Feb. 28

International Journal of advanced studies in Computer Science and Engineering

IJASCSE, Volume 3, Issue 2, 2014

Genetic algorithm: an overview and its application

Vijay Kumar Verma[1]
Asst. Professor Dept. of CSE NSIT, Bihta, Patna

Biresh Kumar[2]
Research Scholar NIT Patna

Abstract: There exist many problems for which there are not any clear and precise method by means which optimal or best solution can be found out. These type of problem fall into category of combinatorial problems as they can only be solved through random search techniques. The difficulty is that many problems have so many possible solutions and trying all of them for optimal solution is practically not feasible. To cope up with these types of problems scientists have applied various search methods and heuristics. These are nothing but optimization process applied in several domains such as computer science, computational biology, drug designing and other fields. Genetic algorithm is one such optimization process inspired by laws of natural evolution. This paper demonstrates the basic understanding of Genetic algorithm and principle that works behind it with some of its application and basic implementation using tool R.

Keywords— GA, Optimization, Search Heuristics, R, Genetics, natural selection, Combinatorial problems, TSP, Knapsack

I. INTRODUCTION

GA (Genetic Algorithm) is an optimization and search techniques based on the principles of Genetics and Natural Selection. Natural selection always tends to pick the fittest individuals dominating over the weaker ones and it always favours the positive adaptation resulting into the best one to survive in the long run. This is what depicted in Darwin's Survival of the fittest theory. GA is a part of the evolutionary optimizing computing inspired by Darwin's survival of the fittest.GA is an adaptive search heuristics that mimics the process of natural evolution which uses techniques like Selection, Crossover, Inheritance and Mutation.GA represents an intelligent exploitation of a random search used to solve Optimization problems. It is one of the rapidly growing areas of Artificial Intelligence.

Basic understanding:

In optimization process solving problems means finding best solution among others. TSP (travelling salesman problem) is one such combinatorial problem where optimization can be applied. TSP states that a *salesman has to go on a tour around several cities. Each city has to be visited only once. What is the shortest tour or path he needs to pick? Mathematically it is proven that for n cities the number of possible tour is (n-1)!; For 9 cities the total possible combination is 362,880.* Solution to a problem like TSP is a search problem as it explores many possible tours to find the optimal one. The set of possible solutions for a given problem is known as **search space** (or **state space**). Optimization is the process of finding the best solution and it is vastly used in Engineering and Mathematics domain. The problem specific to domain is first formulated as mathematics model function and finding the best solution requires the parameter that optimizes the system performance. An **optimization problem** thus can be defined as finding values of the variables that minimize or maximize the objective function while satisfying the constraint. The solution to optimization problem is based on mainly 3-significant points:

1. Optimization Function (Which is to be minimized or maximized).
2. A set of variables that affect the Optimization function.
3. A set of Constrained which must be satisfied.

Genetic Algorithms are one such evolutionary optimization process that takes potentially huge search spaces and parallel processing them, looking for optimal combination of things, the solution one might not find otherwise. GA offer significant benefit over many other typical search optimization techniques like – linear programming, Heuristic, DFS, BFS, Simulated annealing, Hill climbing.

Biological Background:

There are certain set of rules that describe how every organisms are built. All living organism consist of cells which are fundamental unit of life. In each cell there is same set of Chromosome. Chromosomes are strings of DNA which serve as model for whole organism. Computationally it can be considered as array of genes. Each Gene encodes a particular character called traits (e.g. colour of eyes etc.).Each gene has its own position in chromosome called Locus. The complete set of genetic material is called Genome .Particular set of genes in a genome is called genotype. The physical expression of genotype is called Phenotype. When two organism mate they share their genes this process is called Crossing over or Recombination. The newly created Offspring may mutate. The mutation means change in a bit of DNA pattern.

Working Methodologies:

The space of all feasible solution among which the desired solution resides is called search space or state space. Each point in the search space represents one possible solution. Each possible solution can be marked by its fitness for the problem. This task is actually done by fitness function that assigns fitness value to the individual and it is problem specific. The GA looks for the best solution among a number of possible solutions represented by one point in the search space. Looking for a solution is then equal to looking for some extreme value (maximum or minimum) in the search space.GA generates other points (possible solutions) as evolution proceeds.

II. Correlating Terminologies with GA:

Population: set of individuals (having same length) which can be tentative solution for the search problem; **Individuals:** chromosomes; **Gene:** chromosome contains the solution in the form of a gene; **Fitness:** The value assigned to an individual based on how far or close an individual is from solution; **Fitness Function:** A problem specific function that assign fitness value to an individual. **Crossing over:** picking two fittest individual from search space intermingling their chromosome to create two new offspring; **Mutation:** A random change in a bit pattern of a gene; **Selection:** Selecting individual for next generation. GA begins with a set of solutions (represented by the chromosome) for a particular problem called the population. Solutions from population are selected for crossing over based on their fitness value to form new solutions. This is repeated until some condition is satisfied.

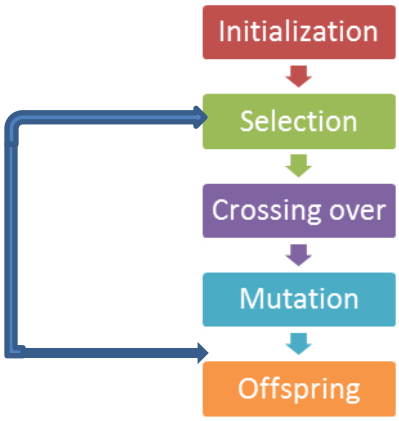
1. Basic Steps:

Start: Generate n-chromosome population (suitable solution for the problem)
Fitness criteria: Evaluate the fitness f(x) of each chromosome x in the population.
Generate population: use the natural evolution process to generate the new individuals:

1. Selection
2. Crossover
3. Mutation
4. Acceptance

Test: if the end condition is satisfied stop and return the best solution in the current population otherwise repeat the process from Fitness criteria.

1.1 Logic Flow:



1.2 Pseudo Code:

BEGIN
Initialize
Evaluate
Repeat until (Condition)

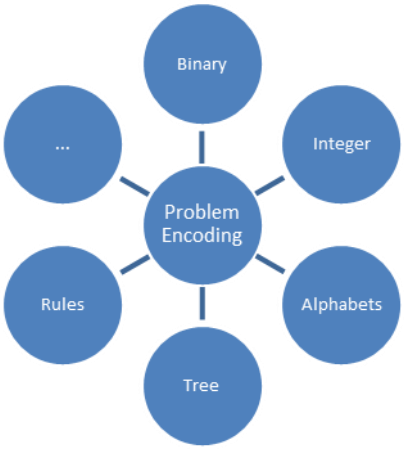
1. Select parents
2. Recombine
3. Mutate
4. Select individuals from next Generation

END

2. Problem Encoding: it concerns with how problems can be formulated. One major challenge in GA is to represent each individual by its genes. The major task is to identify the simplest elements (building blocks) of which a solution may consist, and assign a letter (or a number) to each element, we called it **gene** (each letter/number). A **chromosome** is an array of genes and a gene represent some data. A string of genes can be represented by $(Gene)_i$. and $((Gene)_i)_n$ can represent chromosome of an individual. Binary string is one common approach to encode a solution. In binary form a gene looks like: 11001101 and chromosome representative of an individual looks like:

$Gene_1$	$Gene_2$	$Gene_3$
11001101	10101111	01110011

Each bit in the string represents some characteristics of the solution. A chromosome should in some way contain certain information about solution which it represents. There are many other way of encoding the solution depending upon problem domain (i.e. encoding scheme totally depend on the problem to work on). Some encoding schemes are:

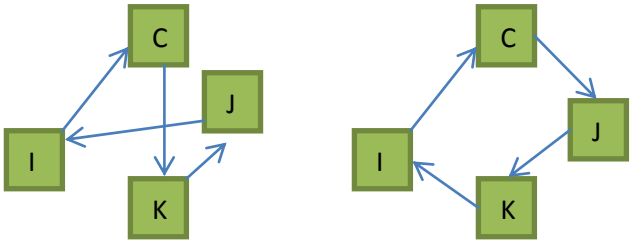


3.Encoding Constraints:
Chromosome should all of the same types (bit string).
Chromosome should be of the same length.
Encoding in some real prospect:

3.1 8-queen's problem: this problem can be encoded as Integer ranging from 1 to 8 where each number represents the position of queen in chess board column. The one possible chromosome configuration might look like: 23451687. Here the first bit represent the position of queen in 2nd column that can have 8-different position ranging from $R_{i=1 \text{ to } 8}C_2$

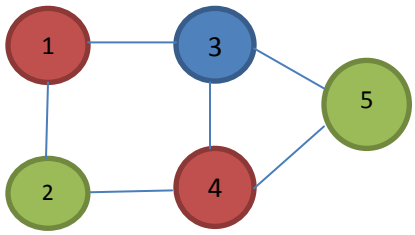
3.2 word guessing problem: here GA has to find the certain word (say "evolution") of fixed length (9 in this case). This problem can be encoded as character ranging from a to z and each chromosome is of length 9.
e.g. evoltunio, evlotkpni.

3.3 TSP (travelling salesman problem): cities can be represented through gene in a chromosome. For instance 4 - countries say India, China, Korea, Japan can be encoded as an individual like:
$Tour_1$=ICKJI
$Tour_2$=ICJK



ICKJI ICJKI

3.4 Graph Coloring Problem: This also known as n-colourability decision problem, it states that whether all nodes of a Graph G can be coloured in such a way that no two adjacent nodes have the same color and only n colours are used. GA can be applied to solve the problem like this. The chromosome string will be of the length equals to the number of node in the graph and each node may take up to n-color. Graph adjacency matrix (n x n) can be used for decision making for fitness evaluation.



in this case length of chromosome will be typically of 5 and in same order as per the graph numbering and each node may accept up to n-colour. For n=3(say R, G, B), the chromosome configuration might look like: RGBRG, RGBGR, GBRGR

4. Crossing Over:
It is one of the most significant features in GA. It is responsible for exchange of genetic material between two parents. It can be single point, double point, uniform.

4.1 Single Point:

P1	P2		O1	O2
100^11011	001^10101	=	10010101	00111011

4.2 Double point:

P1	P2		O1	O2
100^11011^1110	001^10101^1011	=	100101011110	001110111011

4.3 Uniform crossover: uniform crossover means just a random exchange of genes between two parents.

P1: 1001101111110	=	O1: 1000101111111
P2:0010101101101	=	O2: 0011010110100

^: crossing over point, Pi=Parents, Oi=Offspring

5. Mutation: A randomly chosen gene (or several genes) is changed to some other gene (mutate). For example, mutation of genes 3 in the offspring O will give:

O = 1 0 0 0 1 1 1

O_m =1 0 1 0 1 1 1

Mutation helps in avoiding local maximum and it occurs at variable rate.

6. Selection: Once the fitness of whole population in calculated there should be a selection function which tries to pick the fittest individual for reproduction.

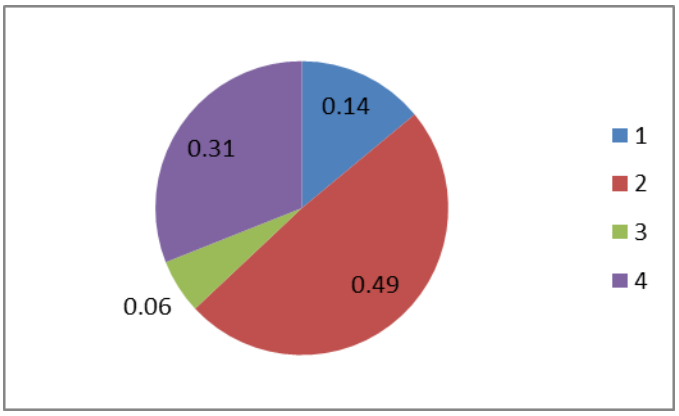
There are several strategies for selection some of them are **Roulette Wheel, Ranked selection.**

Roulette Wheel: In this selection process each member assigned a space in roulette according to their fitness. The member with the highest fitness score has the greatest chances to be picked. This selection techniques works for the GA which maximizes the function.

7. Basic demonstration:

Finding the global maximum of **function 0<x<=31 for x^2** This problem can be best represented as binary encoding with 5 bits pattern.

Individual No	*Encodings(genotype)*	*Phenotypes (X value)*	*F(X)=X²*	*Pselection (fi/∑fi)*
1	0 1 1 0 1	13	169	0.14
2	1 1 0 0 0	24	576	0.49
3	0 1 0 0 0	8	64	0.06
4	1 0 0 1 1	19	361	0.31



Roulette wheel probability selection chart.

∑f(x)=1170
Avg=293 and Max=576

As the probability selection favours highest value, the two fittest individual selected for reproduction are individual 2 and 4 respectively. However a random selection might result into individual say 1, 4, 2, 2.

The next generation table can be formulated as

Selected Individuals	*Crossover Point(random)*	*New Population*	*X*	*F(X)=X²*
1 1 0 0^ 0	4	11001	25	625
1 0 0 1^ 1	4	10010	18	324
1 1^ 0 0 0	2	11101	29	841
0 1 ^1 0 1	2	01000	8	64

∑f(x)=1854
Avg=463 and Max=841

The above result shows that GA always favours the fittest individual and it approaches global maximum.

Ranking methods: choose individuals according to fitness rank.

Tournament selection: select best among a randomly selected subset.

III. Complete Illustration (Knapsack Problem):

In this problem we have to fill the knapsack of capacity W with a given set of items **$I_1,I_2,I_3…In$** having weight **$W_1,W_2,W_3…W_n$** in such a manner that the total weight of the items should not exceed the knapsack capacity and maximum possible value can be obtained.

1. Simplified overview:

There is a Knapsack with maximum weight capacity of 20 kg. There are 7 items that one can carry in knapsack. Each item has certain benefits and weight attach to it and you can carry only one of each item. Your objective is to maximize points seeing in mind the weight constraint with objective of scoring maximum points.

Items	Benefits	Weight
A	10	1
B	20	5
C	15	10
D	2	1
E	30	7
F	10	5
G	30	1

2. Problem encoding:

This problem can be best encoded as binary bit patterns where each bit represents the presence or absence of item in knapsack, for instance the one possible configuration can be represented as 1100111 where each bit represent whether to carry or not to carry item in knapsack and here the total no. of bit is equals to the total no. of item.

I.e. Encoding: 0 = not exist, 1 = exist in the Knapsack

Item	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Configuration (Chromosome)	1	1	0	0	0	1	1
Exist (in knapsack)?	Y	Y	N	N	N	Y	Y

3. Population Generation: Generate random 7 bit population of n-chromosome.

1) 1010101
2) 1110001
3) 1001110

4. Fitness evaluation:

Item	A	B	C	D	E	F	G
Chromosome (1)	1	0	1	0	1	0	1
Points	10		15		30		30
Weight	1		10		7		1

∑Weights=19, ∑points=85

Item	A	B	C	D	E	F	G
Chromosome (2)	1	1	1	0	0	0	1
Points	10	20	15				30
Weight	1	5	10				1

∑Weights=17, ∑points=75

Item	A	B	C	D	E	F	G
Chromosome (3)	1	0	0	1	1	1	1
Points	10			2	30	10	30
Weight	1			1	7	5	1

∑Weights=14, ∑points=82

5. Crossing over:

The two fittest individual selected for crossing over are:

Selected Individuals	Crossover Point(random)	New Population	∑Point	∑Wt
101^0101	3	1011111	97	15
100^1111	3	1000101	70	9

The new offspring generated (1011111) has highest fitness value. From one generation to next GA tends towards optimal solution by eliminating the weaker one and fittest always have the chance to reproduce.

In GA fitness function and Selection process eliminates the bad solution from the good one. Offspring inherit the properties of good solutions are retain. The good individual may survive for many generations (elitism) and the good part of chromosome may be kept intact known as schema.

6. R implementation [25]:

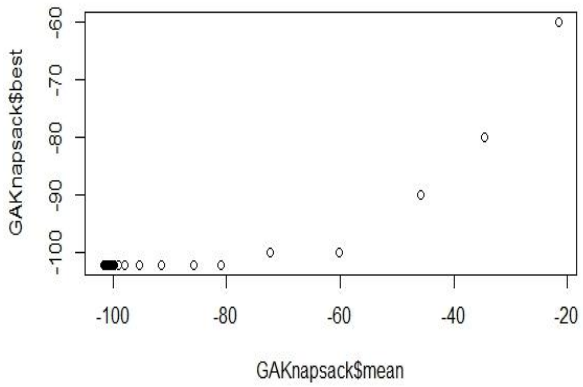
Library used: genalg Editor: RStudio

Function use for demonstration **rbga.bin** R based genetic algorithm for binary configuration. The major parameter requires are:

Size	the number of genes in the chromosome.
PopSize	the population size.
Iters	the number of iterations.
mutationChance	the chance that a gene in the chromosome mutates. By default 1/(size+1).
Elitism	the number of chromosomes that are kept into the next generation. By default is about 20% of the population size.

Assigning basic items, benefits and weight constrain in Knapsack	**ItemsKnapsack=c("A","B","C","D","E","F","G")** **ItemsBenefit=c(10,20,15,2,30,10,30)** **ItemsWeight=c(1,5,10,1,7,5,1)** **WeightConstrain=20**
Frame generation using dataset .frame and assigning it to variable dataset.	**dataset=data.frame(ItemsKnapsack,ItemsBenefit,ItemsWeight)**
Calculating chromosome(1110011) benefits	**chromosome=1110011** **cat(chromosome %*% dataset$ItemsBenefit)**
Fitness function declaration	**FitnessEval = function(x)** **{** **Chromosome_ItemBenifits = x %*% dataset$ItemsBenefit** **Chromosome_ItemsWeight = x %*% dataset$ItemsWeight** **if (Chromosome_ItemsWeight > WeightConstrain)** **return(0)** **else** **return(-Chromosome_ItemBenifits)** **}**
Iteration parameter	Iter=100
Assigning arguments to main function of GA	**GAKnapsack = rbga.bin(size = 7, popSize = 200, iters = Iter, mutationChance = 0.01, elitism = T, evalFunc = FitnessEval)**
Checking statistics of model	**cat(summary.rbga(GAKnapsack))**
Checking solution	**cat(summary(GAKnapsack))**
Output	**GA Settings** **Type = binary chromosome** **Population size = 200** **Number of Generations = 100** **Elitism = TRUE** **Mutation Chance = 0.01** **GA Results** **Best Solution : 1 1 0 1 1 1 1** **Benefits=102**

If Knapsack weight is constraint to 18 the GAKnapsack model predict the best solution : 1 1 0 1 1 0 1 with benefits point 92.



IV. Application of GA:

Some important areas where it applied significantly are [13],[24]:

Conformational analysis: in medicinal chemistry; **Docking:** in drug designing; **Scheduling:** Facility, Production, Job, and Transportation Scheduling; **Design:** Circuit board layout, Communication Network design; **Machine Learning:** Designing Neural Networks, Classifier Systems, Learning rules; **Robotics:** Trajectory Planning, Path planning; **Combinatorial Optimization:** TSP, Bin Packing, Set Covering, Graph Bisection, Routing; **Image Processing:** Pattern recognition; **Business:** Economic Forecasting; **Medical:** Studying health risks for a population exposed to toxins; **Bioinformatics**: MSA[20], Gene finding, Motif discovery; **Clustering[19]**: using genetic algorithms to optimize a wide range of different fit-functions.

V. Conclusion:

Genetic algorithms provide a very strong and effective means to model population genetics for different problem domains. It provides a flexible platform influenced from naturals evolution to model various diversified problems in an effective manner. It is one of the highly growing areas in the area of artificial intelligence and lots of research is going on to exploit its potential to minimize the huge search space. GA can potentially be applied with other optimization techniques or heuristics to minimize the computation task once candidate with strong fitness scores are identified.

This paper provides very basic and simple concept working behind the principle of Genetic algorithm. Researchers have applied the potential of GA to solve several computationally challenging issues like Drug designing, Image processing and several other domain mentions in application section. This paper explains a simple review of GA using many scholastic articles, books and open contents over internet with basic demonstration using R.

References:

[1] Practical genetic algorithms Randy L. Haupt Sue Ellen Haupt A
[2] Genetic algorithm Dr. Roman V Belavkin BIS3226.
[3] Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA, 1989.
[4] Koza, J. R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge, MA, 1992.
[5] Devillers J Ed. Genetic algorithms in molecular modeling. Academic Press: New York, 1996.
[6] Kenneth A. De Jong Evolutionary Computation A Unified Approach.
[7] Banzhaf, W., P. Nordin, R. E. Keller, and F. D. Francone (1998). Genetic Programming –An Introduction; On the Automatic Evolution of Computer Programs and its Applications. San Mateo, CA: Morgan Kaufmann.
[8] Vose, M. (1995). Modeling simple genetic algorithms. Evolutionary Computation 3(4), 453–472.Whitley, T. Starkweather, and C. Bogart, "Genetic Algorithm and Neural Networks: Optimizing Connections and Connectivity", Parallel Computing.
[9] Back, T. and H.-P. Schwefel (1993). An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation
[10] Michalewicz, Z. (1994). Genetic Algorithms + Data Structures = Evolution Programs. New York: Springer-Verlag.
[11] Fundamentals of Genetic Algorithm by RC chakraborty.
[12] A Field Guide to Genetic Programming Riccardo Poli Department of Computing and Electronic Systems University of Essex – UK , William B. Langdon Departments of Biological and Mathematical Sciences University of Essex – UK Nicholas F. McPhee Division of Science and Mathematics University of Minnesota, Morris USA.
[13] Introduction to Genetic Algorithms Theory and Applications The Seventh Oklahoma Symposium on Artificial Intelligence November 19, 1993.
[14] Foundations of Genetic Algorithms, Gregory Rawlins, Editor, Morgan Kaufmann, 1991.Whitley, T. Starkweather, and C. Bogart, "Genetic Algorithm and Neural Networks: Optimizing Connections and Connectivity", Parallel Computing.
[15] R Based Genetic Algorithm Egon Willighagen.
[16] Radenbaugh, A.J. (2005). Applications of genetic algorithms in computing architectures. San Jose State University, Department of Computer Science, Course 247: Computer Architecture, Dr. Chun.
[17] Cramer, N. L. "A Representation for the Adaptive Generation of Simple Sequential Programs." In Proceedings, International Conference on Genetic Algorithms and their Applications, July 1985 [CMU]. pp. 183-187.
[18] A Field Guide to Genetic Programming by Poli, Langdon, McPhee
[19] Auffarth, B. (2010). Clustering by a Genetic Algorithm with Biased Mutation Operator. WCCI CEC. IEEE, July 18–23, 2010.
[20] Gondro C, Kinghorn BP (2007). "A simple genetic algorithm for multiple sequence alignment". Genetics and Molecular Research **6** (4): 964–982.
[21] Willett P (1995). "Genetic algorithms in molecular recognition and design". Trends in Biotechnology **13** (12): 516–521.
[22] Wong, K., Leung, K., and Wong, M. 2010. Protein structure prediction on a lattice model via multimodal optimization techniques. GECCO 2010: 155-162.
[23] Genetic algorithm theory and application by Ulrich Bodenhofer.
[24] http://en.wikipedia.org/wiki/Genetic_algorithm.
[25] R Bloggers.