CLASSIC SNAKE GAME OPTIMIZATION USING GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORK

Abstract

Genetic algorithms (GAs) draw inspiration from Darwin's theory of natural evolution, which elucidates the emergence and survival of species on Earth. In GA, we initiate a population comprising individual solutions to a given problem within a generation. From this population, we select the most optimal solutions based on their fitness and conduct mating operations among them, introducing a defined mutation rate. Through this process, GA generates a new population for the next generation, with the objective that the new individuals exhibit higher fitness levels than the current generation. This iterative process continues for a predefined number of generations until the solution with the best/optimal fitness is attained, at which point the process halts. We apply this GA approach to a snake game agent/environment and observe the performance enhancements within the game. The underlying concept driving this methodology is neuro-evolution, a fusion of Artificial Neural Networks (ANNs) and Genetic Algorithms. Here, the snake's brain is represented by the neural network, which governs all the snake's activities, while GA aids in discovering the most suitable or optimal neural network configuration for sustainable optimal snake performance. Our proposed neuro evolutionary method can be further used to develop sustainable model to address different societal issues.

Keywords: Genetic Algorithms (GAs), Natural Evolution, Fitness Function, Neuro-evolution, Artificial Neural Networks (ANNs)

Authors

Sourav Sahoo
Department of Computer Science and Engineering, JIS College of Engineering, Kalyani, Nadia, India
souravsahoo73244@gmail.com

Pranati Rakshit
Department of Computer Science and Engineering, JIS College of Engineering, Kalyani, Nadia, India
pranati.rakshit@jiscollege.ac.in

I. INTRODUCTION

This paper is based on developing an evolved and intelligent “Snake-Agent” using combination of Genetic Algorithm and Neural Network which can intelligently play on classic Snake Game Environment. The Paper deals with how by running genetic algorithm for more number of generations than previous trial can generate more higher fitness based Neural Network(offspring)in final generation as expected but also sometimes it doesn’t work like this, it observed that in a current trial after running genetic algorithm for more number of generations than previous trial till the fitness received from best chromosome in final generation has less fitness value than the previous trial’s final generation’s best chromosome’s(in this case Neural Network acting like chromosome/offspring/DNA) fitness value. This show there is a great uncertainty exist in Genetic Algorithm. our paper covered/showed both aspects of Genetic Algorithm above discussed, and we will explain why such optimal search based heuristic algorithm like genetic algorithm which generally work on a complex solution space sometimes give unexpected results after running it for more numbers of generations than previous trial.

The primary motivation behind employing Genetic Algorithms (GA) to optimize the performance of the snake-agent in the classic snake game environment stems from the inherent complexity of such real-world gaming applications. These types of gaming problems entail numerous strategies, making it challenging for the game agent to discern the optimal strategy using conventional methods like brute force. The sheer magnitude of potential strategies renders brute force impractical, as it would be unfeasible to track all possible strategies within a reasonable timeframe. Similarly, algorithms falling under the category of "greedy search" are also inadequate for identifying optimal strategies due to the vast solution space and complexity inherent in these applications.

In contrast, evolutionary search algorithms offer a compelling solution due to their heuristic approach, which enables them to navigate complex and expansive solution spaces efficiently. By leveraging evolutionary principles inspired by natural selection, these algorithms can iteratively refine solutions and converge upon optimal or near-optimal strategies within a polynomial time frame.

This highlights the strength and efficacy of evolutionary search algorithms in addressing real-world problems characterized by intricate solution spaces. Their ability to efficiently explore and exploit solution landscapes makes them a preferred choice for optimizing strategies and finding optimal solutions across diverse application domains.

II. LITERATURE SURVEY

Shen Hau Hor et al. [1] developed a Snake Game Agent by combining Genetic Algorithm with Artificial Neural Network. They focus on how by tuning different parameters of both Genetic Algorithm and Artificial Neural Network it makes effect on the performance on Snake Game Agent. One such important parameter is mutation rate. The Consistency and Performance of “snake-agent” both monitored continuously as when several parameters of both Genetic Algorithm and Neural Network Changed for testing purpose. This Researchers at last found out each parameter involve in Genetic Algorithm and Neural Network has a strong impact in the performance of “snake-agent”.

Manisha et al. [2] showed how a better playing "Snake Agent" can be possible to make by using best evolved neural network with optimal performance which possible to get by using Evolutionary Algorithm like Genetic Algorithm which inspired from Darwin's theory of "natural selection" and "Survival Of the Fittest" concept. This paper showed by using better selection strategy how by running Genetic Algorithm for several generations can give optimal weight based neural networks on which "Snake-Agent" will perform best on its environment. This Author plotted graphs and showed how the performance improvement happened after each generation of the "Snake-Agent"

Piotr Bialas et al. [3] applied the technique known as "Neuro-Evolution" which is combination of Neural Network with Genetic Algorithm, to get best evolved "Snake-Agent". Neural Network basically act like DNA or chromosome for genetic algorithm and by the evolutionary process of genetic algorithm which mimics the idea of darwin's 'natural selection' and 'survival of the fittest', it creates population of such neural networks generation after generation, where each generations neural network will give better performance than previous generation(expected)until final generation reached, and optimal neural networks received by which "snake-agent" will perform best.

Prashanth Sridhar et al. [4] discussed about the importance of computational intelligence-based algorithms and how such algorithm helping to make human level intelligence based "video games agent", this present work also includes discussions on an intelligent "Snake-Agent" development with the help of Genetic Algorithm and Neural Network which can intelligently play on "classic snake" game environment.

L.Haldurai et al. [5] discuss about different important parameters and operators of genetic algorithm in detail and came to a conclusion how this algorithm is well suited for application/problem have a complex solution space or changing environment.

T.Boris et al. [6] trained a game agent using neuro-evolution approach (Neural Network+Genetic Algorithm) to play properly in a game environment, the game known as "game 2048".They Showed on their paper for at least after running this approach for hundred generations the agent able to properly play on the game environment.

J. Carr et al. [7] discussed what are important parameters need for Genetic Algorithm to properly function and this paper also discussed about the history/source/birth of this algorithm, current areas/applications where it is using and about its future scope. The Author Using 'MATLAB' application software ran this algorithm on several problems 'Travelling Salesperson' one of them.

B. Halmosi et al. [8] created an artificial intelligent snake agent using "Neuro-Evolution" approach (Combination of Neural Network and Genetic Algorithm) and tested the limitation (how much they can achieve by their approach).

O. Kesemen et al. [9] used a multi-layered Genetic Algorithm (GA) to solve 'the cross-matching puzzle problem' but due to the expanding size of this puzzle the multi-layered GA fails to find good solution for this problem but using an intelligently improving GA as the size of the puzzle in problem expanded, GA at last able to give good solution for this problem. So, this paper highlighting about an intelligent/self-improving/problem specific knowledge-based GA which can tune all its parameters as per the changes happened on the problem's environment thus such GA is very successful on finding solution for such problems where changes on problem environment can occur.

Y. Mishra et al. [10] used the famous Neuro-Evolution approach of Machine Learning to train famous "Flappy Bird" game-agent for optimal performance in its environment. They later stage study the effect on performance of game-agent by changing some important parameters at training process like- Increasing No. of neurons on hidden layers of neural network, speed of agent,gravity,distance between trees etc.

N. Nezamoddini et al. [11] discussed about a unique approach on their paper i.e. how Artificial Neural Network (ANN) acts as a brain for Genetic Algorithm (GA), how ANN can help to optimal solution finding meta heuristic algorithm like GA to quickly reach to global optimal solution in less computational time complexity comparison to when GA independently run for finding optimal solution of a problem.

III. METHODOLOGY

Our present work aims to develop an intelligent "Snake-Agent", which will perform in best possible way on classic snake environment. For developing it, we are using a very important mathematical function popular on the area of 'Machine Learning and Artificial Intelligence' Known as 'Artificial Neural Network'('ANN') in its simplest form and also using one optimal search based meta-heuristic type of mathematical algorithm which mainly use for solving such problems which has a very complex solution space and by any other approach to find best solution for such problem is huge time consuming and one way impossible, and this algorithm come under the big umbrella of Evolutionary Algorithms known as Genetic Algorithm(GA).

In our present work, 'ANN' acted like the brain of the playing "Snake-Agent"[2][3][4][6][8],by getting proper information as in the form of output value from one of the output unit of 'ANN' which giving the biggest value comparison to all other output nodes or units ranging between '0' and '1', "Snake-Agent" will perform certain act in its environment, so if we want the 'Agent' will perform as much good as possible on its environment then fulfilling that expectation the quality of output coming from 'ANN' must be as much perfect or accurate as possible. In one simple way 'ANN' is like brain of the 'Agent' (Fig. 1), So if the brain doesn't able to give right or accurate instruction then 'Agent' will not able to perform good way on its environment. So, for getting right and proper output from 'ANN' at right time we need to train up 'ANN' model function by tuning all the connection weights value between a single functional unit or also known as 'Neuron of ANN' with all others functional units/Neurons of previous layer of the 'ANN' function model with proper and accurate value (Fig. 2). So, for achieving this, there is one method available known as back-propagation technique which will try to fine tune all weights value on 'ANN' by doing this technique again and again epoch after epoch until the weights on 'ANN' will fine tune and model will start to give accurate output at right time. But main problem with this approach is, this approach requires training dataset and for solving problem like making an intelligent 'Snake-Agent' which will perform good in its environment require an unimaginable complex training dataset for 'ANN'. So, instead of taking this burden we used an algorithm of 'Evolutionary Search' category known as Genetic Algorithm, mainly expert in giving optimal solution for problems which has complex solution spaces by mimicking the concept of 'Natural Selection' and 'Survival of the fittest' which comes under Darwin's Natural Evolution Theory. So, in our case 'GA' will considers sequence of weights of 'ANN' as a single solution or offspring on its large solution space which in first generation fill with random different possible sequences of weights with different values and all of them acting like a different possible solution [2]. After this GA using it's evolutionary approach by calculating the fitness values of each such solutions by playing a game with using each such solutions or sequence of weights and using a proper defined fitness function will select only some number of best fitness based solution and using operator like crossover and mutation it will mate those best solution of the current generation to create new better fitness based child offsprings for next generation(expected) and at the same time we used elitism selection[5] method which also transfer parents as offspring in next generation, this method will help best fitness based solutions of current generation not become lost due to the process of crossover and mutation. So, in this way, GA (Fig. 3) will run for a fixed number of generation and at last generation we will get a very good fitness based offspring with the help of this evolutionary approach of GA [2][3]. In this case, we will get a possibly one of best sequence of weights which is offspring in GA. Our aim is to show in this paper that if we run this GA for more number of generations than previous trial then higher fitness based offspring activation is possible, which in our case more well-tuned and perfect sequence of weight values, by using output from such Neural Network which has such fine-tuned perfect weight values the 'Agent' will perform better than previous trial [3]. Fig. 4 depicts the architecture of an Artificial Neural Network.

❖ **Important Parameters in ANN And GA**

1. **Artificial Neural Network(ANN)**
   No. of Hidden Layers: 2
   No. of functional units/Neurons used in input layer: 7
   No. of functional units/Neurons used in hidden layer 1: 9
   No. of functional units/Neurons used in hidden layer 2: 15
   No. of functional units/Neurons used in output layer: 3
   Activation function used in 1st hidden layer: tanh
   Activation function used in 2nd hidden layer: tanh
   Activation function used in output layer: softmax

2. **Genetic Algorithm(GA)**
   Type of Selection Operator: Roulette Wheel
   Type of Crossover Operator: Uniform Crossover
   Type of Mutation Operator: Random Resetting Mutation
   Total No of Generations for Which GA Ran (1st Trial): 100
   Total No of Generations for Which GA Ran (2nd Trial): 120
   Total No of Generations for Which GA Ran (3rd Trial): 150
   Total No of Generations for Which GA Ran (4th Trial): 200
   Total No of Generations for Which GA Ran (5th Trial): 220
   Total No of Generations for Which GA Ran (6th Trial): 250
   Total No of Generations for Which GA Ran (7th Trial): 300
   No of Solutions in A Population On Each Generation: 50

No. of Parents Involving In Mating On Each Generation To Create New Offsprings or Solutions For Next Generation: 12

No. of Newly Created Solutions or Offsprings in Every Next Generation: 38
Selection Type: Elitism Selection [5]

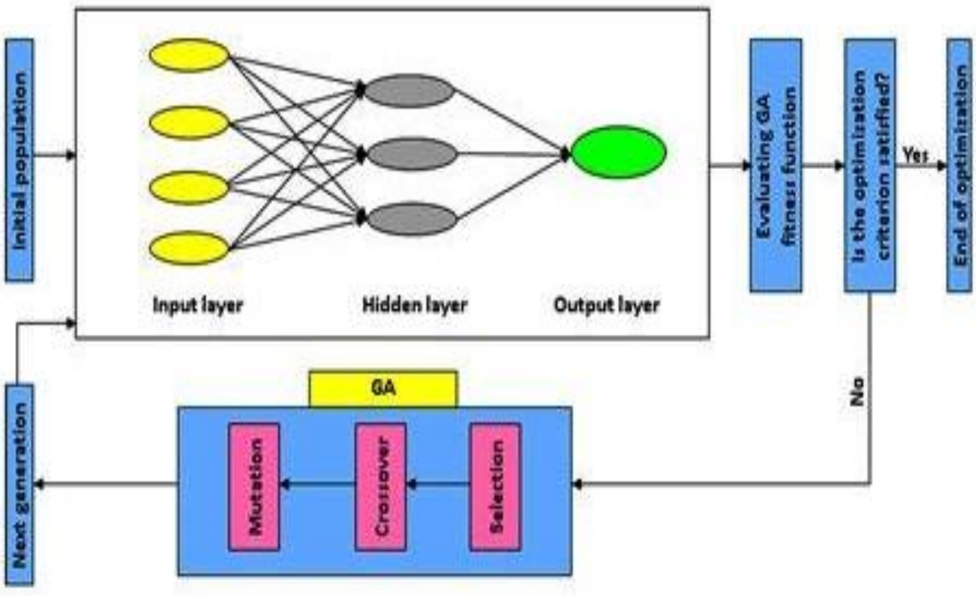

Figure 1: Whole Model for Snake Agent (Genetic Algorithm + Artificial Neural Network)

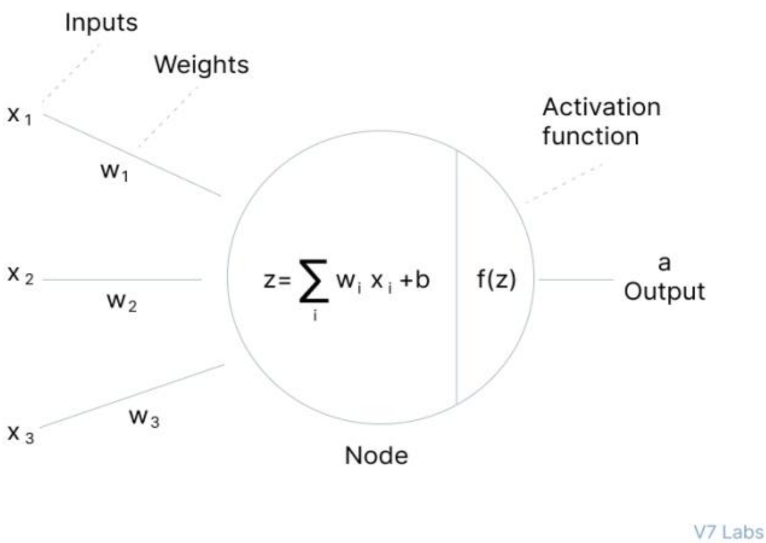

Figure 2: Mathematical Formulation of a Single Functional Unit/Neuron of Artificial Neural Network

Table 1: Fitness value from best solution in final generation

Total No of Generations for Which GA Ran	The Fitness Value Received from Best Solution in Final Generation
100	50620
120	174424
150	180000
200	194630
220	165000
250	175000
300	139672

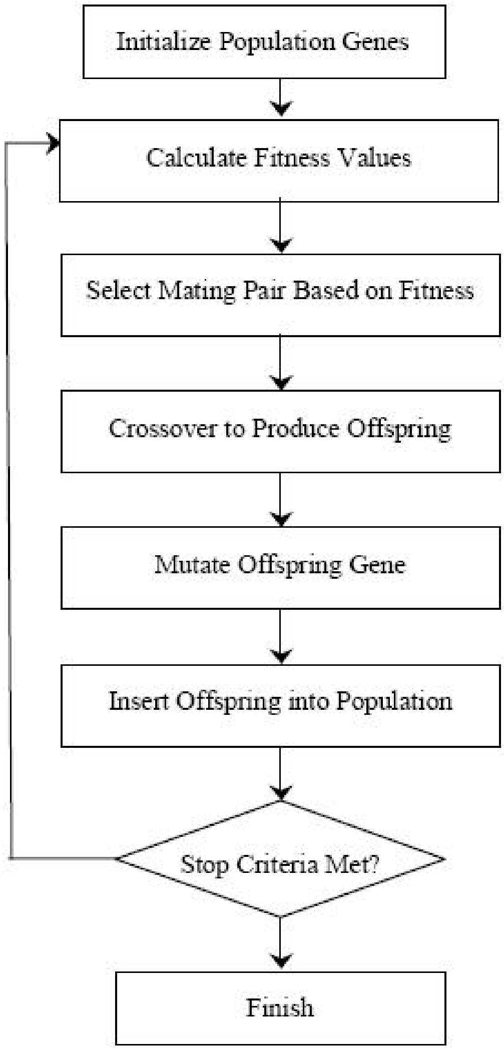

Figure 3: Working Flowchart/Diagram of Genetic Algorithm

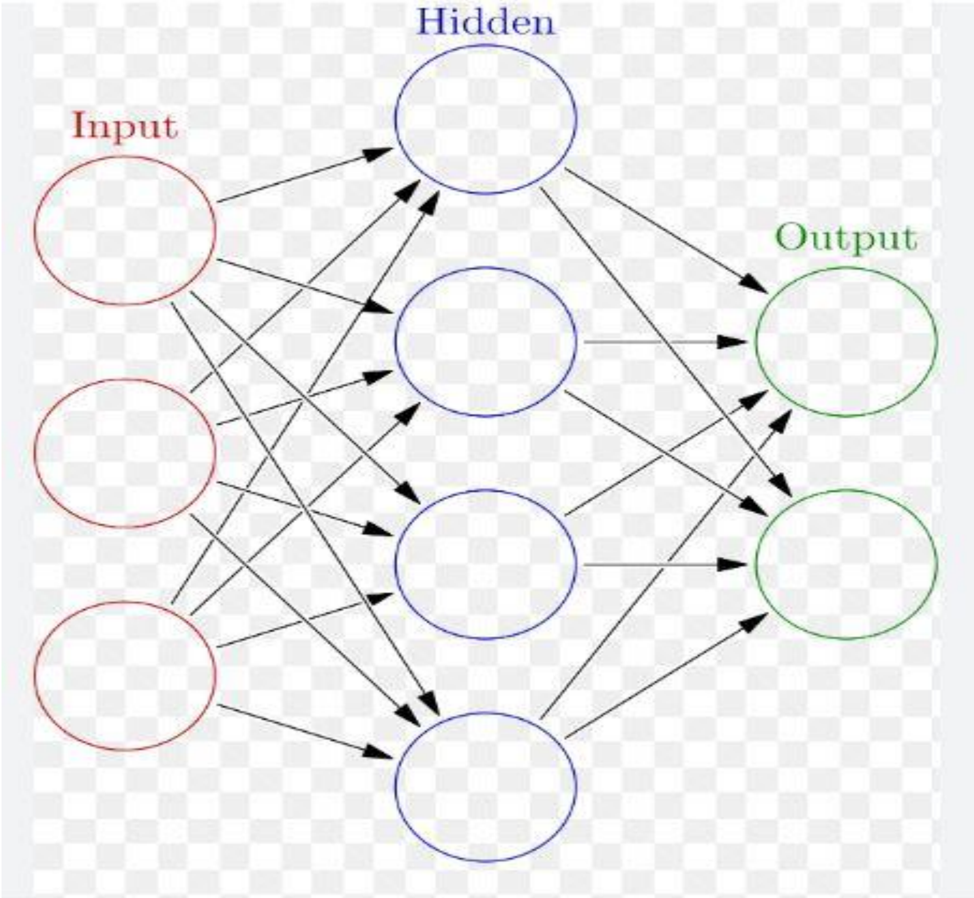

Figure 4: Architecture of an Artificial Neural Network

IV. RESULT AND DISCUSSION

The aim is to show in this paper that how by running Genetic Algorithm for more number of generations in current trial than previous trial will give higher fitness based optimal solution/offspring in final generation than previous trial's final generation's best offspring/solution. We applied Genetic Algorithm to find out the best fitted Neural Network (which has optimal sequences of weight values) for a 'Snake-Agent' which will play in a classic snake game environment. We ran this Genetic Algorithm for 4 trials and below we will show how by running Genetic Algorithm for a greater number of generation than previous trial higher fitness based solution/offspring achieved on final generation, in our case offspring/solution is 'weight parameter values sequences' for a Neural Network, which is acting like the brain of our 'Snake-Agent'. 'Fitness score' in our problem is actually the score achieved by the game-agent by playing a classic snake game in the game environment for '2500' steps. There is total 2500 steps on a single game. On every grasp of food (Apple) we awarded snake agent with 5000 reward points and when the agent hit on the wall or with it's own body we rewarded a penalty of 150 points, this is the nature of fitness function in our problem. Table 1 depicts the fitness value received from best solution in final generation. Fig. 5 shows that higher fitness based solution is achieved as genetic algorithm run for more number of generation. Fig. 6 depicts the pathway GA followed to reach Global Optimal position.

- 1st Trial (GA ran for 100 generation) fittest offspring/solution achieved on final generation has fitness value

```
#######  fittest chromosome in gneneration 99 is having fitness value:   50620
```

- 2nd Trial(GA ran for 120 generation) fittest offspring/solution achieved on final generation has fitness value

```
#######  fittest chromosome in gneneration 119 is having fitness value:   174424
```

- 3rd Trial(GA ran for 150 generation) fittest offspring/solution achieved on final generation has fitness value

```
#######  fittest chromosome in gneneration 149 is having fitness value:   180000
```

- 4th Trial (GA ran for 200 generation) fittest offspring/solution achieved on final generation has fitness value

```
#######  fittest chromosome in gneneration 199 is having fitness value:   194630
```

Graphical Representation

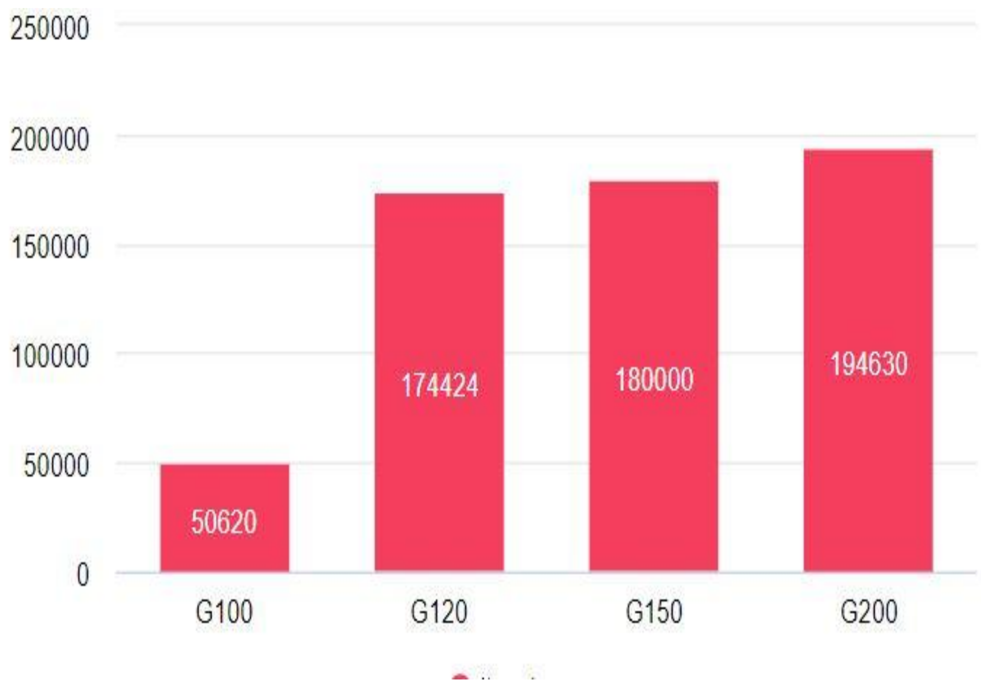

Figure 5: Higher Fitness Based Solution Achieved as Genetic Algorithm Run for More Number of Generation

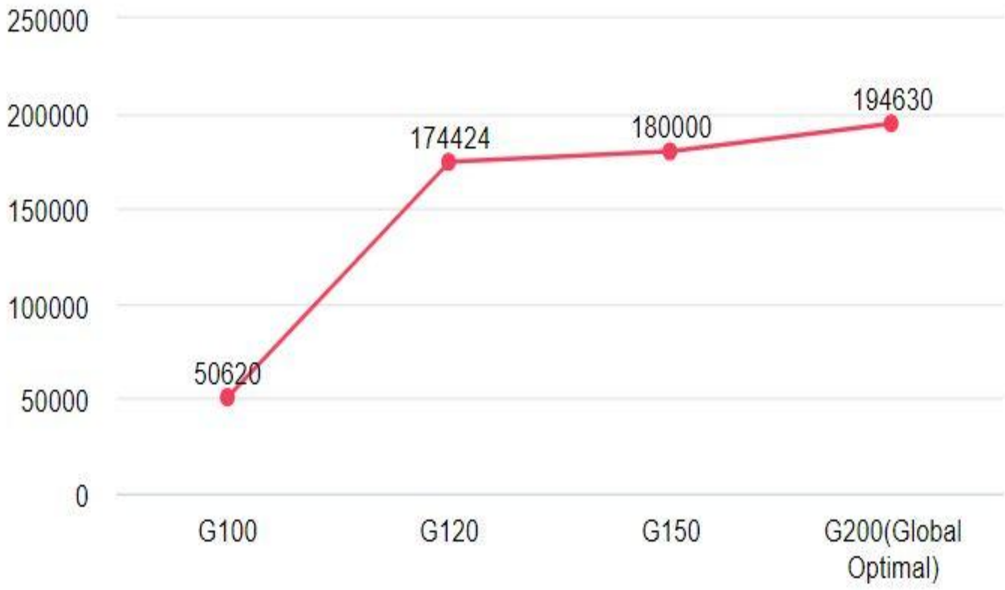

Figure 6: Pathway GA followed to reach Global Optimal position

Proof of Trial 4 Solution as Global Optimal Solution

In subsequent trials, even with a prolonged execution of the Genetic Algorithm (GA) over a greater number of generations than in Trial 4, the best offspring or solutions attained failed to exceed the fitness value of the optimal solution obtained at the conclusion of Trial 4. This observation serves to underscore the significance of the solution derived during Trial 4, suggesting that GA successfully converged upon the most effective solution or strategy for the snake agent. Consequently, it can be inferred that the solution obtained in Trial 4 represents the global optimal solution for the given problem.

- In trial 5 (GA ran for 220 no of generations) the fitness received from best solution

```
####### fittest chromosome in gneneration 219 is having fitness value:   165000
```

- In Trial 6 (GA Ran for 250 No of Generations) The Fitness Received from Best Solution

```
####### fittest chromosome in gneneration 249 is having fitness value:   175000
```

- In Trial 7 (GA Ran for 300 No of Generations) The Fitness Received from Best Solution

```
####### fittest chromosome in gneneration 299 is having fitness value:   139672
```

Figure 7: The Fitness Value Received by Running GA for Different No of Generations, Best Solutions/ Global Optimal at generation 200 which Can't be Surpassed by any other Fitness value

This paper highlights this fact here that though after getting 'global optimal solution' by GA for any problem if we run GA for more number of generations we can't get better fitness based solution than 'global optimal', just because we run GA for more number of generations it don't mean anything because the fact which we already highlighted above that no solution's fitness can surpass the fitness of 'global optimal solution' (as it is best solution of given problem), though they received as best solution by running GA for more number of generations in comparison to total no of generations for which after running GA got 'global optimal solution' (Fig. 7). Fig. 8 depicts the fitness improvement percentage during two generation numbers for which Ga runs till it found Global Optimal.

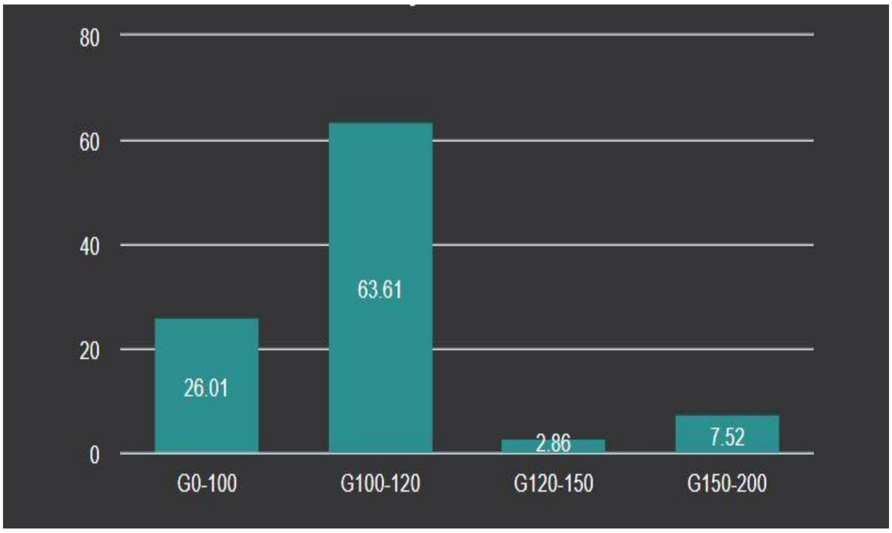

Figure 8: Fitness Improvement Percentage during Two Generation Numbers for Which Ga Runs till It Found Global Optimal

V. CONCLUSION

The results clearly demonstrate that when all the parameters of a genetic algorithm are perfectly defined (such as encoding type of individual solutions, fitness function, selection/crossover/mutation operators, etc.), the genetic algorithm can find a better optimal solution for any problem when it runs for more generations than in the previous trial. In this case, it results in a superior optimal neural network with higher fitness for controlling the snake agent in its environment. This illustrates the strength and effectiveness of the genetic algorithm in finding optimal solutions through its evolutionary approach, particularly for problems with large solution spaces. Unlike other algorithms, which cannot efficiently find optimal solutions for such complex problems in polynomial time, the genetic algorithm excels.

However, there is an exception in the genetic algorithm's performance. After running the algorithm for more generations than in the fourth trial, we unexpectedly obtained a solution with lower fitness as the best solution in the final generation. This outcome occurred because the solution obtained in the fourth trial was the global optimal solution. Despite running the genetic algorithm for more generations in the subsequent trial, the fitness value of the best solution found could not surpass the fitness value of the global optimal solution obtained in the fourth trial.

Exploring alternative combinations of selection, crossover, and mutation operators within Genetic Algorithms (GA) presents an intriguing avenue for future research in the optimization of snake game agents within the classic snake game environment. While the current project utilized specific types of evolutionary operators, future investigations could examine the impact of employing different combinations of these operators.

By varying the selection, crossover, and mutation strategies and running GA across multiple trials, researchers can assess whether alternative approaches yield superior results compared to the current maximum fitness score achieved by the snake game agent. This comparative analysis could shed light on the efficacy of different evolutionary operator configurations in optimizing the performance of game agents within complex gaming environments.

Furthermore, exploring the interplay between different operator types and their effects on convergence speed, solution quality, and robustness could provide valuable insights into the optimization process. By systematically evaluating various operator combinations and their impact on GA performance, researchers can advance our understanding of evolutionary algorithms' behaviour and effectiveness in addressing real-world optimization challenges for sustainable development.

The same kind of problem solving approach as discussed in our paper can possibly to be implemented in some real world problem solving by any other types of robotic agents also. Like in present days there are many kind of humanoid robots working in big industrial complexes in different domains, we can certainly use a deep neural network for decision making of these robots in complex dynamic work environment and the performance of such robots we will optimize by optimizing neural network’s parameters using Genetic algorithm, so basically what we can do before using these robots in real industries, we will train and optimize those robot’s performance first in a virtual environment of the same industry model, and we will deploy these robots in physical environment only after observing their optimal performances in virtual environment.

REFERENCES

[1] Shen Hau Hor, Sheng Jeh Tan, MunKye Yan, ZailanArabee bin Abdul Salam & Yoke Shin Sim (**2022**). "Snake Game: A genetic neural network approach". In Journal of *Applied Technology and Innovation (e – ISSN: 2600-7304)* **Volume.6**, no. 1.

[2] Manisha (**2020**) "An Implementation of Neural Network with Evolutionary Algorithm in Gaming Application". In *JETIR* June 2020, **Volume 7**, Issue 6

[3] Piotr Bialas "Implementation of artificial intelligence in Snake game using genetic algorithm and neural networks" from Faculty of Applied Mathematics, Silesian University of Technology.

[4] Prashanth Sridhar, Bachelor's Thesis of Prashanth Sridhar. "Implementation of the Environment of a Practical Work in a Computational Intelligence Course"

[5] L.Haldurai,T.Madhubala & R.Rajalakshmi."A Study on Genetic Algorithm and its Applications". In *International Journal of Computer Sciences and Engineering* **Volume-4**, Issue-10

[6] T.Boris and Goran.S, (**2016**), November. "Evolving neural network to play game 2048". In *24th Telecommunication Forum(TELFOR)*.

[7] J. Carr (**2014**). "An Introduction to Genetic Algorithms".

[8] B. Halmosi and C. Sik-Lanyi (**2019**). "Learning to play snake using genetic neural networks". *Pannonian Conferences on Advances in Information Technology (PCIT 2019)*, **Volume.4**, no. 5,pp. 126-132

[9] O.Kesemen and E. Ozkul(**2016**). "Solving cross-matching puzzles using intelligent genetic algorithms". *Artificial Intelligence Review*, **Volume.49**, no.2, pp. 211-225

[10] Mishra.Y,V.Kumawat and K.Selvakumar, (**2019**), May. "Performance Analysis of Flappy Bird Playing Agent Using Neural Network and Genetic Algorithm." *In International Conferences on Information, Communication and Computing Technology (pp. 253-265), Springer, Singapore*.

[11] Nezamoddini.N & A. Gholami (2019). "Integrated Genetic Algorithm and Artificial Neural Network". In (**2019**) *IEEE International Conference on Computational Science and Engineering(CSE)* and *IEEE International Conference on Embedded and Ubiquitous Computing (EUC) (pp. 260-262). IEEE.*