Homework #7
Search Tool (Version 3)

AI Programming

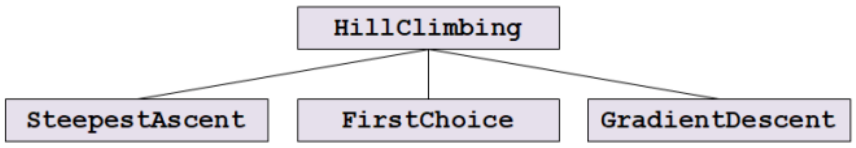
Overview

In this assignment, you are asked to have all your main programs united into a single program by defining and using a new class named 'HillClimbing' in which you can put together all the search algorithms. To have a single main program, you need a new user interface to query the user the type of problem to be solved and the type of algorithm to be used. A typical outcome of numerical optimization is shown below.

```
Select the problem type:
  1.  Numerical Optimization
  2.  TSP
Enter the number: 1
Enter the file name of a function: problem/Griewank.txt

Select the search algorithm:
  1. Steepest-Ascent
  2. First-Choice
  3. Gradient Descent
Enter the number: 3

Objective function:
1 + (x1 ** 2 + x2 ** 2 + x3 ** 2+ x4 ** 2 + x5 ** 2) / 4000 -
math.cos(x1) * math.cos(x2 / math.sqrt(2)) * math.cos(x3 / math.sqrt(3)) *
math.cos(x4 / math.sqrt(4)) * math.cos(x5 / math.sqrt(5))

Search space:
 x1: (-30.0, 30.0)
 x2: (-30.0, 30.0)
 x3: (-30.0, 30.0)
 x4: (-30.0, 30.0)
 x5: (-30.0, 30.0)

Search Algorithm: Gradient Descent

Update rate: 0.01
Increment for calculating derivatives: 0.0001

Solution found:
(25.12, -8.877, 10.866, -18.812, 21.022)
Minimum value: 0.407

Total number of evaluations: 34,219
```

Defining 'HillClimbing' Class

You can let the search algorithms become subclasses under the 'HillClimbing' class, where the body of each search algorithm becomes the body of the 'run' method of the corresponding subclass. The 'displaySetting' function that was previously part of each main program should be moved to the 'HillClimbing' class because what it displays are about the settings of the search algorithms that are now the ‘run’ methods of ‘HillClimbing’. You should figure out how to distribute the codes of ‘displaySetting’ within the class hierarchy considering the mechanism of inheritance. Codes for printing information common to all the algorithms should be placed at the base class, while those for printing the algorithm name and additional setting information specific to that algorithm should be placed at the respective subclasses.



‘HillClimbing’ must have variables to store information necessary for ‘displaySetting’ and the search algorithms. For example, you may need a variable named ‘limitStuck’ to take the role of the previous named constant LIMIT_STUCK, because ‘FirstChoice’ is under the control of this value.

You are recommended to store the class hierarchy of ‘HillClimbing’ in a separate file named ‘optimizer.py’.

Adding a Superclass ‘Setup’

Having defined the ‘HillClimbing’ class in addition to the already existing ‘Problem’ class, you should be aware of that there are some information needed by both classes. The mutation step size, for example, is needed by not only the problem-specific methods of the ‘Problem’ class but also the ‘displaySetting’ method of the ‘HillClimbing’ class. Therefore, it is recommended that you define a superclass named ‘Setup’ as a parent of both ‘Problem’ and ‘HillClimbing’, so that such shared information can be stored in the variables of ‘Setup’ and let them be inherited to the subclasses.

You are recommended to store the ‘Setup’ class in a separate file named ‘setup.py’.