Development of an autonomous agent for the game **Rush Hour**

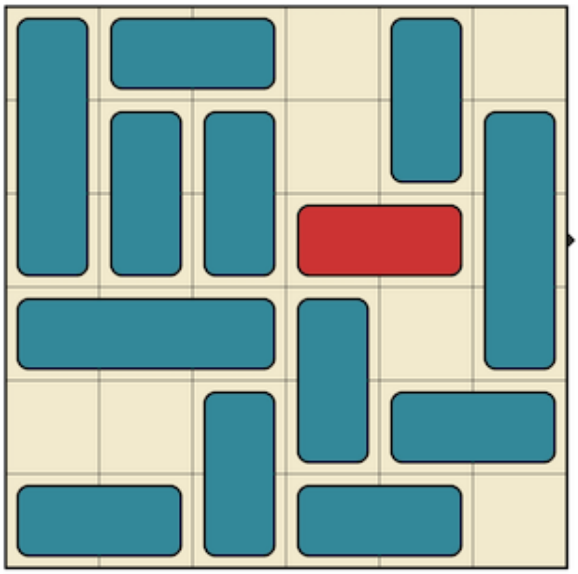
Group Assignment

Inteligência Artificial

Ano Lectivo de 2022/2023

Diogo Gomes
Luís Seabra Lopes

September 28, 2022



I Important notes

1. This work must be carried out in groups of 2 students. In each Python module submitted, you must include a comment with the authors' name and mechanographic number.

2. A first version of the program must be submitted by 11th of November 2022. The final version must be submitted by 2nd of December 2022. In both submissions, the work can be submitted beyond the deadline, but will be penalized in 5% for each additional day.

3. Each group must submit its code through the *GitHub Classroom* platform. In the final submission, include a presentation (Powerpoint type) in `.pdf` format and named `presentation.pdf`, with a maximum of five pages, where you should summarize the architecture of the developed agent.

4. The code should be developed in at least Python 3.7. The main module should be named `student.py`.

5. If you discuss this work with colleagues from other groups, include a comment with the name and mechanographic number of these colleagues. If you use other sources, cite them as well.

6. All code submitted must be original; though trusting that most groups will comply with this requirement, tools will be used to detect plagiarism. Students involved in plagiarism cases will have the assignment canceled.

7. The project will be evaluated taking into account: performance; quality of architecture and implementation; and originality.

II Overview

This work involves the application of concepts and techniques from three main chapters of the AI course, namely: Python programming; agent architectures; and search techniques for automated problem solving.

Within the scope of this work, you should develop an agent capable of playing intelligently the game Rush Hour, a puzzle with sliding blocks or "cars"invented by Nob Yoshigahara in the 1970s.

In *Rush Hour*, the player must drive the red car out of the map through the right side. The agent has access to a cursor, through which it can select any car and move it. Cars are parked traditionally in a 6x6 map (our game can nonetheless have different dimensions) and can only move in one direction according to their orientation: horizontal cars only in the left/right direction; and vertical cars only in the up/down direction. Only one car can be moved at a time, and cars may not overlap each other.

The game runs at a predefined speed of 10 steps/second, and the agent can only execute one move/action per step.

The game's score takes into account the number of steps. The time the agent takes to find a solution and the time it takes to execute are all accounted for. Each map has a base score from which points are deducted.

1 Objectives

- To obtain a positive mark, the agent must be able to score more than 10000 points.
- Score as much as possible. (see below details on marking)

III Game Rules

- *Rush Hour* starts with a square tiled map of parked "cars". The player owns the red car and must leave the parking lot through the right side.
- The *Rush Hour* player (student agent) has access at all times to the entire map, with the location of all occupied positions. The agent controls a cursor, through which he can select a car and move it either vertically or horizontally (depending on the car direction).
- *Rush Hour* allows you to move the cars using the commands **"w"** (*move up*), **"s"** (*move down*), **"a"** (*move left*), **"d"** (*move right*) and **" "** (*select/deselect*).
- The game is composed of several maps of incresed difficulty. For each map, the player must move the red car through the right side of the map.
- During the game, cars might move randomly.
- Each map in the game has a timeout value. If you don't find a solution before timeout you loose the game.

IV Code and Development Support

A *Rush Hour* game engine written in Python is available at `https://github.com/dgomes/ia-rush`.

All game entities are represented by classes.

Each group develops an agent creating a client that implements the exemplified protocol in the `client.py` file. No modifications to other files are necessary (you can't change `game.py`), but you can create new files, folders, etc.

If you implement a new feature or implement any improvement to the game engine and/or viewer, you can create a "Pull Request"(PR) on the GitHub platform. If your change is accepted, you will be credited with a bonus on the final evaluation up to a maximum of 1 point (in 20).

The developed agent must be delivered in a module named `student.py` and the agent should connect to the local game server, using as *username* the mechanographic number of one of the elements of the group (any).

There is a support channel in `https://detiuaveiro.slack.com/messages/ai/` where students can ask questions and receive notifications of changes.

Given the novelty of the game engine, it is expected that there are some bugs and tweaks during the course of work. Be on the watch out for server modifications (configure the professor repository as an upstream repository and fetch/merge often) and notifications in *e-mail*, *Slack* and *e-learning*.

To start the work, you must form a group with colleagues and access the link `https://classroom.github.com/a/qQ9M_PnH` which will *fork* the code for the group. Only one *fork* should be done per group. One of the elements of the group creates the group associating the other elements. After this step, the *fork* will be created automatically (do not create a new *fork* without all elements being registered).

V Recommendations

1. Start by studying `client.py`. The code is very basic and simple, so start by *refactoring* the client to something more oriented to an AI Agent.
2. Periodically `git fetch` from the original repository to update your code.
3. Run `git log` to keep track of small changes that have been made.
4. Follow the channel #ai on Slack

VI Clarification of doubts

Clarifications on the main doubts that may arise during the performance of the work will be placed here.

1. **Question**: How will the performance of agents be evaluated?

   Answer: Agents will be evaluated on their performance in a set of games based on the sum of the final scores in these games. The final score in a game is the one the agent has at the time of *gameover* (when the agent fails to exit the current map within the time limit) or when all maps have been completed.

2. **Question**: How will the practical work be evaluated?

   Answer: Total game scores for each submitted agent are mapped to marks taking into acount the distribution of total scores. A total score of 10000 is mapped to 10/20. The median of the total scores is mapped to 16/20. The maximum total score is mapped to 20/20. Other total scores are mapped linearly.

   1st delivery evaluation

   - In this delivery, only the agent code must be submitted.
   - Each agent will play 10 games and the average of these 10 games will be obtained.

   2nd delivery evaluation

   - Each agent will play 10 games.
   - In this delivery it is necessary to submit a presentation (see point I.3 above).
   - The evaluation of the second delivery reflects the agent's performance (90%), the design quality (according to the delivered presentation, 7.5%) and the quality of the implementation (2.5%).