Ms Pac-Man vs. Ghosts

Jiacheng Geng, jg4754

October 25, 2016

Abstract

This is a report for applying tree search including depth-first search, breadth-first search, iterative deepening and A*, and optimization methods including hill-climber, simulated annealing, mutation and crossover to the Ms Pac-Man vs. Ghosts in the game state space, recording their performance, analyzing their limit and giving out some possible solving or improving methods for future.

1 Introduction

Ms. Pac-Man is a very interesting game and many people try to create AI agents to play the game automatically.There can be several different targets for the game like living longer from ghosts, eating more pills, or getting higher score. Besides, there can also be different priorities like avoiding ghosts no matter what or eating power pills no matter what. Moreover, there can be different basic algorithms in their single pure form or complex hybrid form. As a result, after we combine so many different elements together, there will be a lot of different agents and the corresponding game result. In this report, I will provide my version of different methods including depth-first search, breadth-first search, iterative deepening and A*, hill-climber, simulated annealing, mutation and crossover required in the assignment in their pure form no matter how bad they perform, record their performance, analyze their problem and corresponding possible solution one by one.

2 General implementation idea

First, I am doing all the tree search method in game state space, which means the tree node is a certain game state and they are connected by a possible move. And for optimization, I am doing mutate, crossover or other operation upon movement sequence and judging from the game state the sequence reach from the present game state.

Second, I am doing all the algorithm in their pure and form, which means there won't be any combination of two algorithm or two steps of algorithm. For example, in the StarterPacMan, at first, it searches and tries to run away from the nearest ghost when it may be eaten by the ghost. Then it tries to eat a ghost when the ghost is edible. Then it tries to eat pills and power pills. This is a three-step combined strategy. It has priority from high to low, which is complex. Such combined strategy never happens in my strategy, each of my strategy is pure and simple no matter how bad they perform.

Third, I simply use score for criteria or fitness for all of my algorithm except A*. We will talk about A* in its case later.

3 Different algorithm and result

3.1 Depth-first search

I just followed the example given on the note. I limited a certain depth, 12,then used depth-first search to get all the possible game state after 11 movements, discarding those game state where Pac-man was eaten by ghosts. Then I compared all the path, chose the path that leading to the highest score. In the end, the first movement of that path would be my choice for this step.

This strategy usually have more than 1000 score and near 1500. It will go away from a ghost

when the ghost is approaching it in order to survive. Meanwhile it may jam in some jeopardy if there are two paths leading to a same score(usually in no-pill-around case).

3.2 Breadth-first search

Breadth-first search is just like depth-first search, I just replaced the stack with queue with the same depth limitation, 12.

This strategy also performed almost the same as depth-first search.

3.3 Iterative deepening

Iterative deepening is just a growing-limit depth-first search. I applied depth from 1 to 10 and still chose the highest-score path and used its first movement. A slight technical difference is that Pac-man may get eaten for a higher score if he has no choice but to die in the limited depth.

This strategy also performed almost the same as depth-first search.

3.4 A*

In my opinion, this is actually a bad algorithm if it is used in its pure form. The A* we talked about on the class is aiming at finding a shortest path to a certain goal. Meanwhile, in Pac-Man, the goal may be the state where all pills are eaten, so the pills you can eat is always a constant, and there is no "shortest path" or any "least cost"in any simple definition I can figure out. So, I can only choose my heuristic function as

$$h[state] = 10 * (pills\ left\ to\ eat) - 500 * (ghosts\ just\ eat)$$

and cost function as

$$g(state) = \begin{cases} 100000 & \text{if Pac-Man is eaten} \\ 0 & \text{if Pac-Man is not eaten} \end{cases}$$

Under this, I can choose the smallest node to expand, and the depth limit is 10. The result is a little better than the three algorithms above, around 2000 maybe because it really wants to eat a ghost when it can. But it jammed in some jeopardy happened for more times.

3.5 Hill-climber

For hill-climber, we are actually evaluate the movement sequence. So I generate a random movement sequence with 10 step movement as the initial movement. Then I defined the movement sequence with only one step movement different from the original movement sequence is called the neighbor of the original sequence. So all the neighbor of my movement sequence is all those movement sequence have exactly one different movement at exactly one step(could be 1 to 10) from my original sequence. I measure the score after applying the sequence to the game state and use it to judge the goodness of the sequence. If no neighbor is better, I choose the first movement of current sequence as my choice. Otherwise, I choose the best neighbor to "climb" until the round limitation is reached.

The result is very bad. The pac-man was always going forth and back while moving slowly on some certain path full of pills and it never avoided ghosts. The score it got is usually only near 800.

3.6 Annealing

All other elements are the same as hill-climber. But I only choose one neighbor to measure this time. I set T as 1000, which is a constant. And repeat until the round limitation is reached.

The result is extremely bad. Instead of slowly moving on some certain path full of pills, which is done in hill-climber, it just moved forth and back at the start point and waited for ghosts to eat it. The score was only below 100.

3.7 Mutation

All other elements are the same as hill-climber. I generate the initial movement sequence population randomly with a size of 10 and each has a depth of 10. The fitness function is just the score after the sequence. The variation is just becoming one of the neighbor defined in hill-climber. I leave the best half of population as the survivors to mutate until round limitation is reached. Then I choose the first movement of the best sequence.

The result is just like hill-climber.

3.8 Crossover

All other elements are the same as mutation. I just randomly choose two sequence A and B from the survivors to produce a child by the first 5 movements from A and the last 5 movements from B and keep the population size constant until round limitation is reached.

The result is bad. I don't even know what's the point of this method, so how could it possibly make sense? The score is always under 300. And the Pac-man sometimes just moved forth and back near some certain point even there were some pills near it to eat.

4 General problem and their possible solution

4.1 Moving forth and back

All the algorithms have this problem. This can be solved simply removing the backward movement of the last movement from the possible movement choices. Meanwhile, if you do this, your Pac-man will never avoiding the ghost coming towards its face because you remove that choice at the first place. So this must be done by some complex structure like ghost detection, which is far beyond my pure form.

4.2 Avoiding ghosts

I only use score to measure in some of my algorithms, like those optimization algorithms. As a result, pac-man don't care about getting eaten. Meanwhile, surviving is at the most priority in this game. So ghost detection and avoiding must be introduced for better game result.

4.3 Random ghosts move

I am using StarterGhosts to advance my game. But there is some random part in its strategy. So if the ghosts' move in my simulation is different from in running. My pac-man may be in danger. My friend choose the most possible movement after getting the ghosts movements for several times for their algorithms. This method looks simple, but it is actually the knowledge in machine learning, which is even too far beyond this class.

5 Conclusion and future

As I mentioned in the begin, I only try pure form of each algorithms assigned. Generally, searching is better than optimization in my way. Maybe it's because I can judge condition more easily in the former. But they are all not so good. As I analyze, three major problems and solutions are listed. But they all need more complex structure. When too many layers of decision are added onto a simply DFS, is the DFS actually working? Or is the layers added doing most of the decision? This is another question. As a course assignment, I can't go that far. So I prefer to strictly stick to the assignment demands, write a algorithm as purely as I can and just see the result rather than spend unlimited time and add unlimited possible methods for better performance. But they can all be done in future.