Teach the Pacman with Searching and Optimization

Zilong Liang
School of Mathematical Sciences
zlliang15@fudan.edu.cn

1 Introduction

How to teach a Pacman to find a path to his success? In the context of Artificial Intelligence, giving wisdom to the Pacman can be formulated as a search problem. Figure 1 shows a basic version of the game, where the map is a table of positions and the Pacman can move towards four directions. Obviously, it is intuitive to adapt graph search algorithms here.

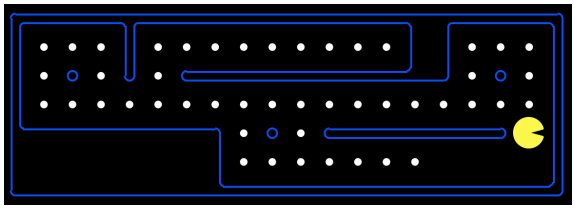

Figure 1: An example of Pacman game

However, the goal of the Pacman often varies, like looking out a shortest path to a fixed position, eating all the dots or getting rid of all the ghosts. As a result, we have to help the Pacman always keep his goal in mind. Then, solution of the problem can be divided into the following two steps:

- Construct an abstract graph (or state space), including the map itself and status of goals. Fortunately, in this project we can access the information of the Pacman world at any time, like the positions of walls and food dots.

- Adapt graph search algorithms on the abstract graph, to find a path with least cost to the goal. In this project, least cost often means fewest actions.

According to the project instruction, we should formulate several goals and implement several search algorithms. To make these two parts cooperate seamlessly, we define the problems according to a single interface in `searchAgents.py` and implement general versions of search algorithms in `search.py` which is able to solve any problem through the interface.

Consequently, this project consists of four working parts:

1. Read the existing codes and understand the framework that every formulation of search problems should follows.

2. Implement general versions of Depth First Search (DFS), Breath First Search (BFS), Uniform Cost Search (UCS) and A* Search.

3. Formulate two search problem that the Pacman should touch all four corners of the maze and the Pacman should eat all the food dots. Also, we should design appropriate heuristics for these problems.

4. Solution of suboptimal search, due to the heuristic-driven A* Search is hard to find the optimal path sometimes.

2 Framework of Search Problems

In line 22–62 of `search.py`, an abstract class gives methods that every search problem has to implement. And by reading the problem class `PositionSearchProblem` in `searchAgents.py` which defines a problem that the Pacman should find the shortest path to a fixed position, we can understand data structures used in the problem classes.

Firstly, the state space consists of tuples of states, every tuple including sufficient information like current position and positions of rest food dots. A start state is given by the method `getStartState`. Then, the method `getSuccessors` gives a list of successors of a state, which is a list of tuples in the format `(next state, move direction, move cost)`. Through this method, we can construct the state space recursively (lazy loading), avoiding compute the whole space at the beginning, which is often impossible in finite time. Finally, the method `getCostOfActions` returns total cost of a list of actions, and `isGoalState` determines the goal state, which is the terminal rule of a search algorithm.