COMP329 ARTIFICIAL INTELLIGENCE
Assignment 2 (Weight: 20%)

Evolutionary Algorithms for Adversarial Game Playing

Due: 11:55 am, Nov 06, 2017 (Monday, Week 13)

You should upload soft copy of your solution on iLearn in the form of four files. *Make sure that you submit the right file at the right site.*

1. *A pdf file to be submitted as a Turn-It-In assignment (Assignment 2).* This file should be named `<FirstName_LastName>.A2Report.pdf`. For instance, if your name is *John Smith*, it should be named `John_Smith.A2Report.pdf`. This report should contain your response to the tasks of the assignment as specified. This document *must be in the* `.pdf` *format*, and should incorporate the appropriate cover page. Make sure that the document is meant to be printed on A4 size paper with adequate margin all all four sides.

2. The following three Python programs should be submitted at the Assignment 2 auxiliary site (Assignment 2 aux). These files will be used for testing purpose. *They will also be used to determine any bonus marks (up to 2 marks) that you might earn.*

   (a) `<FirstName_LastName>.GA.py` containing the Python program for generating the strategy for playing Iterated Prisoner's Dilemma as a "chromosome" such as a bit-string.

   (b) `<FirstName_LastName>.IPD.py` containing the Python implementation of the strategy that your genetic algorithm generated and selected. This implementation must be "Axelrod-compatible" in the sense that it should compile and run when played in a match/tournament using the Axelrod package (see the Prac for Week 10 for example).

   (c) `<FirstName_LastName>.IPD_test.py` containing the Python program that you used to test (that the strategy you implemented indeed runs as expected), and analyse the strategy.

Late submissions will not be accepted after the submission deadline. No request for extension will be considered unless request for special consideration is formally submitted online.

What this assignment is about?

We know what *Prisoner's Dilemma (PD)* and *Iterated Prisoner's Dilemma (IPD)* are. There are a number of strategies for playing *Iterated Prisoner's Dilemma* such as *Tit Fot Tat*. You can find Python implementation of a large number of such strategies at:
`https://axelrod.readthedocs.io/en/stable/reference/`
`all_strategies.html`.
Most such strategies have been invented by researchers and other enthusiasts. Indeed, as

part of Pracs in Week 10 you have written such a strategy and played it against a couple of strategies available in the Axelrod strategy library. We want to see whether (and how) Genetic Algorithm can be employed to generate such strategies.

To be of any interest, a strategy for IPD must be able to look back and see what the opponent played in the previous move(s). For instance, *Tit Fot Tat* looks back just at the opponent's move made in the previous *turn* and decides what move to make. It has a *memory depth* of 1. The strategy you wrote in Week 10 Pracs has $memory\_depth = 2$. We require that the strategy you will generate and submit **will have** $memory\_depth \geq 2$. Furthermore, the program should be flexible enough so that the we can easily generate strategies of higher or lower memory depths, i.e. the memory depth should not be hard-coded, but be used as a parameter whose value can be easily modified.

The Genetic Algorithm you employed will generate the "fittest strategy" for playing IPD. But this algorithm would have the form of a "chromosome" (such as a bit-string), and is not easily "understood" by us. To better appreciate (and analyse) it, we would like this strategy to be (manually) translated to a Python program. And if it is done the right way, we can play it against other available strategies for IPD, and analyse its nature better.

It is expected that it will be good learning experience for you, and in the process you will gain a better appreciation of evolutionary algorithms and adversarial game playing.

Before you start coding

It is best you think about the task at hand before starting coding. In particular, you need to think about the representation issue: the "structure of the chromosome" that will represent your solution strategies for IPD. You might want to take a concrete strategy, say *Tit Fot Tat*, and think about how you will represent it as a bit-string. Note that *Tit Fot Tat* has *cooperate* as its first move, and also it can check the opponent's last move. What kind of representation will facilitate such "functionalities"? Once you know how to represent *Tit Fot Tat*, you would have a very good idea of how to represent strategies for IPD in general. Feel free to consult material available online (but don't forget to acknowledge them.) It is likely that you will need to use the payoff matrix for Prisoner's Dilemma. Use the matrix provided in Table 1 for consistency with the Axelrod convention.

Table 1: *This payoff matrix for Prisoner's Dilemma should be used.*

	D	C
D	(1, 1)	(5, 0)
C	(0, 5)	(3, 3)

The next thing you will need to think about the *fitness function*. As we know, the representation issue and the fitness function are inter-linked. Think about how you are going to evaluate the strategies generated. Also, think about how you would decide when you think the strategy (solution) you have gotten is of a "good quality".

We have used the DEAP package earlier. You can use this package to implement the genetic algorithm you have in mind and generate the strategies (in chromosomal form). Alternatively, you can implement the genetic algorithm independent of DEAP if you feel adventurous enough.

If you have carefully thought about the representation issue, it will not be difficult (although tedious) to interpret the "fittest candidate" found. Translating it to Python code of the desired specification should not be very difficult. But make sure the code you write is easily readable. ((Think carefully and try to avoid avoidable "cases" in a very long *if-elif-else* statement.)

The Code

At the top of each file, as comments, write your name (as author) and give any acknowledgment due. Make the codes flexible and readable. Give adequate (but not excessive) comments.

Report structure and Preparation

The report should function as a reflective statement of what you have done as part of this assignment, why, and how. Sufficient information should be available in this report to assess your complete work for this assignment. (The Python files will be used to test/verify your claims.)

The report will have a main "body" and a number of appendices as described below. Try to mimic the format (font size, font style, margin size, spacing, inter-line space, etc) of *this document* (the assignment specification). The "body" of the report should not exceed five pages in this format.

It is being noted that the marks allocated to different sections (indicated below) total to 18 marks. The missing 2 marks are reserved for clarity of exposition (1 mark) and aesthetically pleasing formatting (1 mark). You may earn up to 2 bonus marks based on the performance of the strategy you submit.

Introduction [1 mark]

Describe the problem. In the process describe Genetic Algorithm, Prisoner's Dilemma, Iterated Prisoner's Dilemma, and to some extent their significance. Refer to any work in the area that you have consulted, for instance, [2] and [4] are citations to two books; [3] is a citation to a journal article, and [1] is one to a paper in a conference proceedings. Briefly outline what you did, and what result you obtained.

Genetic Algorithm [6 marks]

In this section you describe what you did with respect to generating the strategies in "chromosomal format". In particular, in separate subsections, describe:

1. [2.5 marks]
   How you approached the problem of representing the solution (strategy) space, and why. Did you face any specific problem for this – for instance, choosing between different possible representations? How did you overcome it/them? Give as example how, in your approach, *Tit For Tat* will be represented, and explain clearly how one can read off the *Tit For Tat* strategy from this "chromosome".

2. [2.5 marks]
   What fitness function you chose to go with your approach, and why. As before, describe if you faced any specific problem for this, and you overcame it/them.

3. [1 mark]
   What parameters for Genetic algorithm did you choose? For instance what type of crossover did you choose, with what probability, and why. Also outline how you determined that the solution you received is good enough.

Iterated Prisoner's Dilemma [5 marks]

In this section you describe:

1. [2 marks]
   The strategy generated and chosen via genetic algorithm is to be interpreted (and understood). You might want to display the "chromosome" in a more visually pleasing form (say as a number of blocks) instead of a long chain of 0's and 1's, and explain what each block roughly signifies.

2. [2 marks]
   Briefly describe how you (manually) translated the chromosome at hand to friendly Python code in an Axelrod-compliant manner. Describe (in **English**) this strategy in a simple way.

3. [1 mark]
   Briefly describe the IPD tournament (using the Axelrod package) you ran, what were the other strategies in the tournament, and how your strategy performed. Do a bit of analysis, as to its strengths and weaknesses.

Concluding Remarks [1 mark]

Now briefly outline what went right and what could have been improved. If you have some idea, indicate how things could have been improved.

Appendices [5 marks]

The marks for these tasks assume that the codes you submit compile and run in the expected manner. If any one of your `IPD.py` or `GA.py` programs fail to compile (or is a dummy program), the maximum mark from this section you can expect is 3. If both of the `IPD.py` and `GA.py` programs fail to compile (or is a dummy program), you will be awarded 0 for this section.

Appendix A [2 marks]
Your genetic Algorithm Python file (`<FirstName_LastName>.GA.py`) goes here. (You would have copied your code to the Word file – or other source file – before generating the pdf.)

Appendix B [2 marks]
Your Python file `<FirstName_LastName>.IPD.py`, that implements the code for playing IPD, goes here. (As before, copy it to Word file before generating pdf.)

Appendix C [0.5 mark]
Your Python file `<FirstName_LastName>.IPD_test.py` goes here. (As before, copy it to Word . . .)

Appendix D [0.5 mark]
Include relevant test results here. Have screen capture of relevant outputs from running `<FirstName_LastName>.GA.py`, and `<FirstName_LastName>.IPD_test.py`. In particular you should include screen captures showing:

1. The bit-string that you translated into the Python-strategy `IPD.py` indeed is one that was generated and chosen by your `GA.py` program, and
2. The tournament/match result of your running your `IPD.py` strategy.

References

[1] Robert Axelrod. The future of cooperation. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2004, 19-23 June 2004, Portland, OR, USA*, 2004.

[2] R Dawkins. Oxford University Press, Oxford, UK, 1976.

[3] Piotr Faliszewski, Jakub Sawicki, Robert Schaefer, and Maciej Smolka. Multiwinner voting in genetic algorithms. *IEEE Intelligent Systems*, 32(1):40–48, 2017.

[4] Kevin Leyton-Brown and Yoav Shoham. *Essentials of Game Theory: A Concise Multidisciplinary Introduction*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2008.