Report of Theory of Computer Games project 4, 2022

109550027 紀竺均

a. Implementation

Based on project 3, I implemented a RAVE MCTS and improved my time management function.

I create two classes – “MCTS” and “mcts_agent” which inherited from class “agent”.

In MCTS class, there are four mainly function for constructing MCTS tree – select, expand, simulate, and update. And a function “best_action” that decide the action based on the tree.

In class “MCTS”:

1. select():

   In function “select”, I select the current node’s child with the highest uct value. I calculate the uct value using MC-RAVE:

   Beta function: (set b = 0.025)

$$\beta = \frac{\tilde{n}}{n + \tilde{n} + 4n\tilde{n}\tilde{b}^2}.$$

```
float exploitation;
if(myturn)
    exploitation = (1 - beta) * winRate + beta * raveWinRate;
else
    exploitation = (1 - beta) * (1 - winRate) + beta * (1 -
raveWinRate);
float exploration = sqrt(log(cur_visittime) / (float)(child.visittime +
1));
exploitation + c * exploration;
```

2. expand():

   In function “expand”, I append every legal move in current space to the node’s child list.

3. simulate

   In function “simulate”, I apply the legal moves based on rand_move function, which return a random legal movement drawn from current space, and return the win/lose result.

4. update():
   (1) Update current node's visit time and the count of win time.
   (2) Update RAVE visit time and RAVE win time based on a vector that store the position and child node.

5. best_action():
   If the childs list is empty, it means that there isn't any legal move for current root, so we simply return a random action.
   If there are childs for root, we could choose the child with greatest visit time to be our next position.
   Once we have the next position, we can traverse thru current space to find what is the legal move that can lead to that position, and finally return that legal move.

b. Improvements

(1) Two player MCTS

Then, I tried to improve the function by adding who's turn is it while calculating uct value because if it's the opponent's turn, their win rate should be our loss rate (1-winrate).

(2) MC-RAVE calculation

Considered that the position matters more than the time we placed it, I use RAVE to calculate UCT value. Detailed implemented method is in part(a) select() function.

(3) Time distribution

Because there are total 73 spaces in the board, there are at most 73/2 = 37 actions that we have to think. Considered the 300 seconds thinking time, I tried to spend more simulations in the middle of the game.

```
 if(sims_count<=3) clocktime = 4;
        else if(sims_count<=8) clocktime = 7;
        else if(sims_count<=15) clocktime = 10;
        else if(sims_count<=20) clocktime = 11;
        else if(sims_count<=25) clocktime = 9;
        else if(sims_count<=30) clocktime = 5;
        else clocktime = 3;
```

(Here, the sims_count is the number of actions that I have placed.)