i.e. the positive part. The point to be noted, we will never omit the negative part at any instant. The evaluation process will go till end as usual, the final result will the mod of final value. The graph below shows the plot.



From the graph, it is clear that, the values are mirror images across the x axis. Now we proceed to the next step, the mutation. Mutation is very important phase in data centric applications. From mutation, the data mining gets robust and data so created becomes logically connected, but not knowledgeable. Here I have used two consecutive mutation scenarios. During the first mutation deep connected pattern is established, while the second one narrows down the logical connection and sorts out the healthiest connections.

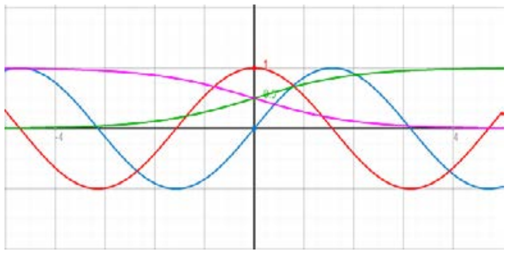
1) GA Mapper 1: $(\sin x * \cos x)$

   The traditional sine cosine association, create a finite correlation between the inputs. The correlation is very deep and hence as summation is done for inner values, very small floating point data is also included, hence we must use at least one function that creates an intersection of the sine and cosine and hence narrow down the search results and henceforth pick up only healthy associations.

2) GA Mapper 2: $\sum_{n=0}^{p} {}^{(-1)^n x^{(2n+1)}}/_{(2n)!} + x \ln x + ceil(sinx)$

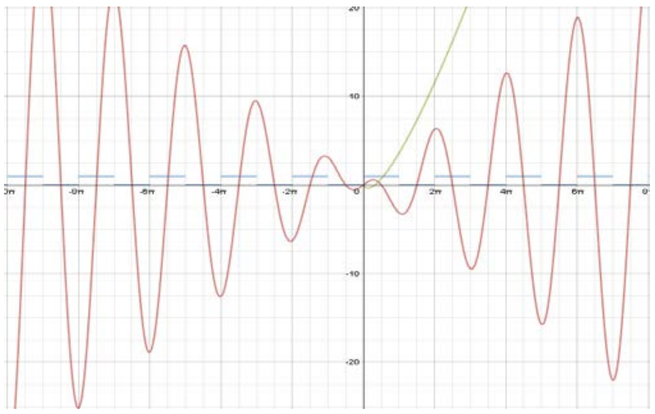
   The first term of the equation is the well know Taylor's series, the second is the logarithmic series and the last is the ceil function. The Taylor's series creates a subset that makes finer correlation. The logarithmic series and the ceiling function restrict the data subsets, thereby making only useful data count. Graphically:

   Mapper 1:
   Shows how the sine and the cosine graphs cut the sigmoid graphs. As we stated before we are only interested in positive graphs.
   Inference:
   Data sets which are correlated by sine, cosine and the sigmoid graph. Theoretically, this data contains all possible combinations that have any inter dependence between them. If we do a permutation of among all data sets, the result will be a super set of this mapper.



   Mapper 2:
   Shows the narrowing down of sample space. The space is now reduced to a small region visible between graph color orange, green and blue.
   Inference:
   All logically related data sets are achieved here. Those that were feebly related or casually related are sorted out. Also sorted out are those which were potential dangers against knowledge acquisition. The property of knowledge acquisition describes a set of data that are defined by many to many onto function, meaning each

data has a relation to every other data and position of any data is dependent of every other data.



Now, moving towards the steps ahead, we calculate the delta, the degree of permissible error. This PE [permissible error] shows our deviation from the actual result and hence decides the contestants for the next iteration. This process comes under the acceptance of the GA. If any datum goes well inside the PE, then it becomes the successful candidate for the actual population.
For establishing the validation of equation, we will use the concept of loop invariant, commonly used inside loops. Well we are here constantly going through a cycle of different phases of GA, hence loop invariant concept is quite applicable here.

4.2 Validation of Equation:

Initialization:

We must start by showing a base of validation just before applying GA. This initialization is done using the population gathering. Each of three sets, as described before, contains elements that fall into their domain of application. Let the first set contains elements $\{x_1, x_2, \ldots\ldots, x_n\}$. In worst case complexity each of this n elements qualify for final population, hence the final outcome is the super set of this set. This shows that loop invariant concept holds just before the iteration of GA starts.

Maintenance:

To validate on the concept of loop invariant, we must show its consistency between i th and (i+1) th step. At the i th step, then result is obtained via (i-1) th step. Let this be α, now for i th step:

$$\text{The sigmoid: } {}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}$$

After applying the sigmoid, we a probability density function (pdf) wrt all members of other respective set. Now we go on applying the respective mapper function one by one.

Mapper1

$$\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]$$

Mapper2

$$\sum_{n=0}^{p} {}^{(-1)^n \left\{\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}\right] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]\}^{(2n+1)}}\Big/_{(2n)!}$$

$$+$$

$$\left\{\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}\right] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]\} ln \left\{\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}\right] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]\} +$$

$$ceil(sin\left\{\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}\right] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]\})$$

Now partially differentiating the above equation (say ρ) wrt α:

$$\frac{\partial \rho}{\partial \alpha i} =$$

$$\lim_{\alpha i \to n}\{\sum_{n=0}^{p} {}^{(-1)^n \left\{\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}\right] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]\}^{(2n+1)}}\Big/_{(2n)!}\}$$

$$+$$

$$\lim_{\alpha i \to n}\{\left\{\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}\right] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]\} ln \left\{\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}\right] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]\}\} +$$

$$\lim_{\alpha i \to n}\{ceil(sin\left\{\sin[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}\right] * \cos[{}^{1}/_{1+e^{\alpha i}} + {}^{1}/_{1+e^{-\alpha i}}]\})\}$$

We repeat the above process for both (i-1) and (i+1). Let $\omega_{i-1,I}$ numerical difference between the (i-1) th and i th step. The negative should be neglected as we are concerned only with magnitude. Hence

$$\omega_{i-1,I} = \frac{\partial \rho}{\partial \alpha i} - \frac{\partial \rho}{\partial \alpha i - 1}, \quad \omega_{i,i+1} = \frac{\partial \rho}{\partial \alpha i + 1} - \frac{\partial \rho}{\partial \alpha i}$$

Now, plot the obtained values on previously obtained values .i.e. from 1 to (i-2) th step. The plot must be continuous in nature, as all the function we are using is continuous and covergible henceforth the continuous nature of association of function alone is sufficient to generate validity.

Termination:

The termination is established at the last iteration. Let this last iteration be n th, here also we plot the $\omega_{n-2,n-1}$ and $\omega_{n-1,n}$ values and check continuity. In this way we create the validation of the equation.

Thus, in the way shown above we can analyze big data and create a great degree of accuracy.

5. References:

- Rich and Knight, Introduction to Artificial Intelligence
- Thomas H Coreman, Charles E Leiserson, Ronald L Rivest, Clifford Stein, Introduction to Algorithms
- Kristian Guillaumier, Generic Chromosome Representation and Evaluation for Genetic Algorithms
- Fozia Hanif Khan, Nasiruddin Khan, Syed Inayatullah, Shaik Tajuddin Nizami, Solving TSP Problem By Using Genetic Algorithm
- William H Hsu, Genetic Algorithms
- SM Gabber, Nour ElDinn, Genetic Algorithm and Random Number Generators

- Brain Farrell, John Konya, Meeting the Big Data Challenge
- Jiayu Zhou, Youfang Lin, Xi Wang, Visualization of Large Scale Weighted Clustered Graph: A Genetic Approach
- Behrouz Mineai, William Punch, Using Genetic Algorithm for Data Mining Optimization
- Vitoantonio Bevilacqua, Giuseppe Mastronardi, Filippo Menolascina, Angelo Paradiso and Stefania Tommasi, Genetic Algorithms and Artificial Neural Networks in Microarray Data Analysis: A Distributed Approach
- Paul O Lewis, A Genetic Approach for Sequence Data
- Sufal Das, Benani Saha, Data Quality Mining using Genetic Algorithm
- Shital S Shah, Andrew Kusiak, Data Mining with Genetic based Selection
- Li Lin, Longbin Cao, Application of Genetic Algorithm in Stock Market Data
- Randy L Haupt, Practical Genetic Algorithm
- Matlab-Reference, http://www.mathworks.com/help/
- Terry J Woodfield, Predictive Modeling in Insurance Industry
- Magnus Lei Hetland, Mastering Basic Algorithms in Python
- Flechter, C Gardner, Guide to Financial Modeling using Python

Ref (1) Thomas H Coreman, Charles E Leiserson, Ronald L Rivest, Clifford Stein, Introduction to Algorithms

6 Biography:

Munawar Hasan:

He is an Algorithm Developer at Computer Sciences Corporation (CSC) for last 2.7 years. Most recently he developed a Predictive Algorithm for Financial Modeling to detect several types of Fraud in Banking and Insurance. He has been active contributor to CSC Innovation Lab and provided numerous design techniques for financial data. He also has a year experience working in Cloud Computing and came up with his own Computing framework for Cloud to facilitate true virtualization. He has written several white papers on topics related to computational analysis and cloud simulation
He was born and brought up in Bodh Gaya and Graduated from Dehradun. He currently lives in New Delhi