CS 421 Homework #4: Genetic Algorithm

Overview: Your assignment is to use a genetic algorithm to discover a good opening layout for a random Antics agent that is playing against Booger.

Specification: Start with the random agent provided with Antics. Modify the code for your agent so that the initial layout of the Antics constructions (anthill, tunnel, grass and enemy's food) is determined by a "gene" selected from a population. You can determine the fitness of a gene by having it player multiple games against the Booger agent. Once all the genes have been evaluated, you can "mate" the genes to create a new population. Over time, your layouts should improve.

Before you begin coding, careful design the mapping of layout and gene (and vice versa). Try to design a gene so that the standard mating and mutation process won't create a gene that defines an invalid layout.

The following elements are required:

1. Three instance variables:
   a. A list of lists to store your current population of genes. This should probably be an instance variable.
   b. An index to track which gene in the population is next to be evaluated
   c. A second list to store the fitness of each gene in the current population

   You are likely to need several other instance variables. These are just the required ones.

2. A method to initialize the population of genes with random values and reset the fitness list to default values.

3. A method to take two parent genes and generate two child genes that result from the pairing. This mating should include a chance of mutation. *Note:* If you use a for-loop for this routine, you are forgetting about slices in Python. Read up on them and do this without a loop declaration.

4. A method to generate the next generation of genes from the old one. The details of selecting the fittest parents for mating is up to you.

5. The `getPlacement()` method should contain the code that uses the current to-be-evaluated gene to define the layout it returns to the game.

6. Override the `registerWin()` method you inherited from `Player.py` and use it to manage the population. Each time a game ends, the following actions should be performed:
   a. Update the fitness score of the current gene depending on how well it performed
   b. Judge whether the current gene's fitness has been fully evaluated. If so, advance to the next gene.
   c. If all the genes have been full evaluated, create a new population using the fittest ones and reset the current index to the beginning.

Hints:

1. Try to design your gene so child genes so they don't need adjustment after crossover and mutation.
2. Use a unit test to verify each method as you write it. If you come to me for help with a bug in this assignment, my first answer will likely be "have you tried using a unit test?"
3. To ease debugging, start with a very small population size and use only one game to evaluate the fitness of a gene. Then increase to a larger size once everything is working

Once you are certain that your genetic algorithm is working, create an evidence file by following these steps:

1. Modify the agent to print out the layout defined by the gene with the highest fitness. It should do this at the end of each generation. Use the `asciiPrintState()` method provided in AIPlayerUtils.py for this so I can easily read your output. Your agent should have no other output.
2. Run your program with the program output piped to a file. This will be your evidence file.
3. Train the agent for at least 20 generations by running a very long tournament. The number of games you need to play is:
   (population size) • (# of games to evaluate each gene) • (number of generations)
   Beware: This is likely to take several hours and you may find that you have to debug it and try again if no learning is apparent. Plan accordingly so you do not miss the due date for this assignment.
4. Rename your evidence file so that the filename contains the UP usernames of all team members.
5. Compress your evidence file into a .zip file with the same filename but .zip extension.

Report

Complete an informal homework report as described in homework #1.

Turning In Your Assignment: Follow these steps:

1. Turn in your agent's source code by following the steps defined in homework #1.
2. Turn in the compressed evidence file (.zip) via the second Turn it in Here link that is provided.

Grading Guidelines

These are the same as homework #1.