ISSN No:-2456-2165

An Overview of Genetic Algorithms: A Structural Analysis

Satish Kumar Gupta
Assistant Professor, Department of Information Technology,
Gopal Narayan Singh University, Jamuhar Sasaram (Rohtas), Bihar-821305

Abstract:- The Genetic Algorithm (GA) is a genetic and natural selection-based search-based optimization technique. It is often employed in the search for optimal or near-optimal solutions to complex problems that could take a lifetime to solve. It's widely used in science and machine learning to solve optimization problems. In computer science and operations research, a genetic algorithm (GA) is a metaheuristic influenced by natural selection that belongs to the broader class of evolutionary algorithms. The aim of this article is to look about the Genetic Algorithm, Basic Terminologies and the it's various applications in different fields. This paper also focuses on explaining the various types of representation of Genetic Algorithm and various Merits and Demerits of Genetic Algorithm (GA).

Keywords:- *Algorithm, Operator, Terminology, Presentation.*

I. HISTORY

A. Rechenberg introduced evolutionary computing in his work "Evolution strategies" in the 1960s (Evolutions strategies in original). Other researchers expanded on his definition. John Holland invented Genetic Algorithms (GAs), which he and his students and colleagues created.
B. John Koza used a genetic algorithm to evolve algorithms to perform specific tasks in 1992. His system was dubbed "genetic programming" by him (GP). Since LISP programs can be represented in a "parse tree," which is the object the GA deals with, they were used.

II. INTRODUCTION

The Genetic Algorithm (GA) is genetics and natural selection-based search-based optimization technique. It has often been employed to quest for ideal or near-optimal solutions to complex problems that will take an eternity to solve otherwise It's a popular tool for optimization, analysis, and machine learning.

Nature has always been a source of great inspiration for all of humanity. GAs is search-oriented algorithms that are based on natural selection and genetics concepts. GAs is a subset of Evolutionary Computation, which is a much larger computing branch.

GAs was invented at the University of Michigan. The algorithms were developed by John Holland and his students and colleagues, most notably David E. Goldberg, and have since been successfully evaluated on a variety of optimization problems.

In GAs, we have a population or reservoir of potential solutions to a crisis. Recombination and mutation are then applied to these solutions (as in natural genetics), resulting in the birth of For decades, the technique has been used. A fitness value is given to each individual (or candidate solution), (based on the value of its objective function), and the fitter it is, the better ones have a better chance of mating and producing more "fitter" individuals. This is consistent with Darwin's "Survival of the Fittest" theory.

In this manner, we continue to "evolve" Before we reach a point where we must quit, we must develop great people or alternatives over years.

Despite being sufficiently random, genetic algorithms outperform random local quest (in which we simply pursue various random solutions) and keep track of the best so far) since they also use historical data.

Genetic Algorithm:

"Genetic Algorithms are good at exploring big, potentially enormous search spaces, searching for ideal combinations of items that you may not find in a lifetime."

Basic Terminology:

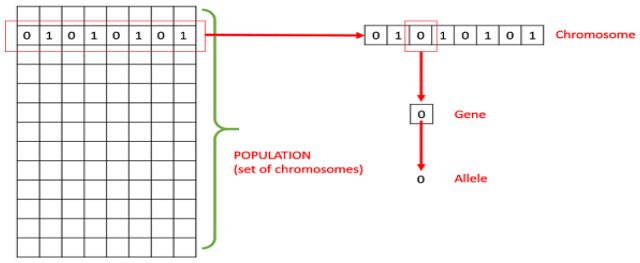
Before diving into the topic of Genetic Algorithms, it's important to understand some of the vocabulary that will be used in this tutorial.

- **Population**: The algorithm begins with a population of solutions (represented by chromosomes). A population's Solutions are sought and implemented in order to establish a new population. This is driven by a desire for the new population to be superior to the existing. Solutions chosen to form new solutions (offspring) are chosen based on their fitness; the better they are, the more likely they are to replicate.
- **Chromosomes**: Cells are the building blocks of all living organisms. The chromosomes in each cell are identical. Chromosomes are DNA strings that function as a blueprint for the entire organism. A chromosome is made up of chromosomes, which are DNA blocks. Each gene codes for a specific protein. Essentially, each gene codes for a characteristic, such as the color of one's eyes. Alleles are the variouscharacter's settings (for example, blue and brown). Each gene is found on a different chromosome.

ISSN No:-2456-2165

This is a challenging job opportunity is referred to as locus.

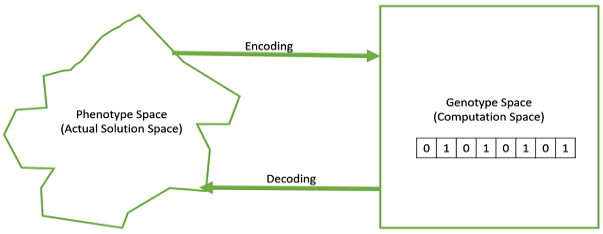
- **Gene**: Potential theories in a GA are made up of chromosomes made up of genes in turn. In a GA, chromosomes are usually represented as binary strings, a series of 1s and 0s that denote the inclusion or exclusion of specific items based on their position in the string. Within such a chromosome, a gene is a single piece.
- **Allele**: It is the weight a gene has on a specific chromosome.



- **Genotype**: In the computation space, the population is known as genotype. The solutions are So that a computing machine can efficiently interpret and control them, they are represented in the computation space.
- **Phenotype**: The community in the actual real-world solution space, where solutions are described exactly as they are in the model the actual world conditions, is referred to as phenotype.
- **Parent**: We will have a population of candidate solutions at any point in the algorithm. Some candidate solutions will be chosen in each iteration, or generation, of the algorithm to survive into the next generation and produce some children.
- **Children:** Selected parents from the previous generation have created new candidate solutions.
- **Decoding and Encoding**: Basic problems have the same phenotype and genotype spaces. The phenotype and genotype spaces, on the other hand, are rarely the same. Encoding is transforming a solution from genotype to phenotype space, while decoding is transforming a solution from genotype to phenotype space, the transformation from the space of phenotype to the space of genotype since encoding occurs several times in a GA during the computation of fitness values, it should be fast.

Consider the 0/1 Knapsack Problem, as an illustration The Phenotype space is made up of only the component numbers of the objects to be chosen.

It can be represented as an n-byte binary string in genotype space, however (where n is the number of items). The next object is selected when a 0 appears at position x, while a 1 indicates the opposite. In this case, the genotype and phenotype spaces are not the same.

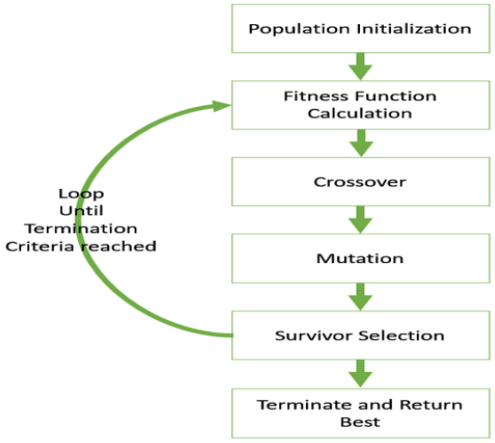


- **Fitness Function:** A fitness function takes a solution as an input and outputs its suitability. The health and objective functions can overlap in some cases are the same; however, depending on the problem, they may be different.
- **Genetic Operators:** Genetic Operators affect the offspring's genetic structure. This includes things like crossbreeding, mutation, and selection.

III. BASIC STRUCTURE

A GA's basic structure is as follows:

We begin by selecting parents for mating from an initial population (which may be produced randomly or seeded by other heuristics). To create new offspring, on the kin, use crossover and mutation operators. Finally, these offspring replace the new individuals in the population, and the cycle begins again. Genetic algorithms aim to approximate human evolution in this manner to some extent.



How they Work:

They're primarily interested in the natural selection process, which means they extend natural selection's basic properties to whatever issue we're attempting to solve.

A genetic algorithm basic procedure is as follows:

1. **Initialization:** Make a starting population. This population is typically produced at random and can range in size from a few individuals to thousands.
2. **Evaluation:** After that, we test. Each person in the population is given a "fitness" ranking. The importance of exercise is measured by how well it does our ideal criteria. These specifications may be straightforward, such as "faster algorithms are better," or more complicated, such

ISSN No:-2456-2165

as "Sturdier materials are preferable, but they should not be too so."

3. **Selection:** We want to continue to improve our population's physical health. We can achieve this with the help of filtering, which removes the poor designs from the population and leaves only the best individuals. There are many selection approaches, but the core idea remains the same: make it more likely that fitter individuals will be selected for our next generation.
4. **Crossover:** We merge components of our selected individuals to create new individuals during the crossover. This is comparable to how sex behaves in nature. The hope is that by combining these traits from two or more people, we will be able to create a more 'fit' offspring who will inherit the best traits from both parents.
5. **Mutation:** We ought to inject some randomness into the genetics of our populations, or else any possible answer will end up in our initial population. Mutation usually works by making minor changes to a person's genome at random.
6. **And repeat:** Now that we have our next generation, we can go back to phase two and repeat the process before reaching a state of termination.
7. **Termination:**

You may want to halt the genetic algorithms search for a solution for a variety of reasons. The possible explanation is that algorithm has discovered a satisfactory solution and satisfies a set of minimum requirements. Constraints such as time or money may be offered as grounds for terminating.

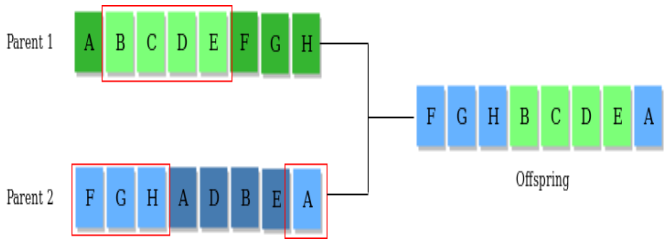
Operators of Genetic Algorithms:

Following the development of the initial generation, the algorithm evolves the generation using the following operators –

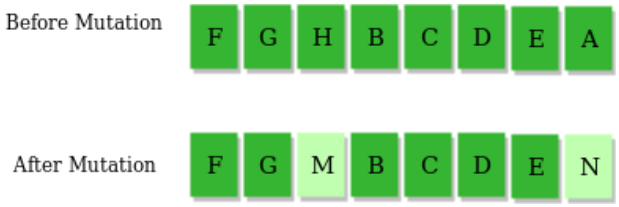
1) Selection Operator: The goal is to give people who have poor health scores a leg up to inspire them to keep going pass on their genes to future generations.

2) Crossover Operator: This describes person mating. The crossover sites are chosen at random, and two individuals are chosen using the selection operator. The genes at these crossover sites are then swapped, resulting in creating an entirely new body (offspring).

For instance-



3) Mutation Operator: The central idea is to insert random genes into embryos to maintain population variability and avoid premature fusion. For instance –



Genotype Representation:

The representation we will use to represent our solutions when applying a genetic algorithm is one of the most crucial choices to make. It has been discovered that poor representation can result in poor GA results.

As a result, picking the suitable representation and defining the mappings between the phenotype and genotype spaces is critical to a GA's success.

Binary Representation:

In GAs, this is one of the most basic and commonly used representations. The genotype in this form of representation is made up of bit strings.

The binary representation is normal for complex problems where the solution space consists of Boolean decision variables – yes or no. Think about the 0/1 Knapsack Problem. If there are n products, a solution can be expressed as a binary string of n elements, with the xth element indicating whether item x is selected (1) or not (0).

0	0	1	0	1	1	1	0	0	1

The binary representation of numbers can be used to describe them in other problems, especially those involving numbers. Since various bits have different definitions in this form of encoding, mutation and crossover operators can be ineffective unexpected results. Gray coding can help with this to some degree, as a change of one bit has a minor effect on the solution.

Real-Valued Representation:

The real-valued representation is the most natural when Rather than discrete variables, we'd like to describe genes using continuous variables. These genuine or floating-point numbers, however, are limited by the computer's precision.

0.5	0.2	0.6	0.8	0.7	0.4	0.3	0.2	0.1	0.9

Integer Representation:

We can't limit the solution space for discrete-valued genes to binary "yes" or "no" forever. E.g., the four distances can be encoded as 0,1,2,3 if we want to encode North, South, East, and West. In such instances, it is preferable to use integer representation.

1	2	3	4	3	2	4	1	2	1

ISSN No:-2456-2165

Permutation Representation:

The solution to an order of elements is used to express several problems. Permutation representation is the best choice in these contexts.

The travelling salesman problem is a well-known example of this kind of representation (TSP). Before returning to the starting city, the seller must tour all cities, visiting each one exactly once.The tour's overall distance must be kept to a minimum. Since the TSP's solution is a standard ordering or permutation of all the towns, it makes sense to use a permutation representation for this issue.

1	5	9	8	7	4	2	3	6	0

IV. POPULATION

The population is now a subset of solutions in the modern era. It is also known as a chromosome set. There are a couple of things to keep in mind when dealing with GA populations:

• It is critical to preserve demographic diversity, as premature convergence will result.

• A large population can cause a GA to slow down, and a small population cannot have a sufficient mating pool. As a result, trial and error must be used to decide the best population scale. A population is typically described using a two-dimensional array of population size, size x, and chromosome size.

Population Initialization:

In a GA, there are two critical methods for initializing a population.

• **Random Initialization:** To populate the initial population, completely random solutions are used.

• **Heuristic initialization:** Using a known heuristic for the dilemma, populate the initial population.

It has been discovered that using a heuristic to initialize the entire population results in a population with identical solutions and no diversity. Random solutions have been discovered to be the ones that move the population to optimality. As a consequence, rather than flooding the population with heuristic-based solutions, we seed it with a diverse set of solutions and fill in the gaps with random solutions using heuristic initialization.

It was also discovered that heuristic initialization only influences the population's initial fitness in certain situations; Optimality is aided by the variety of options.

Population Models:

The steady-state and fluctuating demographic models are the two most popular population models.

Steady-State:

Each version of steady-state GA produces one or two descendants, each replacing one or two individuals in the population. Stable condition GA is also known as incremental GA.

Fluctuating Population:

In a generational model, we produce "n" heirs, where n is the population size, and after each cycle, the whole population is replaced.

Application of the Genetic Algorithm:

Genetic algorithms are often used in some applications, but they are most widely used in optimization problems of various kinds.

- **Genetic Algorithms for Optimization:** When solving optimization problems, we must optimise or minimise the value of an objective function, genetic algorithms are most commonly used while adhering to a set of constraints. Throughout the tutorial, the approach to solving Optimization problems has been illustrated.
- **In Economics:** Models like the cobweb model, game theory equilibrium resolution, and asset pricing are all described using Gas, and so on.
- **Recurrent Neural Networks:** GAs is also used to train recurrent neural networks.
- **Image Processing:** GAs is also used for dense pixel matching in digital image processing (DIP).
- **Vehicle Routing Problems** − with a diverse fleet, numerous soft time windows, and multiple depots
- GAs is also used to solve various **Scheduling Problems**, especially the timetabling problem, in scheduling applications.
- **Machine Learning GBML (Genetics-Based Machine Learning)** is a niche field of machine learning, as previously stated.
- **Robot Trajectory Generation:** GAs is used to plan a robot arm's path from one point to the next.
- **Parametric Design of Aircraft:** By varying criteria and findings, GAs has been used to manufacture aircraft.
- **Travelling Salesman Problem and its Solutions:** Using novel crossover and packaging techniques, the TSP, a well-known combinatorial puzzle, has been solved using GAs.

Advantages of Genetic Algorithm:

GAs has several advantages that have helped them become highly successful.

a. It is impractical to use derivative knowledge (which might not be usable for certain real-world problems).
b. As opposed to conventional methods, it is quicker and more effective.
c. Optimizes both continuous and discrete functions, as well as multi-objective problems, thanks to its solid, parallel capabilities.
d. Rather than providing a single "good" alternative, it provides a list of "good" choices.
e. Gets a solution to the problem often and it gets better over time.
f. Useful when there are several parameters to consider and the search space is enormous.

V. LIMITATIONS OF GENETIC ALGORITHM

While genetic algorithms have proven to be a quick and efficient problem-solving method, they do have some disadvantages. Any of these flaws will be discussed later.

1. Determining a representation for the problem is the first and most critical factor in developing a genetic algorithm. The terminology used to specify candidate solutions must be robust because it must be able to withstand random changes without causing fatal errors.
2. Coding the fitness (evaluation) function to achieve higher fitness and provide better solutions to the problem at hand is one of the most challenging genetic algorithms. A poor fitness function choice can result in serious problems, such as being unable to solve a problem or, even worse, returning the incorrect solution to a problem.
3. In addition to choosing a good fitness function, a Genetic Algorithm's other parameters, such as population size, mutation rate, and crossover rate, must be carefully chosen. The Genetic Algorithm would not be able to achieve accurate results with a small population size because there would not be enough solution space. A high rate of genetic change, or a faulty selection system, will destroy the beneficial schema, and the population will hit error catastrophe, evolving too rapidly for selection ever to achieve convergence.
4. Computational problems cannot be solved with genetic algorithms. Although genetic algorithms can find accurate solutions to these problems, traditional analytic methods can faster and with fewer computational steps.
5. When developing Generic algorithm solutions, GA scientists must also consider premature convergence. An individual could be physically fitter than any of its competitors. As a result, rather than searching the fitness landscape extensively enough to find the global optimum, this individual may replicate many more new individuals, reducing the diversity of the new population and causing the algorithm to converge on the local optimum that represents that specific individual. This type of inefficiency is most common in small problems, where even minor differences in reproduction rate can lead to one genotype dominating the others.

VI. CONCLUSIONS

The richness of genetic algorithms and the various parallel models is one of the most striking features of this kind of computing. Minor changes to the algorithm will often result in unpredictable emergent behaviour. Our understanding of genetic algorithms has also improved due to recent scientific advances, allowing us to use more sophisticated analytical methods.

After study of this article, readers are well known about the how many Terminologies are used for GA, How the GA Model working and what are the Advantages and Disadvantages of GA. This Article also addresses a wide range of current issues.

REFERENCES

[1]. Azadivar, F. and Tompkins, G., Simulation optimization with qualitative variables and structural model changes: a genetic algorithm approach. *Euro. J. Opt. Res.*, 1999, **113**(1), 169–183.
[2]. Chan, K.C., and Tansri, H., A study of genetic crossover operations on the facilities layout problem. *Computers & Ind. Eng.*, 1994, **26**(3), 537–550.
[3]. Chen, C.L., Vempati, V.S. and Aljaber, N., An application of genetic algorithms for flow shop problems. *Euro. J. Opt. Res.*, 1995, **80**(2), 389–397.
[4]. Meyer, L., Feng, X.: A fuzzy stop criterion for genetic algorithms using performance estimation. In: Proceedings of the Third IEEE Conference on Fuzzy Systems. (1994) 1990–1995
[5]. Howell, M., Gordon, T., Brandao, F.: Genetic learning automata for function optimization. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 32 (2002) 804–815
[6]. Poli, R.: Exact schema theorem and effective fitness for GP with one-point crossover. In Whitley, L.D., Goldberg, D.E., Cant´u-Paz, E., Spector, L., Parmee, I.C., Beyer, H.G., eds.: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann (2000) 469–476
[7]. De Jong, K.A.: Genetic algorithms are NOT function optimizers. In Whitley, L.D., ed.: Proceedings of the Second Workshop on Foundations of Genetic Algorithms, Morgan Kaufmann (1993) 5–17
[8]. B¨ack, T., Hammel, U., Schwefel, H.P.: Evolutionary computation: Comments on the history and current state. IEEE Transactions on Evolutionary Computation 1 (1997) 3–17
[9]. Burkowski FJ (1999) Shuffle crossover and mutual information. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 1999, pp. 1574–1580
[10]. Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Systems 9:115–148

Websites:

[1]. https://www.tutorialspoint.com/genetic_algorithms/index.htm
[2]. Darrell Whitley, Computer Science Department Colorado State Universit, Fort Collins CO
[3]. https://www.cs.jhu.edu/~ayuille/courses/Stat202C-Spring10/ga_tutorial.pdf
[4]. https://www.researchgate.net/post/How_can_I_decide_the_stopping_criteria_in_Genetic_Algorithm
[5]. https://www.cs.rochester.edu/u/nelson/courses/csc_173/genetic-algs/algorithm.html
[6]. https://www.sciencedirect.com/topics/engineering/genetic-algorithm
[7]. https://www.geeksforgeeks.org/genetic-algorithms/
[8]. https://ocw.mit.edu/courses/institute-for-data-systems-and-society/ids-338j-multidisciplinary-system-design-optimization-spring-2010/lecture-notes/MITESD_77S10_lec11.pdf
[9]. http://pages.cs.wisc.edu/~dyer/cs540/notes/ga.html