CS512– Artificial Intelligence

Instructor : Shashi Shekhar Jha (shashi@iitrpr.ac.in)

Assignment - 2 | Due on 16/03/2020 2400 Hrs (70 Marks)

Submission Instructions:

All submission is through google classroom in one zip file. In case you face any trouble with the submission, please contact the TAs:

- Armaan Garg, 2019CSZ0002@iitrpr.ac.in
- Rahul Kumar Rai, 2018csz0004@iitrpr.ac.in

Your submission must be your original work. Do not indulge in any kind of plagiarism or copying. Abide by the honour and integrity code to do your assignment.

As mentioned in the class, late submissions will attract penalties.

Penalty Policy: There will be a penalty of 20% for every 24 Hr delay in the submission. E.g. For the 1st 24 Hr delay the penalty will be 20%, for submission with a delay of >24 Hr and < 48 Hr, the penalty will be 40% and so on.

You submission must include:

- A legible PDF document with all your answers to the assignment problems, stating the reasoning and output.
- A folder named 'code' containing the scripts for the assignment along with the other necessary files to run yourcode.
- A README file explaining how to execute your code.

Naming Convention:

Name the ZIP file submission as follows:
Name_rollnumber_Assignmentnumber.zip
E.g. if your name is ABC, roll number is 2017csx1234 and submission is for lab1 then you should name the zip file as: ABC_2017csx1234_lab1.zip

Assignment

This assignment is divided into two sections. First section tests your mettle on local search techniques while the second section of the assignment brings out your AI ingenuity with adversarial search on developing AI game agents.

Trivia:

- **All technical instructions is based on UBUNTU OS**
- **Work on your assignment with Python3**

SECTION 1: Local Search

As we discussed in the lectures, local search is a heuristic method for solving computationally hard optimization problems. Local search can be used on problems that can be formulated as finding a solution maximizing a criterion among a number of candidate solutions. Local search algorithms move from solution to solution in the space of candidate solutions (the search space) by applying local changes, until a solution deemed optimal is found or the time bound has elapsed.

Problem: The N queen puzzle

The **8-queens puzzle** is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens cross each other, thus, a solution requires that no two queens share the same row, column, or diagonal. The 8-queens puzzle is an example of the more general ***n*-queens puzzel** of placing *n* non-attacking queens on an *n*×*n* chessboard, for which solutions exist for all natural numbers *n* with the exception of *n* = 2 and *n* = 3.

For section 1, download the template code provided alongside the assignment document.

Files you'll edit:	
simulated_annealing.py	To implement the code for local search based on simulated annealing.
hill_climbing.py	To implement the code for local search based on hill climbing.
GA.py	To implement the code for local search based on GA.
Files you might want to look at:	
NQueens.py	The main file that defines the search space and the state space for n queen problem setup.
TestLocalSearch.py	The file that is used to run the local search algorithms. This file is also responsible for building the board and printing the position of the queens based on the choosed algorithms.

Files to Edit and Submit: You will fill in portions of simulated_annealing.py, hill_climbing and GA.py during the assignment. You should submit this file with your code and comments. Please *do not* change the other files in this distribution or submit any of our original files other than these files.

Evaluation:
Please *do not* change the names of any provided functions or classes within the code, or you will wreak havoc on the evaluation system. However, the correctness of your implementation -- will be the final judge of your score. We will review and grade assignments individually to ensure that you receive due credit for your work.

[PART 1] Simulated Annealing…….....**[10 marks]**

- Implement the algorithm.
- Choose five different temperature schedule and observe the change in heuristic value (number of unattaching queens) with in every iteration of your search for all five temperature schedules

[PART 2] Hill climbing……**[10 marks]**

- Implement the algorithm.
- Take five different starting points and observe the ascent of the heuristic value (#non attacking queens) with each iteration of your search

[PART 3] GA……...…….**[15 marks]**

- Implement the algorithm.
- Take 5 different values for the hyperparameters viz. population size, crossover probability and mutation probability and observe the changes in the best fitness value in the population across different generations for each combination of population size, crossover and mutation probability.

NOTE: Write all your observations supplementing them with graphs in the PDF file. Also, compare the results across the three algorithms based on number of iterations to get the best solution.

SECTION 2 : Adversarial Search

In **Adversarial search**, we focus on finding a policy for an agent which is competing with other agents for supremacy in an adversarial multi-agent environment .

Examples: Chess, business, trading, war.

Opponents would change the next state in a way i.e.:

1. unpredictable
2. *hostile* to you

You only get to change (say) every alternate state.
Problem: Create a 4x4 **tic-tac-toe** game agent where the agent employs minimax search to generate the policies to choose its next move.

What is Minimax?

Being an AI genious, you must know it well. For any confusion, refer to the lecture slides.

How does it work?

The algorithm searches, recursively, the best move that leads the *one of the* players to win or not lose (draw). It considers the current state of the game and the available moves at that state, then for each valid move it plays (alternating *min* and *max*) until it finds a terminal state (win, draw or lose).

Coding 4x4 tic-tac-toe game agent

Remember, you will be competing in a 4x4 grid (not 3x3) for the tic-tac-toe game.

Create the game agent wherein you pitch two game agents against each other and observe how they perform. You must put a time limit for choosing your moves (use some clever heuristics).

You have to create different versions of your game agent, starting with the basic implementation using minimax search, then applying alpha-beta pruning recording and finally the heuristics you may have used to improve the performance of your game agents. Record all the observations about the effects of pruning and heuristics. You can do that by calculating the effective branching factor.

How about pitching your game agent against that of your friends in the class?

Can your game agent beat a human player? Pitch your game agent for a tournament of 5 games against a human player (choose amongst your friends/colleagues). Record the wins/losses and number of moves played in each game.

[PART 1] Basic Implementation with minimax..…….**[10 marks]**

[PART 2] Alpha beta pruning ...……...**[10 marks]**

[PART 3] Use Heuristics and time limited decision making……...…...**[15 marks]**

Happy Spring Break !!!