|  |
| --- |
| **CMP304 Coursework**  **Project Report**  **Genetic Algorithm Pong** |
| **by: Ewan Stewart**  **1900598** |

**Introduction**

This project will explore a genetic approach to optimizing a paddle’s movement in an adapted version of Pong. Through genetic techniques, the paddle’s movement will be manipulated to reach the goal of creating a paddle that can effectively and consistently collide with a ball. To achieve this, the algorithm will traverse through a large combination of values to find a set of values that can satisfy the end condition of the algorithm.

This algorithm implements the genetic techniques of simulation, evaluation, selection, crossover and mutation to produce its result. These techniques were chosen as combined they are an effective way to explore a large space of number combinations in a short space of time over other artificial intelligence methods.

This algorithm will have three end conditions. If the paddle can make thirty successful collisions within a generation. If there is no meaningful progress within ten generations or if the generation count exceeds one hundred.

To hypothesis, using a genetic algorithm to solve this solution will take longer on average (number of generations) depending on the difference of speed between the paddle and the ball.

Background

Genetic Algorithms are a heuristic search algorithm that models itself on theorized real life evolutionary patterns. Mimicking genetic structure and chromosomes within nature, genetic algorithms can be implemented to search through and optimise data by learning from past data to satisfy a solution.

Genetic Algorithms in their most basic form require two evolutionary concepts: cross-over and mutation. In present wildlife as well as in how mankind evolved, survival of the fittest principles are in play. Members of a population which have the best traits and characteristics naturally produce the strongest offspring. Therefore, throughout generations weak and undesirable characteristics die out and the current generation should contain the best characteristics of the previous generations.

For most of the generation, they gain their genetic structure through cross-over from their genetic parents, this is the most important part of the algorithm as it combines the characteristics and traits of the past generation to create new line of strong offspring. In a genetic algorithm, cross-over can be implemented using several different operators however this solution will implement single point cross-over which takes two members of the population swapping and mixing their characteristics to produce a new member.

Mutation also plays a vital role in creating a strong and diverse population. Mutation in a species is defined as a small random tweak in a trait or characteristic which produces a different genetic structure in a member of the population. Mutation maintains and introduces diversity into a population preventing the full convergence of a population. Full convergence being that the population has identical genetic structure. As with cross-over there are many options in implementing mutation however this solution will make use of random resetting where a random value is assigned to a random characteristic.

Through these concepts, an algorithm can be created to solve the paddle’s movement in the implemented Pong game.

Methodology

This algorithm implements three fundamental objects: the paddle; the object directly modified by the algorithm, the ball; the object which evaluates the paddle, and the generation; the object which controls the genetic algorithm’s flow. See Appendix 1A for game structure.

***APPENDIX 1A***

Chart

Description automatically generated

The paddle is the critical object within the algorithm and contains the genetically controlled movement function. The paddle is genetically structured with a float and an integer value which manipulates how the paddle responds to the moving ball. These values are generated in relation to how the previous generation performed, using the genetic techniques of cross-over and mutation to produce them.

To create the simplest paddle that can infinitely collide with the ball, the paddle’s intercept position should match the incoming ball’s position as shown in Appendix 2A.

***APPENDIX 2A***

******

However, to create a solution, that a genetic algorithm can solve, the paddle’s intercept position has been skewed. The goal of the algorithm is to generate a set of values that can mitigate the result of the modification. Using genetic techniques, an integer and float value is generated, and the paddle’s intercept position is calculated as shown in Appendix 2B.

***APPENDIX 2B***



Using the paddle’s intercept position the paddle can now move. The paddle remains idle until the ball reaches halfway across the screen, then it begins to move. If the paddle is beneath the intercept position, then the paddle moves up and if the paddle is above the intercept position, then the paddle moves down. Moving until it reaches the defined intercept position.

If the paddle collides with the ball, then the paddle’s score is increased. A paddle with a higher score is more likely to be considered to have its values carried onto the next generation.

The paddle can move 4 units in either direction where the ball moves at 4.8 units diagonally. Adjusting the difference in units between the paddle and the ball will impact the average speed at which the algorithm solves the problem. A smaller unit difference increases the range of values that can solve the algorithm therefore decreasing the average algorithm solve time. Where larger unit differences decreases the range of values acceptable to solve the problem therefore on average increasing the average solve time.

The ball object isn’t controlled by genetics and its movement is predictable with some random elements to allow the paddle to predictably track it. The ball is essential as colliding with the paddle creates the main objective of this system and allows the genetic algorithm to deem which paddles should live onto the next generation.

The implementation of generations facilitates the genetic process. Each generation has a population size of ten, so that within each generation there are ten paddles with ten corresponding balls. The population size relates to the number of combinations of values that can be explored, which theoretically should increase the likelihood of solving the solution. Each paddle relates to a ball, and both have the same RGB value to denote this. A paddle exists independently from other members of the population and cannot interfere with other member’s fitness.

Genetic algorithms rely on the past genetic data from previous generations in order to generate the current generation’s genetic structure however, with the initial generation that is not possible as there is no prior generation. To circumvent this random generation is used to create the initial generation, generation zero. Each paddle is instantiated with a random integer value ranging from (0, 500) and a random float value ranging from (0, 5) and is applied to the paddle’s movement function as shown in Appendix 2B.

A generation is considered over/complete when all paddles fail to collide with their respective ball, or an individual paddle reaches a collision score of thirty. Once the generation has ended, evaluation of the population can take place so that the next generation can be populated. The paddle scores from the generation are passed through a bubble sort algorithm which sorts them in order from the least successful to the most successful using the score data. With the data sorted this information is passed onto selection.

The selection algorithm considers all paddles into the breeding pool that scored the highest score in the generation or had a score greater than five. The selection process favours the highest scoring paddles breeding them the most, with the lesser scoring paddles having a lower chance to breed. The lesser paddles are considered into the breeding pool for genetic diversity and exploration purposes.

During the cross-over period there is a chance of mutation of the paddle. The generated float or integer value have an equal chance of randomly mutating to a new random value discarding its parents’ genetics. The mutation rate is set to 30% to keep exploration high and convergence low. A higher mutation rate should aid in finding a set of values that can satisfy the solution quicker when the previous generation is not near the solution.

Results

Breakdown using unit difference of 1

**Avg. generation count: 11 Avg. integer value: 400 Avg. float value: 2.68**

|  |  |  |
| --- | --- | --- |
| Generations run | Final generated integer value | Final Generated float value |
| 19 | 431 | 1.74 |
| 6 | 439 | 4.79 |
| 4 | 416 | 0.66 |
| 21 | 413 | 3.93 |
| 11 | 412 | 2.26 |
| 11 | 262 | 1.37 |
| 5 | 402 | 3.56 |
| 19 | 433 | 2.25 |
| 7 | 306 | 3.54 |
| 8 | 482 | 2.65 |

Breakdown using unit size of 0.9

**Avg. generation count: 5 Avg. integer value: 290 Avg. float value: 1.85**

|  |  |  |
| --- | --- | --- |
| Generations run | Final generated integer value | Final Generated float value |
| 3 | 442 | 2.9 |
| 3 | 412 | 1.03 |
| 5 | 77 | 0.9 |
| 1 | 383 | 2.03 |
| 1 | 193 | 0.81 |
| 2 | 270 | 2.32 |
| 16 | 171 | 1.15 |
| 4 | 312 | 4.47 |
| 2 | 284 | 1.36 |
| 9 | 357 | 1.53 |

Breakdown using unit size of 0.8

**Avg. generation count: 2 Avg. integer value: 372 Avg. float value: 2.57**

|  |  |  |
| --- | --- | --- |
| Generations run | Final generated integer value | Final Generated float value |
| 1 | 481 | 4.12 |
| 1 | 252 | 1.38 |
| 1 | 343 | 4 |
| 5 | 369 | 2.48 |
| 2 | 282 | 1.31 |
| 1 | 388 | 2.35 |
| 2 | 312 | 1.76 |
| 1 | 364 | 4.23 |
| 2 | 440 | 0.85 |
| 3 | 487 | 3.13 |

Discussion

The results show that the number of generations to solve the solution is directly correlated to difference of unit speed between the paddle and the ball.

There are no discernable patterns between the difference in unit speed and the generated float and integer values however a larger sample size may be more insightful in identifying a pattern.

A glaring issue with the nature of the problem is that the solution can be solved within the initial generation zero. However, these results have been omitted from the averages as these are not formed from the genetic algorithm and only occur due to the randomness applied to form the values for the genetic algorithm to initially work with.

Conclusion

The initial hypothesis of the correlation between average solve time and the difference in speed between the paddle of the ball was true as shown in the results section.

This solution solved a trivial problem in finding and optimizing a set of values to use within a paddle’s movement. However, upon reflection it may be intriguing to explore a genetic algorithm which has greater learning movement mechanics and does not lean on randomness to create a solution. Within this solution the paddle is already aware of how to react to the moving ball and only needs to produce a set of values which correctly adjusts its positional offset to collide with the ball. Although, for a stochastic search technique, this solution does meet its purpose.

GeeksforGeeks. (2022). *Genetic Algorithms.* Available: https://www.geeksforgeeks.org/genetic-algorithms/. Last accessed 28/03/2022.

Apar Garg. (2021). *Crossover Operators in Genetic Algorithm.* Available: [https://medium.com/geekculture/crossover-operators-in-ga-cffa77cdd0c8. Last accessed 28/03/2022](https://medium.com/geekculture/crossover-operators-in-ga-cffa77cdd0c8.%20Last%20accessed%2028/03/2022).

TutorialsPoint. (2021). *Genetic Algorithms - Mutation.* Available: https://www.tutorialspoint.com/genetic\_algorithms/genetic\_algorithms\_mutation.htm. Last accessed 28/03/2022.

GeeksforGeeks. (2019). *ML – Convergence of Genetic Algorithms.* Available: https://www.geeksforgeeks.org/ml-convergence-of-genetic-algorithms/. Last accessed 28/03/2022.