Case Study of Genetic Algorithm in the Game of Flappy Bird

Kin NG
Dept. Computer Science & Engineering
University of South Florida
Tampa, FL.
kin1@usf.edu

***Abstract*—The game of flappy bird consists of a bird that navigates through a pair of pipes. In the game, the score of a match is determined by the number of pipe pairs that a bird manages to pass through without causing a collision with the pipes. This paper is aimed at the exploration of an artificial intelligence for the game by employing the library of neuro evolution of augmenting topologies for the realization of a genetic algorithm that produces a bird AI capable of achieving a score of 200 points.**

***Index Terms*—genetic algorithm, artificial intelligence**

I. Introduction

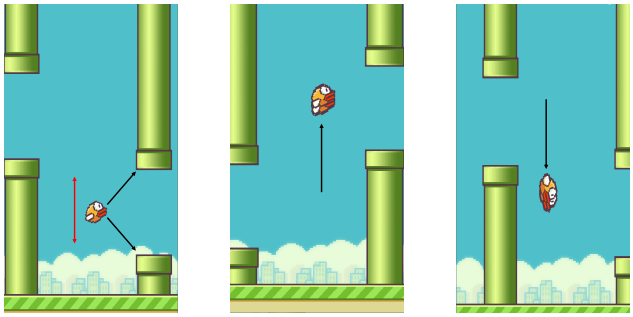
Genetic algorithms are part of the optimization branch of algorithms, and are based on the process observed in natural evolution, where each individual of a population represents a chromosome and evolve through successive iterations that result in a new generation [1]. In the context of flappy bird, each individual bird agent represents a chromosome and each successive population that results from the use of the fitness function is said to be a generation of birds. The fitness function is the metric in which agents are analyzed in order to determine the performance on the environment. Moreover, the environment in flappy bird is defined by the base floor and set of pipe pairs through which the bird must navigate to increase its score. The objective of applying a genetic algorithm to a problem is to converge to a single best individual that represents the optimal or sub-optimal solution to the problem [1]. For the purpose of metric analysis, a score of 200 points has been set as the goal state for the project which indicates the success of the algorithm in the creation of a bird AI.

The paper is divided into four main sections: system design, experimental results, results analysis, and conclusion.

II. System Design

The system design for the project consist of a genetic algorithm based on the neuro evolution of augmenting topologies. In NEAT, individual agents compete primarily within their own niches instead of taking into consideration the whole population size [2]. The aforementioned allows bird AI's to be identified easier within the population created; thus, resulting in a faster acquisition of the best individual bird to pass the goal test.

In terms of percepts, the bird AI obtains the distance between the two pipes and its position relative to the y-plane. The combination of these factors allows the flappy bird AI to improve upon previous generations by getting information that is necessary to achieve a higher performance. In terms of the output layer, the bird AI acts upon the environment with the jump or not jump command. In the system, the module of neuroevolution of augmenting topologies is used to control the percept-output connection. The feed forward neural network assigns a weight for each percept of the bird AI which are composed of the bird y-plane, and the top and bottom pipe positions with respect to the bird. After, a weighted sum of all the percepts with their corresponding weights is computed to obtain an estimate useful for the bird AI positioning with respect to the environment. Moreover, a heuristic bias is applied to the weighted sum to adjust the network percepts of the bird AI. Lastly, the hyperbolic tangent function is used as the activation function which allows the algorithm to map the value computed by the weighted sum to a common domain. In this case, the hyperbolic tangent function will map any neural output value that is above 0.5 as a jump command as shown in Figure (1b) ; otherwise the command will be no jump.



(a) Input Layer (b) Jump Command (c) Hold Command

Fig. 1: Input and output layer representation of the system. Figure (a) shows the input layer where the red arrow represents the y-plane position of the bird AI, and the black arrows represent the position of the top pipe and bottom pipe with respect to the bird AI. Figure (b) shows the output layer jump command. Figure (c) shows the output layer for the hold command

Furthermore, the genetic algorithm design was realized by

a combination of NEAT module functions and algorithmic processes that integrated to the overall functionality of the game. The process to model and experiment the flappy bird AI was as follows:

1) Creation of the flappy bird population through the use of neat configuration file. The population command was varied during experimental testing.
2) Creation of neural network controllers for each of the individual birds by using a feed forward network function from NEAT.
3) The fitness function was based on the score progression of a bird AI. For example, a bird AI that achieved a score of 20 points is better than a bird AI with a score of 5 points.
4) According to the fitness function, a percentage of the best birds gets extracted and crossed to create a new population of birds. Thus, eliminating the ones that performed the worst, and keeping the highest performing birds.
5) The offspring obtained repeat the process from step 2 to 4 until the bird AI reaches the goal state which will then terminate the process.

III. Experimental Results

The experiments where considered under 3 main parameters which include: population size, generations, and score. First, the population size was a factor that was analyzed in order to determine how does the number of agents used to train the model affected the performance of the system. Secondly, the number of generations was studied in order to understand the effects of fitness evaluation decisions as they pertain to the game. Lastly, the game score was the main indicator of performance measure that allowed to evaluate the goal test of the game.

The fitness function was integrated within the game to work in conjunction with the genetic algorithm. The following criteria was followed to increase and decrease the fitness level of an agent:

- Every bird AI started with a 0 as its fitness level.
- For every second the bird AI stayed alive in the game, its fitness level increased by 0.1.
- For every pair of pipes the bird AI managed to pass through, its fitness level increased by 5.
- If the bird AI had a collision with a pipe, its fitness level decreased by 1 and it was removed from the current execution.

For experiment 1 seen in Table I, a population size of 5 birds was used to run the genetic algorithm. Moreover, the top best performing bird AI was extracted within each generation according to the fitness function. Also, the average fitness of the population was noted for each generation.

For experiment 2 seen in Table II, a population size of 20 birds were selected, and the same calculations as experiment 1 were computed. In addition, the algorithm was also executed 5 times for the population size of 20 to compute the generational average of a the mid size population which was found to be around 3 generations to create a bird AI that passes the goal test as seen in Table III.

TABLE I: Genetic Algorithm with Population Size = 5

Generation Number	Best Fitness	Avg. Fitness
1	7.7	5.02
2	7.4	5.88
3	7.6	5.92
4	7.8	4.74
5	7.4	5.38
6	7.5	6.64
7	8.5	3.88
8	8.4	6.08
9	32.8	14.68
10	7.5	4.68
11	7.3	4.88
12	7.5	5.68
13	20.7	10.22
14	26.9	9.84
15	19.0	8.24
16	8.5	4.9
17	13.8	5.94
18	8.2	7.08
19	7.9	6.6
20	21.2	9.92
21	58.8	15.8
22	33.3	10.6
23	inf	N/A

TABLE II: Genetic Algorithm with Population Size = 20

Generation Number	Best Fitness	Avg. Fitness
1	7.9	4.30
2	27.5	5.74
3	inf	N/A

TABLE III: Average Generational Success (Population Size = 20)

Test	1	2	3	4	5	Average Generational Success
Generations	2	6	5	2	2	3

Lastly, experiment 3, employed 100 agent birds to recreate the algorithm. The results showcased in Table IV show that from 5 total executions the algorithm only went to the second generation in 2 occasions. Moreover, those two test had very high fitness levels for their first generation as seen in Table V.

TABLE IV: Average Generational Success (Population Size = 100)

Test	1	2	3	4	5	Avg. Generational Success
Generations	1	0	1	0	0	0

IV. Results Analysis

From the results obtained from all the experiments, it was noticed that as the population number of agents used for the genetic algorithm increased, the more likely it was to find a bird AI that achieve the goal of 200 score points in fewer generations as seen in Figure 2. In particular, this result is clearly illustrated in Table V where the fitness levels of the

TABLE V: Genetic Algorithm with Population Size = 100

Test	Best Fitness	Avg. Fitness
1	39.1	4.77
3	81.8	4.78

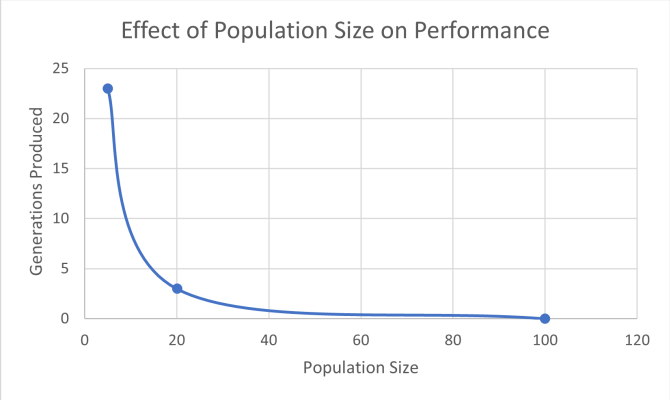


Fig. 2: The effect of population size on the genetic algorithm performance.The x-axis represents the population size used to train the algorithm. The y-axis represents the generations produced given a population size.

first generations are very high, but the average fitness of the population is low, this indicates that although many agents failed to pass in between the pipes, there were a few birds out of the population that learned faster to avoid those mistakes. This is an interesting discovery since the incentive given by increasing the fitness function of an agent as they manage to pass through a pipe or stay alive in the game suggests that they have a more rapid growth in terms of generations needed to find the optimal bird. Contrarily, it can be seen from Table I that the lower the population number of agents used for the algorithm, the more likely it is to get more generations in order to find the optimal bird that satisfies the goal state. Although there are some high peaks from generation 13 to generation 15, the genetic algorithm took 7 more generations to find the optimal agent bird. The aforementioned is indicative of the poor performance of the algorithm when lower number of agents are used, but at the same time is suggestive of the high potential of the algorithm of generating an optimal bird for a big population.

In terms of average generational success, it was noticed from Table III that for a population of 20 bird agents, the optimal bird AI was usually produced within 3 generations. More surprisingly, for a population of 100 bird agents, the optimal bird AI that completed the goal test of 200 score points was found on the first generation, and in the second generation in the worst case. This factor is due to the properties of NEAT neuro-controllers to have topological innovations protected and optimized before they have to compete against other niches in the general population [2]. Therefore, the success of finding the optimal bird AI in fewer generations is also dependent on the number of bird agents used to start the algorithm.

Time complexity of genetic algorithms can vary depending on the implementation, but they usually consist of a set of potential solutions to the problem in the form of individual chromosomes (population), the number of generations needed to produce the optimal chromosome, the fitness function to compute performance, and the crossover and mutation process that might be added to create the next population [3]. The aforementioned implies that in general the complexity of genetic algorithm is given according to the parameters n used for the function O(g(n)). For the project, a total of m generations where computed and for each generation a total of n individual agents where tested. Moreover, the fitness function was integrated into the game and it only consisted of constant assignments to the fitness value of a particular bird taking O(1). The total time complexity for the algorithm is O(nm) where n represents the population size and m represents the number of generations.

Lastly, in the experiments a limit of 50 generational constructions was set to stop the program in case no optimal bird AI was found that could surpass a score of 200 points. For all the experiments, it was surprising to see that all populations regardless of the size found the optimal bird agent; however,as it was mentioned previously the number of generations in which the optimal bird AI was found is closely related to the number of agents used to execute the algorithm and thus using a population size lower than 5 agents might not be suitable to produce optimal results. Overall, it was notice that as the variation for the neuro-controllers increased, the rate at which the AI learned also increased.

V. Conclusion

In the game of flappy bird, the application of genetic algorithms were found to be useful in the exploration of an artificial intelligence that could fly infinitely without hitting a pipe. Genetic algorithms are based on the principle of evolution and as such they are a perfect fit for games that can have a semi-deterministic environment and where the output layer consists of a binary command. In this paper, the results obtained showcase that optimal bird agents can be obtained very fast with the use of the neuro-controllers from the NEAT module with surprising outcomes such as being able to find the optimal bird AI in only 1 generation given that the population size is large enough to satisfy the learning curve of the algorithm. Thus, it is shown the utmost importance and relevance of genetic algorithms for simple games such as flappy bird, but also for development and study of learning agents within the domain of artificial intelligence.

References

[1] P. Guo, X. Wang, and Y. Han, "The enhanced genetic algorithms for the optimization design," in *2010 3rd International Conference on Biomedical Engineering and Informatics*, vol. 7. IEEE, 2010, pp. 2990–2994.

[2] K. O. Stanley and R. Miikkulainen, "Efficient evolution of neural network topologies," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2. IEEE, 2002, pp. 1757–1762.

[3] Z. Nopiah, M. Khairir, S. Abdullah, M. Baharin, and A. Arifin, "Time complexity analysis of the genetic algorithm clustering method," in *Proceedings of the 9th WSEAS International Conference on Signal Processing, Robotics and Automation, ISPRA*, vol. 10, 2010, pp. 171–176.