ECM2423 - Coursework exercise

Deadline: 17th March 2022 12:00 (midday, Exeter time)

Lecturer: Dr. Ayah Helal (a.helal@exeter.ac.uk)

This continuous assessment (CA) is worth 40% of the overall module assessment. This is an individual exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

Coursework submission The submission of this coursework is via **eBART**. The code needed to answer the questions must be written in Python and must run without errors. Make sure that your code is clear, commented, and that your answers are clearly labelled.

In **eBART**, you should submit a **single compressed (zipped) folder** which contains all the relevant files for this coursework exercise. The compressed folder can contain **either** a PDF file with the answers and your Python scripts, or a Jupyter notebook containing both code and answers if that is what you prefer using.

Marking criteria The two questions you will find below account for 75 marks. Each of the questions is further divided in sub-questions to guide your answers, and specific marks are provided for each of the sub-questions. Additionally, for each sub-question I have clearly defined what the deliverable should be, so that it is clear what you should produce for your answer. A further 25 marks will be awarded according to the following three criteria:

1. code documentation and comments, with all the relevant information needed on how to run the code (including the names of scripts so that they can be clearly linked to a specific question): **5 marks**
2. clear presentation of your answers: **5 marks**
3. additional excellence, for example, further experimentation, depth of analysis or discussion, or any relevant additional figure: **15 marks**.

The sum of the marks that you can obtain in this coursework exercise cannot exceed 100. In general, all questions will be marked according to the following criteria:

- For questions which require code (question 1.2 part 3; question 1.3; question 2.1. part 2): have you correctly implemented the algorithm? Does your implementation solve the problem as required in the question? Does your code run without errors? Is the code clearly structured?
- For questions requiring a written answer (question 1.1; question 1.2 part 1, 2 and 4; question 1.3; question 2.1; question 2.2): is your answer clear, coherent and well-structured? Does it give the correct answer to the correct question? If applicable, have you made the most of possible figures and plots to support your answers, and are those figures and plots appropriately designed and clear? Is your answer succinct and does it only contain information relevant to the question?

Question 1 | Implement a heuristic search algorithm: $A^\star$ and the 8-puzzle game.

The 8-puzzle consists of a 3 x 3 board. Eight tiles in the board are numbered from 1 to 8, and one tile is blank. Any tile adjacent to the blank space can slide into that space. The goal of the game is to reach a given goal configuration from the given initial state. Example initial and goal configurations are given in Figure 1. In this question, you will be asked to phrase the 8-puzzle problem as a heuristic search and to implement the $A^\star$ algorithm to solve it.

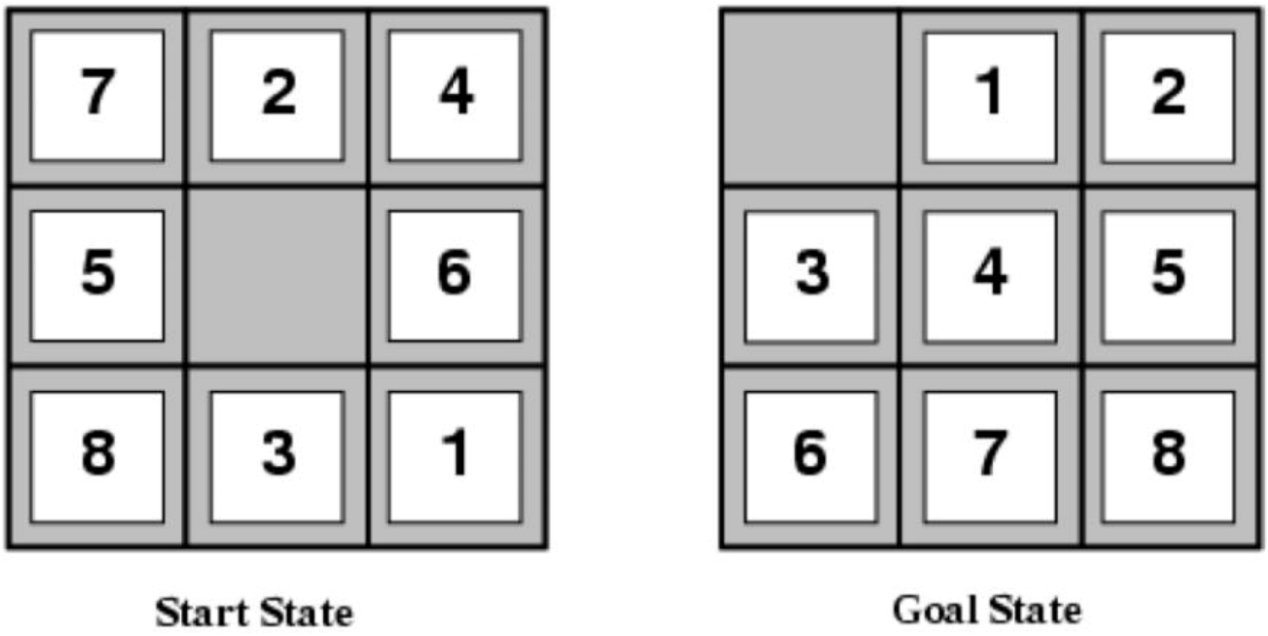

Figure 1: Examples of an initial and goal configuration of the 8-puzzle. Figure taken from Russell, S.J. and Norvig, P., 2016. Artificial intelligence: a modern approach

Question 1.1: Describe how you would frame the 8-puzzle problem as a search problem.

In your own words, write a short description of how the 8-puzzle problem can be seen as a search problem.

(5 marks) Deliverable: written answer.

Question 1.2: Solve the 8-puzzle problem using $A^\star$.

This question is further divided in the four parts described next.

1. In this question, you should first briefly outline the $A^\star$ algorithm.

   (5 marks) Deliverable: written answer.

2. Describe **two** admissible heuristic functions for the 8-puzzle problem and explain why they are admissible. Make sure you also explain why you chose these two heuristic functions in particular amongst all the possible ones.

   (5 marks) Deliverable: written answer.

3. Then, you should implement **two** versions of the $A^\star$ algorithm in Python to solve the 8-puzzle using the two heuristic functions you have chosen. You can either implement the two versions in the same Python script, letting the user select which one to use before running the code, or you can have two different scripts if you prefer. To test that it works, you can use the start and goal state of Figure 1 (however, note that this may take a few minutes to run depending on your computer, implementation and choice of

heuristic functions), or you can specify your own initial and goal state. If you specify your own initial and goal state, select states which are **at least** five moves apart from each other and **write** these states in your report. You will **not** be penalised if you do not use the start and goal configurations of Figure 1, but you will not receive full marks if the configurations you select are too easy.

(10 marks) Deliverable: code and brief written answer if needed to discuss start and goal configuration.

4. Briefly discuss and compare the results given by $A^\star$ when using the two different heuristic functions in question 1.2.

(5 marks) Deliverable: written answer (including figures if appropriate).

Question 1.3: General solution of the 8-puzzle using $A^\star$.

Write a general version of the $A^\star$ algorithm (using either of the two heuristic functions described above) to solve a generic version of the 8-puzzle where the user can input any start and goal state. **(5 marks)**
(Hint: can this be done for any generic pair of configurations...?)

Deliverable: code and brief written answer to discuss whether the code can solve any pair of configurations.

Go to the next page for question 2 of the coursework.

Question 2 | Evolutionary Algorithm.

This question focuses consists of designing, implementing, analysing experimentally and writing a report on an *evolutionary algorithm* to solve *Sudoku puzzles*.

In a *Sudoku puzzle*, the objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid contains all of the digits from 1 to 9, from an initial partially competed grid.

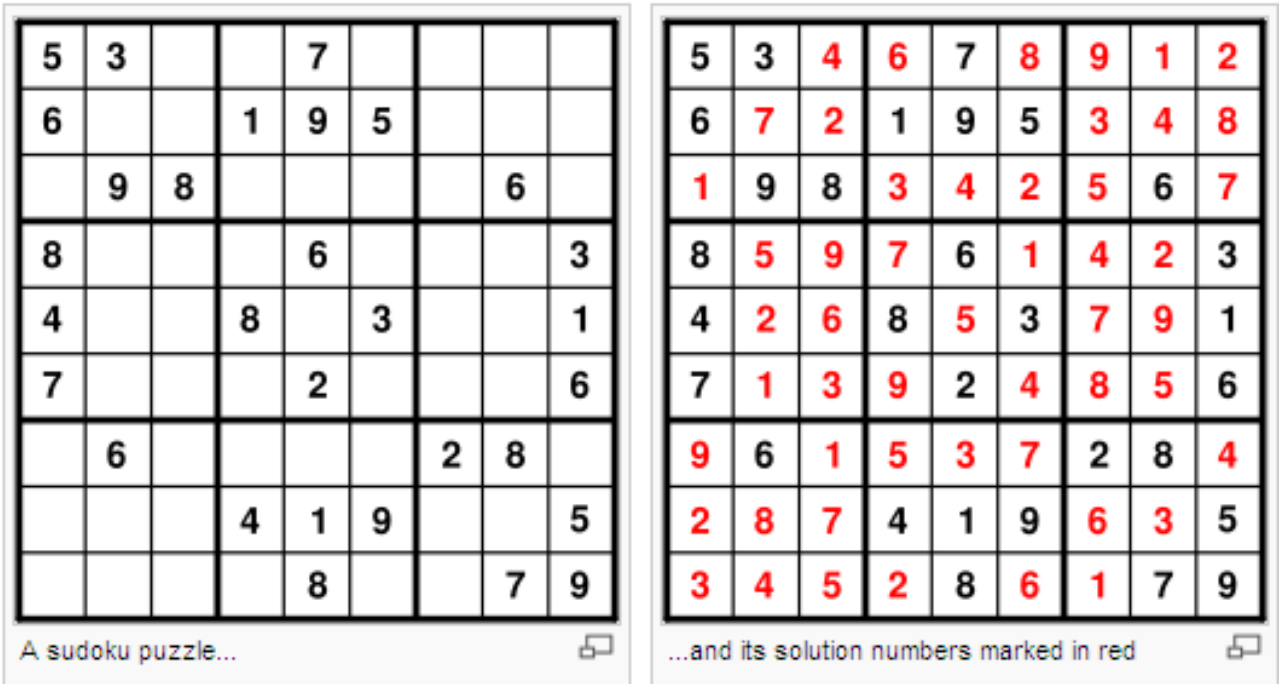

Figure 2: Examples of a sudoku puzzle. Figure taken from Wikipedia

An evolutionary algorithm is a general problem solving framework inspired to biological evolution. The pseudocode of an evolutionary algorithm is reported below.

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Question 2.1: Design and implement the Sudoko problem using Evolutionary algorithm.

This question is further divided in the two parts described next.

1. In first problem design your evolutionary algorithm addressing the following points in your design process

   (a) Choose an appropriate solution space and solution representation.
   (b) Define an appropriate fitness function.
   (c) Define a crossover operator for the chosen representation.
   (d) Define a mutation operator for the chosen representation.
   (e) Decide how to initialise the population.
   (f) Decide selection and replacement methods.
   (g) Choose an appropriate termination criterion.

   you should first briefly outline the how you representing Sudoko problem.

   (10 marks) Deliverable: Written answer.

2. Then, you should implement the evolutionary algorithm in Python to solve the Sudoku problem. You will need to run experiments for the three Sudoku grids provided on the ELE page, for population sizes 10, 100, 1000, 10000. Each experiment (i.e., a specific combination of grid and population size) needs to be ran 5 times (each one with a different random seed) and average performance across runs considered. In total these amount to 3 x 4 x 5 = 60 runs.

   (15 marks) Deliverable: code and table for the average performance on each grid and population size.

Question 2.2: Analysis the Sudoko problem using Evolutionary algorithm.

This question is help guide you to analysis your results based on the following questions.

1. What population size was the best?
2. What do you think is the reason for your findings in question 8.a?
3. Which grid was the easiest and which the hardest to solve?
4. What do you think might be the reason for your findings in question 8.c?
5. What further experiments do you think it may be useful to do and why?

(15 marks) Deliverable: Written Answer, including figures and graphs.