Shreeraj R. Pawar. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 10, Issue 12, (Series-III) December 2020, pp. 25-29

RESEARCH ARTICLE OPEN ACCESS

A Survey on Genetic Algorithms

Shreeraj R. Pawar
**(Department of Computer Engineering, Fr. C. Rodrigues Institute of Technology, Vashi-400703.*

ABSTRACT
This is a summary paper of a simple genetic algorithm. A genetic algorithm is used to find a way out of them maze generated by walls. Here a player is simulated in a population, and then evolved to further as per the fitness of each player. The child with the highest fitness is carried on the next generation and is breed to give a new population. This paper covers the applications of genetic algorithms and how it was used to simulate a player running through a maze of walls trying to reach its goal and few applications of genetic algorithms.
Keywords – Artificial intelligence, Genetic algorithm, generation, fitness

Date of Submission: 08-12-2020 Date of Acceptance: 24-12-2020

I. INTRODUCTION

Evolution is the change in characteristics of biological population over successive generations. These characteristics are the expressions of genes that are passed on from parent to offspring during reproduction. The process of evolution was vastly explained by Sir Charles Darwin in his book " On the Origin of Species" in 1859. Darwin in his theory later known - Darwin' s Theory of Evolution, says that evolution by natural selection is the process by which organisms change over time as a result of changes in heritable physical or behavioral traits. Changes that allow an organism to better adapt to its environment will help it survive and this benefits the future generations. Evolution by natural selection is one of the best substantiated theories in the history of science, supported by evidence from a wide variety of scientific disciplines like geology, genetics and developmental biology.

In computer science, evolutionary computation is a subfield of artificial intelligence and soft computing inspired by biological evolution. In technical terms, they are a family of population-based trial and error problem solvers with stochastic optimization character. Here the initial set of candidates are generated and iteratively updated. Each new generation is produced by removing less desired solutions, and introducing small random changes. In biological terminology, a population of solutions is subjected to natural selection (or artificial selection) and mutation. As a result, the population will gradually evolve to increase in fitness. Evolutionary algorithms form a subset of evolutionary computation which involve techniques implementing mechanisms inspired by biological evolution such as reproduction, mutation, recombination, natural selection and survival of the fittest.

Evolutionary computing techniques mostly involve optimization algorithms which are designed to find, generate or select a heuristic that may provide sufficiently good solution. In a broad sense we are talking about fields like:

- Genetic Algorithms
- Genetic Programming
- Neuroevolution
- Ant Colony Optimization
- Artificial Immune Systems
- Swarm intelligence

II. GENETIC ALGORITHM

In Computer science, a Genetic algorithm (GA) is a randomized search and optimization technique inspired by natural selection that belongs to a larger class of Evolutionary algorithms. The GA was introduced by John Holland in 1975. He was inspired by the concept of Darwinian' s principle of survival of the fittest individuals and natural selection and developed the theory of genetic algorithm. They are categorized as global search heuristics as they give a optimal global solution to a problem. In genetic algorithm, a population of candidate solutions called individuals or phenotypes are evolved towards a better solution. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered. Every genetic algorithm has a few components which are as follows:

A. Generation

Evolution starts with a population of randomly generated individuals and is an iterative process with population of each iteration is called as

Generation. A generation can of any size depending on the computing capabilities of the system the algorithms will be running on. For our Maze runner I created a population of 500 candidates, thus each generation has 500 individuals with completely random set of chromosomes. Before we head further, we need to understand terminologies like genotype and phenotype.

• Genotype: A genotype is an organism' s complete set of heritable genes or genes that can be passed down from parent to offspring. These genes help encode the characteristics that are physically expressed in an organism such as hair color, height, weight, etc.

• Phenotype: A phenotype is the term used in genetics for the composite observable characteristics or traits of an organism. The term covers organism' s morphology or physical features, structure, its developmental processes, its behavior, etc.

Our program has candidates which as essentially dots on a screen. These dots represent an array of randomly generated vectors which tell the dot the direction in which it will move. So, our phenotype is limited to only the shape, size and color of the dot, whereas the genotype consists of 3 arrays of long set of directional vectors which signify position, velocity and acceleration.

B. Fitness Function

A fitness function is a particular type of objective function that is used. A figure of merit is a quantity used to characterized the performance of a device, system or method relative to its alternatives. The fitness function tells the algorithm how close a particular individual is to achieving the desired outcome. Fitness functions are used in genetic programming and genetic algorithms to guide simulations towards optimal design solutions.

When a population is created it contains individuals or candidates with completely random set of genes. The population is simulated to do a task in our case traverse the maze without dying and reach the goal. The dots traverse the maze with directions corresponding to the vectors generated during initialization. Fitness is given by the inverse of the square of the distance from its goal. It is as follows:

fitness=1/(distanceToGoal * distanceToGoal)

The fitness function changes when a candidate or individual of a generation reaches the goal. The fitness function is then as follows:

$$\text{fitness} = \frac{1}{16} + 1000(step * step)$$

Here ' step' is the number of steps taken by an individual before dying. Each step is incremented based on the velocity and acceleration of each dot. Constants like 16 and 1000 act as hyperparameters which were tuned during testing of the fitness function.

C. Selection

Natural selection is a key mechanism in which only the strongest genes are selected and passed on to the next generation. During an individual' s lifespan based on environmental factors some genes get suppressed and some get activated to a large extend. The activated genes are key to survival of a species and are then passed on to the next generation. These new genes give rise to different characteristics or phenotype in the next generation.

Selection is the stage of genetic algorithm in which individual genomes are chosen from a population for later breeding. Selection is done based on the fitness calculated by the fitness function. The top 2 individuals are selected as the best candidates for gene crossover. There are many methods to Selections namely Roulette Wheel Selection, Rank Selection, Steady State Selection, Tournament Selection, Elitism Selection, Boltzmann Selection. For our Maze Runner problem, we used Elitism Selection where the best individual from last generation is carried over to the next one.

D. Crossover and Mutation

In genetic algorithms and evolutionary computation, crossover, also called recombination is a function where genetic information of two or more individuals is used to generate a new offspring. Our Maze runner Dot contains various vectors for position, velocity, and acceleration. The directions array at the initialization consists of random angles from 0 to 2π. After the end of generation 1 that is when all the individuals of the generation die the fitness is calculated for each individual and the top 2 are selected. The best individual is simply carried on to the next generation and the 2nd best is cross-overed with the best one. Here first half of the best individual's genome is taken and combined with the second half of the 2nd best candidate. Thus, the resultant genome is a recombination of the best and 2nd best candidates of the previous generation.

Mutation is the alteration of a genome of an organism. It alters one or more genes thus generation variation within a species. Variations can be subtle or may entirely change the characteristics of an organism from its parent. In genetic algorithms, mutation is introduced to bring about some genetic diversity from one generation to next. The mutation within a generation is governed by a parameter called the rate of mutation or mutation rate. It is the probability of whether a given candidate will

undergo mutation or not. In our Maze runner, mutation occurs when a random generated variable ' n' is less than the mutation rate. If so then all the values in the directions array are added with the random number ' n' . Mutation rate should be very low as we don' t want our new generation to be completely different than this previous as this may lead to diverting from the main goal.

III. APPLICATIONS

Genetic Algorithms can be used to solve many problems which traditional search algorithms would take many hours or days or years to get to the optimal solution. Some application of genetic algorithms are as follows:

1)NEAT Algorithm

NEAT is an acronym for NeuroEvolution of Augmenting Topologies is an algorithm or a method of optimizing weights that are used in neural networks.

Weights in a neural network are randomly initialized and then changed after each epoch. Using a loss function and learning rate the weights are adjusted so as to give correct output for a given input. Genetic algorithm can be used to optimize the weights on between connections and significantly enhance the performance of the NN. Here a population is generated with each candidate having a NN with randomized weights. After a generation the candidates with the maximum fitness or the best performing candidates of a generation are cross overed and mutated. During crossover the weights are recombined and not the input and output. Thus, over generations the weights get optimized such that the NN always reaches the desired output. Mutation in NEAT can change both connection weights and network structures. Connection weights mutate as in any NE system, with each connection either perturbed or not at each generation. New nodes may also get added to the NN and the resultant NN after several generation may look completely different than its initialization. NEAT is a powerful method for artificially evolving neural networks. It demonstrates evolving topology along with weights can be a major advantage.

2)Solving Travelling Salesman Problem

The Travelling Salesman problem (TSP) is one of the old problems in computer science. Given a network with ' n' nodes (cities), from a starting node say ' S' what path should be taken such that all the nodes can be visited with least amount of cost (cost here can be distance or time). The problem is to find the least cost Hamiltonian cycle. The number of ways which gives minimum cost can increases exponentially with more number of cities. Many real-world problems can be modelled after TSP and thus has attracted many researchers and mathematicians. Finding the optimal solution involves evaluating every possible Hamiltonian path, but this is highly inefficient even for a moderate size graph. Since practical applications require solving larger problems, hence emphasis has shifted from the aim of finding exactly optimal solutions to TSP, to the aim of getting, heuristically, ' good solutions' in reasonable time. Genetic algorithm (GA) is one of the best heuristic algorithms that have been used widely to solve the TSP instances. Many crossover operators and mutation operations have been implemented that give a heuristically ' good solution' and traditional search algorithms.

3)Signal Processing

Genetic Algorithms being a powerful optimization tool have a large number of applications in signal processing. Signal processing involves noise cancellation, time-delay estimation, etc. GA are well suited for techniques like Adaptive filtering where filter coefficients are needed to be optimized. Gas can be applied to solve time-delay problems where any modeling of gain factors and delay is not required and direct estimates of both parameters are obtained through the genetic optimization procedure. GA also can be used for Active noise cancellation (ANC). ANC is a technique that uses secondary sound sources to generate sound waves that create destructive interference canceling the undesired sound. GA can be applied to find optimally controlled secondary sound signals such that minimum noise is perceived.

IV. OBSERVATIONS

The entire project was made in a sketching tool called processing3. Processing is a JavaScript based sketching tool used for creative coding. The project consists of a window with a starting point of the player and a goal point. The path between is made up of walls both horizontal and vertical. If the player hits the window boundary or any wall, the player dies. When we start the program, in generation 1 as the vectors are randomly assigned, the players move around randomly and die quicky. After generation 15, the player moves in a particular path and less player die from random movements.

At generation 32 the player always avoids the initial walls and boundaries and reach near the goal. It is until generation 38 that one of the candidates in the population reaches the goal. Now the fitness function changes as described earlier and we seen a large number of candidates reaching the goal. By generation 44, 425 out of 500 reach the goal avoiding all the obstacles. I kept the algorithm running and later found a limitation or a bug within my program which is covered in the next section. It can be seen in the following graph fig1.0 that after generation 55 the number of candidates reaching the

goal does not improve but reduces with increasing generations.

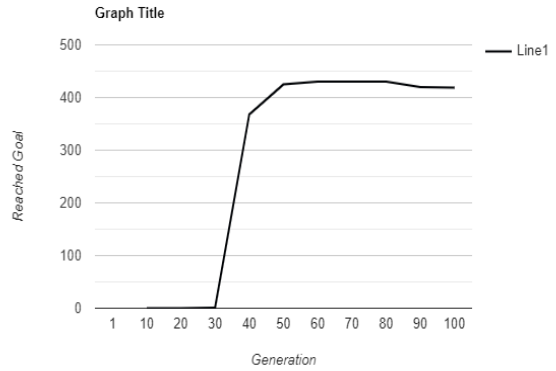

Fig1.0: Line 1 indicates the number of candidates

V. LIMITATIONS

Although evolutionary approach and genetic algorithms seem like a faster alternative to traditional search algorithms, they have their own set of limitations. Due to the randomness of the GA operation, it is difficult to predict its performance and time of operation which are essential factors for hard-deadline real-time applications.

Finding the right fitness function can be challenging at times. It may take hours or days to tweak the fitness functions hyperparameters to find the optimum fitness of each individual. Moreover, if large population exits calculating fitness of each individual can be computationally heavy, so efficient functions needs to be programmed.

Genetic algorithms can be endless as in they will keep on running until we manually stop them. This may lead to converging towards a local optimum rather than a global optimum. During our testing we found that after generation 50 the dots used to take a path which led to the death of many candidates as compared to previous generations. Due to constant mutation after every generation even the best dot who took the shortest path in generation 44 to reach the goal died in the next generation. Since a stopping condition was not added the algorithm kept on running. The global maxima as achieved when a dot takes the shortest path which in a rectangular window is the diagonal. As there are obstacles in the path candidates simply can' t traverse diagonally hence maximum fitness could not have been achieved.

VI. CONCLUSION

Thus, genetic approach can be seen as a viable approach towards an underlying problem. Genetic algorithms can be used for various other applications and due to their versatility, they are used to find optimum solutions of problems across various fields.

Evolutionary computation is ought to improve with advancements in parallel computing and by developing more efficient algorithms.

ACKNOWLEDEMENTS

This paper is based on a project I participated in during our 2nd year of our bachelor' s degree. I would like to thank my professors who encouraged me to write a research paper based on the project. This has been an educational journey and it would not have been possible without the constant support from my mentors.

REFERENCES

[1] S. Forrest. Genetic algorithms: Principles of adaptation applied to computation. Science, 261:872{878, Aug. 13 1993.

[2] Bäck T. Optimal mutation rates in genetic search. InProceedings of the fifth international conference on genetic algorithms 1993.

[3] Janikow, C.Z. A Knowledge-Intensive Genetic Algorithm for Supervised Learning. Machine Learning 13, 189– 228 (1993).

[4] de Jong, K., 1988. Learning with Genetic Algorithms: An Overview. Machine Learning, 3(2), pp.121-138.

[5] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," in Computer, vol. 27, no. 6, pp. 17-26, June 1994, doi: 10.1109/2.294849.

[6] K. S. Tang, K. F. Man, S. Kwong and Q. He, "Genetic algorithms and their applications," in *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 22-37, Nov. 1996, doi: 10.1109/79.543973.

[7] Ahmed, Z.H., 2010. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *International Journal of Biometrics & Bioinformatics (IJBB)*, 3(6), p.96.

[8] Tang, K.S., Man, K.F., Kwong, S. and He, Q., 1996. Genetic algorithms and their applications. *IEEE signal processing magazine*, 13(6), pp.22-37.

[9] Stanley, K.O. and Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), pp.99-127.

[10] Grefenstette, J., Gopal, R., Rosmaita, B. and Van Gucht, D., 1985, July. Genetic algorithms for the traveling salesman problem. *In Proceedings of the first International Conference on Genetic Algorithms and their*

Applications (Vol. 160, No. 168, pp. 160-168). Lawrence Erlbaum.

[11] Koza, J.R. and Koza, J.R., 1992. Genetic programming: on the programming of computers by means of natural selection (Vol. 1). MIT press.

[12] Montana, D.J., 1995. Strongly typed genetic programming. Evolutionary computation, 3(2), pp.199-230.

[13] Schmitt, Lothar M. "Theory of genetic algorithms." *Theoretical Computer Science* 259, no. 1-2 (2001): 1-61.

[14] Clack C., Yu T. (1997) Performance enhanced genetic programming. In: Angeline P.J., Reynolds R.G., McDonnell J.R., Eberhart R. (eds) Evolutionary Programming VI. EP 1997. Lecture Notes in Computer Science, vol 1213. Springer, Berlin, Heidelberg.

Shreeraj Pawar. "A Survey on Genetic Algorithms." *International Journal of Engineering Research and Applications (IJERA)*, vol.10 (12), 2020, pp 25-29.