UCLouvain

epl

LINGI2261: Artificial Intelligence

Assignment 3: Project: Adversarial Search (Part2)

Groupe 81
Sanae Abdelouassaa
Jiayue Xue

November 30, 2018

CONTENTS

CHAPTER 1

TAK GAME

1.1 Evaluation function

1.1.1 Question 5

What are the weak points of the evaluation function that you implemented for the basic agent?

Answer:

The evaluation function of the basic agent is designed to return the difference between the numbers of pieces that the two players control if the game is not over. However, this practice can never precisely reflect the real situation of the current game because it does not take some important properties of the TAK game into consideration. To illustrate, suppose one player A controls three pieces. In the first situation, the three pieces lies in the same line and are adjacent to each other, which is obviously more desirable more a situation where A controls three pieces but they lies in three different corner of the board. In addition, if one player controls one piece having a higher height, he may have a bigger chance to convert current game to one that he favours.

1.1.2 Question 6

As described in the class, an evaluation function is often a linear combination of some features. Describe new features of a state that can be interesting when evaluating a state of this game.

Answer:

We would like to add some useful features that are closely related to the TAK game. To be more concrete, we create one feature that counts the height of every piece. The intuition is quite straightforward: if one player controls a high stack, he can move one or more pieces in that stack so that he is able to capture more space. In addition, another possible feature we can add is to count how many adjacent pieces one player is controlling. The final goal of this game is to create a road, which is a string of one player's pieces connecting opposite sides of the board. In consequence, not only the number of pieces that one player controls, but also the way these pieces lie will be the key factors that influence the final result of the game.

1.1.3 Question 7

Describe precisely your evaluation function.

Answer:

As is described before, our evaluation function will first verify whether the game is over. If so, the function will return a flag variable indicating which player is the winner. If the game is still running,this function will count not only how many pieces one player controls, but also the height of the corresponding piece that will add another bonus score to the total score, which effectively provides useful instructions to the player in order to help him to choose a better action.

1.2 Successor function

1.2.1 Question 8

Give an upper bound on the maximum number of actions that can be performed in any given state. If you cant, explain how could you estimate what is the average number of successors and estimate it.

Answer:

In our game setting, our board size is five and each player has twenty-two pieces. Since we have one constraint that the maximum height of one stack is six. We assume the most extreme situation. We have seven stacks with six pieces in each stack. All the stacks are placed at the edge of one boarder

and are controlled by the same player A. Now A can move any one of these stacks. For each stack, he can move them to the opposite boarder with four or five pieces. If he takes four pieces, he can put them in the adjacent space or drop some of them and move to another space. The total number of possible movement is seven. If he takes five pieces, the same rules apply to this situation. In that, the total number of possible movement is fourteen. Finally, all the possible combination of the movement is $(7 + 14) * 7 = 147$.

1.2.2 Question 9

What do you loose by ignoring successors?

Answer:

We filter out effective successors by discarding successors with very low scores. The score corresponding to every successor is derived by calling the evaluation function. Since our evaluation function can honestly reflect the realistic situation, we is able to conclude that these successors with low scores will have little chance to be one of steps that leading the player to win the game. Out of this observation, we can simply discard them and only focus on the successors that are more promising.

1.2.3 Question 10

In what order should the successors be returned in order to help Alpha-Beta prune the search tree?

Answer:

Since we use the evaluation function to help us choose promising successors, we should return the successors in a descending order. For example, we should first return the successors with the highest scores.

1.3 Cut-off function

1.3.1 Question 11

The cutoff method receives an argument called depth. Explain precisely what is called the depth in the minimax.py implementation.

Answer:

First of all, the MiniMax algorithms explore the search space by utilizing a tree structure. In consequence, the "depth" variable represents our current level in the search tree. Furthermore, the optimal situation for MiniMax algorithms is that our search is able to reach the final state of the game. However, in the TAK game, we cannot search that deep in the search tree because it will consume quite a long time. To make a trade-off, we have to stop the search in advance and make a movement according to the state that we can find. In that, the "depth" variable also provides us with the most powerful evidence that help us decide whether continue the search procedure or stop right now.

1.3.2 Question 12

In the Tak contest your agent will be credited a limited in time.How do you manage that time? How can you make sure that you will never timeout? Explain how you can use iterative deepening to avoid timeouts.

Answer:

One simple implementation of the time management is that we limit every action to a certain number of seconds. However, in our point of view, this practice is not very sophisticated. The beginning of the game is so important. If we can make a wise action in the very early time, the action will directly give us many advantages. Out of this consideration, we arrange more time for the beginning actions. As the game continues, we will dynamically adjust the time limit. For example, when we just have a fifty seconds remaining, we will decrease the depth that we want to search. If we just have twenty seconds, we will make actions by only search one level, which is so rapid that it can guarantee that our agent will never time out.

1.3.3 Question 13

Describe your cut-off function.

Answer:

We improve our cut-off function together with the get-action function. At the beginning of the game, since we still have a lot of time, we tend to explore as many levels of the search tree as we can so that we can make the optimal action in our limited observation. As time is consumed gradually, we will

decrease the search level so that we can still make rational movements and save enough time to defeat our enemy. When we just have several seconds remaining, we will adjust the search level to one which can totally guarantee our agent will not time out.

1.4 Contest

1.4.1 Question 16

Describe concisely your super tough agent in your report. Your description should not be longer than two A4 paper pages.

Answer:

In order to create a tough agent to compete with others, we utilize all the aforementioned methods to improve our agent.