Module Code
COM00037H

BEng, BSc Degree Examinations 2023–24

DEPARTMENT OF COMPUTER SCIENCE

Evolutionary & Adaptive Computing

Open Individual Assessment

Issued: 12 noon on 22 February 2024
Submission due: 12 noon on 23 May 2024
Feedback and marks due: 27 June 2024

All students should submit their answers through the electronic submission system: http://www.cs.york.ac.uk/student/assessment/submit/ by 12 noon on 23 May 2024. An assessment that has been submitted after this deadline will be marked initially as if it had been handed in on time, but the Board of Examiners will normally apply a lateness penalty.

Your attention is drawn to the section about Academic Misconduct in your Departmental Handbook: https://www.cs.york.ac.uk/student/handbook/.

Any queries on this assessment should be addressed by email to Simon O'Keefe and Dimitar Kazakov at simon.okeefe@york.ac.uk or dimitar.kazakov@york.ac.uk. Answers that apply to all students will be posted on the VLE.

Your exam number should be at the top of your Jupyter or Google Colab notebook and the corresponding PDF printout, and at the start of the Info section of the NetLogo file. You should not be otherwise identified anywhere on your submission.

Rubric

Your submission should be a single zip file named after your exam number, `Yxxxxxxx.zip`, which contains:

- For Part 1 of this assessment
  - a single Jupyter or Colab notebook `EVAC1.ipynb` combining code and explanations
  - a PDF `EVAC1.pdf` of the state of the same notebook after all of its code has been executed.
- For Part 2 of this assessment
  - a pair of NetLogo files `EVAC2-train.nlogo` and `EVAC2-test.nlogo`, possibly accompanied by additional image and data files. The first of these files should allow to run the evolutionary algorithm (with a relevant description in its Info section). The second file should contain an already evolved behaviour for every dog, and allow for the performance of the evolved herding dogs to be tested. It should also contain an Info section with a report on the results and all remaining comments.

If there are discrepancies between the contents and output of the Jupyter/Colab notebook and what is shown in the PDF, the former will be used for marking.

Do not use archive formats other than zip. **Your submission MUST be a single zip archive**. Your code should assume any auxiliary files that you may generate are in the same folder as the notebook from which they are accessed.

1 Evolution of a function (this part is worth 50 marks)

1.1 Scenario

The scenario for this part of the assessment is the approximation of the function to predict the number of e-scooters that will be hired given a particular date, hour, holiday and weather information. The dataset contains a count of e-scooters rented per hour, with the corresponding weather data and holiday information.

The data for this assessment may be found on the Assessment page of the module VLE site.

1.2 Task

This assessment requires you to use an evolutionary algorithm to evolve a function that will generate a prediction of the number of e-scooters required, given a time, day and weather information. The evolutionary algorithm may be **EITHER** Genetic Programming **OR** a Genetic Algorithm evolving a neural network.

You must follow these rules of implementation:

(a) You **MUST** use Python and the DEAP library to implement an evolutionary algorithm.

(b) You **MUST** use either a Genetic Programming approach or a neural network representation.

(c) You **MUST** submit a Colab or Jupyter notebook containing runnable code for your evolved solution.

(d) You **MUST** evaluate thoroughly the performance of the evolved solution in terms of accuracy, precision, and so on as appropriate to the problem.

1.3 Data

For this assessment, you will use the data in the file eScooterDemand.csv, a file that contains the following variables:

1. Date : day/month/year
2. Count - Count of e-scooters rented

3. Hour - Hour of the day
4. Temp - Temperature in Celsius
5. Humidity - %
6. Windspeed - m/s
7. Visibility - 10m
8. Dew point - Celsius
9. Sunshine - MJ/m2
10. Rain - mm
11. Snow - cm
12. Seasons - Winter, Spring, Summer, Autumn
13. Public Holiday - Yes/No (Public Holiday or not)
14. HireAvailable - Yes/No (hire scheme operating)

1.4 To do

You should report on this part of the assignment by adding comments to the Colab/Jupyter notebook you have created.

The marking criteria assume there is working code. Partial marks may be allocated for design alone. No results will be accepted if the corresponding parts of the code involved in their generation cannot be executed.

1. Description of the algorithm implemented [15 marks]
   Give a full description of the chosen representation for individuals, and of the algorithm that you finally have chosen. You should give a clear justification for all the design choices made.
2. Quality of code [5 marks]
   You must provide runnable code and a solution for the evolutionary algorithm using Python and DEAP. Your code should be well commented, and demonstrate the competent use of features of DEAP.
3. Investigation of parameters and representation [10 marks]
   You should make a systematic investigation of the effect of parameters and of the

choice of representation in order to improve evolution of the solution. Where you have tested ideas experimentally you should present statistics to support decisions about parameters.

4. Evaluation of your solution [20 marks]
   You should evaluate your algorithm or algorithms thoroughly. You should present appropriate summary statistics and plots in your notebook to support statements about the effectiveness of the solution, and give appropriate interpretation and discussion of results.

1.5 Marking criteria

Q	Marks	Sophisticated	Satisfactory	Needs work
1.1	15	Description of the algorithm implemented		
		15 - 11	10 - 6	5 - 0
		Algorithm design is very clearly explained. Justification for all design choices is robust. Choice of representation is clearly motivated.	Explanation of the algorithm is mostly clear. Use of evolution is appropriate. Representation is clear and well justified. Overall design choices justified.	Algorithm is unclear or does not use evolution, or uses it inappropriately. Representation is unclear. Little or no justification for design choices.
1.2	5	Quality of code		
		5 - 4	3 - 2	1 - 0
		Evidence of runnable code, with competent use of a range of DEAP features. Implementation is very clear, with good use of comments to explain what has been done.	Evidence of runnable code, with straightforward use of DEAP implementation is mostly clear, with use of comments to explain what has been done.	No code, or code not runnable. Code does not use DEAP for evolution. Code not documented. Code runs but does not appear to evolve the solution.
1.3	10	Investigation of parameters and representation		
		10 - 7	6 - 4	3 - 0
		Clear investigation of the effect of parameters and the choice of representation, clearly linked to principles taught across the module (or beyond).	Some investigation of how to improve the solution Some lack of clarity on what has been done or the degree of improvement Not clearly linked to what has been taught.	Basic algorithm. No evidence of investigation of parameters or the representation to improve performance.
1.4	20	Evaluation of solution		
		20 - 14	13 - 8	7 - 0
		Thorough evaluation of what has been attempted. Clear presentation of statistics supporting insightful interpretation and discussion of results.	At least partial evaluation, although it may be more thorough in some aspects than others. Some presentation of statistics supporting interpretation and discussion of results.	Little or no attempt to evaluate what has been done. No presentation of statistics. No or little interpretation or discussion.

2 Evolving Herding Dogs (this part is worth 50 marks)

2.1 Scenario

You need to implement a simple multi-agent environment with two types of agents, sheep (plural) and dogs. All agents are placed on a square, two-dimensional, wrapped around an orthogonal grid made of $N \times N$ patches. At the beginning of each run, S sheep and D dogs are placed in the environment in a (pseudo-)random way. Both sheep and dogs can choose from five possible actions: moving in one of the four cardinal directions (North, South, West, East) or staying in the same square. The environment is updated in rounds. All agents are prompted in turn to plan their next move, after which all moves are carried out simultaneously. There is no limit on the number of agents that can reside in the same patch.

Sheep will be using a simple, fixed behaviour, choosing exactly one of the following actions, in order of preference (from highest to lowest):

1. If there is a dog in your current patch, move, if possible, to a patch without a dog;
2. If there is a dog in any of the four adjacent patches (i.e. North, South, East or West of the current one), move, if possible, to an adjacent patch that does not contain a dog;
3. Move to a patch with no sheep, but which is adjacent to a patch with sheep;
4. Move to an adjacent patch containing fewer sheep than the current patch;
5. Make a stochastic choice of action as follows: choose the same action as the last one with 50% probability, or choose one of the remaining four actions, each with 12.5% probability. For the first move, assume for all sheep that their previous move was to stay put.

The initial behaviour for all dogs is to choose one of the five possible actions with the same 20% probability. Dogs can see the position (coordinates of the current patch) of all other dogs and all sheep, and they can tell which species each agent belongs to.

2.2 Task

On the most general level, the goal here is to evolve a team ("pack") of herding dogs, which would constantly strive to keep all sheep in as tight a formation as possible. Numerically, we can say the dogs will be aiming to minimise the score defined as the sum of the variances of X and Y coordinates for all sheep:

$$Score = \frac{\sum_{i=1}^{S}(x_i - \bar{x})^2 + \sum_{i=1}^{S}(y_i - \bar{y})^2}{S}$$

Design, implement, describe and evaluate a procedure that employs evolution to produce herding behaviour for the dogs that outperforms as best as possible their default behaviour. Each of the dogs should be able to evolve a separate behaviour, which is potentially different from the rest. The size of the environment can be changed at will for evolution purposes, but the resulting behaviour should be tested for 50 sheep, 5 dogs, and an environment of size $N = 49$, i.e. where patches have coordinates ranging from (-24,-24) to (24,24).

2.3 To do

1. Representation of dogs' behaviour [5 marks]
   Choose a simple and efficient representation of your dogs' behaviour that would also allow for adaptation. Describe the chosen representation and explain the reasons behind it.
2. Implementation of default behaviour and fitness estimation [10 marks]
   Provide the necessary, working code implementing the simulation where your chosen representation of dogs' behaviour is set to their default behaviour, then describe how running the simulation is going to provide data for estimating the fitness of your dogs.
3. Design and implementation of adaptation [20 marks]
   Describe the design of (10 marks), and implement (10 marks) a procedure that uses adaptation to optimise the behaviour of your agents.
4. Design of evaluation procedure [10 marks]
   Design and describe an evaluation procedure that allows you to compare the behaviour obtained through adaptation to the initial, non-adaptive behaviour, and draw conclusions that are grounded on sound statistical arguments.
5. Experimental evaluation [5 marks]
   Collect experimental evidence, carry out, and show the results of the evaluation procedure described above.

2.4 Marking Criteria

1. Three marks for a rational encoding making use of the most relevant information, with up to another 2 marks for a well-argued effort to find a good trade-off in the choice of representation that reduces the search space of distinct behaviours with as little loss of expected performance as possible.

2. Up to 5 marks for a running, easy to use, and well-described implementation of the simulator, and up to 5 marks for the design of a setup that evaluates the dogs' fitness.

3. A maximum of 10+10 marks for a well-implemented, described and argued procedure that provides at least the functionality of a generic GA (which would score 4+4 marks max), explores systematically the meta-parameter space to achieve best performance (2+2 marks), and implements at least 3 of the following: fitness scaling or equivalent (e.g. ranking), elitism, niching/crowding, adaptive mutation rates, fitness sharing (up to 4+4 marks).

4. Four marks for a solution that includes an unbiased exploration of the space of initial configurations, six marks for a sound choice and detailed description of a statistical tool/mechanism for the comparison (e.g. choosing the most appropriate statistical test, well argued for).

5. Full marks for a substantial range of experiments, well presented and summarised through the evaluation procedure.

All of the above assumes there is working code implementing the individual objectives. Partial marks may be allocated for design alone. No results will be accepted, if the corresponding parts of the code involved in their generation cannot be executed.

The criteria to be used in the marking also include clarity, simplicity, generality and rigour of the methods chosen, as well as the ability to describe, analyse and visualise experimental results in an effective way.

End of examination paper