AI332

Computational Cognitive Science

Task1 Report

Written By:

Omnya Ahmed Mohamed [ID: 20210074]
Soha Ashraf Hassan [ID: 20210589]
Alaa Azazi Abd elhamed [ID:20210071]

Genetic algorithm

our genetic algorithm has 2 main classes:

in Chromosome class we have 3 functions, we initialize the weights and the score of the chromosome in the constructor first

- **calc_fitness function**, we set the score to be the 3rd element of the list game_state which has weights that represents the game state like:

- **calc_best_move function**, that takes the board status and the piece to play with and determines the best move for a given Tetris piece on a game board. It finds the best x-position, rotation, and y-position by simulating each possible move and calculating a movement score based on the chromosome's weights….

In Genetic class we have 6 methods , we initialize a list containing the chromosomes in the population and initialize the population with random chromosomes and evaluate their fitness.make function to print weights and score for each chromosome

- **selection function**: is to choose a subset of chromosomes from the current population for the next generation. By selectively favoring chromosomes with higher fitness scores, the method facilitates the evolutionary process by promoting the propagation of beneficial traits.

  take parameter: 1) list of chromosomes representing the current population.
  2) number specifying the number of chromosomes to be selected for the next generation.

  Then,extract the fitness scores of all chromosomes in the population.and normalize the fitness scores to obtain probabilities for each chromosome.

Generate a cumulative probability distribution (roulette wheel) based on the normalized fitness scores.
Randomly select chromosomes from the population based on their respective probabilities.then Continue selecting chromosomes until reach to number of selection.finally, return selected chromosomes as a list

- **crossover function**: is to generate new offspring by combining genetic material from parent chromosomes

  take parameter:1) a list of chromosomes representing the selected individuals from the current population
  2) value representing the probability of crossover occurring between two parent chromosomes.
  3) value indicating the probability of mutation occurring in the offspring chromosomes.

  Initialize a list to store the offspring chromosomes.then for each pair of parent chromosomes selected from selected population , determine whether crossover should occur based on the cross rate probability and If crossover is to be applied iterate over each gene in the chromosome and generate a random value a between 0 and 1 then apply arithmetic crossover to combine the genetic material from the parents by (for each gene, calculate the weighted average of the corresponding genes from the two parents,and update the genes of both parents with the calculated values)
  finally return resulting offspring chromosomes.

- **mutation function**: is to introduce diversity and exploration into the population of chromosomes by randomly altering individual genes within the chromosomes.

  take parameter:1) a list of chromosomes representing the individuals in the population
  2) value indicating the probability of mutation occurring for each gene in the chromosome.

For each chromosome in the population,iterate over each gene in the chromosome and generate a random number between 0 and 1 then If the generated random number is less than the mutation_rate, perform mutation on the current gene and replace the current gene with a new random value sampled from a predefined range.finally ,return a list of chromosomes after applying mutation

- **replace function**: is to update the population of chromosomes by replacing a subset of existing chromosomes with new ones.

  take parameter:1) a list of chromosomes representing the new individuals to be incorporated into the population

  doing sort the existing population of chromosomes based on their fitness scores in descending order then replace the last n chromosomes in the sorted population with the new chromosomes generated from genetic operations then make shuffle the entire population to introduce randomness and avoid bias towards specific individuals in subsequent generations.

Playing the game

First we set up the screen, The screen size for the game window is set to 560x360 pixels then :-

- The `run_tetris-game` function initializes, this function manages the game.

  It takes three parameters: chromosome(It contains weights) , speed (determines the game's speed), and max_score (sets the maximum score for the game) ew set up the max score to 200000.

  It updates the falling pieces, calculates scores, and manages game states.

  The score is calculated on this basis: 40 points for 1 line, 120 for 2 lines, 300 for 3 lines,1200 for 4 lines as shown in the game file attached to the project.

  The game runs until the game ends,the game ends when a new piece can not be placed at the top of the board because it's full or when the score exceeds the maximum score, it means that the player has achieved victory or completed the game.

  the user can quit the game so It handles user events such as quitting the game.

- **draw_terteis_game function:**

This function handles rendering the game elements on the screen.

It clears the screen, draws the game board, displays the current score and level, shows the next falling piece, and updates the display to reflect any changes.

The pygame.display.update() function is called to refresh the display, ensuring that any changes are visible to the player.

The main function

- **Parameters and general configurations:**

    - We chose the random seed to be 20
    - NUM_GENERATION = 350:
        Stands for the number of generations to evolve. This is the total number of iterations for which the genetic algorithm will run, with each generation representing one complete cycle of selection, crossover, mutation, and evaluation.

    - NUM_POPULATION = 15:
        the number of chromosomes in the population at the start of each generation.

    - GAP = 0.3:
        The proportion of the population that will be selected to create new offspring (children). This is used to calculate the number of children to generate in each generation.

    - NUM_CHILD = round(NUM_POP * GAP):
        The actual number of children to create in each generation. In this case, it's calculated by rounding NUM_POP * GAP. Given NUM_POP = 15 and GAP = 0.3, the number of children is 5.

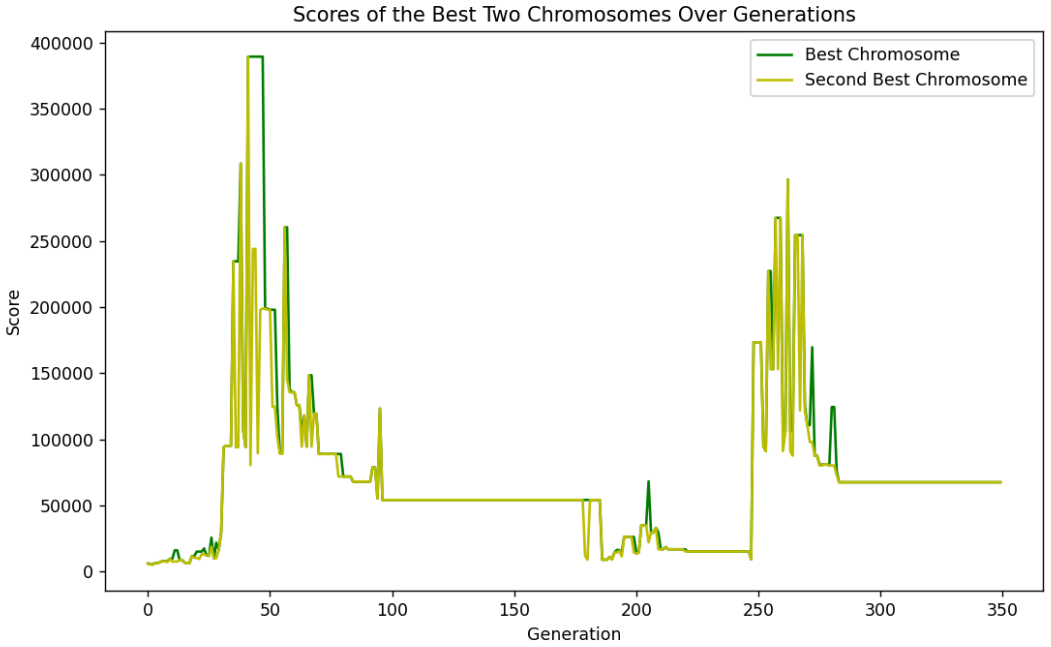
- **Training Process:**

    - The loop runs for NUM_GENERATION = 350 iterations.
    - Each iteration represents a generation and with each generation:
        - The current population is added to generations

- The chromosomes in the population are sorted based on their scores in descending order.
- The best and second-best scores in the population are recorded for getting their progress later
- Then the algorithm decides which chromosomes from the current generation will be carried over to the next generation without applying crossover or mutation (this helps getting high scores in the training process) so it takes the best three chromosomes.
- Parent chromosomes are selected from the population using a selection method (roulette wheel selection).
- Crossover and mutation are applied to create a new set of offspring chromosomes (new_chromo).
- The fitness of each offspring chromosome is evaluated by running the game and calculating the fitness.
- The new offspring chromosomes replace the old population.

After completing all generations, the results are saved in experiments.

- **The progress of the best two chromosomes:**



- **The Two best chromosomes and their scores:**

```
Best Score 1: 389455
Corresponding Weight 1: [-0.20041426 -0.60715724 -0.80930414 -0.36989255  0.68018247 -0.310204
  0.89338506]
Best Score 2: 308594
Corresponding Weight 2: [-0.17648189 -0.72643006 -0.81065326 -0.04991283  0.68003517 -0.31088426
  0.50498654]
```

- **The score of the optimal factor in the final test run:**

```
 Score in 15 iteration :  248079

********The game has reached the highest score
********
Max Score is :  248079

[Done] exited with code=0 in 3615.946 seconds
```