COMP90054 AI Planning for Autonomy
The University of Melbourne
School of Computing and Information Systems

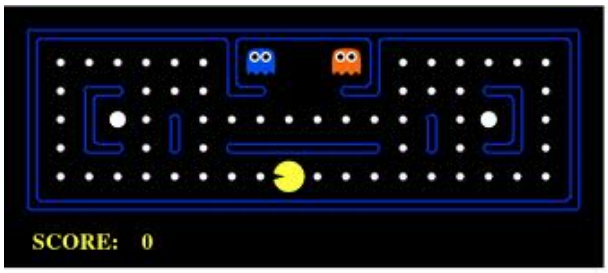
Project 2, 2018
Contest: Pacman Capture the Flag

Deadline: 23:59 Sunday 14 October 2018

This project counts towards 40% of the marks for this subject.

This is an **team** project-assignment and **has to be done in groups of 3 (or 4)**.

The purpose of this project is to implement a *Pac Man* Autonomous Agent that can play and compete in a *tournament*.



http://ai.berkeley.edu/contest.html

Note that the Pacman tournament has different rules as it is a two teams game, where your Pacmans become ghosts in certain areas of the grid. Please read carefully the rules of the Pacman tournament. Understanding it well and designing a controller for it is part of the expectations. To help you develop your solution you must provide:

(i) A working Pac Man Agent that is capable of playing Pac-Man and competing in the tournament. Your Agent can use any technique, or combination of techniques, that you choose. For example using a classical off-the-shelf planner, Reinforcement Learning, Heuristic Search, Monte Carlo Tree Search or a purpose built decision tree of *your own making* (25 marks).

(ii) A recorded 5-minute oral presentation that outlines the theoretical or experimental basis for the design of your agents (that is, why you did what you did), challenges faced, and what you would do differently if you had more time. Your presentation must end with a live demo of your different implementations, i.e. showing how the different techniques your tried work. The video Will be shared with us through an unlisted youtube link. (10 marks).

(iii) A WIKI: describing the approaches implemented, a small table comparing your different agents showing their performance in several scenarios. Discuss briefly the table. The link for the recorded oral-presentation should be included in the wiki. (5 marks).

This project follows up from your Search agents for project 1. You should be familiarized with the pacman environment, and you can use variations of your search agents as a starting point for this project.

Corrections: From time to time, students or staff find errors (e.g., typos, unclear instructions, etc.) in the assignment specification. In that case, corrected version will be uploaded to the course LMS as quickly as possible and an announcement will posted in the course LMS and also to the forum (if they issue was related to a forum message). The date of the latest specification can be found in the bottom right of this document.

Silent Policy: A silent policy will take effect **48 hours** before this assignment is due. This means that no question about this assignment will be answered, whether it is asked on the newsgroup, by email, or in person.

Team Registration & GitLab Repo Setup

This is a group/team project assignment (groups of 3 students). There will be no need to explicitly submit anything for this assignment project, except filling the certification, as the repository on Bitbucket will represent the team's submission (more details below).

A team should do the following (only one student member needs to do these steps):

1. Set-up project GIT repository:
   - One of the members of the new team should **fork <u>privately</u>** the following template project repository in UoM GitLab using their Unimelb student account:

     `https://gitlab.eng.unimelb.edu.au/nlipovetzky/comp90054-pacman`

   This repository has the initial template for the contest under subdirectory `pacman-contest/`. Make sure:
   - Click on the gitlab link above and **Fork** the repository forllowing these instructions.
   - Repository must be slightly changed to "`comp90054-pacman-<student number>`", where `<student number>` is the student number of the owner of the repository. In order to **change the name of the project** follow these instructions
   - Set and keep your repository **private**. Respect the wishes of the creators of the UC-Pacman: ***"Please do not distribute or post solutions to any of the projects."***
     - Go to your forked project gitlab page and *click on Poject's Settings → General → Expand permissions tab → Change "Visibility Level" to Private.*
   - **Give write permissions** to all the other students who are part of team. Follow these instructions. Add each member as a *mantainer* role.
   - Give *mantainer* **permission to me** (Nir) by registering my username: nlipovetzky.
   - **Set up a Wiki** in the repo: Wiki page should list team name, team members names and student numbers, and can have information for each project. See a template here.
   - Do not alter the directory structure of the forked repository.

   <u>Note:</u> The team is expected to develop their solution incrementally in this repository and apply good Software Engineering practices and group working by regularly updating this repository (for example, a submission containing the final solution in just a few commits is *not* good practice). The repository will be used as a way to gauge any anomalies in the effort put in the semester for any student. This could lead to fairly different marks within the same team if deemed necessary.
2. Register the group team for this project by filling this form:

   `https://goo.gl/forms/onQaCAWqJaLxUUSE2`

   Once your team is registered, the team is ready to work and will be able to submit its solutions!

Deliverables

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

(1) a **working Pacman agent** that is capable of competing in the tournament in Python by suitably modifying file `myTeam.py` as per instructions in the UC Berkeley Contest page. The code should be internally commented at high standards and be error-free and never crash.

In your solution, You **have to use at least 2 of the techniques (3 techniques at least for groups of 4)** that have been discussed in COMP90054. Feel free to combine them in any form. The candidate techniques are:

(a) Heuristic Search Algorithms (using general or pacman specific heuristic functions)
(b) Classsical Planning (PDDL and calling a classical planner, see subsection below)
(c) Policy iteration or Value Iteration (Model-Based MDP)
(d) Monte Carlo Tree Search or UCT (Model-Free MDP)
(e) Reinforcement Learning – classical, approximate or deep Q-learning (Model-Free MDP)
(f) Goal Recognition techniques (to infer intentions of opponents)
(g) Game Theoretic Methods

(2) a *group.txt* plain text file listing your group members' student number, full name, and email, one per line,

(3) a *Wiki* in your repository, including the link to your **youtube** presentation at the top of the Wiki. Make your youtoube video unlisted if you don't want it to be searchable. The Wiki should explain and analyse critically

your Pacman agent system. Besides explaining the technique used, the wiki can include, but is not limited to, experiments to show how different variations of your agent perform against each other, assumptions that you made, analysis of the strengths and weaknesses of your solution, techniques tried but not used in the final version, etc.

The above files must all be placed in folder `pacman-contest/` of your repo and with the exact name and capitalization as described above. Please check submission details below.

Submission Instructions

To submit you must follow the following 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 marking process, we will automatically extract files `myTeam.py`, `group.txt` in the folder `pacman-contest/` of your repository. You can still import other files if needed; see item 5 below.
2. **Certify your submission** and **contribution details** by filling the following form:

   Contest - Certification of Submission

   Each member of the team must certify in order to get a mark for the assignment project. Lack of certification will attract zero marks for the whole assignment project (for the corresponding student).
   As part of the certification, each team member is requested to submit a brief **individual contribution statement** outlining two things:
   (a) What they contributed to the project.
   (b) What they perceive their team members contributed to the project.

   This individual statement is intended to give everyone the opportunity to inform subject staff of the contributions of their team. While the project can be done as a team, we reserve the right to assess members of a team individually. Note that you will not be given a chance to submit this information after the deadline, even if your mark is affected by one of your team members report.

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 2.7**. Staff will not 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 will not debug or fix code that crashes.
3. The deliverables **must be placed** in folder `pacman-contest/` of your repo (i.e., not in other folders).
4. 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.*
5. Your code will be copied into a directory called `teams/<your_teamname>/` in the contest package. This means that if you import from other files outside `myTeam.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 `myTeam.py`:

   ```
   import sys
   sys.path.append('teams/<your_team>/')
   ```
6. Your code will be run by the following command:

`python2.7 capture.py -r teams/<team1>/myTeam.py -b teams/<team2>/myTeam.py`,

please make sure your `AgentFactory` is defined in `myTeam.py`.

7. Do *NOT* use the current working directory to write temporary files; instead, redirect all output to your own folder `./teams/<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 `./teams/<your_teamname>/` folder.
8. If you want to use any other 3rd-party executable please discuss with us before submission.
9. Finally, submit your project **substantially before** the deadline, preferably one day before. Submitting close to the deadline could be risky and you may fail to submit on time, for example due to lose of Internet or server delays. There will be no extensions based on these unforeseen problems.[1]

Submissions not compatible with the instructions above will attract zero marks and do not warrant a re-submission. Staff will not debug/fix your submission.

General Comments (optional read)

1. Implementation of Pac Man Agents

You can always use hand coded decision trees to express behaviour specific to Pac-Man, but they won't count as a required technique. You are allowed to express domain knowledge, but remember that we are interested in Autonomy, and hence using techniques that generalise well. The first 7 techniques can cope with different rules much easier than any decision tree (if-else rules).

If you decide to compute a policy, you can save it into a file and load it at the beginning of the game, as you have 15 seconds before every game to perform any pre-computation.

While a classical planning approach is perhaps the simplest way to get a working agent (quick prototype), it is unlikely to do well in the tournament play if not combined with other techniques. That is, you should think about each possible situation that may arise during the game, and use the best technique you know. You do not need to use classical planning for each situation, actually you don't need to use it at all. Just use at least 2 (3 if groups of 4) different techniques from the list in Deliverables Section.

2. Pac Man as Classical Planning with PDDL

Typical applications of planning consist on one or several calls to a planner. The instances are generated *on the fly* by a *front–end* (the pacman engine), and the solutions (plans) are interpreted as executable instructions. As the pacman is not a classical single agent problem, you could implement two points of view: The point of view of the pacman, where its goal is to stay alive while eating all the dots of the grid, and The point of view of the ghost, whose goal is to kill pacman. Assume that the game is turn-based, so at each step an instance is generated with the current state of the world, i.e. the dots and ghosts locations in the grid. From the point of view of pacman, the ghosts don't move, and vice-versa, that is, the environment is static.

At each step the planner would come out with a plan to eat all the dots while avoiding static ghosts, and plans to enable ghosts to kill the static pacman. A simple interpretation of the plans by the pacman engine is to execute only the first action of the plan, ignore the remaining actions, and call the planner in the next step with a new updated instance accounting for the new locations of the ghosts and the pacman.

The axiomatisation should define the state model for pacman using PDDL, and another PDDL for a ghost state model. If you try this approach, explain clearly the assumptions made, e.g. pacman do not move to cell X when Y holds, Ghosts are static, etc., and describe several initial states or goals to illustrate interesting situations.

[1] Extensions will only be permitted in exceptional circumstances. Note that workload and/or heavy load of assignments will not be accepted as exceptional circumstances for an extension.

Use one PDDL domain file for pacman, and one domain file for the ghost containing the predicates and the actions of the world. The problem file describes the 'initial' state and goals. Therefore, with a single domain for either the pacman or the ghost, several problems can be generated by only updating the problem file.

By reading the state of the Pacman from the engine and converting this into PDDL predicates, you can describe the state of the game in PDDL and, at each step that an action is required, call your favourite planner using that state as the initial state. Then, parse the solution in order to choose the best action.

Different domains can be used to encode different strategies.

Make sure that your PDDL files can be solved using the online solver in `http://editor.planning.domains`.

Preliminary Contest Submission

We will be running informal tournaments based on preliminary submissions in the weeks before the final project submission. We will start once **five teams have submitted** their preliminary agents (by tagging their repos).

Participating in these pre-contests will give you a lot of insights on how your solution is performing (by downloading and re-playing each game) and how to improve it. Results, including replays for every game, will be available only for those teams that have submitted. You can re-submit multiple times, as long as your repository has a submission tag, and they carry no marking at all; they are just for debugging and improving your solution! You do not need to certify the preliminary submissions, only the final one (you do need to register your team though).

Marking criteria

A *final contest* using many layouts will be run juts after final submission. The top-8 will play quarterfinals, semi-finals and finals, time permitting live in the last day of class or in week 13 in a day specified for that (these final phases will not be part of the marking criteria, just bonus marks).

The *final contest* and the *quality of the Wiki and Presentation* will be used to derive the final marks for the project.

Part (i) – 25 marks

Marks will be given according to final position in the tournament with respect to *staffTeam*:

- Pacman competitor finishes above the *staffTeamBasic* agent [11 marks].
- Pacman competitor finishes above the *staffTeamMedium* agent [7 marks].
- Pacman competitor finishes above the *staffTeamTop* agent [7 marks].
- Final competition place (up to 2 bonus marks).
  - Top-3 teams in the final competition will receive 1 bonus mark.
  - Winner of the Final Tournament will receive 1 bonus mark.

  The precise mark will depend how far your agent system is from these reference agents in the final contest.

Part (ii) – 10 marks

- A clear presentation of the design decisions made, challenges experienced, and possible improvements [3 marks]
- A clear demonstration and understanding of the subject material [2 marks]
- Demo of the different agents implemented across a variety of scenarios, showcasing pitfalls and benefits of each approach. No need of full game demo, just edit interesting parts and explain your insights [5 marks]

Part (iii) – 5 marks

- A clear written description of the design decisions made, approaches taken, challenges experienced, and possible improvements [3 marks]
- An experimental section that justifies and explains the performance of the approaches implemented, including a table and discussion [2 marks]

The *staffTeams* are the reference baselines: the farther an agent is from the base reference agents (Basic and Medium), the more marks it will attract. The only exception is *staffTeamTop*, any team that finishes above it, will earn full marks (25 points). This together with the quality of the Wiki and the presentation will determine the final mark, then adjusted as per individual contribution[2] . So this means that if an agent system is between `staffTeamMedium` and `staffTeamTop` and has a VERY GOOD (all marks earned) Wiki and Presentation, then it will score between 82.5% and 100% overall (the closer to `staffTeamTop`, the closer to 100%).

Inter-University Competition

The top teams of the tournament will qualify to the yearly championship accross RMIT and The University of Melbourne, which will run each semester along with the best teams since 2017 onward (given you grant us permission). Note that the top-8 will play Quarterfinals, semifinals and finals, time permitting live in the last day of class. This is just "for fun" and will attract no marks, but is something that previous students have stated in their CVs!

I hope you enjoy this open-ended contest-based project and learn much from it. If you still have doubts about the project and/or this specification do not hesitate asking in the Course Forum.
GOOD LUCK!

Academic Misconduct

The University misconduct policy[3] applies. Students are encouraged to discuss the assignment topic, but all submitted work must represent the individual's understanding of the topic.

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.

[2] We will use the *team code repository* as a way to gauge any anomalies in the effort put in throughout semester for any student. This could lead to fairly different marks within the same team if deemed necessary. Plagiarism detection software will also be used.

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