ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ

BÀI TẬP LỚN

Học phần : NHẬP MÔN AI

CHỦ ĐỀ: TIC TAC TOE SỬ DỤNG THUẬT TOÁN MINIMAX

NHÓM BÁO CÁO: NHÓM 11

SINH VIÊN LÀM BÁO CÁO: LÊ SỸ VĂN
NGUYỄN VĂN HÙNG
NGUYỄN TRỌNG TRƯỞNG
PHẠM TRUNG KIÊN

Hà Nội - 2023

Mục Lục

Mở Đầu 3
Chương I. Các thuật toán 4
1. Thuật toán Minimax 4
2. Cắt tỉa Alpha-Beta 4
3. Giới hạn độ sâu 5
Chương II. Ứng dụng trò chơi Tic Tac Toe 7
1. Thuật toán Các hàm chức năng chính trong game 7
a. Minimax 7
b. Minimax Alpha-Beta 9
c. Giới hạn độ sâu 11
2. Demo 13
Kết Luận 19
Tài Liệu Tham Khảo 21

Mở Đầu

Trò chơi Tic Tac Toe là một trò chơi đơn giản, nhưng nó cũng là một bài toán thú vị trong lĩnh vực trí tuệ nhân tạo. Mục tiêu của bài toán là tìm cách để máy tính chơi Tic Tac Toe với chiến lược thông minh, trong đó máy tính sẽ luôn chọn nước đi tốt nhất để đảm bảo thắng cuộc hoặc giữ cho trận đấu hòa.

Trong đề tài này, chúng ta sẽ tập trung vào việc tối ưu hóa thuật toán Minimax trong trò chơi Tic-Tac-Toe bằng cách áp dụng các kỹ thuật như cắt tỉa Alpha-Beta và giới hạn độ sâu tìm kiếm. Bằng cách kết hợp những cải tiến này, mục tiêu của đề tài là nâng cao hiệu suất và hiệu quả của thuật toán Minimax trong việc tìm kiếm nước đi tốt nhất trong trò chơi Tic-Tac-Toe. Các cải tiến này giúp giảm số lượng nút được khám phá bởi thuật toán và tăng tốc độ tính toán, đồng thời giúp máy tính đưa ra quyết định chính xác hơn trong thời gian ngắn hơn.

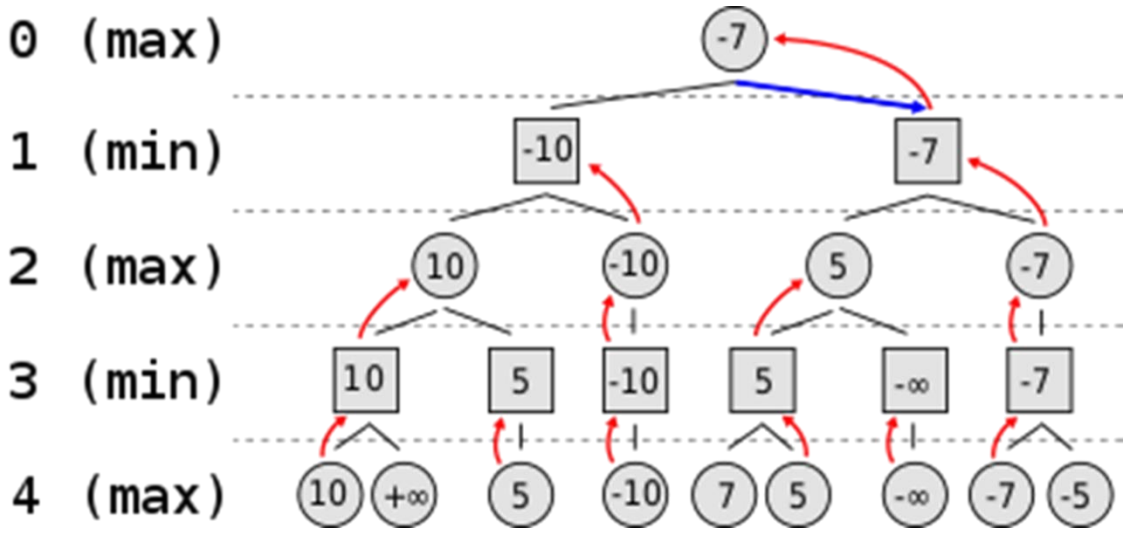
Chương I. Các thuật toán

1. Thuật toán Minimax

a. Khái niệm

Minimax là một thuật toán đệ quy lựa chọn bước đi kế tiếp trong một trò chơi có hai người bằng cách định giá trị cho các Node trên cây trò chơi sau đó tìm Node có giá trị phù hợp để đi bước tiếp theo.

b. Mô tả thuật toán



2. Cắt tỉa Alpha-Beta

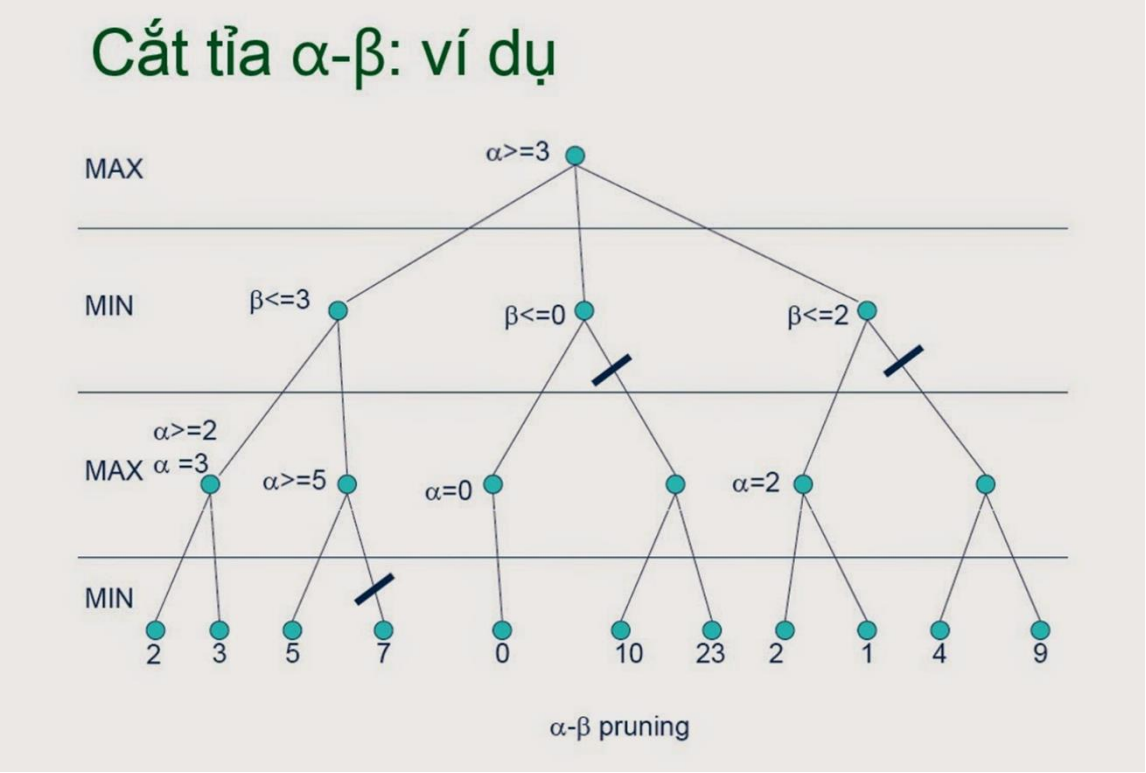
a. Khái niệm

- Alpha-Beta là một phiên bản sửa đổi của thuật toán minimax. Nó là một kỹ thuật tối ưu hóa cho thuật toán minimax.

- Thuật toán làm giảm số nút tìm kiếm để không lãng phí thời gian tìm kiếm những nước đi đã bất lợi rõ cho người chơi. Giải thuật Alpha-Beta cải tiến so với Min-Max bằng cách thêm vào 2 tham số là alpha và beta. Cho biết các giá trị nằm ngoài khoảng [alpha, beta] không cần xét nữa. Thủ tục Alpha-Beta được bắt đầu tại

nút gốc với giá trị của alpha là -100000 và beta là 100000. Thủ tục sẽ tự gọi đệ quy chính nó với khoảng cách giữa các giá trị alpha và beta ngày càng đẹp dần.

b. Mô tả thuật toán



3. Giới hạn độ sâu

a. Khái niệm

Trong trí tuệ nhân tạo hay các lý thuyết đồ thị, thuật toán tìm kiếm có giới hạn độ sâu (DLS) hay depth-limited search algorithm là một thuật toán phát triển các nút chưa xét theo chiều sâu nhưng có giới hạn mức để tránh đi vào những con đường không mang lại kết quả tốt như trong thuật toán tìm kiếm sâu dần.

b. Mô tả thuật toán

1. Đầu tiên, khởi tạo nút gốc và đặt độ sâu hiện tại là 0.

2. Tiến hành tìm kiếm theo chiều sâu từ nút gốc. Duyệt qua các nút con của nút hiện tại theo thứ tự xác định.

3. Nếu độ sâu hiện tại đạt đến giới hạn độ sâu, dừng lại và không tiếp tục duyệt nút con.

4. Nếu đạt đến một nút mục tiêu, trả về kết quả.

5. Nếu chưa đạt đến giới hạn độ sâu và chưa tìm thấy kết quả, tiếp tục duyệt các nút con bằng cách tăng độ sâu hiện tại lên 1 và thực hiện các bước 2-5 cho mỗi nút con.

6. Nếu không còn nút nào để duyệt và chưa tìm thấy kết quả, trả về kết quả không tìm thấy.

Chương II. Ứng dụng trò chơi Tic Tac Toe

1. Thuật toán Các hàm chức năng chính trong game

a. Minimax

```
def minimax(board, depth, isMaximizing):
        global i
        i = i+1
        print(i)
        result = checkWinner()
        if result is not None:
                return scores[result]


        if isMaximizing:
                bestScore = -100000

                for row in range(BOARD_ROWS):
                        for col in range(BOARD_COLS):

                                if(board[row][col] == 0):
                                        board[row][col] = 2
                                        score = minimax(board,depth+1,False)
                                        board[row][col] = 0

                                        bestScore = max(score,bestScore)

                return bestScore


         else:
                 bestScore = 100000
                 for row in range(BOARD_ROWS):
                         for col in range(BOARD_COLS):

                                 if(board[row][col] == 0):
                                         board[row][col] = 1
                                         score = minimax(board,depth+1,True)
                                         board[row][col] = 0

                                         bestScore = min(score,bestScore)

                 return bestScore
```

Hàm này thực hiện tìm nước đi tốt nhất cho người chơi hiện tại, dựa trên trạng thái hiện tại của bàn cờ.

Hàm này nhận 3 đối số:

bord: một danh sách hai chiều đại diện cho trạng thái hiện tại của bàn cờ. Giá trị trong bàn cờ là số nguyên, trong đó 0 là ô trống, 1 là người chơi điền, 2 là máy tính điền.

depth: một số nguyên đại diện cho độ sâu hiện tại của cây tìm kiếm. Hàm sẽ gọi đệ quy với độ sâu tăng dần cho đến khi đạt đến độ sâu tối đa.

is Maximizing: một giá trị boolean cho biết người chơi hiện tại có phải là người chơi tối đa (máy tính) hay không.

Hàm trả về điểm số tốt nhất cho người chơi hiện tại, dựa trên trạng thái hiện tại của bàn cờ.

Thuật toán hoạt động như sau:

- Nếu trò chơi kết thúc, hàm trả về điểm số cho trạng thái hiện tại của bàn cờ.
- Nếu người chơi hiện tại là người chơi tối đa, hàm thử mỗi nước đi có thể có trên bàn cờ. Đối với mỗi nước đi, nó gọi đệ quy với bàn cờ được cập nhật và độ sâu hiện tại cộng thêm 1. Sau đó, hàm chọn nước đi với điểm số cao nhất và trả về điểm số đó.
- Nếu Người chơi hiện tại là tối thiểu, hàm làm điều tương tự, nhưng chọn nước đi với điểm số thấp nhất.

b. Minimax Alpha-Beta

```
def minimax(board, depth, alpha, beta, isMaximizing):
    global i
    i = i + 1
    print(i)
    result = checkWinner()
    if result is not None:
        return scores[result]

    if isMaximizing:
        bestScore = -100000
        for row in range(BOARD_ROWS):
            for col in range(BOARD_COLS):
                if board[row][col] == 0:
                    board[row][col] = 2
                    score = minimax(board, depth+1, alpha, beta, False)
                    board[row][col] = 0
                    bestScore = max(score, bestScore)
                    alpha = max(alpha, bestScore)

                    if beta <= alpha:
                        break
        return bestScore


   else:
       bestScore = 100000
       for row in range(BOARD_ROWS):
           for col in range(BOARD_COLS):
               if board[row][col] == 0:
                   board[row][col] = 1
                   score = minimax(board, depth+1, alpha, beta, True)
                   board[row][col] = 0
                   bestScore = min(score, bestScore)
                   beta = min(beta, bestScore)

                   if beta <= alpha:
                       break
       return bestScore
```

Với thuật toán này ta sẽ được 2 số hạng được gọi là Alpha và Beta được cho vào hàm minimax.

Alpha tượng trưng cho nước đi tối ưu nhất(nước đi lớn nhất) của người chơi max:

alpha = max(alpha, bestScore)

Beta tượng trưng cho nước đi tối ưu nhất (nước đi nhỏ nhất):

beta = min(beta, bestScore)

Nếu bất cứ khi nào *alpha >= beta* thì ta sẽ dừng việc kiểm tra và không cần xét đến các nhánh còn lại.

if beta <= alpha:

break

Lý do: Bởi vì nếu ở hai nốt liên tiếp thì nó sẽ là hai lớp *max* và *min*, nếu như *alpha > beta* tức beta là đang nở node min vì vậy nếu như xét tiếp thì điểm của nở nốt này sẽ phải bé hơn beta mà trong khi đó alpha là lớp max sẽ lấy giá trị lớn hơn.

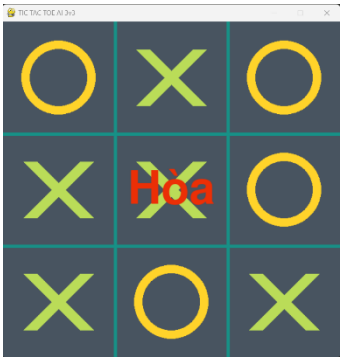
c. Giới hạn độ sâu

```
i = 0
def minimax(board, depth, alpha, beta, isMaximizing):
    global i
    i = i + 1
    print(i)
    n = 5
    # print(n)
    result = checkWinner()
    if result is not None:
        return scores[result]

    if isMaximizing:
        bestScore = -100000
        for row in range(BOARD_ROWS):
            for col in range(BOARD_COLS):
                if board[row][col] == 0:
                    board[row][col] = 2

                    if(depth > n):
                        board[row][col] = 0
                        break
                    score = minimax(board, depth+1, alpha, beta, False)
                    board[row][col] = 0
                    bestScore = max(score, bestScore)
                    alpha = max(alpha, bestScore)

                    if beta <= alpha:
                        break
        return bestScore
```

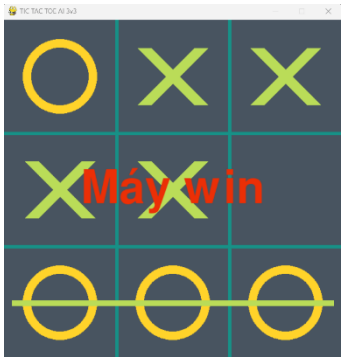
```
else:
    bestScore = 100000
    for row in range(BOARD_ROWS):
        for col in range(BOARD_COLS):
            if board[row][col] == 0:
                board[row][col] = 1
                if(depth > n):
                    board[row][col] = 0
                    break
                score = minimax(board, depth+1, alpha, beta, True)
                board[row][col] = 0
                bestScore = min(score, bestScore)
                beta = min(beta, bestScore)

                if beta <= alpha:
                    break
    return bestScore
```

Độ sâu trong game Tic Tac Toe là số tầng của cây được trò chơi, ta sẽ sử dụng biến n để kiểm soát độ sâu, khi đạt đến độ sâu đã được cho phép thì ta sẽ dừng kiểu tra, tuy nhiên việc giới hạn độ sâu sẽ ánh hưởng đến việc tìm ra nước đi tối ưu, chương trình có độ sâu càng lớn thì càng chơi giỏi nhưng sẽ phải trả giá về mặt thời gian.

2. Demo



Hình 1. Hòa



Hình 2. Máy thắng

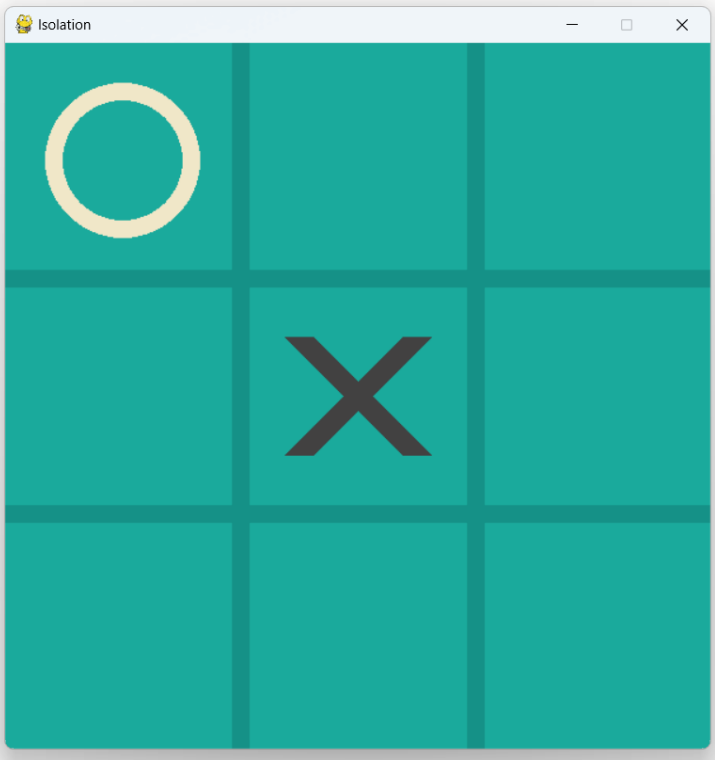
Hình 3. Người thắng

Hòa, người thắng, máy thắng đều là những nút lá của cây trò chơi.

Với mỗi nút lá thì được quy định : Hòa: 0 điểm, Máy thắng :-10, Người thắng 10 từ đấy sẽ tính được điểm tại các nốt.

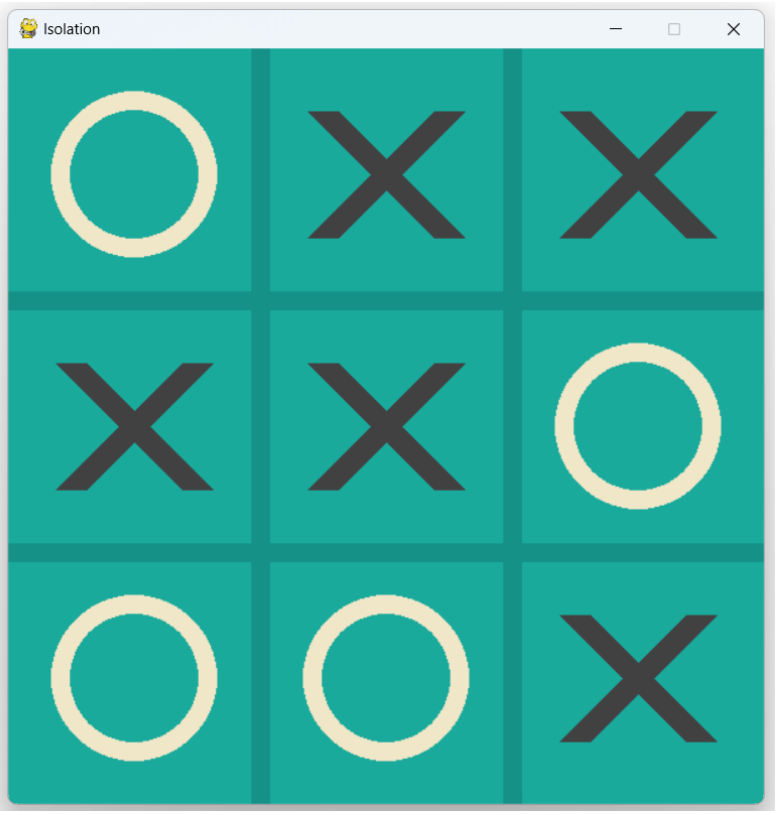
➢ Tic tac toe chỉ có minimax

```
55472
55473
55474
55475
55476
55477
55478
55479
55480
55481
55482
55483
55484
55485
55486
55487
55488
55489
55490
55491
55492
55493
55494
55495
55496
55497
55498
55499
55500
55501
55502
55503
55504
Minimax:time to make first move :1.413651
```



Khi chỉ áp dụng minimax vào trò chơi thì số vòng đệ quy để tìm ra nước đi đầu tiên là 55504 và mất thời gian là 1.41s.

```
56460
56461
56462
56463
56464
56465
56466
56467
56468
56469
56470
56471
56472
56473
56474
56475
56476
56477
56478
56479
56480
56481
56482
56483
56484
56485
56486
Minimax:time to make first move :1.493509
56487
56488
56489
56490
Minimax:time to make first move :1.493509
```

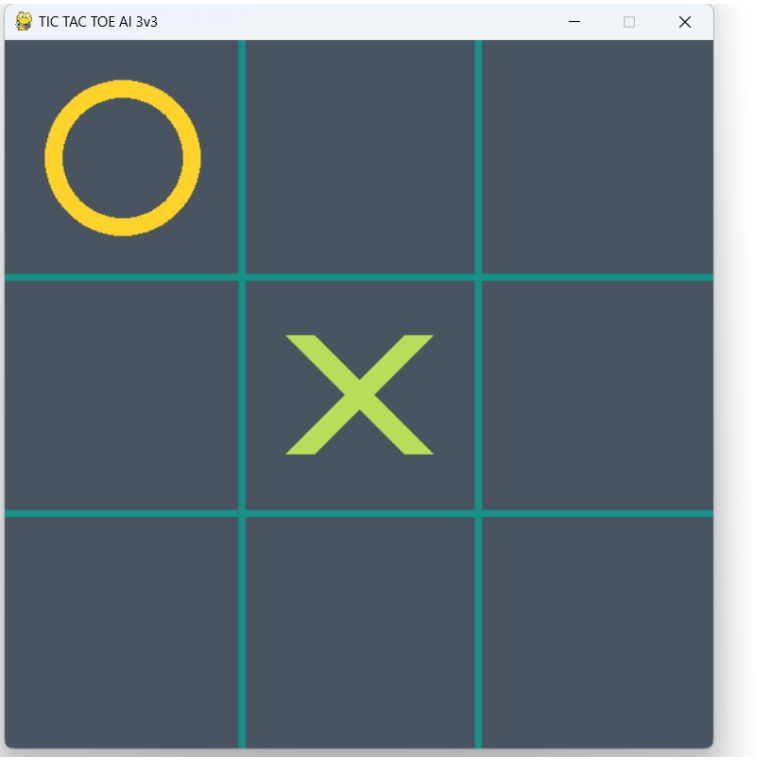


số vòng đệ quy để tìm đánh toàn bộ bàn cờ là 56490 và mất thời gian là 1.49s.

3. Những cải tiến

- **Thêm cắt tỉa Alpha-beta vào minimax**

```
15651
15652
15653
15654
15655
15656
15657
15658
15659
15660
15661
15662
15663
15664
15665
15666
15667
15668
15669
15670
15671
15672
15673
15674
15675
15676
15677
15678
15679
15680
15681
15682
Alpha_beta:time to make first move :0.355453
```

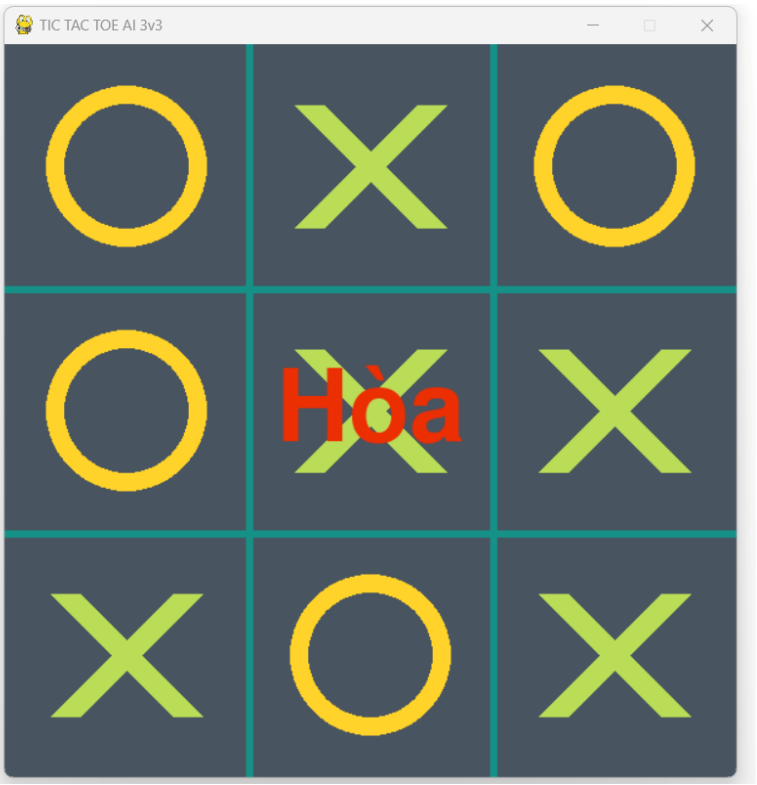


Khi áp dụng cắt tỉa Alpha-beta vào minimax thì số vòng đệ quy và thời gian giảm đi đáng kể cụ thể là:

+ Số vòng đệ quy: 15682

+ Thời gian: 0.35s

```
16254
16255
16256
16257
16258
16259
16260
16261
16262
16263
16264
16265
16266
16267
16268
16269
16270
16271
16272
16273
16274
16275
16276
16277
16278
16279
16280
Alpha_beta:time to make first move :0.374613
16281
16282
16283
16284
Alpha_beta:time to make first move :0.374613
```



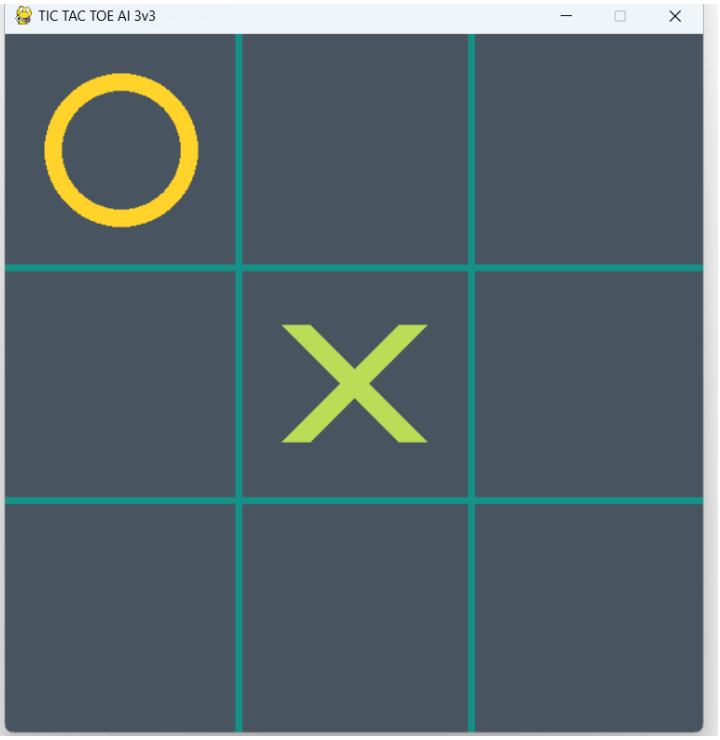
số vòng đệ quy và thời gian đánh đầy toàn bộ bàn cờ giảm đi đáng kể cụ thể là:

+ Số vòng đệ quy: 16284

+ Thời gian: 0.37

- **Thêm giới hạn độ sâu**

```
11992
11993
11994
11995
11996
11997
11998
11999
12000
12001
12002
12003
12004
12005
12006
12007
12008
12009
12010
12011
12012
12013
12014
12015
12016
12017
12018
12019
12020
12021
12022
12023
LimitedDepth:time to make first move :0.302793
```



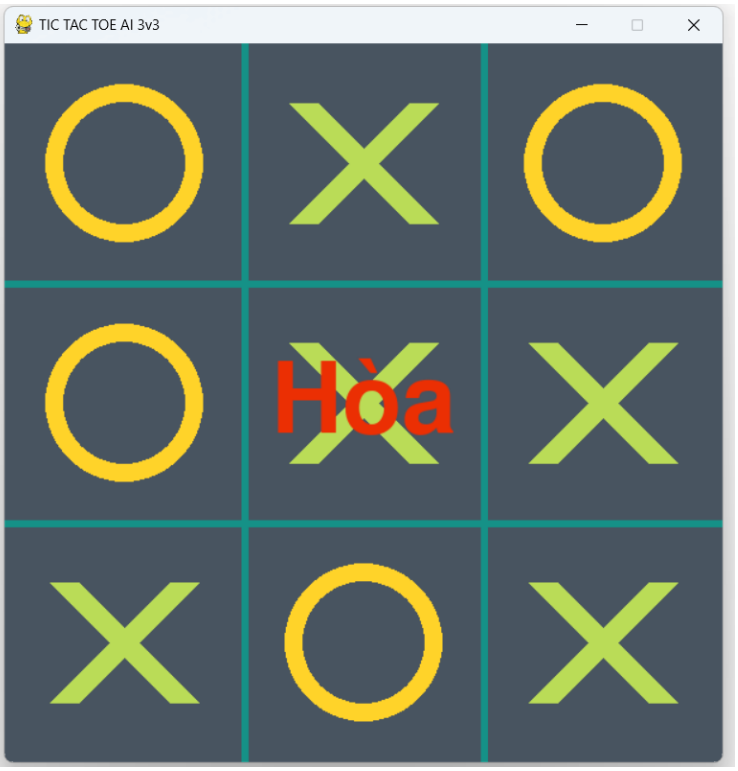
Khi thêm giới hạn độ sâu vào minimax và cắt tỉa Alpha-beta thì số vòng đệ quy và thời gian tìm được nước đi đầu tiên giảm đi rất nhiều, cụ thể là:

+ Số vòng đệ quy: 12023

+ Thời gian: 0.31s

Sau nhiều lần thử nghiệm thì tìm ra độ sâu để không thể đánh bại được máy đó chính là độ sâu n = 5.

```
12595
12596
12597
12598
12599
12600
12601
12602
12603
12604
12605
12606
12607
12608
12609
12610
12611
12612
12613
12614
12615
12616
12617
12618
12619
12620
12621
LimitedDepth:time to make first move :0.318874
12622
12623
12624
12625
LimitedDepth:time to make first move :0.318874
```

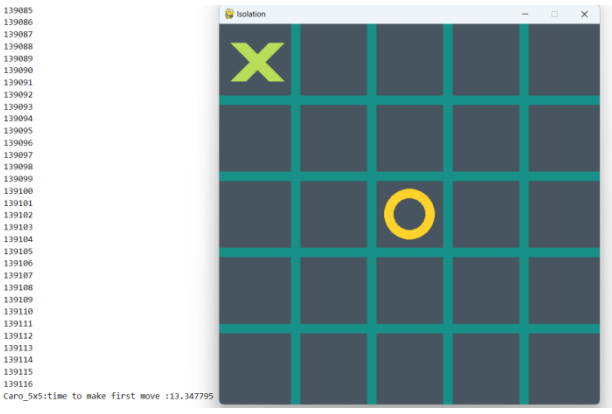


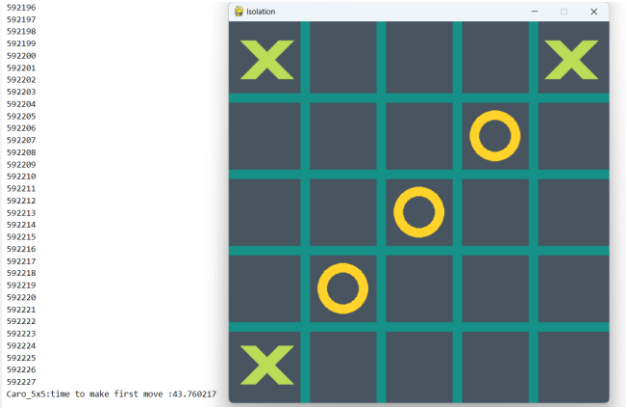
số vòng đệ quy và thời gian đánh đầy bàn cờ giảm đi rất nhiều, cụ thể là:

+ Số vòng đệ quy: 12625

+ Thời gian: 0.318s

➢ Cải tiến số ô lên 5x5





Khi tăng bàn cờ lên 5x5 thì thuật toán chạy khá là lâu hình trên là đã áp dụng các cải tiến Alpha-beta và giảm độ sâu nhưng chỉ đánh được vài nước thì sẽ không tìm được nước đi tối ưu do chưa đủ độ sâu tìm kiếm.

Để tìm ra được nước đi đầu tiên thì phải mất:

+ Số vòng đệ quy: 139116

+ Thời gian: 15.5s

Bảng so sánh tổng thể:

	Miniax		LimitedDepth		AlphaBeta		Caro5x5	
	Nút đầu	Nút lá	Nút đầu	Nút lá	Nút đầu	Nút lá	Nút đầu	Nút lá
Time	1.4	1.49	0.31	0.318	0.35	0.37	15.5	
Đệ quy	55504	56490	12023	12625	15282	16284	139116	

Kết Luận

1. Một số vấn đề khó khăn

Với bàn cờ 3x3 thì số lượng vòng lặp còn ít, dẫn đến thời gian đưa ra nước đi tiếp theo nhanh, tuy nhiên với bàn cờ lớn hơn số lượng vòng lặp quá nhiều dẫn đến mất rất nhiều thời gian hoặc có thể không đưa ra được nước đi tiếp theo, việc này có thể làm cho máy tính bị treo.

Một số biện pháp kháp khắc phục nhưng chưa thành công:

- Chia nhỏ bàn chờ thành nhiều bàn cờ 3x3, đi chuyển bàn cờ 3x3 theo vị trí của người chơi, tập trung vào vị trí của người chơi.

Những biện pháp trên dẫn đến việc đưa ra nước đi khá nhanh chóng, tuy nhiên lại không tìm được nước đi tối ưu do chỉ quan tâm đến xung quanh vị trí của người chơi mà lại không tính toán đến các vị trí khác.

2. Tổng kết và dự định phát triển

Trò chơi Tic Tac Toe là một bài toán thú vị để áp dụng thuật toán Minimax và cắt tỉa Alpha-Beta. Bằng cách kết hợp hai thuật toán này, chúng ta đã tạo ra một trí tuệ nhân tạo thông minh và khó đánh bại trong Tic Tac Toe. Việc áp dụng Minimax và Alpha-Beta giúp cải thiện hiệu suất tính toán và đưa ra các quyết định thông minh trong trò chơi. Với việc sử dụng những kỹ thuật này, người chơi Tic Tac Toe sẽ có trải nghiệm chơi game thú vị và thách thức hơn. Các thuật toán Minimax và Alpha-Beta Pruning không những mở ra cánh cửa cho việc sử dụng trí tuệ nhân tạo trong các trò chơi đã mang lại trải nghiệm chơi game tốt hơn cho người chơi, mà còn giúp cải thiện kỹ năng lập trình.

Dự định phát triển trong tương lai: áp dụng thuật toán thuật toán heuristic, phân chia điểm theo nút lá ở từng độ sâu.

Tài Liệu Tham Khảo

Giải thuật Minimax: https://viblo.asia/p/thuat-toan-minimax-ai-trong-game-APqzeaVVzVe

Giải thuật Alpha-Beta : https://websitehcm.com/thuat-toan-alpha-beta-pruning/

TicTacToe: https://www.youtube.com/watch?v=Bk9hlNZc6sE