International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 04 Issue: 09 | Sept -2020 ISSN: 2582-3930

Efficiency and Optimization Techniques of Genetic Algorithms and its applications in game automation, learning, and artificial intelligence

Archit Garg[1], Anandita Sharma[2]

1Information Technology, ABES Engineering College
2Information Technology, ABES Engineering College

---***---

Abstract –Since the beginning of programming and algorithm, complex logic designing and problem-solving have been an inseparable part of the research domain. This paper serves the purpose of introducing a more conceptual and practical approach to problem solving and automation. Artificial intelligence or the ability of machines to learn has changed the way of approaching any difficult problem in the current era. By observing the natural biological evolution, a theory given by Charles Darwin, programmers came up with a practical solution of solving complex or mathematical problems by letting the machine learn instead of a much straightforward approach which is a program or hard-code.This paper is written to providean introduction to the efficiency and optimization techniques as well as the application of genetic algorithms in gaming automation or automation in general.

Key Words:Genetic Algorithm, Game Automation, Artificial Intelligence, optimizing GA

1.INTRODUCTION

This paper intends to provide AI developers and automation programmers a starting point with an introduction to a genetic algorithm and its application in game automation by learning. Genetic Algorithms are heuristic solution-search or optimization techniques which do not require a set of instructions, rather learn as the algorithm progresses. GA follows the concept of the Darwinian principle of natural biological evolution and starts with any random chromosome binary sequence or a more formally initial population of chromosomes. Apart from being completely random, the initial population also represents a solution to the problem. Since the initial population is a solution to the problem, we must optimize it. The theory of evolution states that biological beings tend to evolve according to the environmental needs. Genetic algorithms perform similarly. The initial population is given a fitness number which shows how close it is to the solution. Genetic algorithms work best when the solution state is known or approximation can be made. Therefore, GA is widely applicable for automation. Genetic algorithm techniques are widely used in optimization problems such as Ant colony optimization and particle swarm optimization.

2.GA Components

A genetic algorithm is made up of many distinct components. Since these components can be re-used in different genetic algorithms, ease the implementation complexities of the solution. These components are briefly explained.

1. Chromosome Encoding

Chromosomes are string encoded solutions to a particular problem. There are many ways in which chromosomes can be represented.

1. Binary Encoding
2. Permutation Encoding
3. Value Encoding
4. Tree Encoding

These encoding techniques can be used to solve different problems. Binary encoding can represent a selection of an item hence can be used in solving knapsack. Permutation encoding shows different permutations therefore, it can be used to solve the traveling salesman problem. Value encoding can be used in finding the weights for neural network and tree encoding can be used for genetic programming.

2. Fitness Function

Fitness function is a measurement of how good a candidate is or how close a chromosome is to a solution. Since fitness function is evaluated for every chromosome, it should be fast to compute but also cover the wide variety of solutions. Fitness function can take the form of very simple mathematical functions to very complex ones. For example, in multidrug cancer chemotherapy fitness function uses differential equations.

3. Selection

Selection is the process of selection genomes or chromosomes for further breeding of genes. Therefore, it is very important to make a selection of fit chromosomes. The selection process affects the overall performance and time taken by the algorithm.
There are many selection methods.

1. Roulette Wheel Selection
2. Rank Selection
3. Steady-State Selection
4. Tournament Selection
5. Elitism Selection

Chromosome which has higher fitness value will be selected. This process keeps on repeating until we reach the solution.

4. **Operators**

Crossover, sometimes also referred to as recombination is a genetic operator to produce a new generation. It is analogous to its biological counterpart. Two or more chromosomes produce new chromosomes, formally former are called parents, and the latter are called offspring. Crossover of two chromosomes of higher fitness value has a higher probability to produce more fit offspring. Crossover can be a single point or multiple point crossover.
Another type of operator is Mutation. In mutation, genes are changed to get a completely random chromosome.

5. **Evolution**

The process of selection and crossover or mutation is repeated over the initial population to produce new offspring, which are closer to solution space. This new generation is then replacing the initial generation. This process of evolution just like its biological counterpart stops after a certain number of generations are produced or the solution is achieved. Some chromosomes which have high fitness value may be allowed to pass in the following generations. The most widely used scheme of evolution is replacement-with-elitism. In this scheme, the best one or two chromosomes are included in the following generations so that possible solutions may not get lost.

6. **Designing Steps**

To design a genetic algorithm, we must consider many steps because these may affect the performance and efficiency of the algorithm.

1. Random initial population (The more random the better)
2. Select a fit fitness function. (Able to identify solutions)
3. Repeat the following steps.
   a. Apply the fitness function to the initial population.
   b. Apply mutation and crossover to generate the successor population.
4. Replace the initial population with the successor population.
5. Check for the terminating criteria.

3.The efficiency of genetic algorithms

Genetic algorithms are having many limitations that hinder the efficiency of the algorithms. We will first look at the most prominent limitations of genetic algorithms and then we will see how we can make our algorithm more efficient.

Limitations

1. The fitness function for complex problems can easily be computationally expensive and their repetitive application is sometimes not even possible.
2. Genetics algorithms tend to solve subproblems rather than problems at large and do not perform well in case of complex problems.
3. Local optima is also a problem in genetic algorithms. Because decisions are made based onthe fitness function, solutions may be biased in one direction.
4. Dynamic data sets create problems because of early convergencetowards solutions.
5. Decision-based problems are not solved effectively.

Efficient approaches

1. Some multi-dimensional problems do not have simple fitness functions. To make it more efficient, we may choose to use an approximate fitness function which is less computationally costly.
2. The fitness function defines where the solution goes. Scaling is the process of reducing the fitness value of a chromosome which is overshadowing others and leading to a local optimum.
3. Triggered hypermutation or random immigrants' techniques are used to increase the efficiency of the genetic algorithm of dynamic data sets.
4. Problems that are well defined may have better solutions.

4. Game Automation

Game theory describes two types of games:

1. Perfect information – Chess, Checkers, Othello
2. Imperfect information – Backgammon, Poker

Perfect information games have well-defined states and optimal choices which may lead a player to victory. Therefore, it is possible to write an exhaustive algorithm for these algorithms but it's not the best approach. Chess has octillion of possible moves in its first ten plays. The Shannon number is an approximation of chess complexity. Therefore, it was believed that it is impossible to build a computer chess game. Genetic algorithms can be used to automate the chess. Genetic algorithms learn as any human does, hence an algorithm can learn chess. In such cases known states also act like hidden states.

Imperfect information games have hidden states hence it is not possible to write a general algorithm for them. Therefore, GA is used for automation purposes.

5. Applications

Genetic algorithms find their use in many fields. The most groundbreaking applications are mentioned.

Feature Selection in ML

Selecting from a list of N features will have 2^N combinations, therefore, it's a combinatorial optimization problem. GA is a stochastic method for function optimization.

Artificial Creativity

Genetic algorithms are being used to replicate the cognitive thinking and creativity of humans. Artificial creativity is a form of artificial intelligence.

NP-Complete Problems

NP-complete problems like Travelling Salesman Problem can be solved using genetic algorithms.

There are many more applications of genetic algorithms.

3. CONCLUSIONS

This paper intended to give and establish the introductory importance of genetic algorithms in the field of programming, game automation, and artificial intelligence. Genetic algorithms are widely used in perfect and imperfect games. Although GA has many advantages, they tend to be slower and are only used because no better solution exists. GA finds extensive use in approximation computation.

ACKNOWLEDGMENT

I would like to thank my professor, Mr. Manish Srivastava, for mentoring and guiding me.

REFERENCES

1. Ling C.X., Sheng V.S. (2011) Cost-Sensitive Learning. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA
2. (Schaeffer 2000; Fürnkranz 2001; Schaeffer and Van den Herik 2002; Lucas and Kendall 2006; Miikkulainenet al. 2006)

3.L. Davis
Handbook of Genetic Algorithms
Van Nostrand Reinhold, New York (1991)
4. M. Mitchell
An Introduction to Genetic Algorithms
MIT Press, Cambridge, MA (1998)