

بسمه تعالی

هوش مصنوعی
جستجوی تخصصی و بازیها
نیمسال اول ۱۴۰۳-۱۴۰۲

دکتر مازیار پالهنک
آزمایشگاه هوش مصنوعی
دانشکده مهندسی برق و کامپیوتر
دانشگاه صنعتی اصفهان

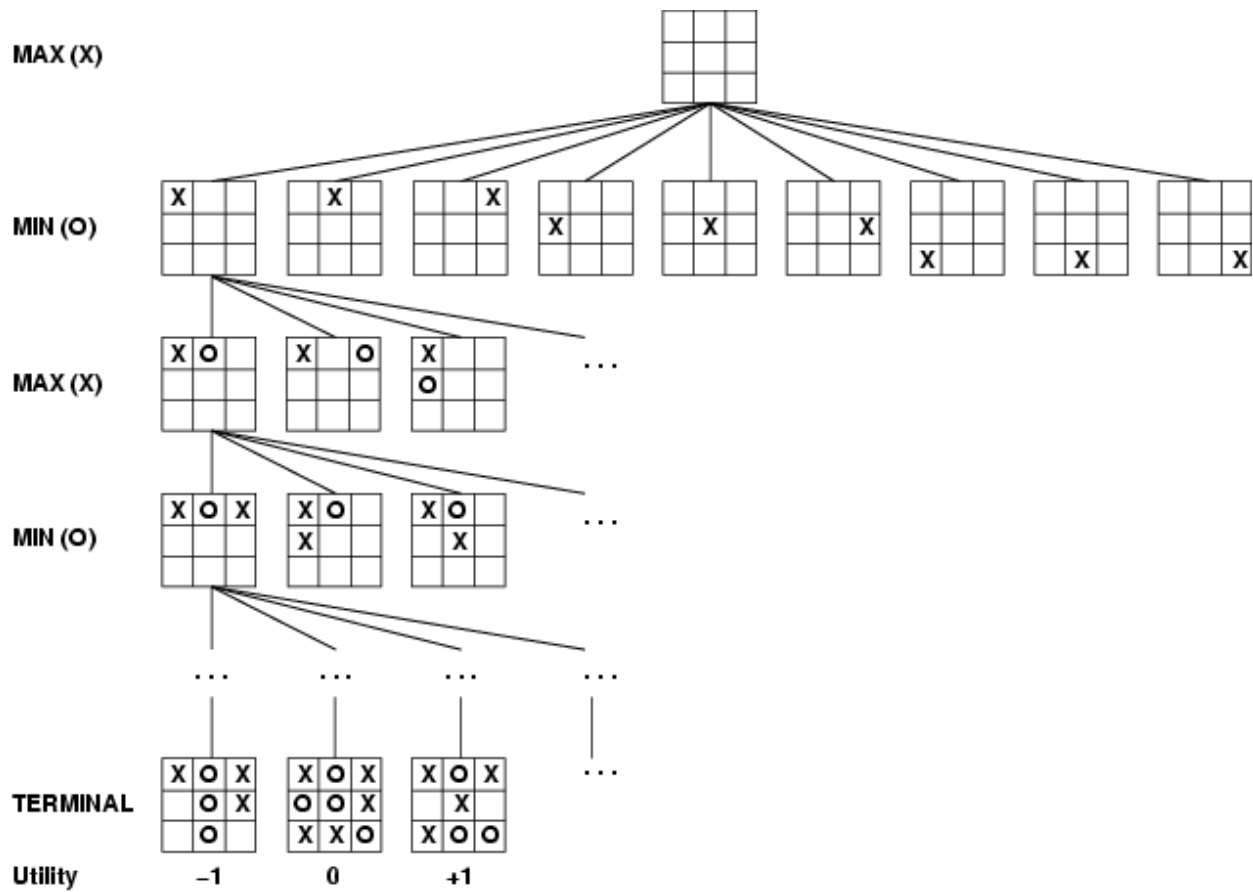
مقدمه

- در محیطهای چندعاملی، عامل ممکن است همکار بوده یا با هم در رقابت باشند.
- هنگامی که عاملها با هم در رقابت هستند، یعنی اهداف متضادی دارند، جستجوی تخصی می که به آن بازی نیز گفته می شود استفاده می شود.
- از ابتدای پیدایش کامپیوتر

بازی دو نفره

- بازی دو نفره هر یک به نوبت
- تعریف به عنوان یک مسئله جستجو
- حالت اولیه: وضعیت تخته S_0
- نوبت کدام بازیکن $To-Move(s)$
- اعمال مجاز در حالت s : $Actions(s)$
- مدل انتقال $Result(s,a)$: نتیجه انجام عمل a در حالت s
- تست هدف: تعیین خاتمه بازی $Is-Terminal(s)$
- حالاتی که بازی در آنها خاتمه می یابد حالات پایانی (ترمینال)
- ارزیابی: امتیاز به وضعیت نهایی بازی $Utility(s,p)$
- مقدار عددی نهایی بازی با حالت پایانی s برای بازیکن p

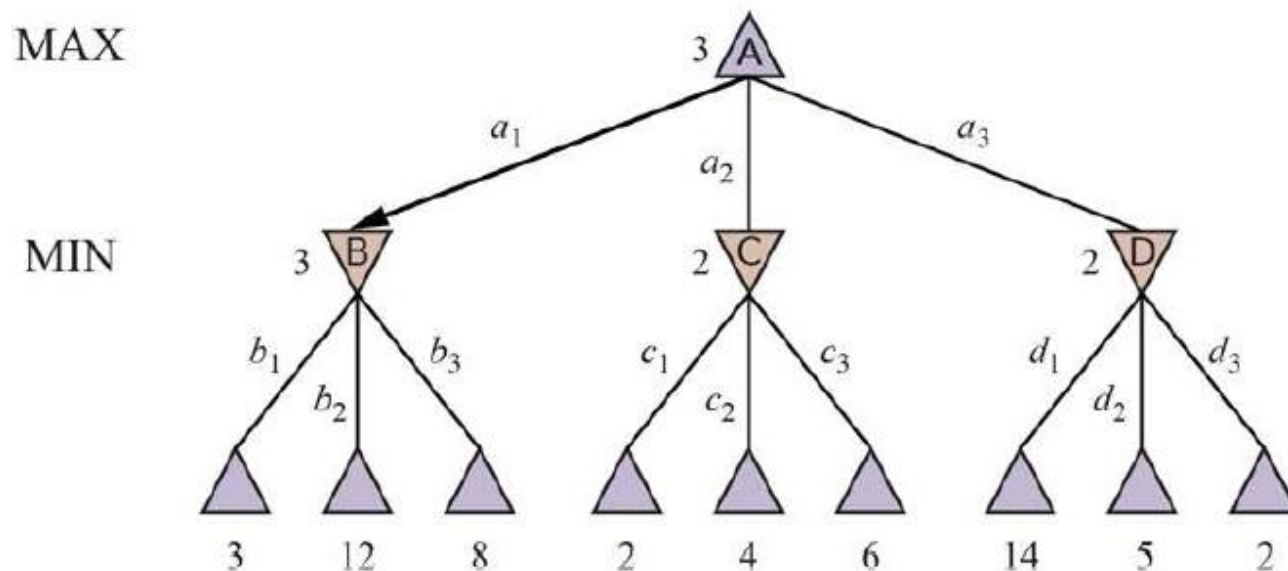
- برای شطرنج نتیجه بازی برد، باخت، یا مساوی $(1, -1, 0)$
- بازی جمع-صفر zero-sum game به بازی گفته می شود که جمع امتیاز بازیکنان برابر صفر شود.
- برای شطرنج $1 + -1 = -1 + 1 = 0 + 0$
- با اغماض به بازیهایی که جمع امتیاز بازیکنان همواره برابر شود.
- اگر امتیازها بصورت $(1, 0, 1/2)$ باشد جمع همواره یک است.
- حالت اولیه و حرکات مجاز یک درخت بازی ایجاد می کنند.



- MAX باید تمام حرکات MIN را در نظر بگیرد.
- به نوعی یک مسئلهٔ اقتضائی است.
- برای بازیهای که نتایج دوتائی (برنده-بازنده) دارند، می توان جستجوی AND-OR را برای تولید طرح شرطی استفاده کرد.
- در واقع همانند وضعیت غیرقطعی می باشد.
- برای بازیهای با نتایج بیشتر جستجوی minimax را می توان استفاده نمود.

Minimax

■ انتخاب حرکتی که بیشترین مقدار کمینه-بیشینه را دارد.





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


```
function MINIMAX-SEARCH(game, state) returns an action  
  player  $\leftarrow$  game.TO-MOVE(state)  
  value, move  $\leftarrow$  MAX-VALUE(game, state)  
  return move
```

function MINIMAX-SEARCH(*game, state*) **returns** *an action*

player \leftarrow *game*.TO-MOVE(*state*)

value, move \leftarrow MAX-VALUE(*game, state*)

return *move*

function MAX-VALUE(*game, state*) **returns** *a (utility, move) pair*

if *game*.IS-TERMINAL(*state*) **then return** *game*.UTILITY(*state, player*), *null*

v $\leftarrow -\infty$

for each *a* **in** *game*.ACTIONS(*state*) **do**

v2, a2 \leftarrow MIN-VALUE(*game, game*.RESULT(*state, a*))

if *v2* > *v* **then**

v, move \leftarrow *v2, a*

return *v, move*

function MINIMAX-SEARCH(*game, state*) **returns** an action

 player \leftarrow game.TO-MOVE(*state*)

 value, move \leftarrow MAX-VALUE(*game, state*)

return move

function MAX-VALUE(*game, state*) **returns** a (utility, move) pair

if game.IS-TERMINAL(*state*) **then return** game.UTILITY(*state, player*), null

$v \leftarrow -\infty$

for each *a* **in** game.ACTIONS(*state*) **do**

$v2, a2 \leftarrow$ MIN-VALUE(*game, game.RESULT(state, a)*)

if $v2 > v$ **then**

$v, move \leftarrow v2, a$

return $v, move$

function MIN-VALUE(*game, state*) **returns** a (utility, move) pair

if game.IS-TERMINAL(*state*) **then return** game.UTILITY(*state, player*), null

$v \leftarrow +\infty$

for each *a* **in** game.ACTIONS(*state*) **do**

$v2, a2 \leftarrow$ MAX-VALUE(*game, game.RESULT(state, a)*)

if $v2 < v$ **then**

$v, move \leftarrow v2, a$

return $v, move$

- کامل: بله (اگر درخت محدود باشد)
- بهینه: بله (درمقابل حریف بهینه)
- اگر حریف غیربهینه باشد، MAX حداقل همانند حریف بهینه عمل می کند.
- با فرض m حداکثر عمق درخت و
- b حرکات مجاز در هر نقطه
- پیچیدگی زمانی: $O(b^m)$
- پیچیدگی فضا:
- اگر همه حرکات بسط داده شوند $O(bm)$
- اگر فقط یکی از حرکات بسط داشته شود $O(m)$

■ چون پیچیدگی زمانی نمائی است برای بازیهای پیچیده غیر عملی است.

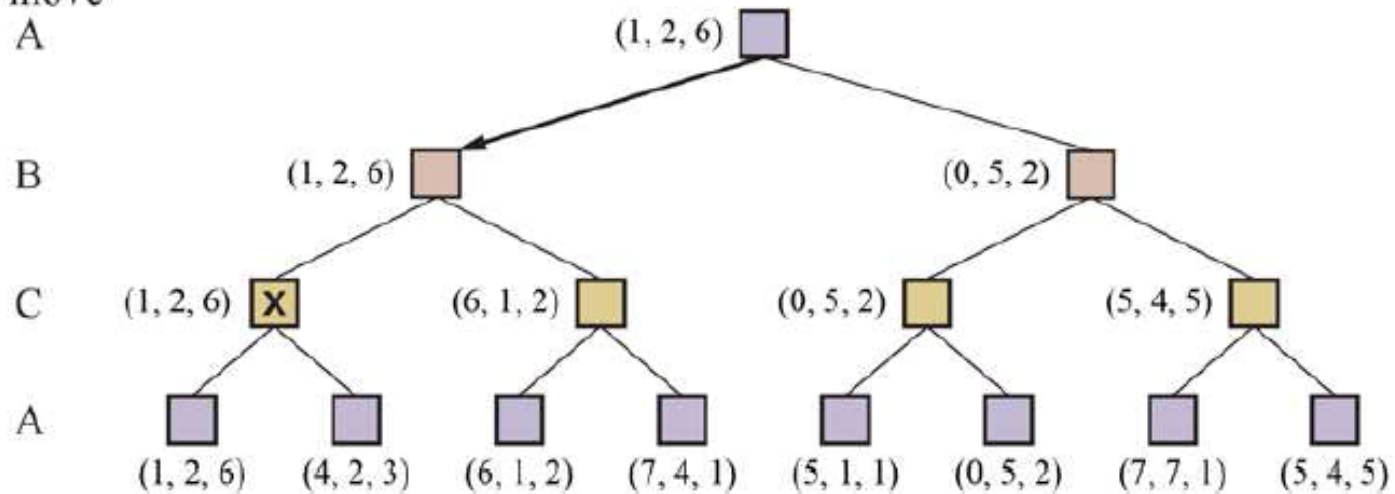
■ برای شطرنج b حدود ۳۵ و m حدود ۸۰

■ مقدار $10^{123} \approx 35^{80}$ غیر عملی

بازی با چند بازیکن

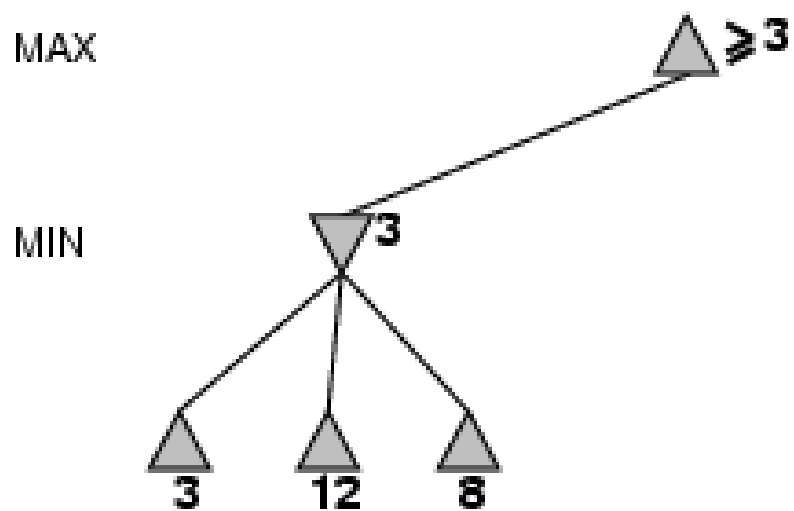
Figure 5.4

to move
A

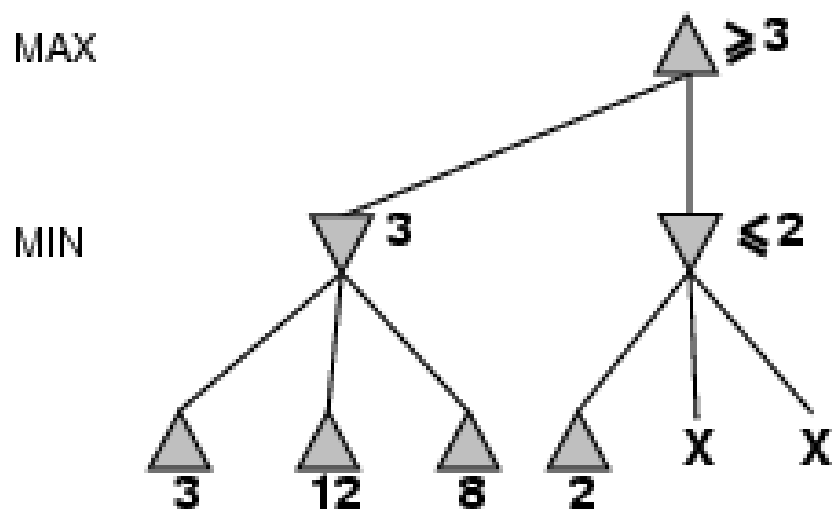


The first three ply of a game tree with three players (A , B , C). Each node is labeled with values from the viewpoint of each player. The best move is marked at the root.

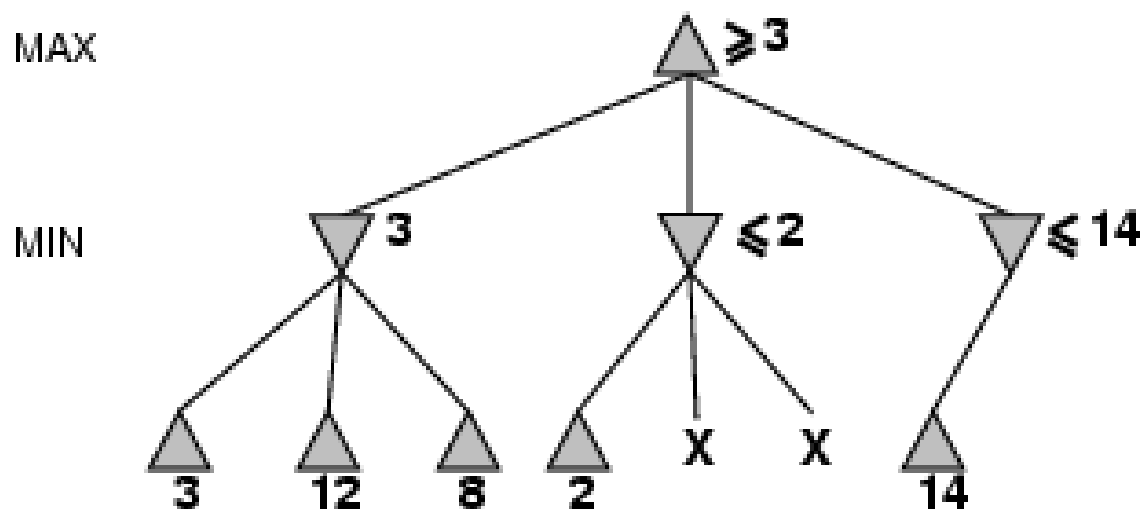
هرس α - β



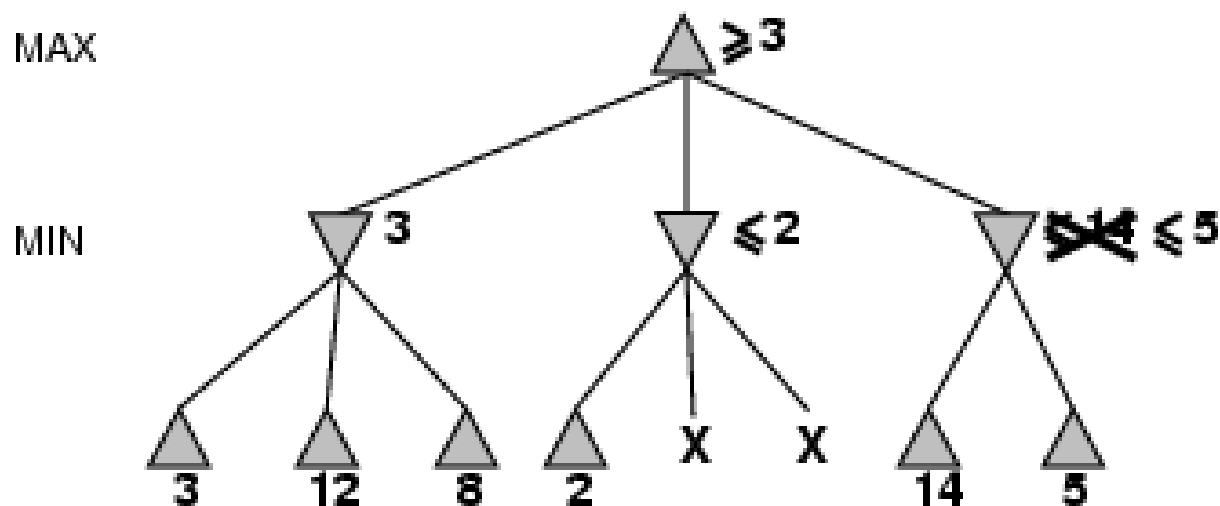
هرس α - β



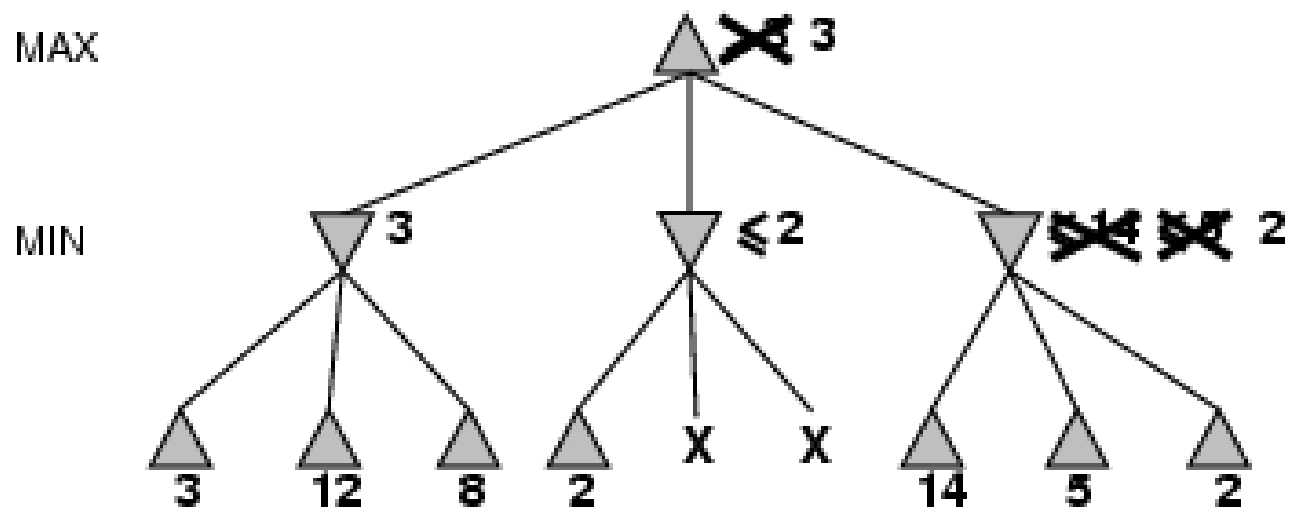
هرس α - β

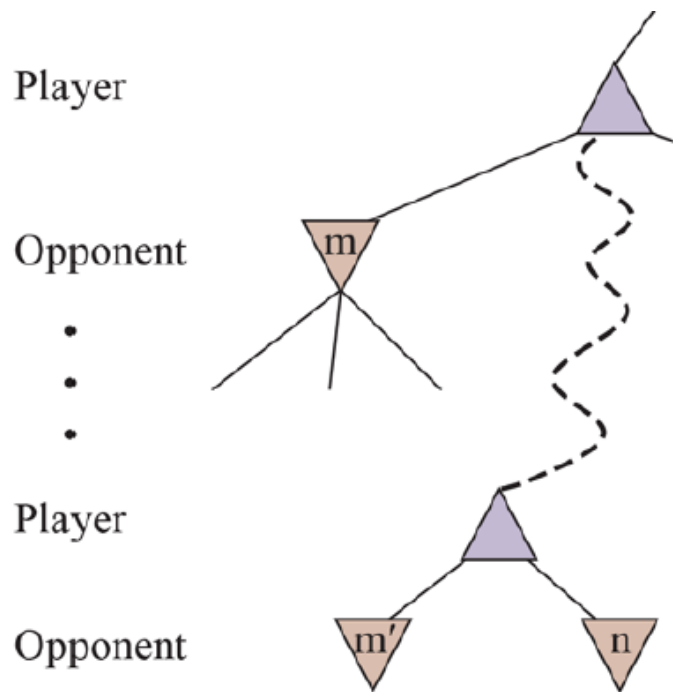


هرس α - β



هرس α - β





- عمل هرس در هر عمقی ممکن است رخ دهد.
- اگر m یا m' بهتر از n برای player باشند، در بازی هیچگاه به n نخواهیم رسید.
- هنگامی که اطلاعات کافی در مورد n بدست آمد می توان آن را هرس کرد.

هرس α - β

- α مقدار بهترین انتخاب (بالاترین مقدار) که در مسیر MAX تاکنون یافته شده است.
- تصور کنید α = حداقل
- β مقدار بهترین انتخاب (کمترین مقدار) که در مسیر MIN تاکنون یافته شده است.
- تصور کنید β = حداکثر

```
function ALPHA-BETA-SEARCH(game, state) returns an action  
  player  $\leftarrow$  game.TO-MOVE(state)  
  value, move  $\leftarrow$  MAX-VALUE(game, state,  $-\infty, +\infty$ )  
  return move
```

function ALPHA-BETA-SEARCH(*game, state*) **returns** an action

player \leftarrow *game*.TO-MOVE(*state*)

value, move \leftarrow MAX-VALUE(*game, state, $-\infty, +\infty$*)

return *move*

function MAX-VALUE(*game, state, α, β*) **returns** a (*utility, move*) pair

if *game*.IS-TERMINAL(*state*) **then return** *game*.UTILITY(*state, player*), *null*

v $\leftarrow -\infty$

for each *a* **in** *game*.ACTIONS(*state*) **do**

v2, a2 \leftarrow MIN-VALUE(*game, game.RESULT(state, a), α, β*)

if *v2* > *v* **then**

v, move \leftarrow *v2, a*

$\alpha \leftarrow$ MAX(α, v)

if *v* $\geq \beta$ **then return** *v, move*

return *v, move*

اصلاح حداقل مقدار

اگر کمترین مقداری که Max برمی گرداند از امتیاز
کنونی Min بالاسر بیشتر است بیشتر جستجو نکن

function ALPHA-BETA-SEARCH(*game, state*) **returns** an action

player \leftarrow *game*.TO-MOVE(*state*)

value, move \leftarrow MAX-VALUE(*game, state, $-\infty, +\infty$*)

return *move*

function MAX-VALUE(*game, state, α, β*) **returns** a (*utility, move*) pair

if *game*.IS-TERMINAL(*state*) **then return** *game*.UTILITY(*state, player*), null

v $\leftarrow -\infty$

for each *a* **in** *game*.ACTIONS(*state*) **do**

v2, a2 \leftarrow MIN-VALUE(*game, game.RESULT(state, a), α, β*)

if *v2* > *v* **then**

v, move \leftarrow *v2, a*

$\alpha \leftarrow$ MAX(α, v)

if *v* $\geq \beta$ **then return** *v, move*

return *v, move*

اصلاح حداقل مقدار

اگر کمترین مقداری که Max برمی گرداند از امتیاز
کنونی Min بالاسر بیشتر است بیشتر جستجو نکن

function MIN-VALUE(*game, state, α, β*) **returns** a (*utility, move*) pair

if *game*.IS-TERMINAL(*state*) **then return** *game*.UTILITY(*state, player*), null

v $\leftarrow +\infty$

for each *a* **in** *game*.ACTIONS(*state*) **do**

v2, a2 \leftarrow MAX-VALUE(*game, game.RESULT(state, a), α, β*)

if *v2* < *v* **then**

v, move \leftarrow *v2, a*

$\beta \leftarrow$ MIN(β, v)

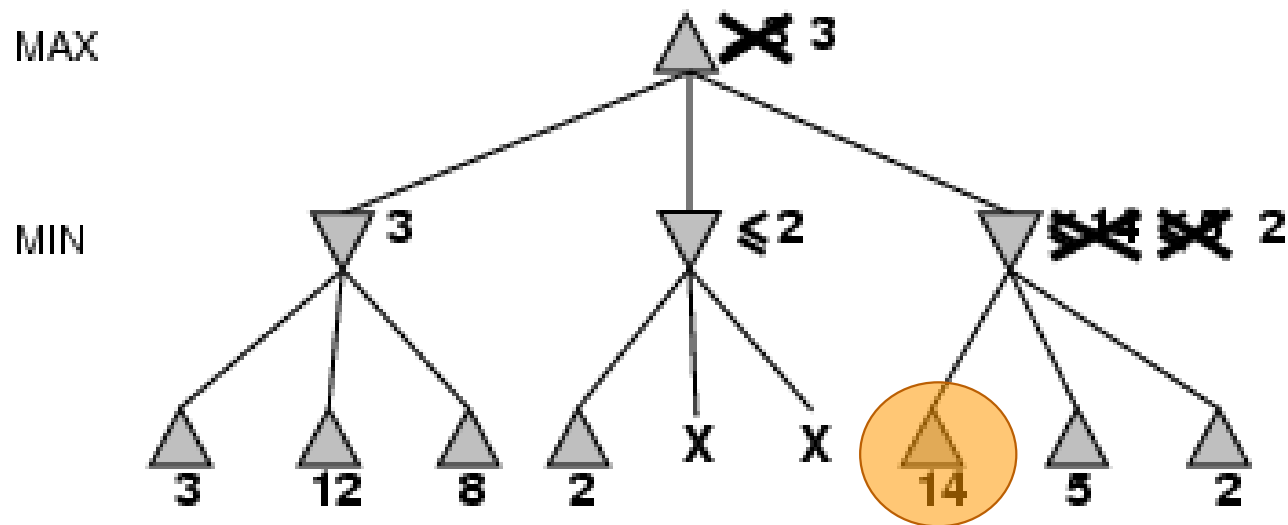
if *v* $\leq \alpha$ **then return** *v, move*

return *v, move*

اصلاح حداکثر مقدار

اگر حداکثر مقداری که Min برمی گرداند از امتیاز
کنونی Max بالاسر کمتر است بیشتر جستجو نکن

■ کار آئی قطع آلفا-بتا تا حد زیادی وابسته به ترتیبی است که تالیها را در نظر می گیریم.



محدودیت منابع

- Minimax همه درخت را جستجو می کند.
- آلفا-بتا حداقل برای جزئی از فضای جستجو باید تا حالت انتهائی ادامه دهد.
- در عمل همیشه امکان ندارد
- قطع جستجو - اعمال یک تابع ارزیابی

تابع ارزیابی

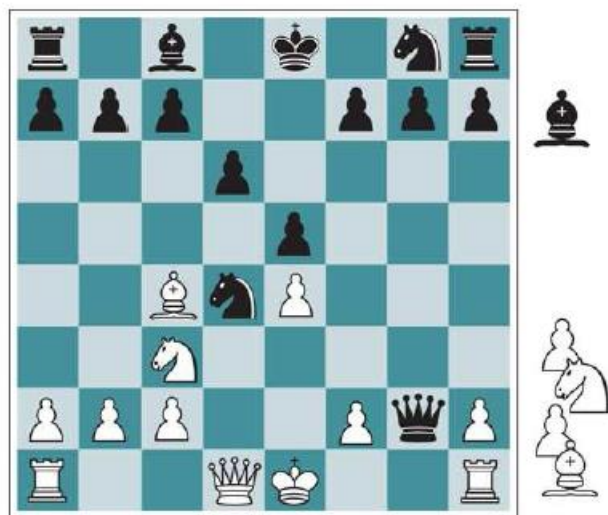
■ مثلاً برای شطرنج

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s) \quad \blacksquare$$

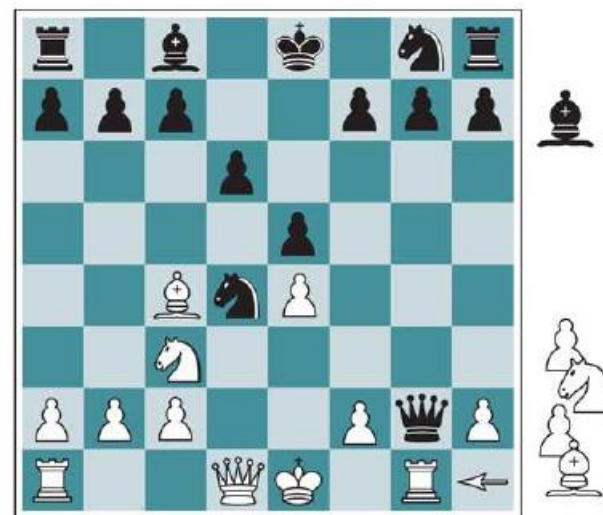
■ مثلاً $w_1 = 3$ و

■ $f_1(s) = (\text{تعداد فیلهای سیاه}) - (\text{تعداد فیلهای سفید})$

■ البته همیشه ممکن است روابط خطی برقرار نباشد.



(a) White to move



(b) White to move

وضعیت

- شطرنج: برنامه IBM Deep Blue در سال ۱۳۷۵ توانست گری کاسپارف را شکست دهد.
- بازی GO: برنامه AlphaGo از شرکت Deep Mind در سال ۱۳۹۶ توانست قهرمان جهان را ببرد.
- استفاده از روش جستجوی درختی مونت کارلو Monte Carlo Tree Search (MCTS)



مازیار پالهنک

هوش مصنوعی

30

- دقت نمائید که پاورپوینت ابزاری جهت کمک به یک ارائه شفاهی می باشد و به هیچ وجه یک جزوه درسی نیست و شما را از خواندن مراجع درس بی نیاز نمی کند.
- لذا حتماً مراجع اصلی درس را مطالعه نمائید.