

AI

TÌM KIẾM NGƯỢC LẠI

VÀ TRÒ CHƠI

Nguyễn Ngọc Thảo - Nguyễn Hải Minh

{nnthao, nhminh}@fit.hcmus.edu.vn

Đề cương

- Trò chơi có tổng bằng 0 dành cho hai người chơi
- Quyết định tối ưu trong trò chơi
- Tìm kiếm cây alpha-beta theo kinh nghiệm
- Trò chơi ngẫu nhiên

Trò chơi có tổng bằng 0 dành cho hai người chơi

Lý thuyết trò chơi và trò chơi AI

- Lý thuyết trò chơi xem bất kỳ môi trường đa tác nhân nào cũng là một trò chơi.
- Tác động của mỗi tác nhân đối với các tác nhân khác là rất đáng kể, bất kể các tác nhân đó hợp tác hay cạnh tranh.
- Mỗi tác nhân cần xem xét hành động của các tác nhân khác và chúng ảnh hưởng đến phúc lợi của chính nó như thế nào.



Trò chơi có tổng bằng 0 dành cho hai người chơi

- Các trò chơi được nghiên cứu phổ biến nhất trong AI là

Thông tin hoàn hảo

Hoàn toàn có thể quan sát được

tổng bằng không

Điều gì tốt cho người chơi này cũng có thể là điều xấu cho người chơi khác.

Không có kết quả "đôi bên cùng có lợi"

xác định

Hai người chơi

Lần lượt lấy

- Các thuật ngữ khác so với các thuật ngữ tìm kiếm.

- Hành động Di chuyển và Trạng thái Vị trí

Trò chơi có tổng bằng 0 dành cho hai người chơi

- Hai người chơi là MAX và MIN. MAX di chuyển đầu tiên.
- Người chơi lần lượt di chuyển cho đến khi trò chơi kết thúc.
- Khi kết thúc trò chơi, người chiến thắng sẽ được thưởng điểm người chơi và hình phạt được đưa ra cho người thua cuộc.
- Hai người chơi đều lý trí, cố gắng tối đa hóa lợi ích của mình.

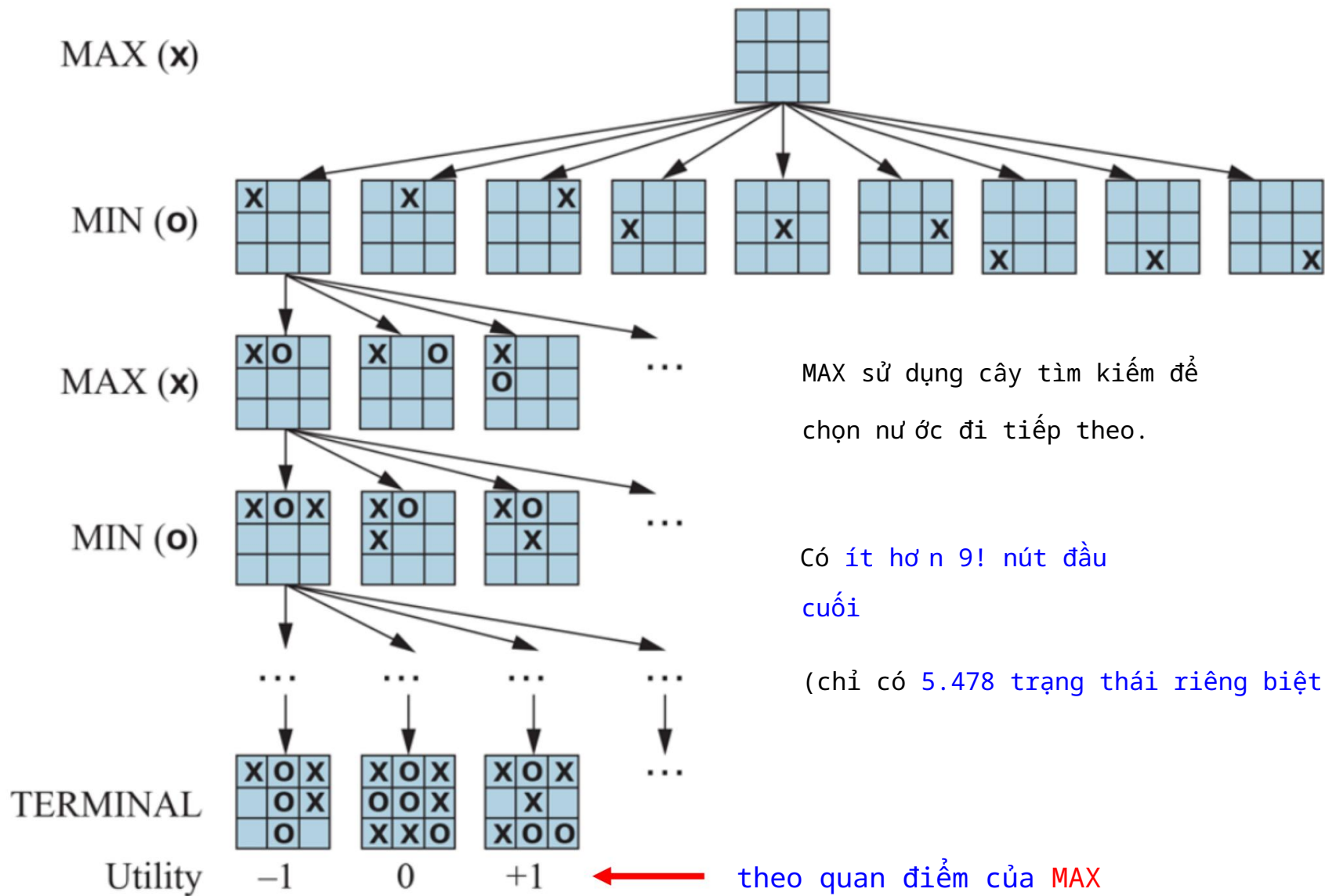
Công thức trò chơi

- : Trạng thái ban đầu, chỉ định cách thiết lập trò chơi khi bắt đầu.
- - $()$: Người chơi đến lượt di chuyển trong trạng thái
- $()$: Tập hợp các động thái hợp pháp trong trạng thái .
- $(,)$: Mô hình chuyển đổi, xác định trạng thái kết quả từ hành động trong trạng thái.
- - $()$: Một bài kiểm tra đầu cuối, điều này đúng khi trò chơi kết thúc và sai khác.
 - Các trạng thái mà trò chơi đã kết thúc được gọi là trạng thái kết thúc.
- $(,)$: Hàm tiện ích xác định giá trị số cuối cùng cho trình phát khi trò chơi kết thúc ở trạng thái cuối
 - Ví dụ: cờ vua: thắng (+1), thua (-1) và hòa (0), cờ thỏ cáo: [0, 192]

Đồ thị không gian trạng thái và cây trò chơi

- Trạng thái ban đầu, chức năng và chức năng xác định đồ thị không gian trạng thái.
- Cây trò chơi hoàn chỉnh là cây tìm kiếm theo sau mọi chuỗi các chuyển động cho đến trạng thái cuối cùng.
 - Nó có thể là vô hạn nếu bản thân không gian trạng thái là không giới hạn hoặc nếu các quy tắc của trò chơi cho phép lặp lại các vị trí vô hạn

Cây trò chơi dành cho Tic-tac-toe



MAX sử dụng cây tìm kiếm để
chọn nước đi tiếp theo.

Có ít hơn 9! nút đầu
cuối

(chỉ có 5.478 trạng thái riêng biệt)

Ví dụ về trò chơi: Cờ đam



- Độ phức tạp

- ~ 10^{18} nút, có thể cần 100 nghìn năm với 106 vị trí/giây

- Chinook (1989-2007)

- Chương trình máy tính đầu tiên giành được danh hiệu vô địch thế giới trong cạnh tranh chống lại con người
 - 1990: Thắng 2 ván đấu với nhà vô địch thế giới Tinsley (chung kết tỷ số: 2-4, 33 trận hòa). 1994: 6 trận hòa

- Tìm kiếm của Chinook

- Chạy trên PC thông thường, chơi hoàn hảo bằng cách sử dụng tìm kiếm alpha-beta kết hợp với cơ sở dữ liệu gồm 39 nghìn tỷ vị trí cuối trò chơi

Ví dụ về trò chơi: Cờ vua

- Độ phức tạp

- b 35, d 100, 10154 nút (!!)

- Hoàn toàn không thực tế để tìm kiếm điều này

- Màu xanh thẳm (11 tháng 5 năm 1997)

- Kasparov thua trận 6 ván trước Deep Blue của IBM (thắng 1 Kasp
– 2 thắng DB) và 3 hòa.

- Trong tương lai, trọng tâm sẽ là cho phép máy tính HỌC chơi cờ thay vì được chỉ dẫn cách chơi



Xanh đậm

- Chạy trên máy tính song song với **30** IBM RS/6000
bộ xử lý thực hiện tìm kiếm alpha-beta
- Tìm kiếm tới **30 tỷ vị trí/di chuyển**, độ sâu trung bình **14** (có thể đạt tới **40** lớp)
- Chức năng đánh giá: **8000** tính năng
 - các mẫu mảnh có tính đặc thù cao (~4000 vị trí)
 - Cơ sở dữ liệu có 700.000 trận đấu kiện tư ớng
- Hoạt động ở **tốc độ 200 triệu vị trí/giây**, ngay cả Deep Blue cũng cần 10100 năm để đánh giá tất cả các trò chơi có thể xảy ra.
 - (Vũ trụ chỉ mới 1010 tuổi.)
- Bây giờ: cải tiến thuật toán đã cho phép các chương trình chạy trên PC tiêu chuẩn để giành chức vô địch cờ vua máy tính thế giới.
 - Cắt tỉa heuristic làm giảm hệ số phân nhánh hiệu quả xuống dưới 3



ĐI

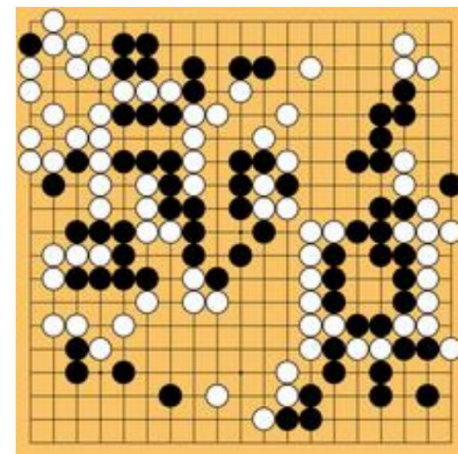
Nhiều hơn 1 triệu nghìn tỷ
nghìn tỷ nghìn tỷ so với
cờ vua!

- Độ phức tạp

- Ván 19x19, b 361, độ sâu trung bình 200
- 10¹⁷4 cấu hình bo mạch có thể có.
- Việc kiểm soát lãnh thổ là không thể đoán trước cho đến khi kết thúc trò chơi

- AlphaGo (2016) của Google

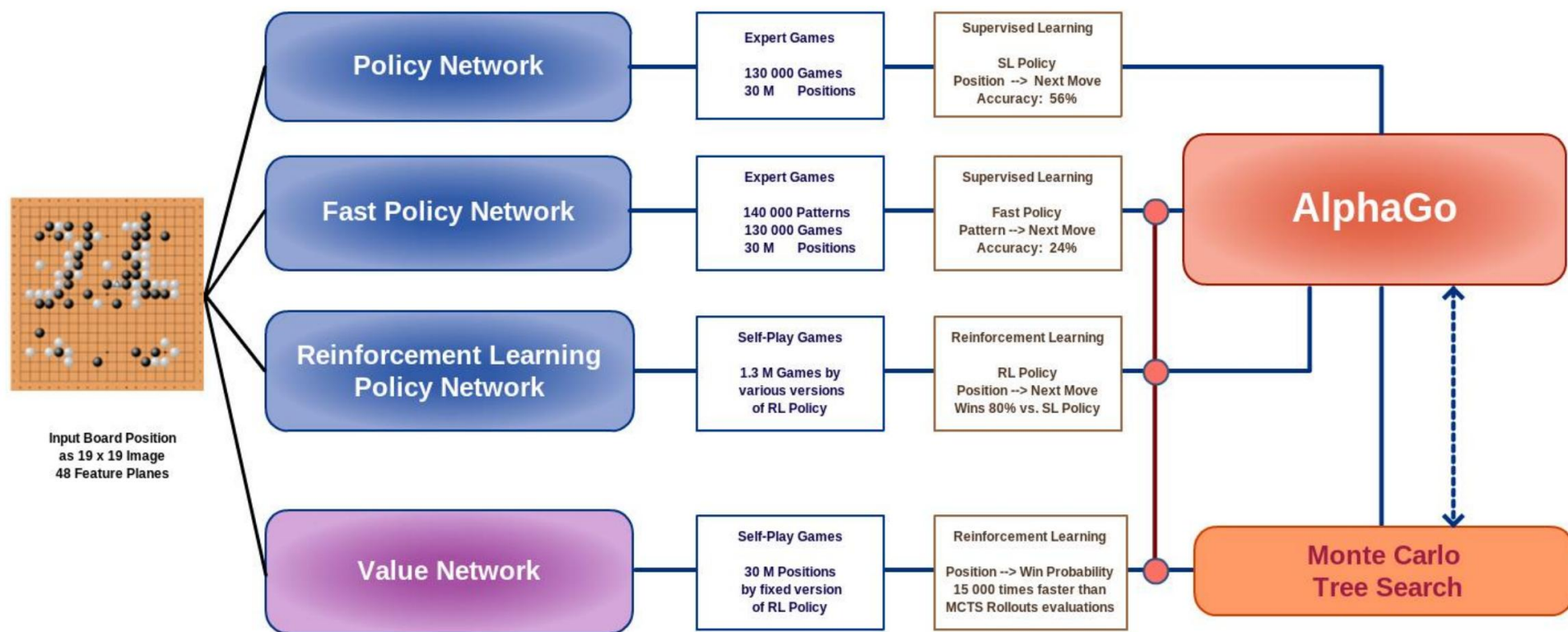
- Đánh bại Lee Sedol chuyên nghiệp 9 đẳng (4-1)
- Học máy + tìm kiếm Monte Carlo được hướng dẫn bởi “mạng giá trị” và một “mạng lưới chính sách” (được triển khai bằng cách sử dụng mạng lưới thần kinh sâu công nghệ)
- Học từ con người + Tự học (trò chơi tự chơi)




Tổng quan về AlphaGo

AlphaGo Overview

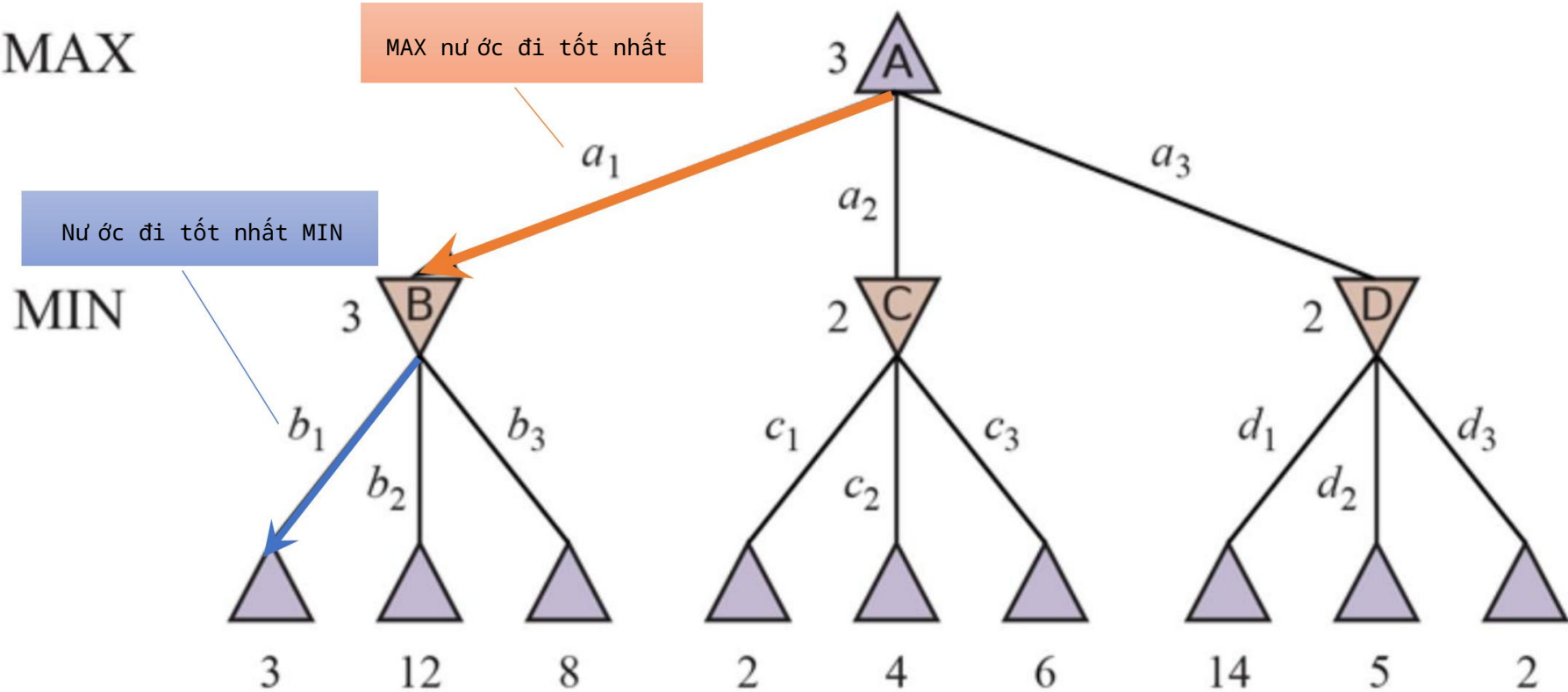
based on: Silver, D. et al. Nature Vol 529, 2016
copyright: Bob van den Hoek, 2016



A close-up, slightly blurred photograph of a Go board. The board is made of light-colored wood with a grid of dark lines. Several black and white Go stones are scattered across the board. In the foreground, there are two white stones and three black stones. The background is dark and out of focus, showing some furniture.

Quyết định tối ưu trong trò chơi

Cây trò chơi hai lớp



Nút đi tốt nhất của MAX ở gốc là giá 1, dẫn đến trạng thái có cực tiểu cao nhất
trị. Câu trả lời hay nhất của MIN là 1, hướng tới trạng thái có giá trị minimax thấp nhất.

nút: Đến lượt MAX di chuyển, nút: Đến lượt MIN di chuyển; các nút đầu cuối hiển thị các giá trị tiện ích cho MAX và các nút khác được gắn nhãn bằng các giá trị cực tiểu của chúng.

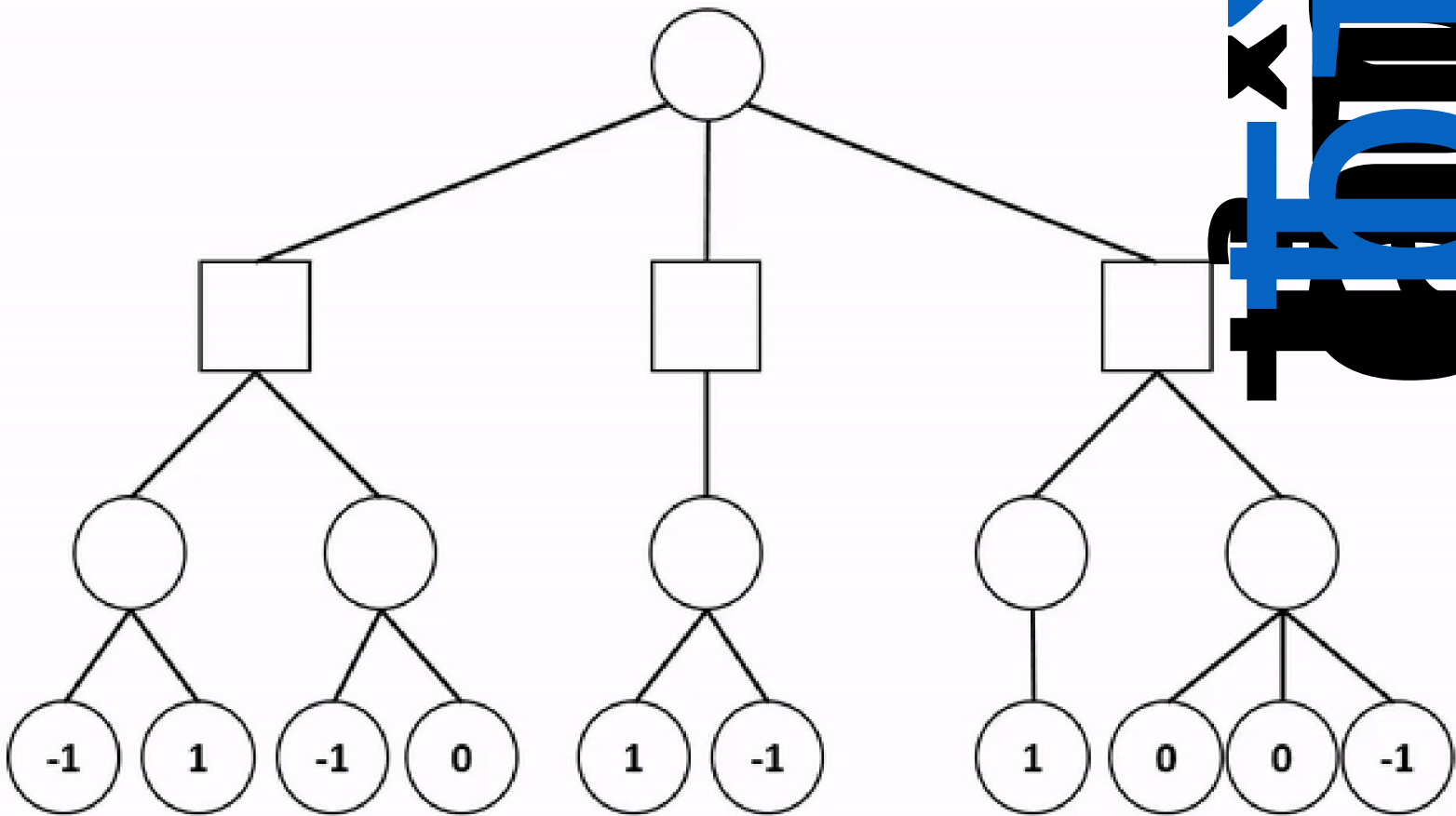
Thuật toán minimax

- Giả sử rằng cả hai tác nhân đều chơi tối ưu từ bất kỳ trạng thái nào đến sự kết thúc của trò chơi.
- Giá trị minimax của trạng thái là tiện ích của việc ở trong .
 - Giá trị cực tiểu của trạng thái cuối chỉ là tiện ích của nó.
- MAX thích chuyển sang trạng thái có giá trị tối đa và MIN thích trạng thái có giá trị tối thiểu.

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$

cho TỐI ĐA

- Tiến hành đệ quy đến tận các nút rồi đi và sao lưu các giá trị cực tiểu khi quá trình đệ quy kết thúc.



hàm MINIMAX-SEARCH(trò chơi, trạng thái) trả về một hành động

```

    người chơi    giá trị game.TO-MOVE(trạng
    thái) , di chuyển    MAX-VALUE(trò chơi, trạng thái)
    di chuyển trở lại

```

hàm MAX-VALUE(trò chơi, trạng thái) trả về một cặp (tiện ích, di chuyển)

```

    nếu game.IS-TERMINAL(state) thì trả về game.UTILITY(state, player), null
    v    -

```

```

    cho mỗi a trong game.ACTIONS(state) do v2,

```

```

        a2    MIN-VALUE(game, game.RESULT(state, a)) if v2 > v
        then v, move

```

```

        v2, a return

```

```

    v, move

```

hàm MIN-VALUE(game, state) trả về một cặp (tiện ích, nước đi)

```

    nếu game.IS-TERMINAL(state) thì trả về game.UTILITY(state, player), null
    v    +

```

```

    cho mỗi a trong game.ACTIONS(state) do v2,

```

```

        a2    MAX-VALUE(game, game.RESULT(state, a)) if v2 < v
        then v, move

```

```

        v2, a return

```

```

    v, move

```

Nếu MIN không chơi tối ưu thì sao?

- MAX ít nhất sẽ có tác dụng tốt khi đối đầu với người chơi tối ưu.
- Tuy nhiên, điều đó không có nghĩa là chơi luôn là tốt nhất nếu đi tối ưu khi đối mặt với một đối thủ dưới mức tối ưu.
- Hãy xem xét tình huống sau.



TIE

Một lối chơi tối ưu của cả hai bên sẽ dẫn đến kết quả hòa.

10 động tác phản ứng có thể có của MIN đều có vẻ hợp lý, nhưng 9 trong số đó là thua đối với MIN và một là thua đối với MAX.

WIN?

MAX thực hiện một bước đi mạo hiểm

MIN không thể khám phá

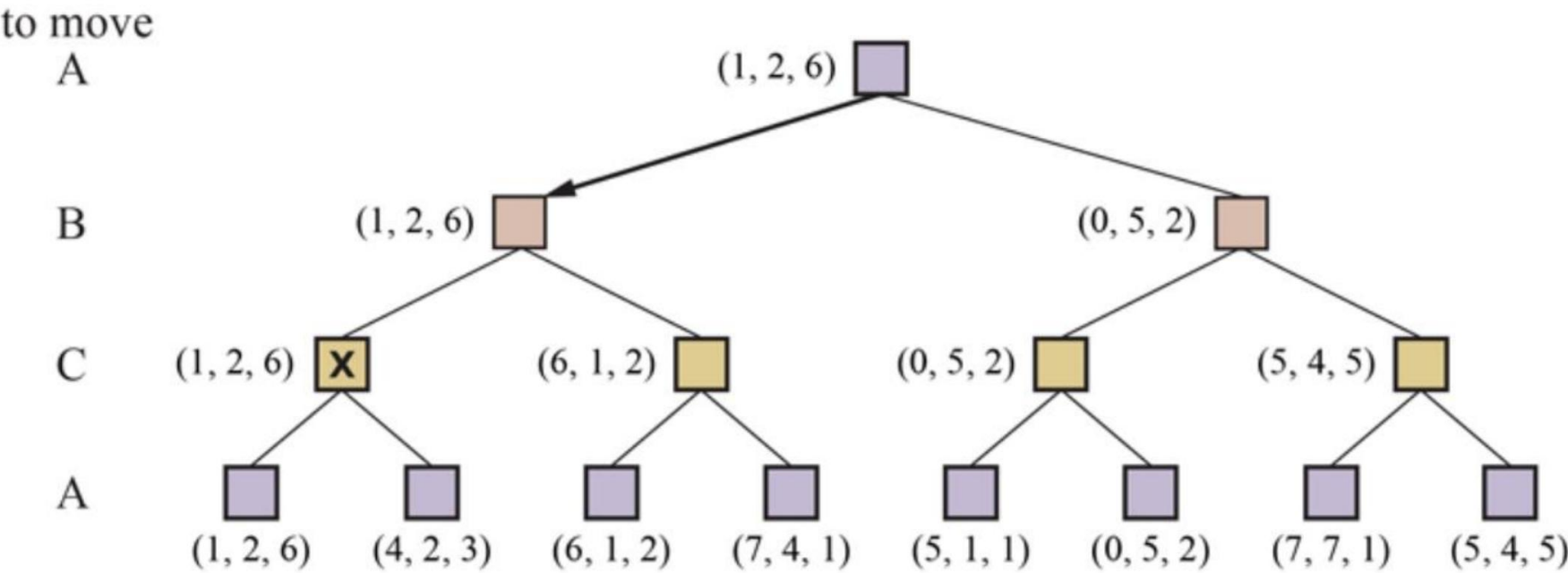
bước đi tối ưu

Đánh giá thuật toán Minimax

- Khám phá cây trò chơi **theo chiều sâu** hoàn chỉnh
- **Tính đầy đủ**: Có (nếu cây hữu hạn)
- **Tối ưu** : Có (trước một đối thủ tối ưu)
- **Độ phức tạp về thời gian**: () **không khả thi đối với trò chơi thực tế**
 - : độ sâu tối đa của cây, : các bước di chuyển hợp pháp tại mỗi điểm
- **Độ phức tạp của không gian**: () (khám phá theo chiều sâu)

Sự tối ưu trong trò chơi nhiều người chơi

- Các chức năng được cải thiện để trả về một vectơ tiện ích.
- Đối với các trạng thái cuối, vectơ này cung cấp tiện ích của trạng thái từ mỗi trạng thái quan điểm của người chơi.



Cây trò chơi ba tầng có ba người chơi (A, B, C). Mỗi nút được dán nhãn giá trị từ quan điểm của mỗi người chơi. Nước đi tốt nhất được đánh dấu ở gốc.22

Sự tối ưu trong trò chơi nhiều người chơi

- Trò chơi nhiều người chơi thường liên quan đến **các liên minh** được tạo ra và bị hỏng khi trò chơi tiếp tục.



A và B yếu trong khi C mạnh.
A liên minh với B.

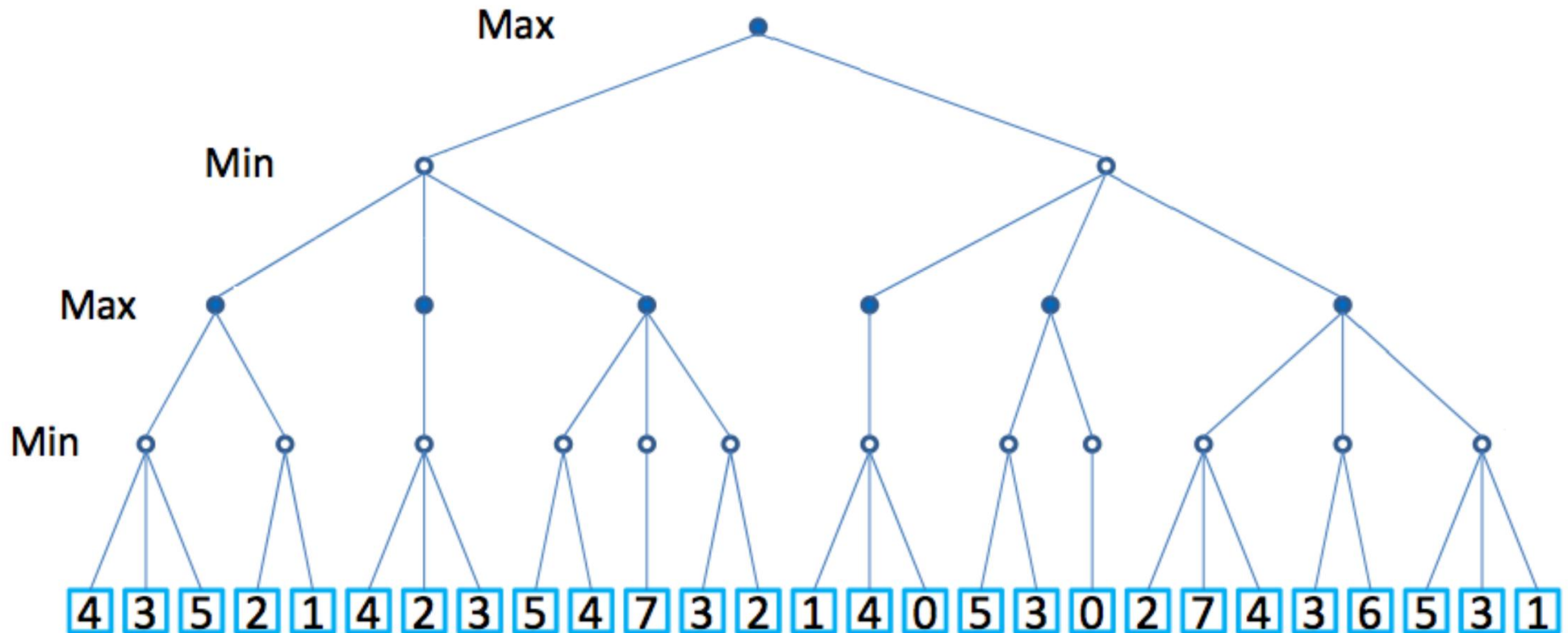


C trở nên yếu đi.
A hoặc B có thể vi phạm thỏa thuận

- Nếu trò chơi không có tổng bằng 0 thì sự hợp tác cũng có thể xảy ra chỉ với hai người chơi.

Câu hỏi 01: Thuật toán Minimax

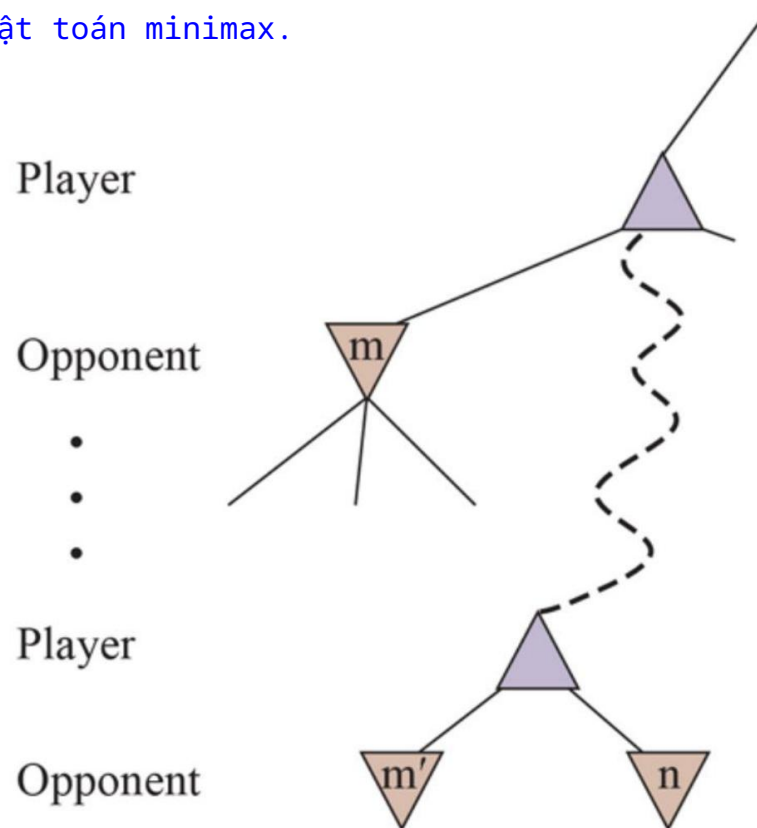
- Tính giá trị tiện ích cho các nút còn lại
- MAX và MIN nên chọn nút nào?



Cắt tỉa alpha-beta

- Việc cắt tỉa alpha-beta nhằm mục đích cắt giảm bất kỳ nhánh nào của cây trò chơi không thể ảnh hưởng đến quyết định cuối cùng.
- Trường hợp xấu nhất của nó cũng tốt như thuật toán minimax.

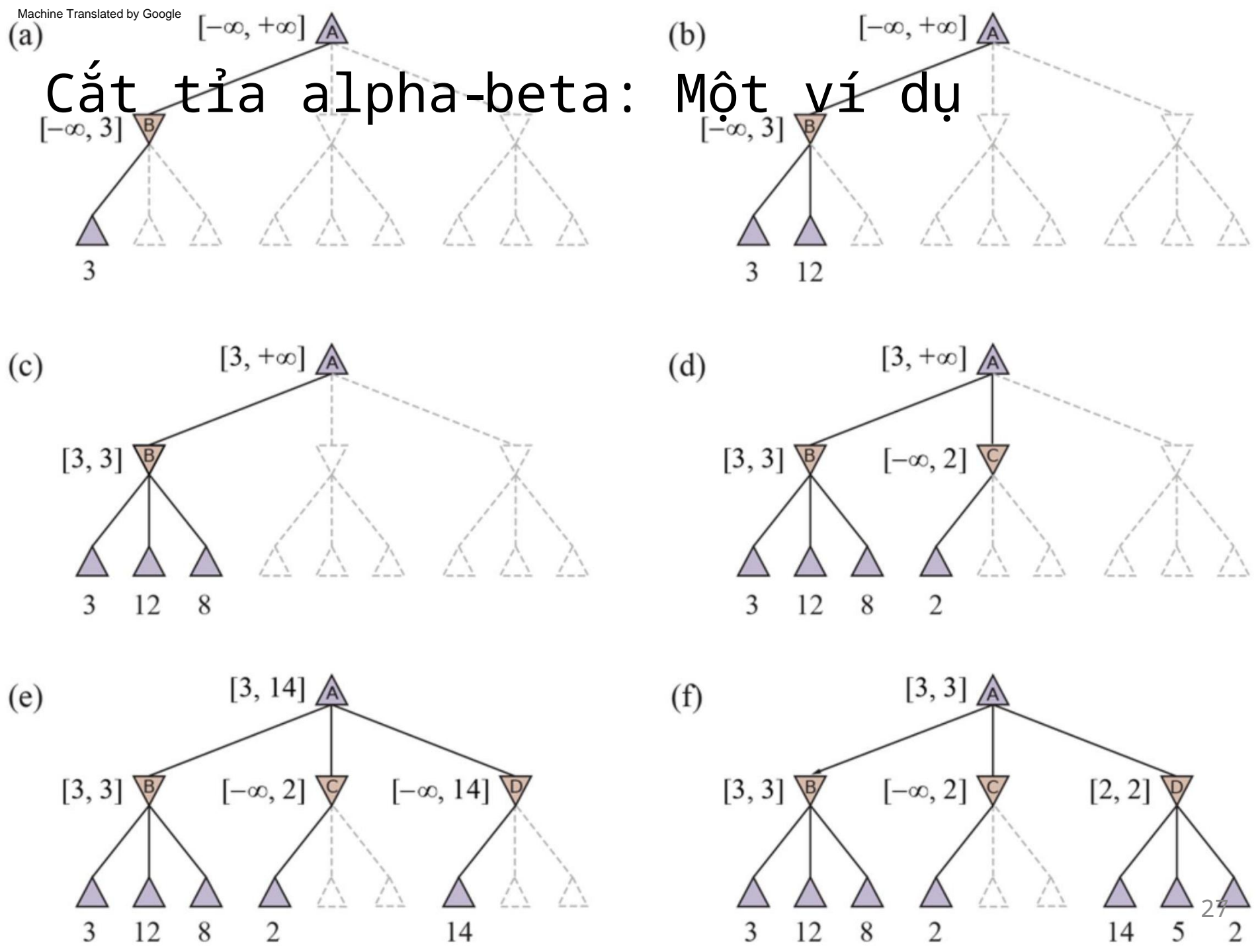
Người chơi có kế hoạch di chuyển đến một nút .
 Nếu anh ta có lựa chọn tốt hơn ở
 cùng cấp độ, ví dụ: nút hoặc ở bất kỳ mức nào
 điểm cao hơn trong cây, ví dụ: nút
 , thì anh ta sẽ không bao giờ di chuyển.

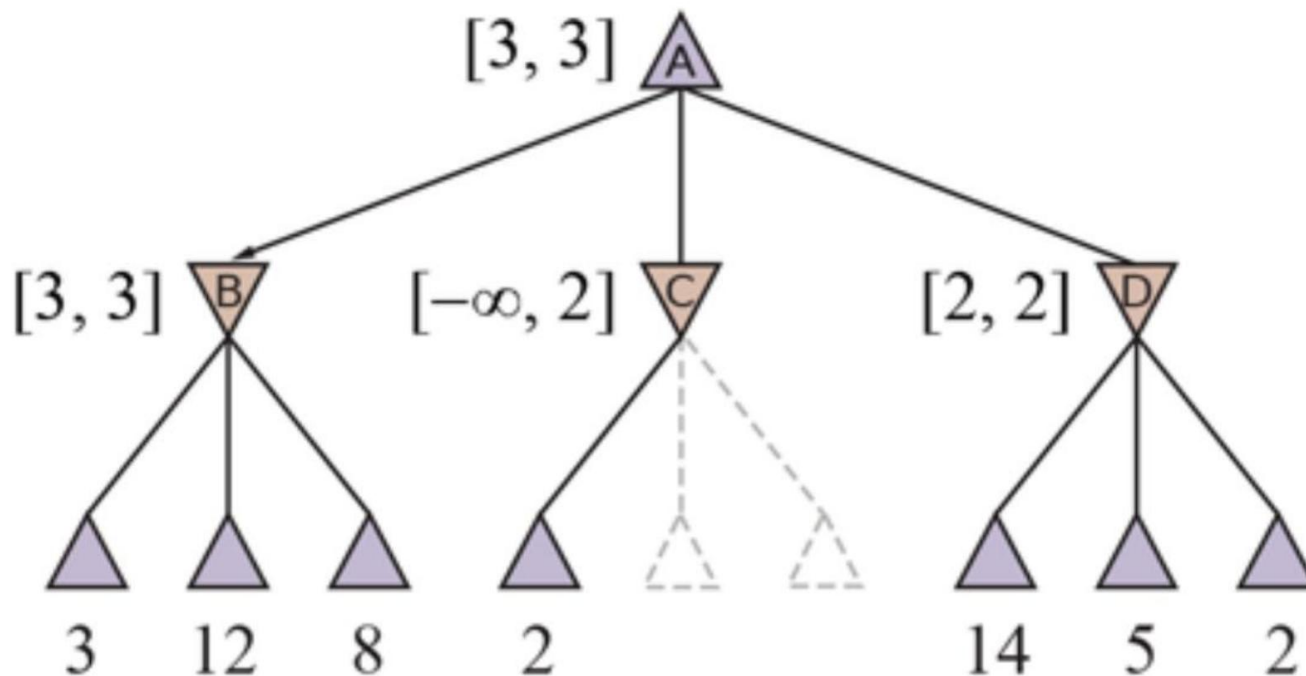


Cắt tỉa alpha-beta

- Hai tham số α và β , mô tả các giới hạn trên xuất hiện ở bất kỳ đâu dọc theo đường dẫn.
 - α = giá trị của lựa chọn tốt nhất (tức là giá trị cao nhất) mà chúng tôi đã tìm thấy cho đến nay tại bất kỳ điểm lựa chọn nào dọc theo đường dẫn cho MAX.
 - β = giá trị của lựa chọn tốt nhất (tức là giá trị thấp nhất) mà chúng ta đã tìm thấy cho đến nay tại bất kỳ điểm lựa chọn nào dọc theo đường đi cho MIN.
- Thuật toán cập nhật các giá trị này trong quá trình thực hiện.
- Việc cắt bớt tại nút hiện tại xảy ra khi giá trị của nó kém hơn giá trị hiện tại hoặc giá trị MAX hoặc MIN tương ứng.

Cắt tỉa alpha-beta: Một ví dụ





- Để hai nút kế tiếp chưa được đánh giá có giá trị và .
- Khi đó giá trị của nút gốc được cho bởi

$$\begin{aligned}
 () &= \max(\min(3, 12, 8), \min(2, \dots), \min(14, 5, 2)) \\
 &= \max(3, \min(2, \dots), 2) \\
 &= \max(3, \dots, 2) \\
 &= 3
 \end{aligned}$$

hàm ALPHA-BETA-SEARCH(trò chơi, trạng thái) trả về một hành động
 người chơi giá trị game.TO-MOVE(trạng
 thái) , nước đi MAX-VALUE(trò chơi, trạng thái, - , +)
 nước đi quay lại

hàm MAX-VALUE(trò chơi, trạng thái, ,) trả về một cặp (tiện ích, nước đi)
 nếu game.IS-TERMINAL(state) thì trả về game.UTILITY(state, player), null
 v -
 cho mỗi a trong game.ACTIONS(state) làm
 v2, a2 MIN-VALUE(game, game.RESULT(state, a), ,) nếu v2 > v
 thì v, di chuyển
 v2, a
 MAX(, v) nếu v
 thì trả về v , di chuyển
 quay lại v, di chuyển

hàm MIN-VALUE(trò chơi, trạng thái, ,) trả về một cặp (tiện ích, nước đi)
 nếu game.IS-TERMINAL(state) thì trả về game.UTILITY(state, player), null
 v +
 cho mỗi a trong game.ACTIONS(state) làm
 v2, a2 MAX-VALUE(game, game.RESULT(state, a), ,) nếu v2 < v
 thì v, di chuyển
 v2, a
 MIN(, v) nếu
 v thì trả về v , di chuyển
 return v, di
 chuyển return v, di chuyển

Thứ tự di chuyển tốt

- Đầu tiên có thể nên xem xét những người kế nhiệm có khả năng là tốt nhất.
- Ví dụ: các nút kế thừa của nút D trong ví dụ trước.
- Alpha-beta với **thứ tự di chuyển hoàn hảo** có thể giải quyết một cây **có độ sâu gần gấp đôi** so với minimax trong cùng một khoảng thời gian. •

Thứ tự nư ớc đi hoàn hảo: $(\sqrt{2})$ hệ số phân nhánh hiệu quả $3 / \sqrt{2}$

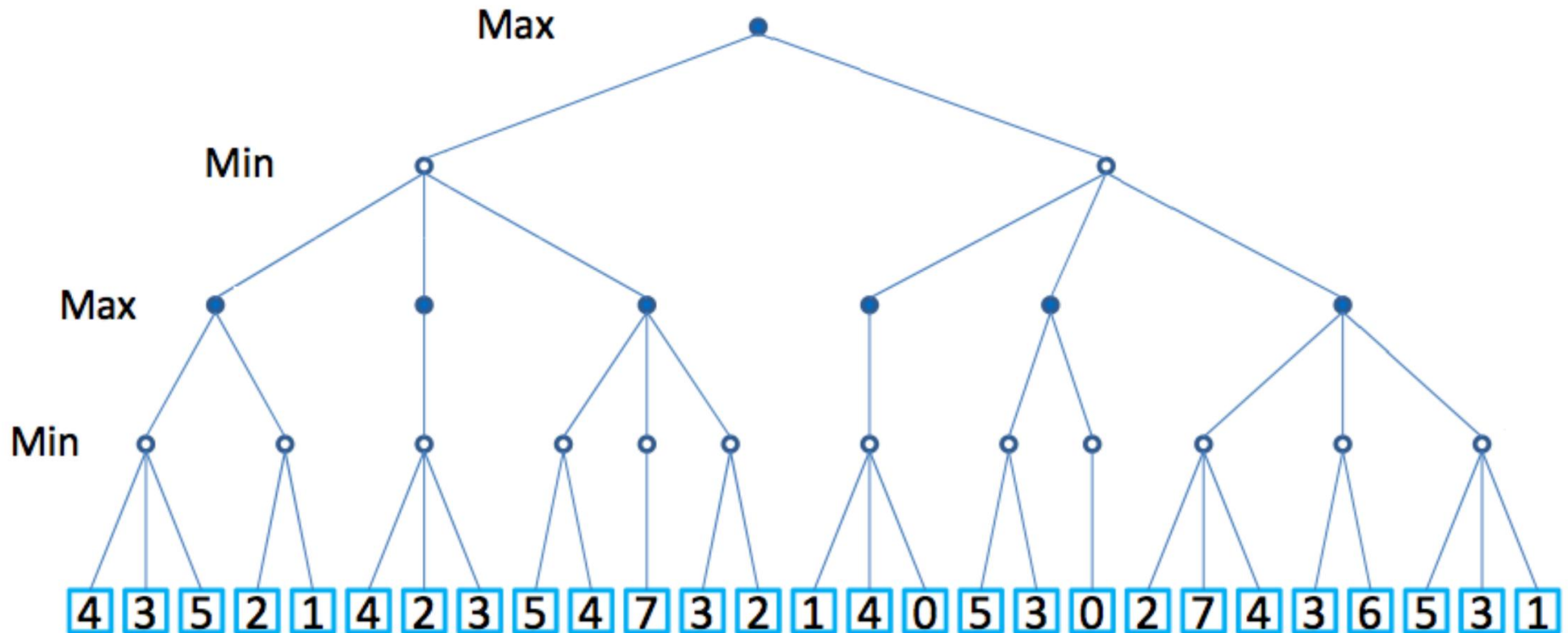
- Thứ tự nư ớc đi ngẫu nhiên: 4) ở mức vừa phải
- (• Rõ ràng, chúng ta **không thể đạt đư ợc thứ tự nư ớc đi hoàn hảo**.

Thứ tự di chuyển tốt

- Các sơ đồ sắp xếp nước đi động đưa chúng ta đến khá gần với giới hạn lý thuyết
 - Ví dụ: trước tiên hãy thử những nước đi được cho là tốt nhất trong quá khứ
- **Heuristic di chuyển sát thủ:** chạy tìm kiếm IDS với độ sâu 1 lớp và ghi lại đường đi tốt nhất, từ đó tìm kiếm sâu hơn 1 lớp.
- **Bảng chuyển vị** tránh việc đánh giá lại một trạng thái bằng cách lưu vào bộ nhớ đệm giá trị phỏng đoán của các trạng thái.
 - Chuyển vị là những hoán vị khác nhau của chuỗi chuyển động kết thúc ở vị trí tương tự.

Câu đố 02: Cắt tỉa alpha-beta

- Tính giá trị tiện ích cho các nút còn lại.
- Những nút nào cần được cắt tỉa?





Quyết định thời gian thực không hoàn hảo

- Chức năng đánh giá
- Cắt tìm kiếm
- Cắt tỉa về phía trước
- Tìm kiếm so với Tra cứu

Heuristic minimax

- Tìm kiếm cắt tỉa cả minimax và alpha-beta các trạng thái đầu cuối.
 - Độ sâu này thường không thực tế vì việc di chuyển phải được thực hiện một cách lưu ý thời gian hợp lý (~ phút).
- Dừng tìm kiếm sớm hơn n với một số giới hạn độ sâu
- Sử dụng chức năng đánh giá
 - Ước tính mức độ mong muốn của vị trí (thắng, thua, hòa?)

$$H\text{-MINIMAX}(s, d) = \begin{cases} \text{EVAL}(s, \text{MAX}) & \text{if IS-CUTOFF}(s, d) \\ \max_{a \in \text{Actions}(s)} H\text{-MINIMAX}(\text{RESULT}(s, a), d + 1) & \text{if To-Move}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} H\text{-MINIMAX}(\text{RESULT}(s, a), d + 1) & \text{if To-Move}(s) = \text{MIN}. \end{cases}$$

Chức năng đánh giá

- Hàm đánh giá này sẽ sắp xếp các trạng thái cuối theo thứ tự giống như chức năng tiện ích thực sự
 - Các quốc gia thắng phải đánh giá tốt hơn các quốc gia hòa, do đó phải tốt hơn là thua lỗ.
- Việc tính toán không được mất quá nhiều thời gian!
- Đối với các trạng thái không kết thúc, lệnh của chúng phải tương quan chặt chẽ với cơ hội chiến thắng thực tế.

Chức năng đánh giá

- Đối với cờ vua, **tổng các đặc điểm** có trọng số tuyến tính thường là

$$() = () + () + . + ()$$

- số của từng loại quân cờ trên bảng là ở đâu,
và có thể là giá trị của các mảnh
- Ví dụ, $() = 9 + 5 + 3 + 3 +$
- Giả định mạnh mẽ tiềm ẩn: sự đóng góp của từng tính năng độc lập với giá trị của các đặc trưng khác.
- Ví dụ: gán giá trị 3 cho một quân tượng mà bỏ qua thực tế là quân tượng mạnh hơn về cuối Sự kết hợp phi tuyến tính

Cắt tìm kiếm

- Điểm cắt Minimax giống với Giá trị Minimax ngoại trừ

1. ? đư ợc thay thế bởi ?

2. đư ợc thay thế bởi

Nếu IS-CUTOFF(trạng thái, độ sâu) thì trả về EVAL(trạng thái)

- Nó có hiệu quả trong thực tế không?

• = 106 , = 35 = 4

• Nhìn trư ớc 4 lớp là một tay chơ i cờ vô vọng!

• 4 lớp \approx ngư ời mới làm quen, 8 lớp \approx PC thông thư ờng, con ngư ời chủ, 12 lớp \approx Màu xanh đậm, Kasparov

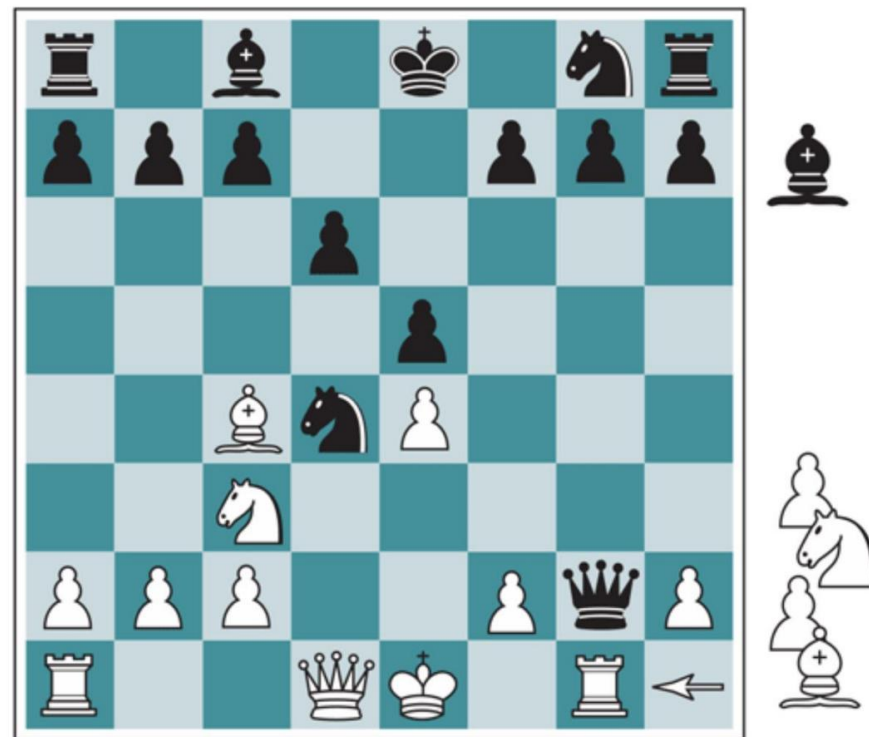
Một thử nghiệm cắt giảm phức tạp hơn

- **Vị trí yên tĩnh** là những vị trí không có khả năng biểu hiện sự dao động mạnh về giá trị trong thời gian sắp tới.
 - Ví dụ, trong cờ vua, các vị trí có thể bắt được lợi thế không đứng yên chỉ đối với vật liệu đếm có chức năng đánh giá
- **Tìm kiếm tĩnh:** mở rộng các vị trí không tĩnh cho đến khi đạt được vị trí tĩnh.

Vị trí tĩnh: Một ví dụ



(a) White to move



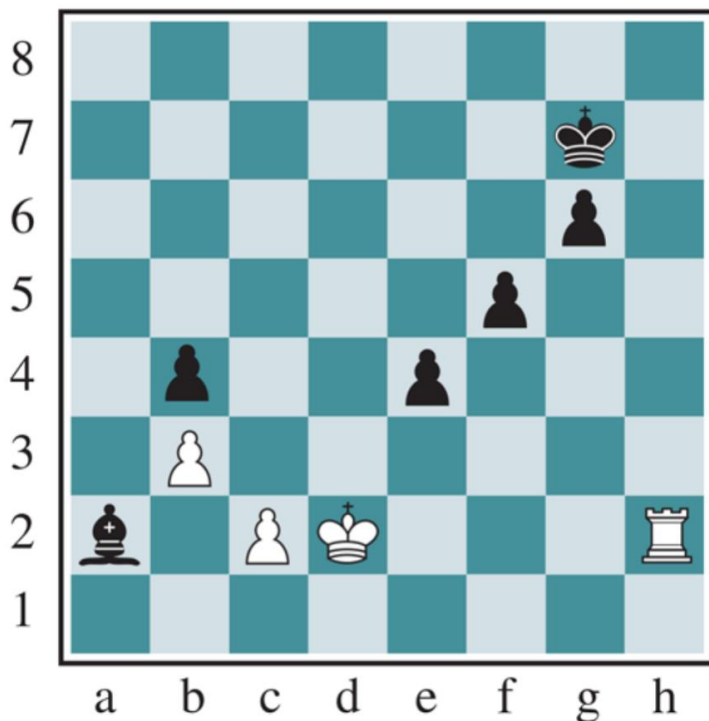
(b) White to move

Hai thế cờ chỉ khác nhau ở vị trí quân xe ở phía dưới bên phải.

Trong (a), Đen có lợi thế về một quân mã và hai con tốt, điều này là đủ để thắng trò chơi. Trong (b), Trắng sẽ bắt được hậu, tạo cho quân hậu một lợi thế đủ mạnh để giành chiến thắng.

Một thử nghiệm cắt giảm phức tạp hơn

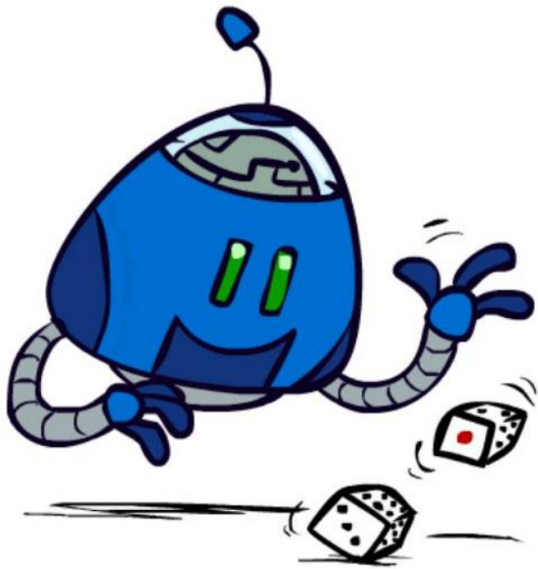
- **Hiệu ứng chân trời:** Chương trình đang phải đối mặt với một vấn đề nghiêm trọng có thể tránh khỏi mất mát và tạm thời tránh nó bằng cách trì hoãn chiến thuật.



Với việc Đen di chuyển, quân tượng đen chắc chắn sẽ phải chịu số phận. Nhưng Đen có thể ngăn chặn sự kiện đó bằng cách kiểm tra quân tốt của vua trắng, buộc vua phải bắt quân tốt.

Một thử nghiệm cắt giảm phức tạp hơn

- **Phần mở rộng đơn lẻ:** một bước đi “rõ ràng là tốt hơn” tất cả di chuyển khác ở một vị trí nhất định.
 - Thuật toán cho phép xem xét sâu hơn về một phần mở rộng số ít hợp pháp cây tìm kiếm sâu hơn, tuy nhiên chỉ có một số phần mở rộng số ít.
- **Tìm kiếm tia**
 - Cắt tia về phía trước, chỉ xem xét một “chùm” nước đi tốt nhất mà thôi
 - Hầu hết con người chỉ xem xét một vài bước di chuyển từ mỗi vị trí
 - PROBCUT, hay thuật toán cắt xác suất (Buro, 1995)
- **Tìm kiếm và tra cứu**
 - Sử dụng tính năng tra cứu bảng thay vì tìm kiếm phần mở đầu và kết thúc

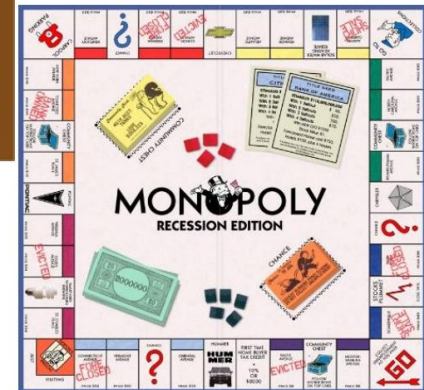
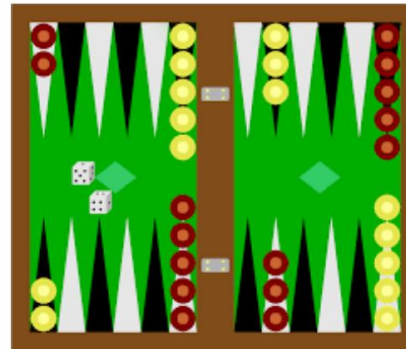
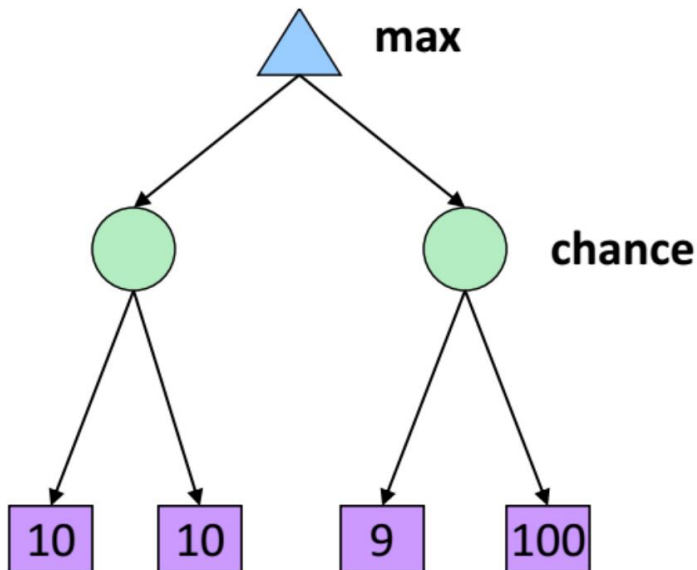


ngẫu nhiên

Trò chơi

Hành vi ngẫu nhiên

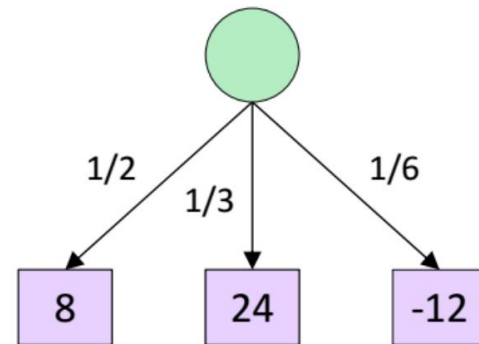
- Kết quả không chắc chắn được kiểm soát một cách ngẫu nhiên, không phải do đối thủ!
- Tại sao chúng ta không biết kết quả của một hành động sẽ như thế nào?
 - Tính ngẫu nhiên rõ ràng: tung xúc xắc
 - Đối thủ khó đoán: các hồn ma phản ứng ngẫu nhiên
 - Hành động có thể thất bại: khi robot đang di chuyển, bánh xe có thể bị trượt



Tìm kiếm mong đợi

- Giá trị phản ánh kết quả của trường hợp trung bình (kỳ vọng), không phải kết quả trường hợp xấu nhất (minimax)

- **Tìm kiếm Expectimax:** tính điểm trung bình khi chơi tối ưu



$$v = (1/2)(8) + (1/3)(24) + (1/6)(-12) = 10$$

- Các nút tối đa như trong tìm kiếm minimax
- Các nút cơ hội giống như các nút tối thiểu, nhưng kết quả không chắc chắn
- Tính toán mức hữu dụng kỳ vọng, tức là lấy bình quân gia quyền của trẻ em
- Đối với minimax, thang đo chức năng đầu cuối không thành vấn đề
 - Các phép biến đổi đơn điệu: trạng thái tốt hơn n để có đánh giá cao hơn n
- Đối với Expectimax, chúng ta cần độ lớn có ý nghĩa

Tìm kiếm Expectimax: Mã giả

```
def value(state):
```

```
    if the state is a terminal state: return the state's utility
```

```
    if the next agent is MAX: return max-value(state)
```

```
    if the next agent is EXP: return exp-value(state)
```

```
def max-value(state):
```

```
    initialize v =  $-\infty$ 
```

```
    for each successor of state:
```

```
        v = max(v, value(successor))
```

```
    return v
```

```
def exp-value(state):
```

```
    initialize v = 0
```

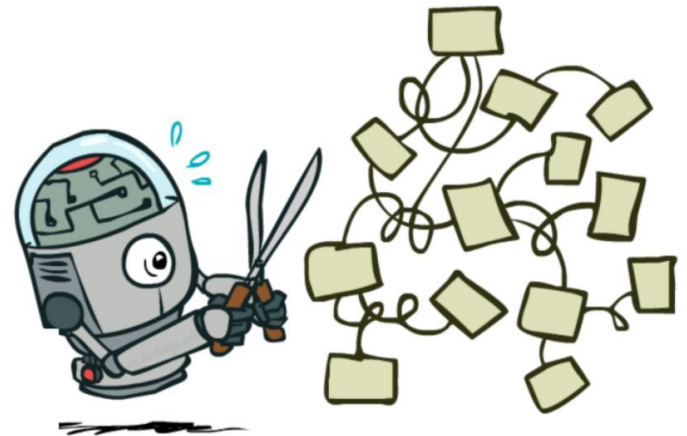
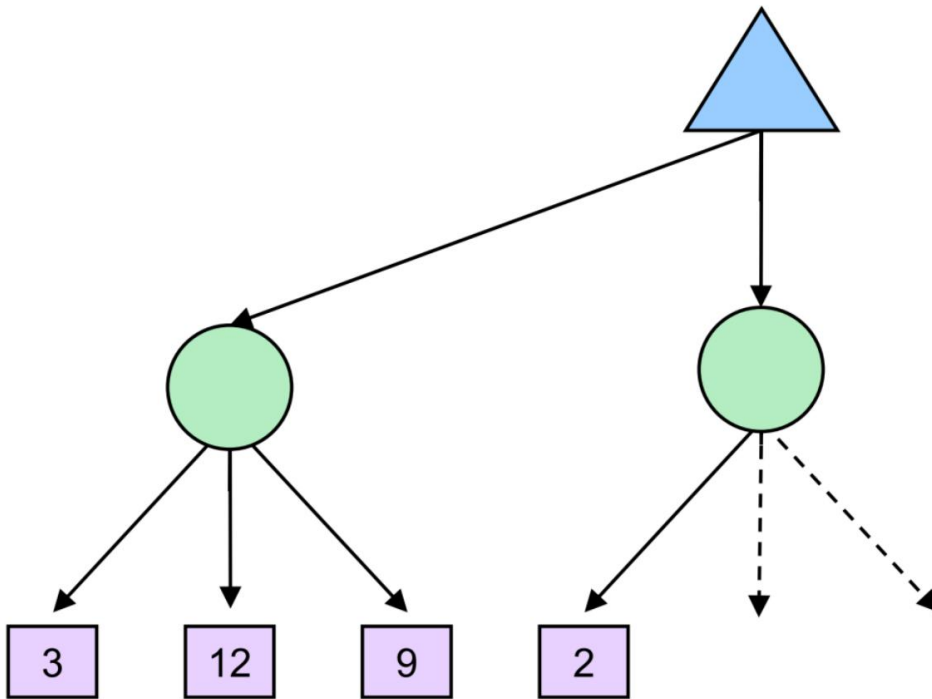
```
    for each successor of state:
```

```
        p = probability(successor)
```

```
        v += p * value(successor)
```

```
    return v
```

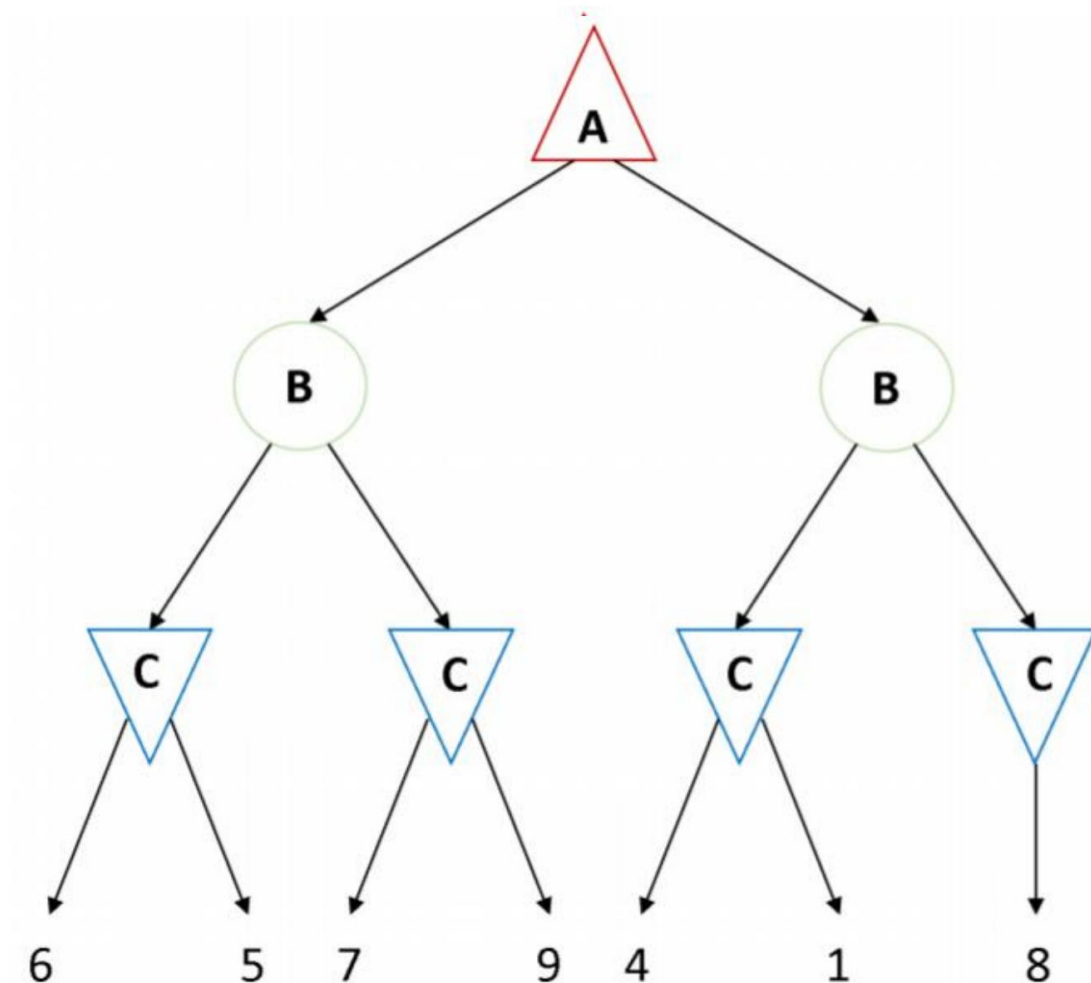
cắt tỉa Expectimax



Có thể thực hiện việc cắt tỉa trong tìm kiếm Expectimax không?

cắt tỉa Expectimax

- Việc cắt tỉa chỉ có thể thực hiện được khi có kiến thức về phạm vi cố định.



Làm thế nào để tỉa cây này?

- Mỗi trẻ có xác suất được chọn như nhau
- Các giá trị chỉ có thể nằm trong khoảng 0-9 (bao gồm).

Kỳ vọng giới hạn độ sâu

