CS420 – Artificial Intelligent

Project 01 – Hide and Seek

Team members:

18125006 – Phạm Lê Thuỳ Dung

18125016 – Tăng Lợi Phát

18125035 – Nguyễn Thành Đạt

18125038 – Hoàng Nguyên Khôi

1. Introduction

In this project, we aim to implement intelligent agents who involve in the game called Hide and Seek, using basic searching and optimization techniques. The project consists of 4 different levels and is developed in four weeks. Details of tasks and assignment are listed in section 2.

We develop the project on Python 3, with Tkinter library for graphical illustration. Section 3 and 4 presents the project structure and describes how to launch the program. Basically, we use a different approach for each level. However, their main idea is for the seeker to minimize his path, and the hider to maximize the seeker’s number of turn to travel. Due to different specifications for each level, there is a variant in how we approach it. Details of them are described in section 5.

Moreover, several experiments on different maps for each level are conducted, and the results are evaluated in section 6. Section 7 presents our conclusion.

2. Assignment Plan

Table I describes the project plan and team assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Team member** | **Start** | **End** |
| Clarify project requirements | Tăng Lợi Phát | 8/11/2020 | 9/11/2020 |
| Setup github  Write assignment plan | Tăng Lợi Phát | 9/11/2020 | 10/11/2020 |
| Work on solution for level 1, 2 | Nguyễn Thành Đạt  Hoàng Nguyên Khôi | 11/11/2020 | 13/11/2020 |
| Generate sample maps  Find graphics asset | Nguyễn Thành Đạt | 11/11/2020 | 13/11/2020 |
| Implement graphics | Phạm Lê Thuỳ Dung | 11/11/2020 | 16/11/2020 |
| Maintain the graphics | Phạm Lê Thuỳ Dung | 17/11/2020 | 29/11/2020 |
| Implement class Game for general game management | Hoàng Nguyên Khôi  Tăng Lợi Phát | 14/11/2020 | 16/11/2020 |
| Implement level 1 algorithm | Hoàng Nguyên Khôi | 17/11/2020 | 19/11/2020 |
| Fix and test level 1 | Tăng Lợi Phát | 19/11/2020 | 20/11/2020 |
| Document level 1 | Nguyễn Thành Đạt | 20/11/2020 | 22/11/2020 |
| Implement level 2 algorithm | Hoàng Nguyên Khôi | 20/11/2020 | 22/11/2020 |
| Work on solution for level 3, 4 | Tăng Lợi Phát  Phạm Lê Thuỳ Dung | 21/11/2020 | 22/11/2020 |
| Fix and test level 2 | Tăng Lợi Phát | 23/11/2020 | 24/11/2020 |
| Document level 2 | Nguyễn Thành Đạt | 23/11/2020 | 25/11/2020 |
| Implement level 3 | Phạm Lê Thuỳ Dung  Tăng Lợi Phát | 23/11/2020 | 25/11/2020 |
| Fix and test level 3 | Tăng Lợi Phát | 26/11/2020 | 27/11/2020 |
| Document level 4 | Nguyễn Thành Đạt | 27/11/2020 | 28/11/2020 |
| Implement level 4 | Phạm Lê Thuỳ Dung  Tăng Lợi Phát | 28/11/2020 | 2/12/2020 |
| Fix and test level 4 | Tăng Lợi Phát | 3/12/2020 | 4/12/2020 |
| Document level 4 | Nguyễn Thành Đạt | 4/12/2020 | 5/12/2020 |
| Revise report | Tăng Lợi Phát | 5/12/2020 | 6/12/2020 |

3. Environment and project structure

Our project is written in Python 3 with Tkinter library for graphcial illustration. We allow two input modes: graphical and nongraphical mode. The nongraphical mode is used to test the efficient of the algorithms with large map size. For better illustration, we recommend that the input map of the graphical mode should be limited to 15x40 (n <= 15, m <= 40).

To experiment with different algorithms for different levels, we divide the project structure into four folders, each folder correspond to one level. All the map that we use for testing and demo are kept in ./map. The assets for graphical illustration are kept in ./asset.

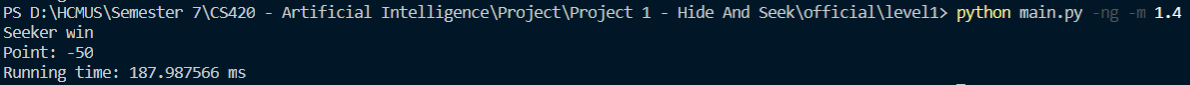
Each level consists of different classes, mainly the Seeker and Hider classes inheritted from Player. All of the agents are managed by Game class. This class is also responsible for passing information to the Gui class for graphical illustration. The config for the game can be stored in class Config.

4. Project setup

First, we should navigate to the specific level (e.g. cd level1). Then, the program can be launched by the following command: python main.py (-ng) -m <map name>.

The -ng flag is added in case we do not want the graphical result to be displayed. In addition, the game winner and seeker point are also presented right before the program terminated. In the nongraphical mode, the total running time are also displayed.

Here is an example:



5. Approach

5.1. Level 1

For this level, there is only one fixed hider that announce randomly in an 7x7 area after 5 turns. Moreover, the seeker is penalized for each irrational move. Therefore, we decide to let the seeker wait 5 turns to verify the surrounded area whether there is an announcement or not. Once the seeker observes either the hider or her announcement, he can cut down the search space to a small number of cells. Then, a simple BFS is enough to catch the hider.

In the unlucky case, the seeker cannot hear the announcement and see the hider, he need to make a random move. We decide to use the heuristic value to help the seeker decide his move, where the heuristic is calculated as the number of nonempty adjacent cells minus the Manhattan distance from that cell to the seeker. We want the seeker to prioritize cells in the corner, while making sure that nearby cells are checked.

In case the hider can hear the announce and know for sure the hider possible positions, but cannot get there, or when all of possible cells are visited, we let the seeker give up. This case happens when the hider are trapped in a disconnect area from the seeker.

5.2. Level 2

5.3. Level 3

Level 3 is an extended version of level 2. Given the hiders the ability to move, the seeker need to refresh the visited map everytime he found a hider. The reason is that a hider might move to the cell that he has visited while he is finding for other hiders.

On the hiders viewpoint, we allow them to know to initial position of the seeker. Combining with the knowledge of the map, they can calculate the shortest path from the seeker initial position to their current positions. Then based on that distance, we let the hiders stay for a short period of time, until just right before the seeker see them. At that time, they begin to move to other position.

The heuristic value that we use for the hider is a combination of number of non-empty adjacent cells, the Manhattan distance to the current hider location, the Manhattan distance to the current seeker location (in case they can see the seeker – the game turns to a chasing game), and the number of steps from the seeker initial position (using pre-calculated BFS map).

To make sure that the chasing game will end, we implement a special rule. When the seeker is saw by a hider, that hider is slown down. It means that they only move one in two consecutive turns. At this level, hiders can only move to decrease the point of seeker. Therefore, we do not use complicated heuristic or heavy memory use to keep track of possible candidates of the seeker next move.

5.4. Level 4

6. Experimental results

6.1. Level 1

6.2. Level 2

6.3. Level 3

6.4. Level 4

7. Conclusion