Assessment Information / Ngā Tohutohu Aromatawai

COMP.7212 Artificial Intelligence

Semester 2, 2023
Tutor: Kim Woods

Student/Ākonga/ID: _____

Student/Ākonga/Name: _____

ASSESSMENT CRITERIA/ Paearu Aromatawai

This assignment must be a product of your own work, except for the use of resources supplied within the course and discussions conducted with the lecturers.

Due Date:	Assessment:	Marks:	Weighting:
Friday, September 22nd (12.00am Midnight)	Report 1	125	40%
Learning outcomes assessed:			
1. Research and analyse the general nature of artificial intelligence and the problems it solves. 3. Apply and evaluate artificial intelligence techniques for solving a variety of real-world problems.			

Marking Rubric:	Marks Allocated:	Marks Awarded:
Part 1: Advanced Search	75	
Part 2: Genetic Algorithms	50	
Total Marks	125	
Weighting	40%	

ASSESSMENT DELIVERABLES / Tuku Aromatawai

You are required to upload a **single file** which may contain multiple files to Moodle on or before the due date. Your zipped file will contain the following files: your **A* implementation** (A_Star_Search.ipynb and maps.py) and **discussion** (Word document), your **Genetic Algorithms implementation** (solve-nurses-v1.py and solve-nurses-v2.py) and **report** (Word document).

ASSESSMENT INSTRUCTIONS/ Ngā Tohutohu Aromatawai
Overall Brief:

Part 1: Advanced Search

You are to implement an example of ***A* Search***. You are to analyse and compare how this algorithm works and compare A* Search with two other algorithms discussed in class (Breadth-first and Depth-first).

You must ONLY edit the Jupyter Notebook as required in the A_Star_Search project and the accompanying map.py file. The Notebook must work with the original project without modification. You will submit the Jupyter Notebook files and your written answers as a Word document.

To check your code runs correctly before submitting, please save your files and then "Reset and Run All" under the Kernel options. Please discuss with your lecturer if you are having any issues getting your code to run.

A. A* Search

You are to implement an A* search algorithm that can find the optimal path across a **40 by 40 grid**. You have been provided with a **map.py** file which can be edited to create your map. You have also been provided with an A* Search Notebook to edit where instructed.

Upon successful completion, your algorithm should:

1. Implement an A* Search that could find the shortest path to the two locations.
2. Ensure your map contains a minimum of 6 objects, with minimum of 1 object being a wall or bespoke shape you have created.
3. Displays the path cost of the solution found as required.

B. Discussion & Comparison

In a word document, describe in your own words:

1. Explain ***this*** implementation of A* Search in Python. i.e. how the code functions, including the libraries it uses.
    a. Please provide explanations for the following parts of the code:
        i. Start_node, goal_node, heuristic, g value, f value, h value, current_node, new_node.
        ii. Describe the role of the get_adjacent_nodes function in the DiscreteMaps class within maps.py.
        iii. Explain why a Priority Queue is used to keep track of open nodes.
        iv. Describe the role of the heuristic in this code. Which heuristic is being used? What is the alternative heuristic that is provided (but currently commented out)?
        v. What is the purpose of the numpy library within our code? What is it used for?

2. Compare A* to Depth-first and Breadth-first searches - which you think is more memory efficient **and why**. (Short paragraph)

3. Compare A* to Depth-first and Breadth-first searches - which algorithm you think is better overall for this task **and why**. (Short paragraph)

Marking Rubric Part 1: Advanced Search	Excellent (4-5 marks)	Good (3-4 marks)	Sufficient (2-3 marks)	Poor (0-2 marks)	Marks Awarded
Jupyter notebook: Start, End nodes selected correctly for both implementations.					
Map.py file: The 40x40 grid has been constructed correctly, with a minimum of six objects, including a wall.					
Map.py file: A print function is used to state when the agent has encountered an object in the room. This prints at least twice during the execution of the loop as objects are found.					
Jupyter notebook: Code errors in Search 2 have been corrected and the code runs successfully.					
Jupyter notebook: The return statement in the A* function has been completed and executes correctly.					
Jupyter notebook: The values of f, g, and h are printed once during the loop.					
Jupyter notebook: A print function displays when the goal node is found.					
Complete code: The full code executes correctly, without errors.					
Written response: Comparison in terms of memory usage, with memory usage being well described.					
Written response: A* overall task performance is accurately compared and well explained.					
Written response: Description 1.a.i					
Written response: Description 1.a.ii					
Written response: Description 1.a.iii					
Written response: Description 1.a.iv					
Written response: Description 1.a.v					

Student ID: _____ **Total Marks: ____ /75**

Part 2: Genetic Algorithms

You are to implement **two** versions of a genetic algorithm (GA) to find the optimal rostering pattern for nurses at **two** different local hospitals, one with 15 nurses, and one with 50 nurses.

This library and the code provided have a lot of features to help you implement your GA. This is primarily about configuring and optimising a GA to solve a difficult problem.

The library we are using: https://github.com/DEAP/deap

You have been provided with **two** versions of the nurse scheduling problem – one for a roster of 15 nurses (**nurses15.py**) and one for scheduling 50 nurses (**nurses50.py**).

You have been provided with the class NurseSchedulingProblem within these files, ***do not edit this class*** but instead modify the code in **solve-nurses-v1.py and solve-nurses-v2.py to implement your two different algorithms**. These files are the ones where you will **import** nurses15 or nurses50 and run your algorithm to get your results. Both algorithms need to be run for each of the two hospitals and used to compare results.

Upon successful completion, BOTH of your TWO algorithms should:

1. Implement a Genetic Algorithm capable of finding the optimal rostering pattern.
2. Calculate the cost of different solutions, according to the number of hard and soft violations (i.e. Take into account the constraints (staff preferences, staffing levels etc.))
3. If found, returns the solution as a schedule for each nurse.
4. Uses a ***different selection function*** from the other GA search algorithm.
5. Uses ***a different cross-over function and mutation probability*** from the other GA search algorithm.
6. Has the optimal settings listed in the report. (e.g. Show a screenshot of the settings in solve-nurses.py: HARD_CONSTRAINT_PENALTY, POPULATION_SIZE, P_CROSSOVER, P_MUTATION, MAX_GENERATIONS, HALL_OF_FAME_SIZE)

A. DNA: The Data Representation

The data representation provided uses a sequence of 1s and 0s to represent whether a nurse is rostered on (1) or off (0). For each day of the week, there are three possible shifts (morning, afternoon, evening).

A roster for one nurse, working a morning shift and evening shift looks like this = [1,0,1]

For a full week, their roster may look like this (with Weekends rostered off):
[1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,0,0,0,0,0,0]

For each shift, we must have a minimum and maximum number of nurses rostered, to maintain safe staffing levels and ensure patients are well cared for. For this reason, any solution we generate we will add up the number of nurses allocated to a particular shift. For example:

Nurses Per Shift = [4,2,4,1,6,6,4,4,5,4,5,5,2,3,4,4,4,3,7,5,3]

If the minimum number we can have on this ward is 3, and the maximum is 6, then any time we have scheduled 2 nurses or 7 nurses is a violation of the problem constraints. However, we might allow different staffing levels in the morning, afternoon and evening.

These settings have been provided to you in the files nurses15.py and nurses50.py.

B. Selection Function

You will need to think carefully about how you choose the individuals to be passed on to the next generation.

- At least one of your algorithms should use Tournament Selection, and the other a different form of selection (i.e. rank or roulette).
- Explain the benefits and downsides of each of these selection processes. Which do you think is most suited to the nurse scheduling problem and why?

Remember that the selection function for each of your algorithms needs to be ***different*** from the other.

C. Cross-Over & Mutation

There are several GA functions you can use for cross-over. You will need to look into what they do and experiment with their use.

While we will be using only **mutFlipBit** to mutate our individuals, the probability of mutation can be adjusted. Experiment with different rates of mutation and note the difference.

You can read the documentation in the source code for DEAP here:
https://deap.readthedocs.io/en/master/
http://deap.gel.ulaval.ca/doc/dev/api/tools.html

Remember that the settings for each of your algorithms need to be ***different*** from the other.

D. Termination

You will need to decide how long you allow your search algorithm to run.

There are two termination conditions for you to choose from:

Maximum Generations **-** The number of generations before the algorithm is just halted.

Stagnation Generations **-** The number of generations where fitness has not improved allowed before the algorithm halts.'

Note: *changing the termination settings* ***does not count*** *towards your algorithms being very different from each other as they do not change the search itself very much at all.*

E. Elitism – Hall of Fame

With at least one of your algorithms, experiment running it both with and without a Hall of Fame. Describe what the impact on the running time of your algorithm is from using Hall of Fame. Explain your findings.

You will list exactly what these settings (e.g. screenshot) were in your report so I can replicate your results.

F. Other Configuration Settings

Other settings include: HARD_CONSTRAINT_PENALTY, POPULATION_SIZE, P_CROSSOVER, P_MUTATION, MAX_GENERATIONS, and HALL_OF_FAME_SIZE. You will need to experiment with these to find the ones that work best. You will need to write about what settings you changed and what they do, so choose carefully.

You will list exactly what these settings (e.g. screenshot) were in your report so I can replicate your results.

G. Report: Describe, Discuss & Analyse

Create a short report discussing how well your algorithms performed. This will be submitted as a Word document via Moodle.

You should include the following sections:

Optimal Settings:

- List all the optimal settings used for each of your two search classes (e.g. As a screenshot)

Method:

1. A brief description of how one of the *cross-over* functions you chose works, why you chose it, and how you configured it.
2. A brief description of how one of the *mutation* functions you chose works, why you chose it, and how you configured it.
3. A brief description of any other settings you changed***, what*** they do and ***why*** they helped your search. (Failure to change any other settings will lose you marks.)

Results:

For each of the nurse rosters*:*

- Display a table of the results of your GA to show how well they worked
- You will need to run your algorithm several times *for each scenario* and take the average (`i.e. same start, goal, settings, etc.`)

The following is a <u>basic</u> example of the data expected (you may choose to provide more data, such as your best overall result):

	SEARCH ONE	**SEARCH TWO**
Roster with 15 staff		
AVG GENERATIONS TO GOAL	550	1000
# OF CONSECUTIVE SHIFT VIOLATIONS	2	1
# OF SHIFT PER WEEK VIOLATIONS	0	1
# NURSES PER SHIFT VIOLATIONS	0	0
# SHIFT PREFERENCE VIOLATIONS	2	1
# TOTAL VIOLATIONS	4	3
Roster with 50 staff		
AVG GENERATIONS TO GOAL	1000	2000
# OF CONSECUTIVE SHIFT VIOLATIONS	10	5

# OF SHIFT PER WEEK VIOLATIONS	2	1
# NURSES PER SHIFT VIOLATIONS	3	2
# SHIFT PREFERENCE VIOLATIONS	15	10
# TOTAL VIOLATIONS	30	18

Marking Rubric Part 2: Genetic Algorithms	Excellent (4-5 marks)	Good (3-4 marks)	Sufficient (2-3 marks)	Poor (0-2 marks)	Marks Awarded
Report Use of a title, contents & references page. APA references were used correctly.					
Genetic Search ONE: Selection Function Implemented a variation of a selection process and discussed the results.					
Genetic Search TWO: Selection Function Implemented a variation of a selection process and discussed the results.					
Settings & Results All settings and results were clearly and professionally presented.					
Cross-Over Function Clear explanation of how one of the cross-over functions you used works, why you chose it, and how you configured it.					
Mutation Function Clear explanation of how one of the mutation functions you used works, why you chose it, and how you configured it.					
Other Settings Changed Description other settings you changed, what they do and why they helped your search					
Comparison A clear explanation of why one algorithm was better than the other.					
Explaining Success Explanation of why you think the settings used resulted in one algorithm outperforming the other.					
Discussion of Learning & Genetic Algorithms Described something interesting learned as part of this project & their opinion on the usefulness of genetic algorithms.					
Student ID: _____				**Total Marks: _____ /50**	