ADVERSARIAL SEARCH
(Tìm kiếm đối kháng)

1. Giới thiệu trò chơi

- Khó tìm ch là giải tốt
- Đòi hỏi:
  + Khả năng quyết định khi k có thể hình toán tối ưu.
  + Chấp nhận tgian chấp nhận đc

2. Các loại dạng trò chơi

- Trò chơi đối kháng. Agent (Tác nhân)
- Trò chơi hợp tác Multi Agent
- Nl trò chơi nằm giữa thường cách từ đối kháng đến hợp tác.

3. Tìm kiếm đối kháng

- Là 1 tìm kiếm ở đó chúng ta khi vấn đề phát sinh khi chúng ta cố gắng lên kế hoạch mới, trong khi các đối thủ đang lên kế hoạch chống lại chúng ta.
- Môi trường có ít hơn 1 tác nhân ch gọi là môi trường đa tác nhân. Mỗi tác nhân cần xem xét hành động của các tác nhân ≠ và ảnh hưởng của hành động đó đến hiệu suất của họ.

Date: ...../...../.....

points

_ Các trò chơi chức mơ hình hoá như bài toán tìm kiếm và các hàm đánh giá heuristic → là yếu tố chính giúp mơ hình hoá và giải quyết bài toán các trò chơi trong AI.

_ Các loại trò chơi trong AI:

	Xác định	Không xác định
Thông tin hoàn hảo	Chess, Checkers, go, Othello	Backgammo, monopoly
Thông tin ko hoàn hảo	Battleships, blind, tic-tac-toe	Bridge, poker, scrabble, nuclear war.

+ Trò chơi quyết định: Tuân theo 1 khuôn mẫu nghiêm ngặt và bộ quy tắc sao cho trò chơi và ko có sự ngẫu nhiên liên quan đến chúng.

+ Trò chơi ko xác định: Có n sự kiện ko thể đoán trước, có yếu tố may rủi hoặc may mắn (đc scl bằng xúc xắc hoặc thẻ). Đây là ngẫu nhiên và mỗi phản ứng hành động không có định.

+ Thông tin hoàn hảo: Tác nhân có thể nhìn vào bảng hoàn chỉnh. Các tác nhân có tất cả các thông tin về trò

chơi và chúng cũng có thể thay nhau di chuyển.

+ Thông tin không hoàn hảo: Các tác nhân không có tất cả thông tin về trò chơi và không biết chuyện gì đang xảy ra.

4. Trò chơi Zero - Sum

- Tìm kiếm đối kháng liên quan đến cạnh tranh thuần túy.

- Mức tăng hoặc mất đi lợi ích (Utility) liên quan của mỗi tác nhân được cân bằng chính xác bởi tổn thất hoặc lợi ích của tác nhân khác.

- Một TC chơi cố gắng tối đa hoá một giải thưởng duy nhất, trong khi người chơi khác cố gắng giảm thiểu nó.

- Mỗi di chuyển của một người chơi được gọi là ply.

- VD: Cờ vua, tic-tac-toe, ...

- Lquan đến suy nghĩ bên trong. Cố gắng tìm ra phản ứng của đối thủ của mình trước hành động của họ. Đôi khi họ duy trì trong hoặc lập luận quay lui để giải quyết các vấn đề như chơi AI.

5. Dạng bài toán

- Một trò chơi có thể được định nghĩa là một loại tìm kiếm

Date: ...../...../.....

hong AI, ở thể chức hình thức hoá các yếu tố sau:

- Initial State: Trạng thái ban đầu
- Player(s): Xđ ai đó nào di chuyển ở ô gian trong thái
- Action(s): Tập các nó di hợp tác pháp ở ô gian hong thá
- Result(s, a): Mô hình chuyển hiệp, chỉ định kết quả của các di chuyển hong ô gian hang thái.
- Terminal-test(as): Thử nghiệm cuối cả chúng nếu hc chó kết thúc, nếu không thì nó là Sai ở mọi trường hợp. Trang thái nó hò chơi kết thúc được gọi là hang thái cuối.
- Utility(s, p): Hàm hiện ích há về giá trị số cuối cùng cho hò chơi kết thúc ở hạng thái cuối s cho ñ chơi p. Nó được gọi là hàm hoàn há.

+ Cờ rua: +1, 0, 1/2 } thắng, thua, hòa
+ Tic-tac-toe: +1, -1, 0 }

6. Cây trò chơi

- Là một cây trong đó các nút của cây là hạng thái trò chơi và cạnh của cây là các bước di chuyển của người chơi. Cây hc chơi luôn quan tâm hang thái ban đầu, hàm hoàn chúng và hàm kết quả.

Cây trò chơi = trạng thái ban đầu + luật di chuyển.

- Tìm đối thủ cho thuật toán minimax hoạt động như sau.

+ Mục đích tìm ra chiến lược tối ưu cho max để giành chiến thắng trong trò chơi.

+ Theo cách tiếp cận của tìm kiếm sâu đầu tiên (DFS)

+ Trong cây trò chơi, nút lá tối ưu có thể xuất hiện ở bất kỳ độ sâu nào của cây.

+ Truyền các giá trị minimax lên đến cây cho đến khi nút đầu của chúng phát hiện.

- Trong một cây trò chơi, chiến lược tối ưu có thể được xác định từ giá trị ~~MINIMAX~~ minimax của mỗi nút, có thể được viết là MINIMAX(n). MAX chuyển sang trạng thái giá trị tối đa và MIN chuyển sang trạng thái giá trị tối thiểu sau đó:

- Cho 1 trạng thái s

$$MINIMAX(s) = \begin{cases} UTILITY & \text{if TERMINAL-TEST}(s) \\ \max_{a \in actions(s)} MINIMAX(RESULT(s,a)) & \text{if PLAYER} = MAX \\ \min_{a \in actions(s)} MINIMAX(RESULT(s,a)) & \text{if PLAYER} = MIN \end{cases}$$

Date: ...../...../.....

7. Thuật toán Mini-max trong AI

- Là thuật toán đệ quy hoặc quay lui thích sd trong lý thuyết trò chơi và ra quyết định. Nó cung cấp một nước đi tối ưu cho người chơi giả định rằng đối thủ cũng đang chơi tối ưu.
- ~~Được~~ Sử dụng đệ quy để tìm kiếm thông qua cây trò chơi.
- Chủ yếu thích sử dụng để chơi trò chơi trong AI. VD: cờ vua, cờ tướng, ... và các trò chơi có người khác. Thuật toán này tính toán quyết định minimax cho trạng thái hiện tại.
  + 1 K chơi là MIN + 1 K chơi là MAX
- Cả hai người chơi đấu với nhau, khi người chơi - đối thủ nhận được lợi ích tối thiểu, trong khi họ nhận được lợi ích tối đa. (MAX sẽ chọn giá trị tối đa, MIN sẽ chọn giá trị tối thiểu)
- Thuật toán minimax thực hiện tìm kiếm theo chiều sâu để khám phá cây trò chơi hoàn chỉnh.
- Thuật toán minimax ~~hthtoán~~ hiện hành tất cả các cách xuống nút cuối cùng cây, sau đó quay lại cây như đệ quy

* Hoạt động của Min-Max Algorithm

- 2 kí chơi: Maxmizer, Minimizer

- Áp dụng DFS. Vì vậy trong cây trò chơi này, ta phải đi hết các kí đến các nút cuối

- Tại nút đứng cuối, các giá trị cuối được đưa ra, vì vậy chúng ta sẽ so sánh các giá trị đó và quay lại cây cho đến trạng thái ban đầu xảy ra.

- Các bước chính liên quan đến việc giải quyết cây trò chơi cho 2 người chơi:

+ Bước 1: Tại bước đầu tiên, thuật toán tạo toàn bộ cây trò chơi và áp dụng hàm tiện ích để lấy các giá trị tiện ích cho các trạng thái đầu cuối

• ~~Trong~~ Chả sử Maximizer thực hiện liệt đầu tiên có giá trị ban đầu trong trường hợp xấu nhất = $-\infty$ và Minimizer sẽ thực hiện liệt kê tiếp theo có giá trị ban đầu trong trường hợp xấu nhất = $+\infty$.

+ Bước 2: Bây giờ, đầu tiên chúng ta tìm giá trị tiện ích là cực đại cho Maximizer, giá trị ban đầu của Maximizer ~~và xấu~~). Vì vậy, chúng ta sẽ so sánh từng giá trị ở trạng thái cuối

Summary

Date: ...../...../.....

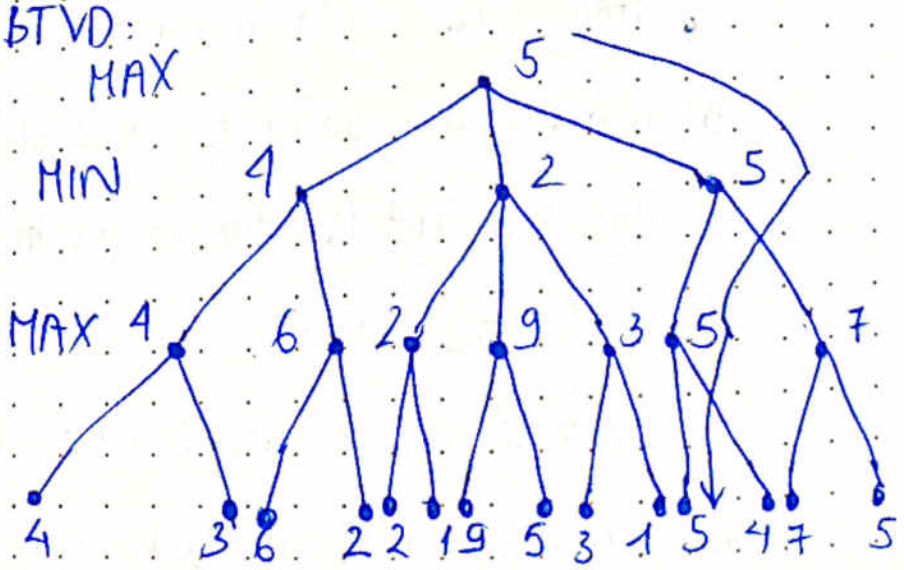
với giá trị ban đầu và xác định giá trị nút cao hơn. Ta sẽ lấy chúc giá trị lớn nhất trong số đó

+ Bước 3: Trong bước tiếp theo, đến lượt Minimizer, do đó, nó sẽ so sánh tất cả các giá trị nút với +∞ và sẽ lấy giá trị nút cho lớp thứ ba

+ Bước 4: Bây giờ đến lượt Maximiner và 1 lần nữa ta sẽ chọn giá trị tối đa của tất cả các giá trị nút và lấy giá trị tối đa cho nút gốc. Trong cây trò chơi này có 4 lớp, do đó chúng ta tiếp cận ngay với nút gốc, nhưng trong các cây trò chơi thực tế sẽ có nhiều hơn 4 lớp

. Ở trạng thái kết thúc, giá trị MINIMAX-VALUE(n) = Utility(n)

BTVD:



Key points

Date: ..../..../....

8. Tính chất thuật toán Mini-max

- Đầy đủ (Complete): Nó chắc chắn tìm thấy 1 lời giải (nếu tồn tại), trong cây tìm kiếm hữu hạn
- Tối ưu (Optimal): Thuật toán tối ưu nếu có hai đối thủ đều chơi tối ưu
- Độ phức tạp thời gian: DFS nên DPT $O(b^m)$ (b là hệ số phân nhánh của cây và m là độ sâu tối đa của cây)
- Độ phức tạp không gian: Cũng tương tự DFS $O(bm)$

9. Giới hạn của thuật toán Mini-max.

- Rất chậm chỉ với các trò chơi phức tạp như cờ vua, cờ vây, vì có yếu tố phân nhánh lớn và ở mỗi lần có rất nhiều sự lựa chọn để quyết định
- Có thể cải thiện hạn chế bằng cách cắt tỉa alpha-beta

10. Cắt tỉa Alpha-Beta

- Là kỹ thuật tối ưu hoá cho thuật toán Mini-Max.
- Ta có số lượng trạng thái trò chơi phải kiểm tra là theo cấp số nhân theo chiều sâu của cây. Vì ta không thể loại bỏ số mũ nhưng có thể cắt nó xuống một nửa. Do đó có một kỹ thuật không cần kiểm tra từng nút của cây trò chơi, chúng ta

Summary

Date: ...../...../.....

Key points

có thể hình toán quyết định Mini-Max của cây trò chơi chính xác và kỹ thuật này được gọi là cắt tỉa

- Điều này liên quan đến hai tham số ngưỡng là alpha và beta để mở rộng sau này, vì vậy nó được gọi là cắt tỉa alpha-beta. Nó cũng được gọi là thuật toán Alpha-beta

- Có thể áp dụng ở bất kỳ độ sâu nào của cây và đôi khi nó không chỉ hỗ là cây mà còn toàn bộ cây phụ

- Hai tham số có thể được định nghĩa:

+ Alpha: Sự lựa chọn tốt nhất (giá trị cao nhất) mà chúng ta đã tìm thấy cho đến nay, tại bất kỳ điểm nào trên con đường của Maximizer. Giá trị ban đầu của alpha là $-\infty$.

+ Beta: Sự lựa chọn tốt nhất (giá trị thấp nhất) của mà chúng ta đã tìm thấy cho đến nay, tại bất kỳ điểm nào trên con đường của Minimizer. Giá trị ban đầu của beta là $+\infty$.

- Việc cắt tỉa alpha-beta theo thuật toán Mini-Max chuẩn hóa về kết quả giống như thuật toán tiêu chuẩn -nhưng nó loại bỏ tất cả các nút không thực sự ảnh hưởng

ory

Key points

Date: ..../..../....

đến quyết định cuối cùng là tìm bài toán chậm. Do đó, bằng cách cắt nhỏ các nút này, sẽ làm cho thuật toán nhanh hơn.

- Đặc điểm để cắt tỉa alpha-beta:
  + Điều kiện chính là: $\alpha \geq \beta$
- Ñ điểm chính về cắt tỉa alpha-beta:
  + Ñ chơi Max sẽ chỉ cập nhật giá trị của alpha.
  + Ñ chơi Min sẽ chỉ cập nhật giá trị của beta.
  + Trong khi quay lại cây, các giá trị nút sẽ được chuyển đến các nút trên thay vì alpha và beta.
  + Chúng ta sẽ chuyển giá trị alpha, beta cho các nút con.
- Pseudo-code cắt tỉa Alpha-Beta:

```
function minimax(node, depth, alpha, beta, maximizingPlayer) is
    if (depth == 0 or node is terminal node) then
        return static evaluation of node
    if MaximizingPlayer then
        maxEva = -infinity
        for each child of node do
            eva = minimax(child, depth-1, alpha, beta, False)
```

Summary

Date: ...../...../.....

```
        maxEva = max (maxEva, eva)
        alpha = max (alpha, maxEva)
        if beta <= alpha
            break
        return maxEva
    else
        minEva = + infinity
        for each child of node do
            eva = minimax (child, depth-1, alpha,
                           beta, true)
            minEva = min (minEva, eva)
            beta = min (beta, eva)
        if beta <= alpha
            break
        return minEva
```

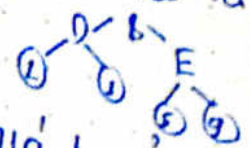
- Hoạt động của cắt tỉa alpha-beta:

+ Bước 1: Ở bước đầu tiên, người chơi Max sẽ bắt đầu di chuyển đầu tiên từ nút gốc trong đó $\alpha = -\infty$ và $\beta = +\infty$, các giá trị này của alpha và beta được truyền xuống nút con

Key points

Date: ..../..../....

trong đó một lần nữa $\alpha = -\infty$ và $\beta = +\infty$ và nút B truyền cùng giá trị cho con của nó.



+ Bước 2: Tại Node D, giá trị của α sẽ được tính là lần lượt cho Max. Giá trị của α được so sánh với đầu tiên là 2 và sau đó là 3 và giá trị tối đa (2, 3) = 3 sẽ là giá trị của α tại nút D và giá trị nút cũng là 3.

+ Bước 3: Bây giờ thuật toán quay lại nút B, trong đó giá trị của β sẽ thay đổi vì đây là lượt của Min. Bây giờ $\beta = +\infty$, sẽ so sánh với giá trị các nút tiếp theo có sẵn, tức là min $(\infty, 3) = 3$, do đó tại nút B bây giờ là $\alpha = -\infty$ và $\beta = 3$.

- Trong bước tiếp theo, thuật toán đi qua nút tiếp theo của nút B là nút E và các giá trị của $\alpha = -\infty$ và $\beta = 3$ cũng sẽ được thông qua.

+ Bước 4: Tại nút E, Max sẽ thực hiện lần lượt và giá trị của alpha sẽ thay đổi. Giá trị hiện tại của α sẽ được so sánh với 5, do đó $\alpha = \max(-\infty, 5) = 5$, khi đó tại nút E: $\alpha = 5$ và $\beta = 3$, trong đó $\alpha \geq \beta$, do đó nút tiếp theo của E sẽ được cắt tỉa, thuật toán sẽ không đi qua nó và

Summary

Date: ...../...../.....

Key points

giá trị tại nút E sẽ là 5.

- Thứ tự di chuyển trong cắt tỉa Alpha-beta, có 2 loại:

+ Thứ tự tốt nhất: Trong một số trường hợp, thuật toán cắt tỉa alpha-beta không cắt tỉa bất kỳ lá nào của cây và hoạt động chính xác như thuật toán Minimax. Nó cũng tiêu tốn tgian hơn vì các yếu tố alpha-beta. Di chuyển tốt nhất xảy ra ở bên phải của cây. Độ phức tạp thời gian cho 1 thứ tự như vậy là $O(b^m)$

+ Thứ tự tồi tệ: Xảy ra khi có nhiều sự cắt tỉa xảy ra trong cây và hành động xảy ra tốt nhất ở bên trái của cây. Áp dụng DFS do đó lần đầu nó tìm kiếm bên trái của cây và đi sâu gấp đôi như thuật toán Mini-max trong cùng 1 khoảng tgian. ĐPT là $O(b^{m/2})$

- Quy tắc để tìm thứ tự tốt trong việc cắt tỉa alpha-beta:

+ Tiến hành di chuyển tốt nhất từ nút nông nhất.

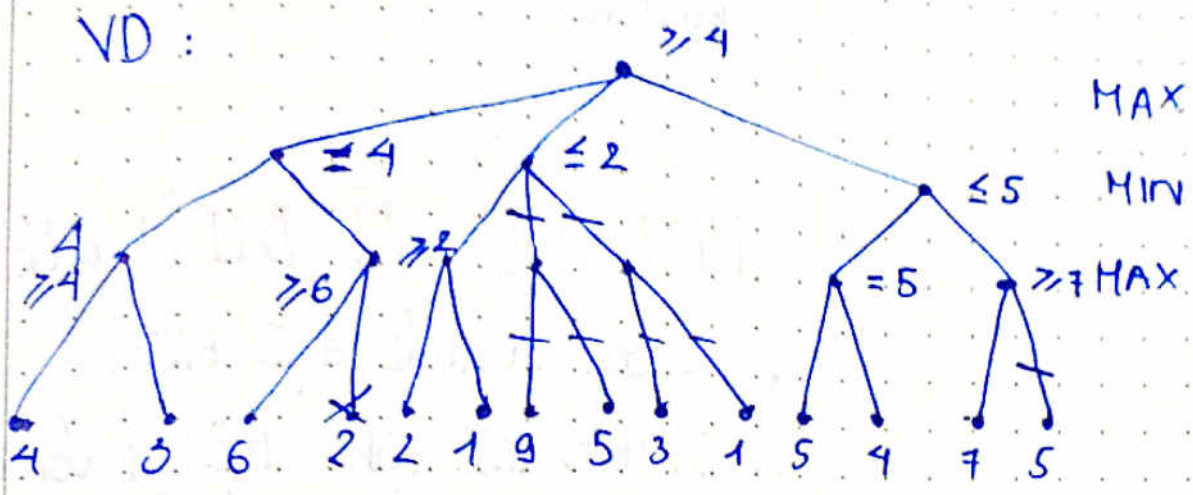
+ Sắp xếp các nút trong cây sao cho các nút tốt

nhất: chốt kiểm tra nước.

+ Sử dụng kỹ thuật miền hàng khi tìm kiếm chỉ chuyển tốt nhất.

+ Chúng ta có thể ghi nhớ các hàng thái, vì có khả năng các hàng thái có thể lặp lại.

VD:



11. Quyết định thời gian thực

- Các trò chơi thường có độ sâu lớn (>35 đối với cờ vua)
- Trong thời gian thực, không thể đi đến hàng thái kết thúc để đánh giá 1 nước đi ⇒ Tìm kiếm giới hạn (Cut-off Search)

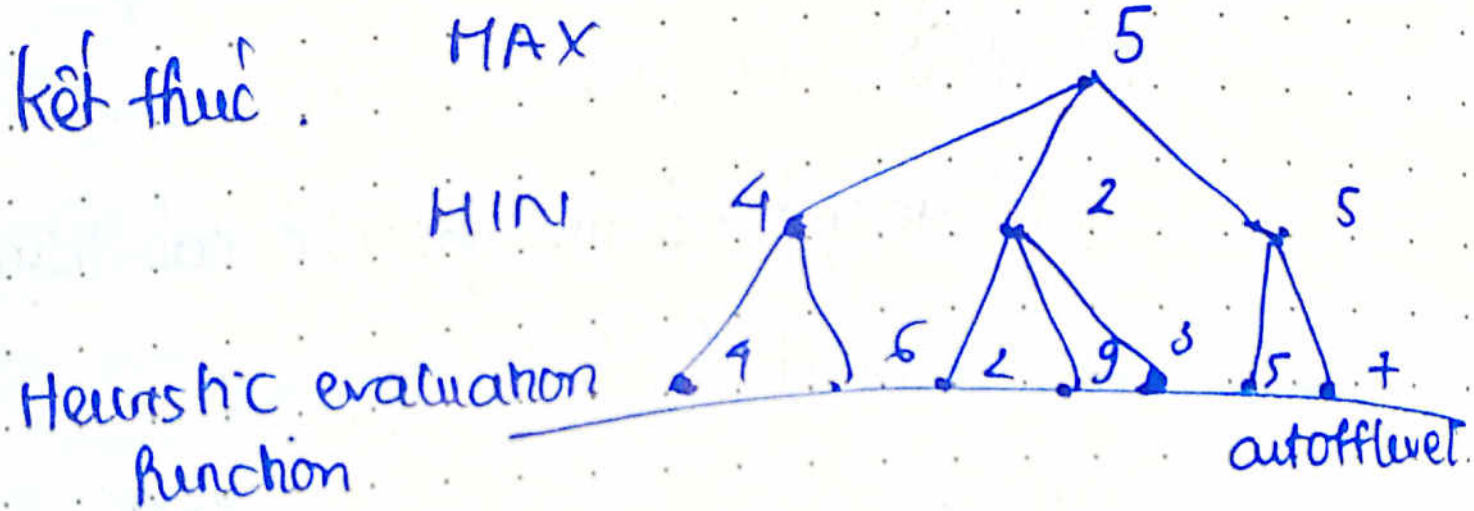
12. Hàm lượng giác

- Ý tưởng:

+ Giới hạn cây tìm kiếm trước khi đến hàng thái kết thúc.

Date: ...../...../.....

+ Sử dụng hàm lượng giá các trạng thái không kết thúc thay cho hàm đánh giá lợi ích của trạng thái kết thúc.



TRI THỨC VÀ BIỂU DIỄN TRI THỨC

* 2 loại tri thức đặc biệt: