The Challenge for the AI component of Module 6 TCS&BIT (Intelligent Interaction)

Adapted version on: Thu 22$^{\text{nd}}$ Nov, 2018 at 11:56.

This report is living document describing the AI challenge for the AI component of Module 6 for TCS and BIT.

Chapter 1

Introduction

The challenge consists of designing and implementing an intelligent game playing agent for two games. The first game is the so-called snake game and the second game is a two player Pacman game. For the Pacman game we will have a final competition at the Design Lab the end of the course. More information on date, time and location can be found at the University Roster. Each team/group consists of two students and TCS students should team up with TCS students, BIT students should team up with BIT students and non TCS and BIT students should also team up with each other.
Your have to deliver a report in pdf format containing your names and student number in which clearly describe how you solved the different assignments on a conceptual level. Your report should contain no code but if needed may contain some snippets of pseudo-code.
You may share ideas, knowledge, information etcetera, but please do not share or distribute code. Please do not let us down!

1.1 Passing the AI challenge

You will already pass the AI challenge if the Snake game part of your scientific report is satisfactory. The second part of this challenge, the Pacman game, is just challenging you for fun!

1.2 Deadline report

Deadline for the final report of the AI challenge is Monday January 21 2019!

Chapter 2

Snake game

The Python code for the snake game can be found on Canvas. Carefully read the included Readme file, this contains information about running the game and the different software components. The critical component is `agent.py` which is the intelligent agent controlling the snake. A typical board configuration of the snake game can be found in Figure 2.1, this configuration, except the initial position snake, is the default configuration of the game which will be used for testing your implementation of the intelligent agent.

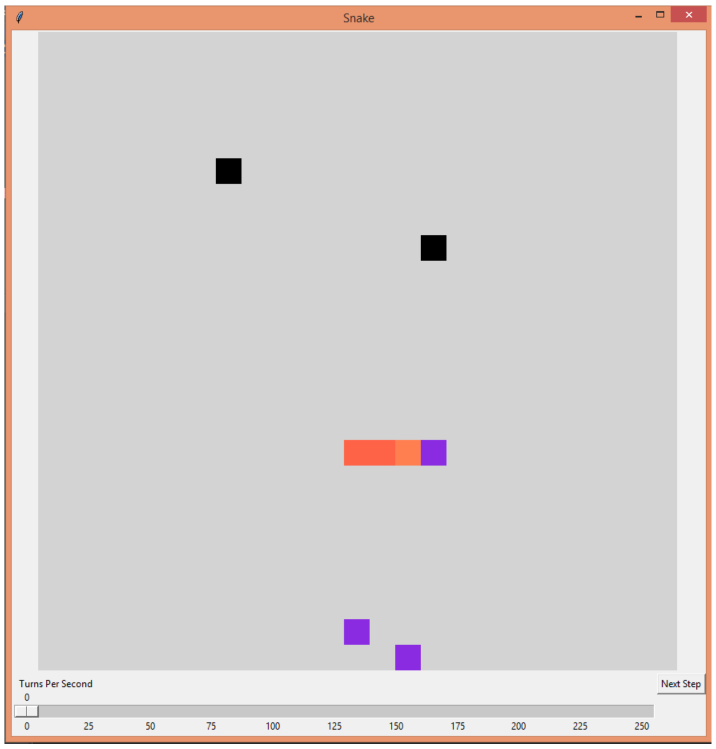

Figure 2.1: Typical board configuration for the default setting of the snake game. Black squares are obstacles, orange squares the snake, and the purple squares indicate food. The slider below regulates the update speed of the game. For a slider value of 0 one can use the `Next Step` button.

Assignment 1: A^* Search

As you may know the goal of snake is to eat as much food as possible. In this assignment you have to design and implement A^*-search to come up with a plan to go from the current state to a goal state in which the snake catches the food by placing the head on the square of the food location.
In your report you clearly describe how you model the above goal and problem formulation. Pay special attention to:

- How you translate the percepts of the agent, described in `agent.py`, into an *internal* state space model of the agent.
- The model for the available actions in each state space, i.e. the transition function.
- How you model and deal with the multiple instances of food.
- How you model the body of the snake and how the position of the body is updated.
- What if the A^* algorithm returns a failure and can this be solved?
- The used cost function and heuristic function.
- Game playing performance.

For the implementation of A^* you may use a standard piece of software available on the internet, but you may also implement your own A^* algorithm. Your design and implementation should reach at least an average performance of 90 (average over 30 trials) on the standard configuration. The reason for the average is the randomness in the placement of new food and the initial position of the snake.

Assignment 2: Complexity of A^* Search for the Snake game

For this assignment you have to investigate the ***time*** complexity of your implementation of A^* search for different sizes of the board and numbers of obstacles.

Adapting the board configuration

The file `main.py` contains the main settings of the game. In the code the meaning of the variables, such as `board_width, board_height, food_blocks_max` is clearly explained. Moreover the game settings can be found between `BEGIN GAME SETTINGS` and `END GAME SETTINGS`. If you want to test with your new configuration then it is **important that `test_config` is set to False**

Part a

Describe and motivate the theoretical expected time complexity your A^* search in terms of board size. You may use the so-called *O big-Oh* notation, see Appendix A of the course book.

Part b

Investigate the time complexity of your A^* search (not the time complexity of the execution of the found plan) for increasing board size. You may assume a square board and determine the time complexity for board sizes ranging from 25 to 75 with steps of 5. For each size take the average over at least 30 trails.

Part c

Investigate the time complexity of your A^* search for increasing obstacle complexity (to be defined by yourself). For each setting take the average over at least 30 trails.

Assignment 3: Applying Reinforcement learning to the Snake Game

The goal of this assignment is to apply Reinforcement Learning (RL) to the Snake game. We will start with simple environment and increase the complexity gradually.

Part a

Reduce the board size to $5x5$ and that only one piece of food is available. For this part the goal is reached when the snake "eats" the food. So we consider only snakes of length 1. You can adapt the game in such a way that the snake does not grow when "eating" food, see the `README` of the new version.
Apply RL to learn good strategy for finding the food. Clearly describe how you model the RL:

- Model of the state space.
- Reward function.
- Discount factor.
- Using U-learning or Q-learning and why.
- Learning parameter(s).

Moreover clearly analyze if the agent *really* learns and the influence of the learning parameters on the learning speed.

Part b

How would you model the RL for the default game of Assignment 1? How to deal with the state explosion?

Part c

Apply RL to the default configuration of Assignment 1 and evaluate and compare the learning and game performance. Clearly describe your approach!
A possible approach is a kind of hierarchical learning. Once the snake has learned a good strategy for level 1 (Part a) freeze it, meaning whenever the snake has length 1 we apply the

learned strategy and do not adapt anymore. Moreover one can copy this strategy also for the snake of length 2 but for this level the snake can still apply learning. After learning this level also freeze the strategy for this level and copy it to level 3. And repeat the learning.

Chapter 3

Pacman game: Not mandatory for passing the challenge

This part is not mandatory for passing the challenge, it is just for fun and to challenge you, see also Section 1.1. Moreover if there are at least 5 teams participating we will organize a competition afternoon and for the best three teams there is a nice price after the tournament. The Pacman game for which you need to design, develop and implement an intelligent agent, or even a team of intelligent agents can be found at UC Berkely Contest: Pacman Capture the Flag.

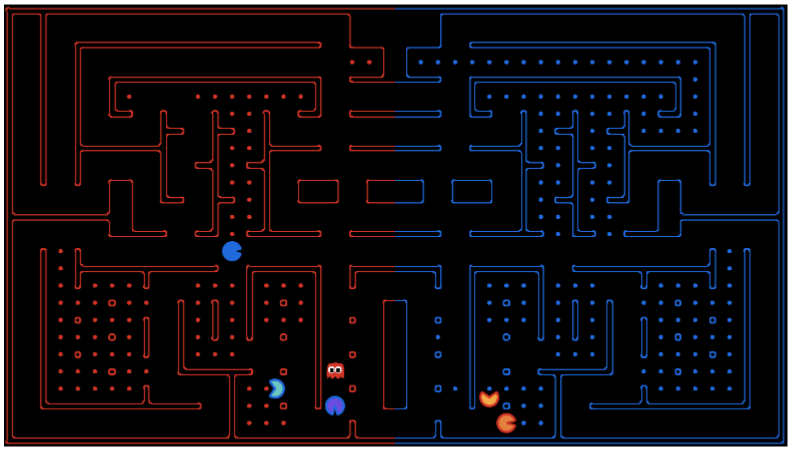

Figure 3.1: Typical board configuration for the two-player Pacman game.

More information, for instance the rules of this version of the Pacman game, and the code (in a zip-archive) is available on this webpage. Some useful hints:

1. Game runs under Python 2.
2. The only files you really need to understand are:
   - `capture.py`, especially the GameState class.
   - `myTeam.py` in which you can program your intelligent agent(s).
   - `captureAgents.py` this class is used in `myTeam.py` and gives access to a lot of useful functions.

- `baselineTeam.py` a baseline team, which can be used as an example and inspiration.

3. Comment the function `observerFunction` in `captureAgents.py`, this gives access to the full game state.
4. Game runs very fast, if needed you can add a small `sleep` in the `run` method van `game.py`.
5. Read the *Getting Started* section on the webpage of the game.

3.1 Pacman Challenge

The challenge you face is to design, develop and implement an (team of) intelligent agent(s) which can play this game in an intelligent (rational) way, meaning obtaining a high as possible score. But of course this depends also on the opponent, so you can design several agents and test which one is the best in a small tournament.

3.2 Tournament

On the afternoon of Tuesday January 29 (see roster for definitive date, time and location) we will have a tournament in the Design Lab between the different teams who participate in the challenge. You need to sign up on Canvas (sign-up list will be available in due time) for participation.

3.3 Requirements report

For this part you need to describe in a scientific way how you tackled this challenge. An important component in this part of the report is the methodology:

1. What is the used performance criteria/measure.
2. Which AI techniques are used in the design of your Intelligent Agent (IA).
3. If applicable: How are the optimal settings or parameters for your techniques determined?
4. How is the *intelligence* of you agent(s) evaluated?

And at the end you need to describe your final solution and design in scientific terms.

Chapter 4

Report

Your have to deliver a scientific report in pdf format containing your names and student number in which clearly describe how you solved the different assignments on a conceptual level. Your report should contain no code but if needed may contain some snippets of pseudocode to clarify your solutions. **Deadline for final report can be found on Canvas.** Requirements for the Snake game part of the report can be found in Chapter 2.