VNU HCMC-UNIVERSITY OF SCIENCE
INFORMATION TECHNOLOGY FACULTY

REPORT LAB02
ADVERSARIAL SEARCH

Lecturer&Instructor:

- Teacher Phạm Trọng Nghĩa
- Teacher Nguyễn Thái Vũ

Subject: Cơ sở Trí Tuệ Nhân Tạo

Class: 20CLC04

Student: Trần Quang An Quốc

Student ID: 20127304

Thành phố Hồ Chí Minh, ngày 28 tháng 06 năm 2022

I. Introduction to the Adversarial search.

Adversarial search is a search which we examine if the problem can arise when we try to predict the whole game. At the same time, other agents tend to against us.

In real life, there maybe some situations where more than one agent participate into the environment so as to search for the solution in the same search space, and this usually occur in the competittive gameplay.

According to Lab02, applying adversarial search to help the computer find the optimal path in caro gameplay with board size 3x3 and 5x5, and I found out that minimax algorithm can solve this problem.

II. Minimax algorithm

Minimax is a kind of backtracking algorithm which is used in decision making and game theory to find the optimal move for a player, assuming that opponent also play optimally. According to tic-tac-toe, minimax is a good method to solve this problem. However, we need to improve as minimax still cause space and time wasting.

1. The idea of the algorithm

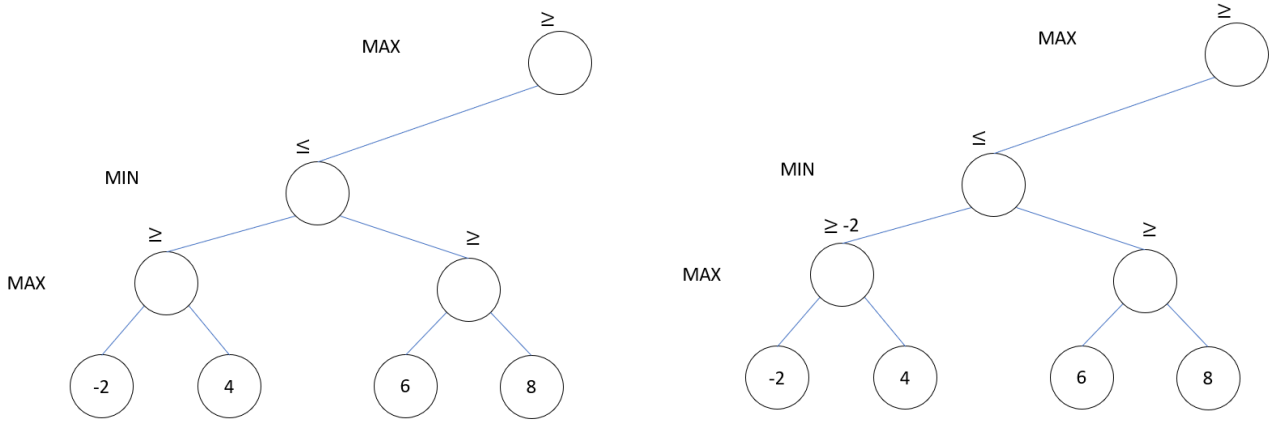
In minimax, two player are called Maximizer and Minimizer. The Maximizer tries to get the highest score possible at each state while Minimizer tries to do the opposite to restrict the Maximizer score (It's mean the lower the Maximizer score is, the better Minimizer get).

The minimax algorithm perform by evaluating each possible state, if this state belong to the Maximizer, we set the maximum value for this state and inversely, the state belong to the Minimizer will be set the minimum value so as to restrict the score of Maximizer. By this evaluation, we can find out which is the optimal state for current step.

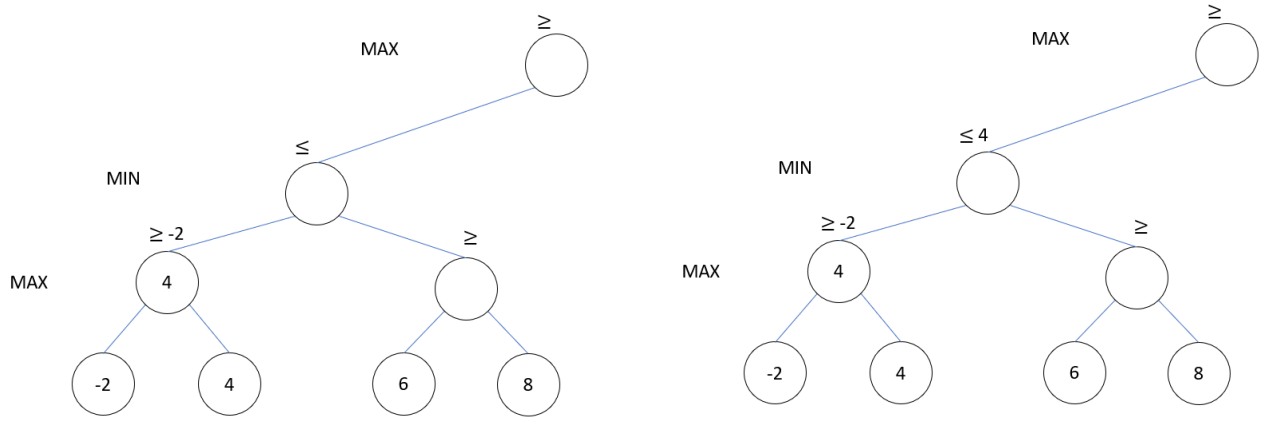
In detail, for each line when one player marks, if in that line the opponent marks, the score would minus itself (We call Minimizer), and when current player marks on the line, the score increase again (We call Maximizer). After all, the computer will choose the way that return the coordinates whose score is the highest one and that is the best move for this current state.

As mentioned above, minimax cause space and time wasting, so we need to improve it. *Therefore, regarding to board size 3x3*, I use the improvement of the minimax algorithm called alpha-beta pruning which prun away branches that cannot possibly influence the final decision or there already exists a better move available. It can reduce a huge of unnecessary situations and allow us to search much faster or explore deeper in the same duration in the game tree.

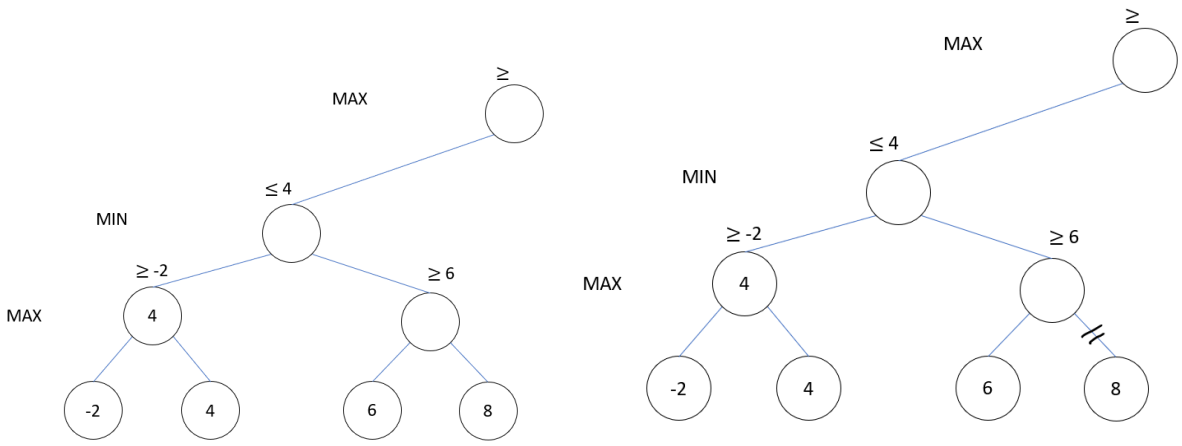
For example, here is a half of binary tree so as to stimulate how alpha-beta pruning perform:



Now we consider the leaves from the left to the right side, at the start we can see the first leave is -2, if there is no number in equation, let's compare the Maximizer and Minimizer node with negattive infinite or positive infinite sequently. The equation of each node depend on if the node is Maximizer or Minimizer, if Maximizer we need to find the bigger value, the other is the smaller one. In this case the parent node is Maximizer, so we have to find the bigger one. Value -2 is larger than negative infinite so we add it into the equation.



The next leave has value 4, and then we check, 4 is bigger than -2, and because this is the binary tree so each node have two nodes, so the parent node of these 2 leaves is 4. Let's see above, 4 is smaller the the infinite number, so we add it into the equation of above Minimizer.



Now turn into the next leave, we have value 6 bigger than negative infinite, so add it into the equation. From now, here is the difference between normal minimax and minimax with alpha-beta pruning, we can see that 6 is now in the equation, and the next leave have to have the bigger value than 6. However, look at above, the parent node of current Maximizer is Minimizer, and it have to smaller than 4, so instead of going deeper in this branch, we don't need to find out more as no number in this branch suitable.

Note: My example is just binary tree, so if we cut off 1 branch in this case may not make any differences significantly. However, in the gameplay, like tic-tac-toe 3x3, the tree have many branches, 9 branches for the first turn. For the next turn, it will have 7 children for 9 parrent node, and so more deepenly. This may takes time and space to solve the problem and now alpha-beta pruning is really efficient.

According to board size 5x5, Alpha-beta pruning can't run although it has improvement as there are too much states for each steps. So I have to combine this algorithm with Iterative deepening search with the limited depth and time search. Because of the limitation, it becomes imperfect real time algorithm. The precision of the algorithm now depending on how long we let it run, maybe it can find the real best move or sometimes the result is just the best move in current moment and return inaccurate value. Therefore, 5x5 board may not be optimal in all cases, we need to test and set the suitable search time for the algorithm.

Moreover, the way 5x5 evaluates the score may have some differences, as in 5x5 board, there are many cases to calculate the score in the line as we just need 4 adjacent elements to win the game, it's much more complicated than 5x5 with 5 adjacent elements to win. So the idea to evaluate for 5x5 in this situation is that take successively line whose length is 4 and then check 4 adjacent elements of that line. It may takes more time to calculate because line with length 5 have to separate into 2 line with length 4 and calculate 2 times. But surely it's easier to make sense than checking various cases lead to winning and calculate those different cases.

2. Completeness

3x3 is a small board, alpha-beta pruning application in this playgame is really efficient both in space and time. Actually, if the opponent play well enough, it may cause draw in the game. Anyway the completeness is still YES.

5x5 is a large board, actually alpha-beta pruning with the limitation in time and space cannot optimize all the path lead to win. The completeness depend on the time we let the computer search. But I think it's still YES if we have suitable time search for the computer.

3. Time/space complexity

Minimax:

- Time complexity: a^b
- Space complexity: a*b

Minimax alpha-beta pruning:

- Time complexity: $a\hat{}(\frac{b}{2})$ in the best case, the worst case is the same as normal minimax
- Space complexity: a*b, the same as normal minimax

Minimax alpha-beta pruning with limitation depth and time:

- Time complexity: depend on the search time we limit
- Space complexity: a*d with d is the current depth at the current time limited.

➔ a is the branching factor (all path to the goal) and b is the maximum depth of the tree.

III. References

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

https://stackabuse.com/minimax-and-alpha-beta-pruning-in-python/?fbclid=IwAR1uj2ONL4mic_TBXW-GFlIFhaIFXiqwquD0VEvggli7_Yd1-LnBOSI2myM

https://github.com/AlejoG10/python-tictactoe-yt/blob/master/tictactoe.py?fbclid=IwAR3r92PUCdMDLSJhWUDYHIZ2wVgmRoScLXw9LA7N133ObEmx9upEucT88jA

https://codefly.vn/tic-tac-toe-5-x-5-in-python-voi-ma-nguon/7446?fbclid=IwAR21UUIovOG_V4E5VimprCpD0Ktg7tmojf7izMmmW2_UGtJefp8TLRDq-dw

https://stackoverflow.com/questions/16328690/how-do-you-derive-the-time-complexity-of-alpha-beta-pruning

Link demo application (Youtube):
https://www.youtube.com/watch?v=gcLGDtelMRg