THE UNIVERSITY OF MELBOURNE
SCHOOL OF COMPUTING AND INFORMATION SYSTEMS
COMP90054 AI PLANNING FOR AUTONOMY

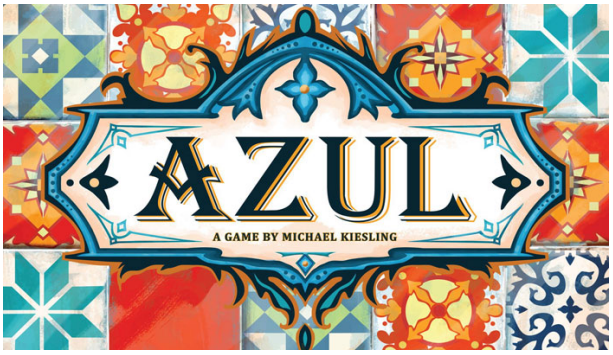
Group Project, Semester 1 2020

Team registration: 18:00 Monday 11 May 2020
Final Submission: 18:00 Wednesday 3 June 2020

This project counts towards 35% of the marks for this subject and **must be done in teams of 3**.

Aim

The purpose of this project is to implement an autonomous agent that can play and compete in a *tournament* for the game Azul[1].



The rules of Azul can be found here: https://www.ultraboardgames.com/azul/game-rules.php

Your final submission will consist of:

1. **[10 marks]** A working agent that is capable of playing two-player games and competing in the tournament.
   Your agent can use any technique, or combination of techniques, that we cover in this subject, as well as more advanced techniques that *extend techniques* covered in this subject. For example, you can use blind search, heuristic search, classical planning, dynamic programming, reinforcement Learning, Monte Carlo Tree Search, game theory, or any extension of these.
2. **[5 marks]** A recorded 5-minute oral presentation that outlines the theoretical or experimental basis for the design of your agents. It should cover:
   (a) the design of your agents; that is, why you did what you did;
   (b) any challenges faced;
   (c) what you would do differently if you had more time; and
   (d) a live demo of your different implementations, i.e. showing how
3. **[20 marks]** A report describing the approaches implemented, an evaluation comparing your different agents showing their performance in several scenarios. The link for the recorded oral-presentation should be included in the report.

[1] See https://en.wikipedia.org/wiki/Azul_(board_game).

Corrections: From time to time, students or staff find errors (e.g., typos, unclear instructions, software faults, etc.) in the assignment specification and source code. In that case, corrected version will be uploaded to the course LMS as quickly as possible.

Silent Policy: A silent policy will take effect 48 hours before this assignment is due. This means that no question about the assignment will be answered, whether it is asked on the newsgroup, by email, or in person. By this stage, you should have a clear idea of what the assignment is about and have figured out any issues that require staff input.

Task 1: Team formation and repository registration – Due: 18:00 Monday 11 May 2020

This is a team project assignment (teams of 2-3 students), which you have already registered.

Each team must set up a GIT repository:

One of the members of the new team should fork privately the following template project repository in UoM GitLab using their student account:

`https://gitlab.eng.unimelb.edu.au/tmiller/comp90054-2020s1-azul.git`

Click on the gitlab link above and fork the repository following these instructions: set and keep your repository *private*: Go to your forked project gitlab page and click on Project settings → General → Expand permissions tab → Change 'Visibility Level' to 'Private'.

Give write permissions to all the other students who are part of team. Add each member as a maintainer role.

Do not alter the directory structure of the forked repository.

Your team must then create a player that is capable of playing in the tournament (see submission instructions), even if this is just a copy of one of the random or naive player.

Then, *one* team member should submit the repository link and team name here: `http://go.unimelb.edu.au/iw3r`

After this, we will run a dummy tournament to ensure we can access all teams' submissions and run them.

Task 2: Implementation of an player for the tournament – Due: 18:00 Wednesday 3 June 2020

Produce a working agent that can play games.

Each member of the team must implement one technique, meaning that the team *must use at least three of the techniques* that have been discussed in the subjects. For the few teams with two members, they should use at least two techniques respectively.

These techniques can be combined into a single player, or can be implement across multiple players. However, you can only run one player in the final tournament.

Note: *The techniques implemented that did not make it to the final submission should have their associated commit tag so we can explore and run the code.*

The candidate techniques are:

1. Blind search algorithms
2. Heuristic search algorithms (using general or specific heuristic functions)

3. Classsical planning (PDDL and calling a classical planner)
4. Value iteration or policy iteration (Model-Based MDP)
5. Monte Carlo Tree Search (Model-free or Model-based MDP)
6. Reinforcement learning – tabular, function approximation etc. (Model-Free MDP)
7. Game theoretic models

You are free to find more advanced versions of these techniques and apply them as well.

If you decide to compute a policy, you can save it into a file and load it at the beginning of the game.

To perform well in the tournament will require experimentation. Recording results of evaluations that you do with you agents will be important in reporting your results later.

Deliverables

A final submission, by the official deadline, your gitlab repository for this project should consist of:

(1) A **working agent** that is capable of competing in the tournament by suitably modifying file `myPlayer.py` as per instructions. The code should be internally commented at high standards and be error-free and should not crash.

(2) In the root folder of your repository, a *group.txt* plain text file listing your group members' student number, full name, and email, one per line, separated by a comman, in the format:
`StudentID, FirstName, LastName, email@student.unimelb.edu.au`

(3) A five-minute (max!) youtube video of the design decisions made challenges experienced, and possible improvements; and a demo of your agent across a variety of scenarios
Some tips and tools for creating video presentations: https://lms.unimelb.edu.au/students/student-guides/record-edit-upload-videos

(4) A four-page PDF report in your repository (no title page), including the link to your **youtube** presentation at the top of the report. Make your youtube video unlisted if you don't want it to be searchable. The report should explain and critically analyse your agent. Your report should consist of:

   (a) A overview explaining which techniques you experimented with and which techniques you used in the final tournament, and techniques tried but not used in the final version. Importantly, each technique, briefly explain *why you thought this technique would work*, based on your understanding of the subject and the game rules.
   As you try new things during the project, take some brief notes on why you are trying those things. You can then form this section of the report based on those notes.
   Note that for this part of the report, teams gain marks by showing that they have formulated different techniques that are suitable for the task. Teams are *not* marked on the actual outcome of the agents – it is perfectly fine to have techniques that were not successful.

   (b) An overview of your final tournament agent, including a critical assessment of its strengths and weaknesses.

   (c) A section that outlines any experiments and evaluation of your techniques against each other or other baselines.

   (d) For each team member, a one page self-reflection that answers the following questions for that team member:
      i. What did I learn about working in a team?
      ii. What did I learn about artificial intelligence?
      iii. What was the best aspect of my performance in my team?
      iv. What is the area that I need to improve the most?

The individual reports do not count towards the four-page limit.

Task 3: Contribution report and Honour Code– Due: 18:00 Wednesday 3 June 2020

All students must fill out the form here about contribution and sign the honour code: `http://go.unimelb.edu.au/jw3r`

Failure to fill out the honour code will result in a late penalty.

Submission Instructions

To submit you must follow the next two steps:

1. **Submit your solution** by simply **tagging** "`submission-contest`" the commit you want to submit:
   - See this guide for tagging using the command line or this video, and here for tagging via GitLab interface directly.
   - To re-submit another version you need to delete previous submission tags
     - First delete it from the GIT server by running: *git push −−delete origin <tagname>*
     - Second, delete the local tag in your repo by running: *git tag −−delete tagname*

   For the tournament, we will automatically extract the `myPlayer.py` file in the folder `players/<your_teamname>/` of your repository. You can still import other files if needed; see item 6 below.
2. **Each student must certify the submission** and **contribution details** by filling the following:

   Contribution report and Honour Code signing: `http://go.unimelb.edu.au/jw3r`

   Each member of the team must certify in order to get a mark for the assignment project. Lack of certification will attract a late penalty for any student that fails to submit.
   While the project can be done as a team, we reserve the right to assess members of a team individually.

Important: When submitting a solution, please make absolutely sure you adhere to the following instructions:

1. Your code **must run** on Linux and adhere to **Python 3.6**. Staff cannot debug or fix any code.
2. At the very minimum, your code should be **error-free**. If your code crashes in any execution, it will disqualified from the contest. Again, staff cannot debug or fix code that crashes.
3. All related files **must be placed** in folder `players/<your_teamname>/` of your repo (i.e., not in other folders).
4. You should **not** create threads or run any parallel code, as this can interfere with other players.
5. You are **not to change or affect the standard output or error** (`sys.stdout` and `sys.sterr`) in any way. These are used to report each game output and errors, and they should not be altered as you will be intefering negatively with the contest and with the other team's printouts. *If your file mentions any of them it will be disqualified automatically.*
6. Your code will be copied into a directory called `players/<your_teamname>/` in the contest package. This means that if you import from other files outside `myPlayer.py` they will not be

found unless you tell Python to look in your team dir. You can do so by having the following code on top of your `myPlayer.py`:

```
import sys
sys.path.append('players/<your_teamname>/')
```

7. There is a useful options 1) `--delay` to slow down the execution if you want to visualize in slow motion, 2) `--output` and `--replay` to save and replay. Use `python runner.py --help` to check all the options.
8. Do *NOT* use the current working directory to write temporary files; instead, redirect all output to your own folder `./players/<your_teamname>/`. For example, if you use a planner online, and generate PDDL files and solutions, redirect your planner call, solution outputs, etc., to your own folder. You can use python code to do it automatically, or you can hardcode it assuming that your team will be located in `./players/<your_teamname>/` folder.
9. The tournament games will be run with the command:
   `python runner.py -r <team-name-1>.myPlayer -b <team-name-2>.myPlayer -s`
   Other options all use default, including a *one second timeout* on all moves and a *five second* buffer for each player at the start of each round to process the layout of the that round. Any player exceeding either of these time limits three times in a game loses the game.
10. If you want to use any other 3rd-party executable please discuss with us before submission. You can assume that `TensorFlow, keras, sklearn, numpy, scipy and neat-python` libraries are installed in our running environment, using the latest version available in Ubuntu 18.04. `ff` executable version 2.1 of the Metric-FF planner (https://fai.cs.uni-saarland.de/hoffmann/metric-ff.html) is available in /usr/local/bin.
11. Finally, it is very important that you submit to the practice tournaments to check that your agents runs successfully and to get some insight into how your agent performs against staff teams and other student teams.

Preliminary Tournaments

We will be running *five* practice tournaments based on preliminary submissions in the weeks before the final project submission. These tournaments will give you an opportunity to play against staff teams and other student teams to determine how your agent performed. Dates for these will be announced on the LMS.

Participating in these pre-contests will give you a lot of insight on how your solution is performing (by downloading and re-playing each game) and how to improve it, but you should not rely on these as your only method to improve your playing. Experimenting yourself and *thinking* are crucial.

Results, including replays for every game, will be made available for these practice tournaments.

Marking criteria

The project is 35% of your final mark, broken into three parts: (1) tournament placing; (2) report; and (3) presentation.

Part (i) Tournament performance – 10 marks

Marks will be given according to final position in the tournament with respect to other competitors and the basic staff teams.

- Agent finishes above the *staffTeamBasic* agent [1 mark].
- Agent finishes above the *staffTeamMedium* agent [1 mark].
- Agent finishes above the *staffTeamAdvanced* agent [2 marks].
- First place in the final competition (not include the staff teams) [1 mark].
- Marks relative to your score on the leaderboard table, scored on a linear scale from lowest score (0) to highest score (5) [5 marks].

The score will be calculate by the points scored in game, plus 50 points for winning, and 20 points for a tie.

Part (ii) Video– 5 marks

- A clear presentation of the design decisions made, challenges experienced, and possible improvements that does NOT exceed five minutes [2 marks]
- A clear demonstration and understanding of the subject material [2 marks]
- Demo of the different agents implemented across a variety of scenarios, showcasing strengths and weaknesses of each technique used. No need of full game demo, just edit interesting parts and explain your insights [1 marks]

Part (iii) Report– 20 marks

- A clear description of the design decisions made, techniques used, strengths and weaknesses of these techniques, challenges experienced, and possible improvements [5 marks]
- A clear demonstration that the design decisions made are suitable for the particular problem being solved [5 marks]
- An experimental section that evaluates the performance of the different techniques applied, both against each other, sensible baselines, and in the trial tournaments [4 marks]
- A description of sensible improvements they would make to the agent given additional time [2 marks]
- Demonstrated attempt to explore different techniques, try various types of techniques, and explore techniques that extend beyond the subject content [4 marks]

We hope you enjoy this challenge and take this as an opportunity to learn and experiment with different techniques. If you still have doubts about the project do not hesitate asking in the subject discussion forum.
Good luck and have fun!

Academic Misconduct

The University misconduct policy[2] applies. Students are encouraged to discuss the assignment topic outside of their teams, but all submitted work must represent the individual's understanding of the topic. Further, any attempts to subvert the tournament by taking advantage of vulnerabilities will be treated as academic misconduct.

[2] See https://academichonesty.unimelb.edu.au/policy.html

Important: As part of marking, we run all submissions via a code similarity comparison tool. These tools are quite sophisticated and are not easily fooled by attempts to make code look different. In short, if you copy code from other teams or from online sources, you risk facing academic misconduct charges. It is ok to take code snippets from online sources if the copyright does not prevent it, but you must acknowledge this in your source code, and the originality multiplier will be used.

Originality multiplier We will be using a code similarity comparison tool to ensure that each team's work is their own. For code that is similar to another submission or code found online, an originality multiplier will be applied to the work. For example, if 20% of the essential parts of the project is deemed to have been taken from another source, the final mark will be multiplied by 0.8.

The subject staff take academic misconduct very seriously. In this subject in the past, we have successfully prosecuted several students that have breached the university policy. Often this results in receiving 0 marks for the assessment, and in some cases, has resulted in failure of the subject.