هوش مصنوعی

جستجوی تخاصمی

(Game Theory)

http://students.madadyar.com

بازی ها چیستند و چرا مطالعه می شوند؟

بازیها حالتی از محیطهای چند عاملي هستند.

- هر عامل نیاز به در نظر گرفتن سایر عاملها و چگونگی تأثیر آنها دارد.
- تمایز بین محیطهای چند عامل رقابتي و همکار.
- محیطهای رقابتی ، که در آنها اهداف عاملها با یکدیگر برخورد دارند ، منجر به مسئله های خصمانه میشود که به عنوان بازی شناخته میشوند

چرا مطالعه مي‌شوند؟

- قابلیتهای هوشمندی انسانها را به کار میگیرند.
- ماهیت انتزاعی بازی ها.
- حالت بازی را به راحتی میتوان نمایش داد و عاملها معمولا به مجموعه کوچکی از فعالیتها محدود هستند که نتایج آنها با قوانین دقیقي تعریف شده اند.

یک نمونه بازی

بازی دو نفره Min و Max

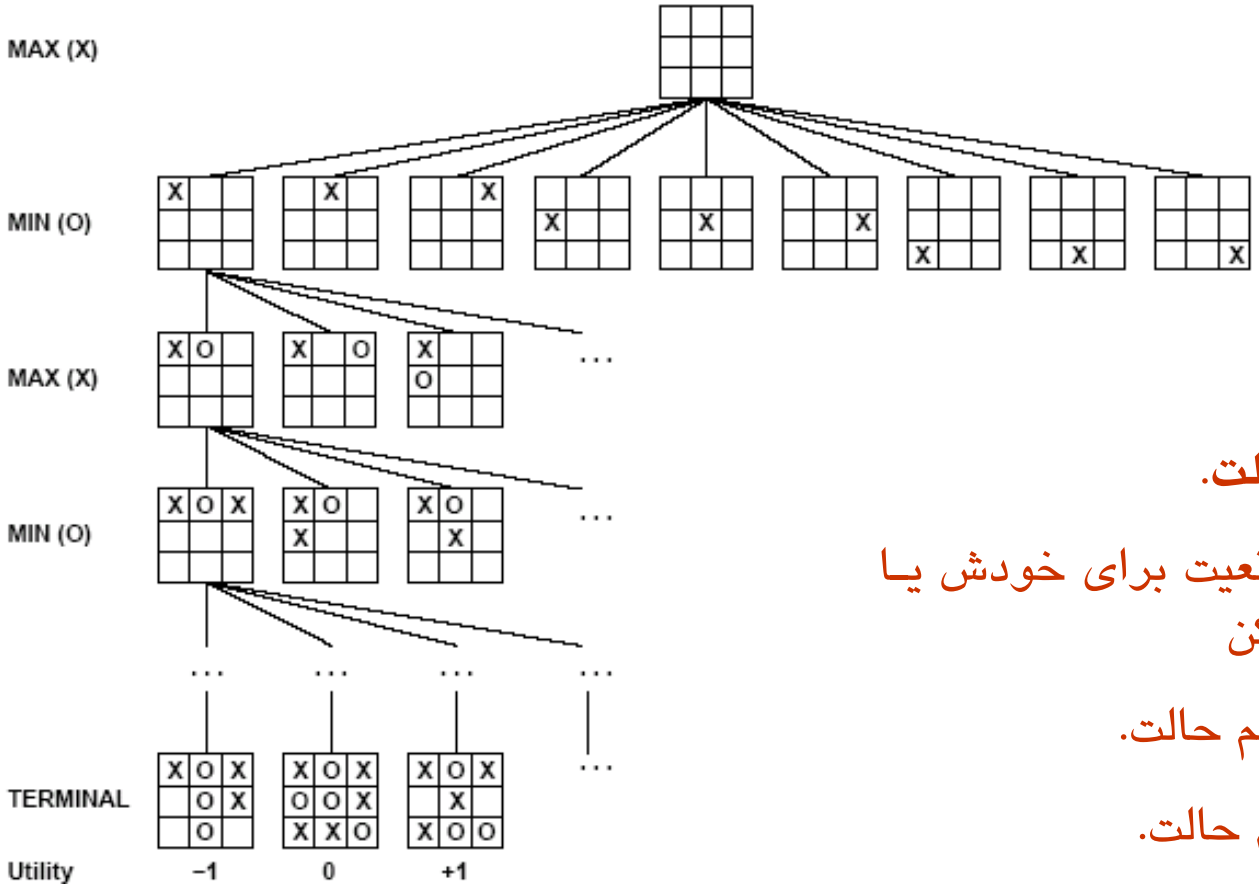
- اول Max حرکت می‌کند و سپس به نوبت بازی می‌کنند تا بازی تمام شود.
- در پایان بازی، برنده تشویق و بازنده جریمه می‌شود.

بازی به عنوان یک جستجو

- **حالت اولیه:** موقعیت اولیه بازی و شناسه های قابل حرکت و شروع کننده.
- **تابع جانشین:** لیستی از (حالت,حرکت) که معرف یک حرکت معتبر است
- **آزمون هدف:** پایان بازی چه موقع است؟(حالتهای پایانی)
- **تابع سودمندی:** برای هر حالت پایانه یک مقدار عددی را ارائه می‌کند. مثلا برنده (+۱) و بازنده (−۱).

حالت اولیه و حرکات معتبر برای هر بازیکن، **درخت بازی** را برای آن بازی ایجاد می‌کند.

یک نمونه بازی

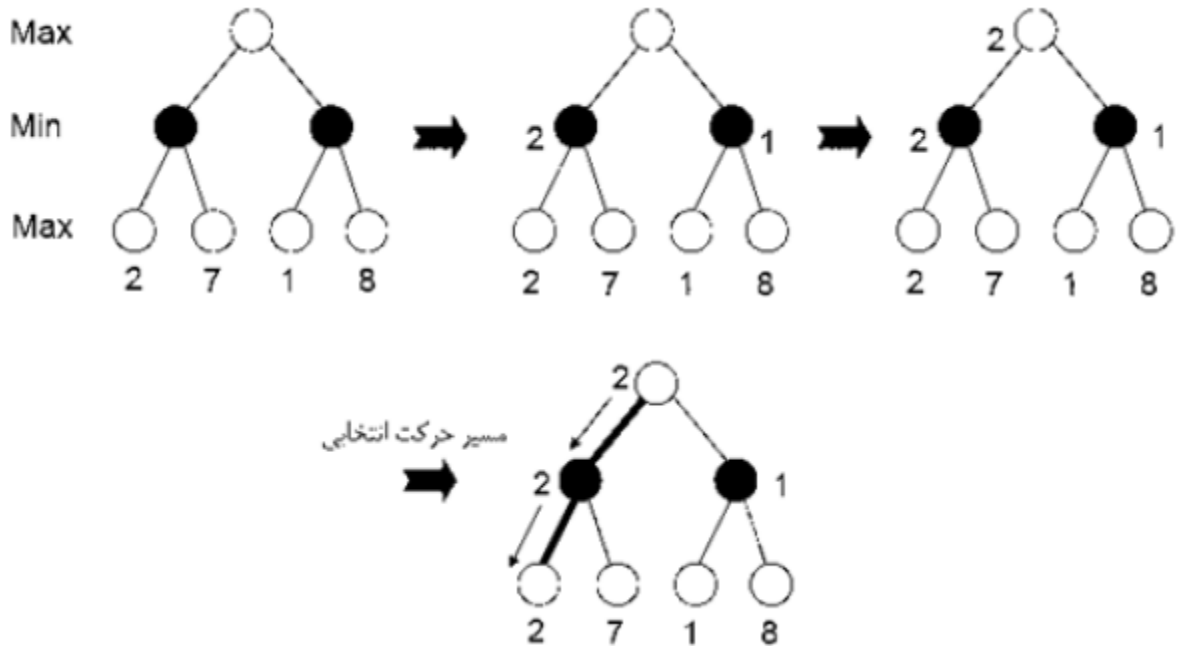


الگوریتم؛

- **بازیکن**: انتخاب **بهترین حالت**.
- **حریف**: انتخاب بهترین موقعیت برای خودش یـا **بدترین وضعیت** برای بازیکن

بازیکن: ماکزیمم حالت.

حریف: مینیمم حالت.

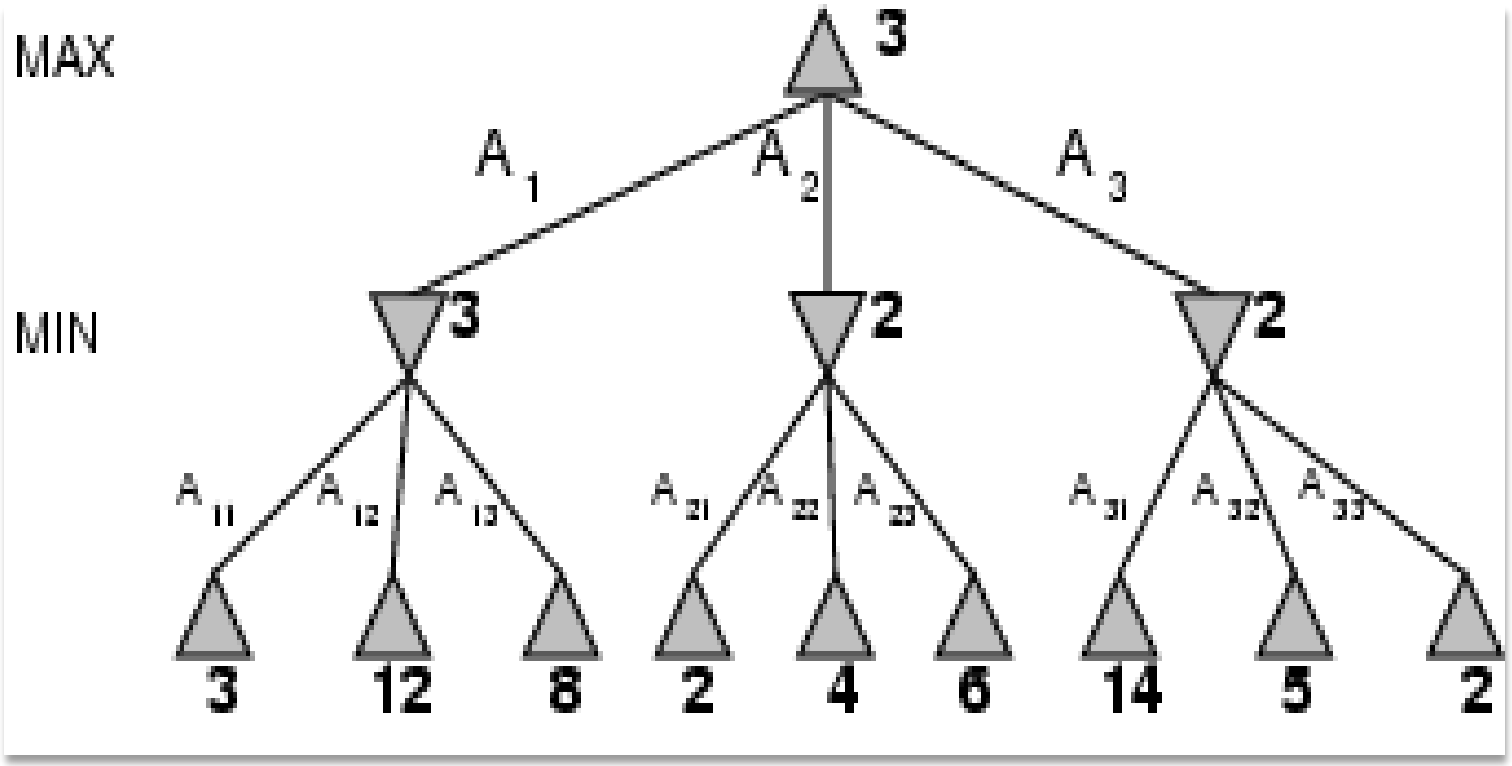

Max
Min
Max
2 7 1 8
2
2 1
2 7 1 8
2
2 1
2 7 1 8
مسیر حرکت انتخابی
2
2 1
2 7 1 8

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

الگوریتم minimax



الگوریتم minimax

کامل بودن: بله (اگر درخت محدود باشد)

بهینگی: بله

پیچیدگی زمانی: $O(b^m)$

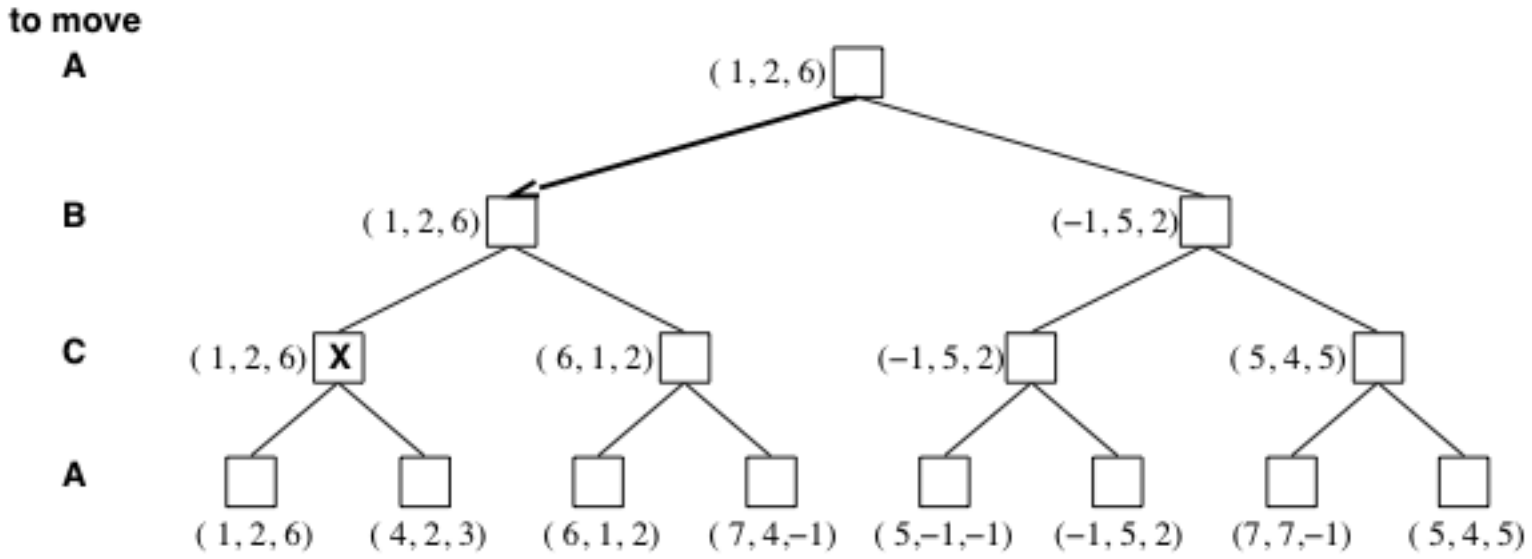
پیچیدگی فضا: $O(bm)$

بازیهای چند نفره

تخصیص یک **بردار** به هر گره، به جای یک مقدار.

بازیهای چند نفره معولاً شامل **اتحاد** رسمی یا غیر رسمی بین بازیکنان است.

- اتحاد با پیشروی بازی ایجاد و از بین میرود.
- بازیکنان بطور خودکار همکاری میکنند، تا به هدف مطلوب انحصاری برسند.



هرس آلفا- بتا

در الگوریتم MaxMin

تعداد حالتهای بازی که باید بررسی شوند، بر حسب تعداد حرکتها، توانی است.

راه حل: محاسبه تصمیم الگوریتم، بدون دیدن همه گره ها امکانپذیر است.

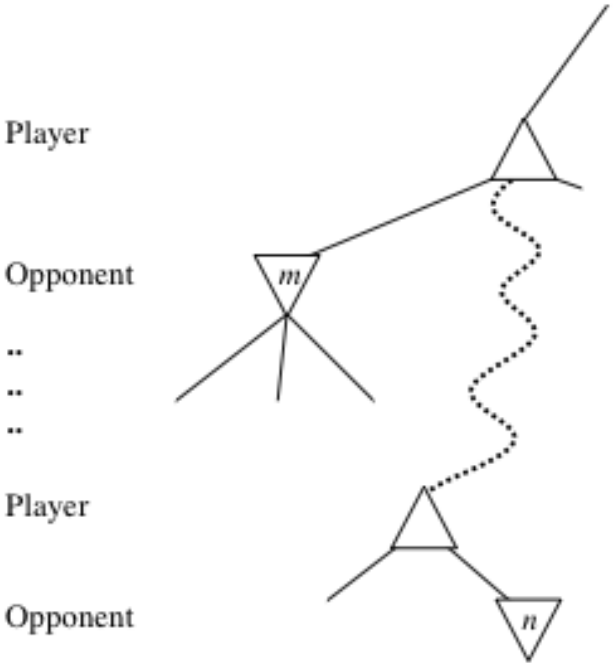
هرس آلفا-بتا

انشعابهایی که در تصمیم نهایی تأثیر ندارند را حذف می‌کند.

- **آلفا:** مقدار بهترین انتخاب در هر نقطه انتخاب در مسیر Max تاکنون.
- **بتا:** مقدار بهترین انتخاب در هر نقطه انتخاب در مسیر Min تاکنون.

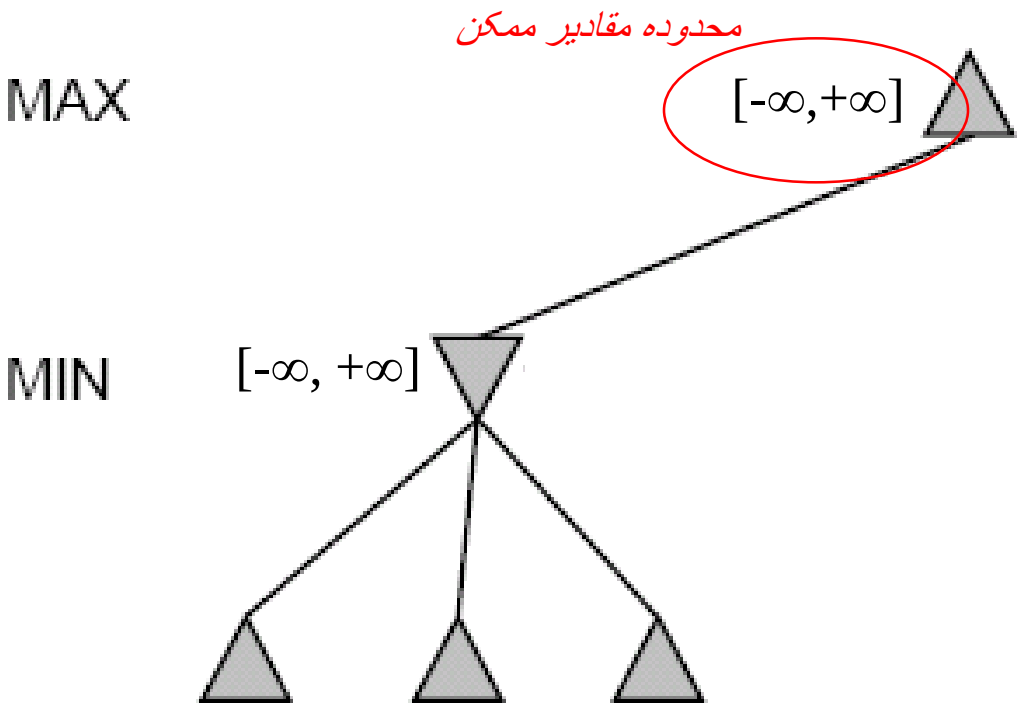
- تعداد گره هایی که باید بررسی شوند به $O(b^{d/2})$ تقلیل میابد

هرس آلفا-بتا

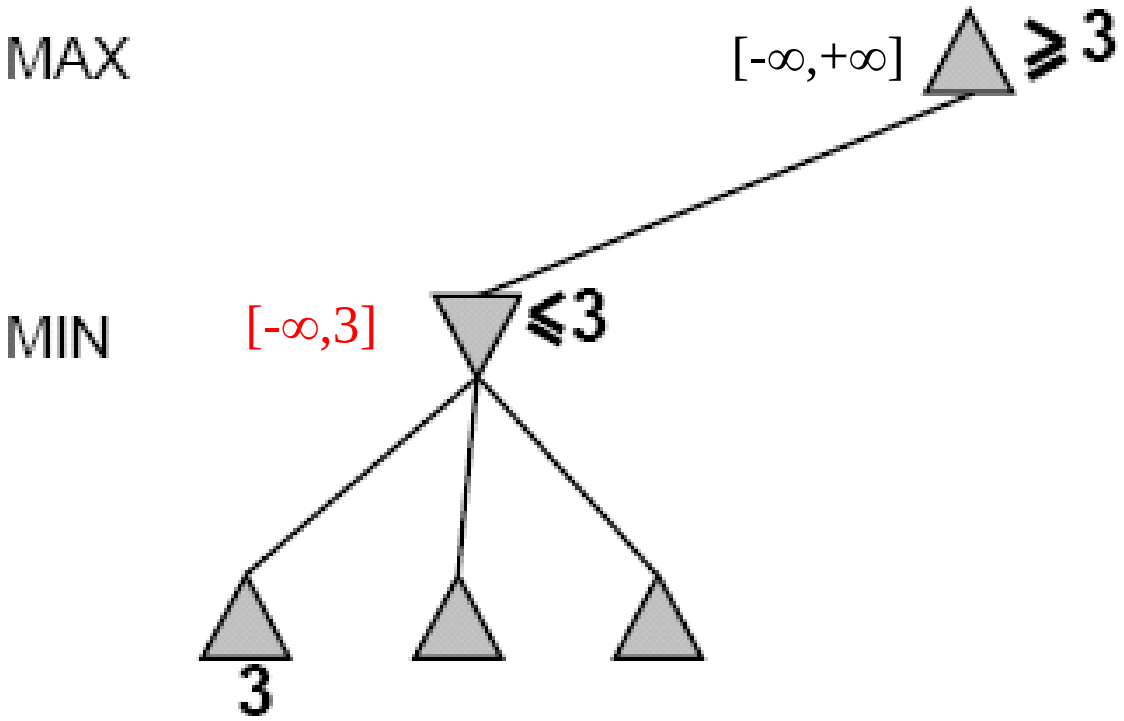


- گره n که هر جای درخت می‌تواند باشد، بررسی می‌شود.
- اگر بازیکن انتخاب بهتری داشته باشد
  - در گره والد n
  - یا هر انتخاب بهتری تا کنون m
- n هیچوقت در بازی واقعی قابل دسترس نخواهد بود. در نتیجه n هرس می‌شود.

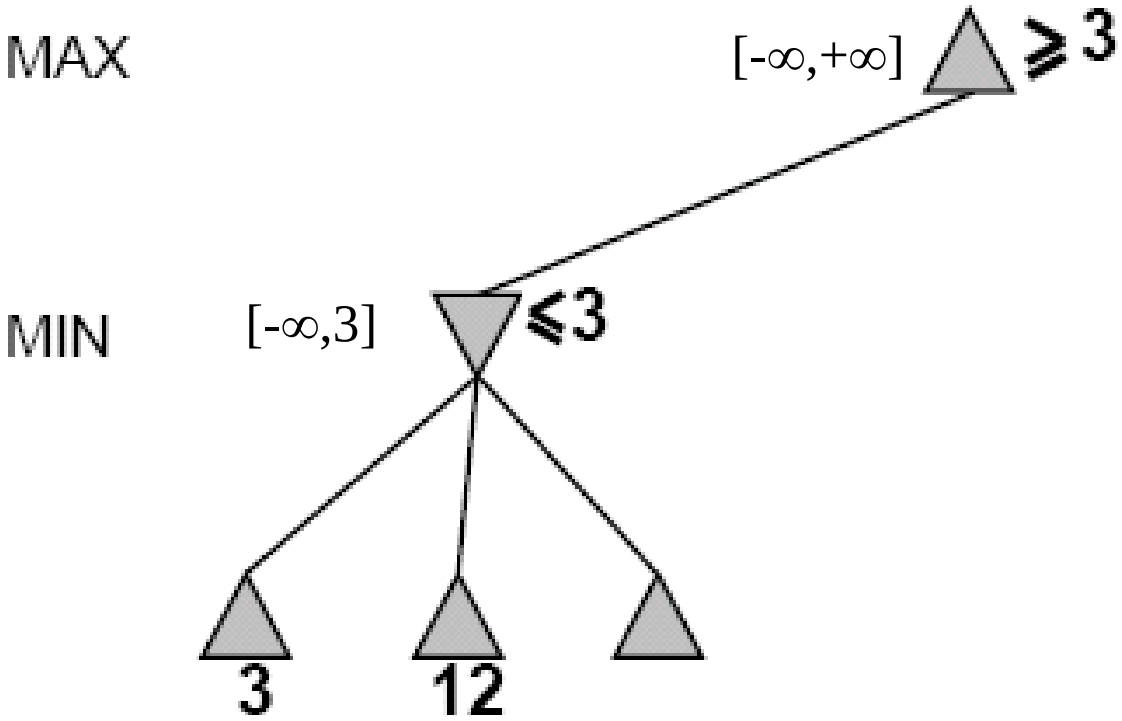
مثال: هرس آلفا–بتا



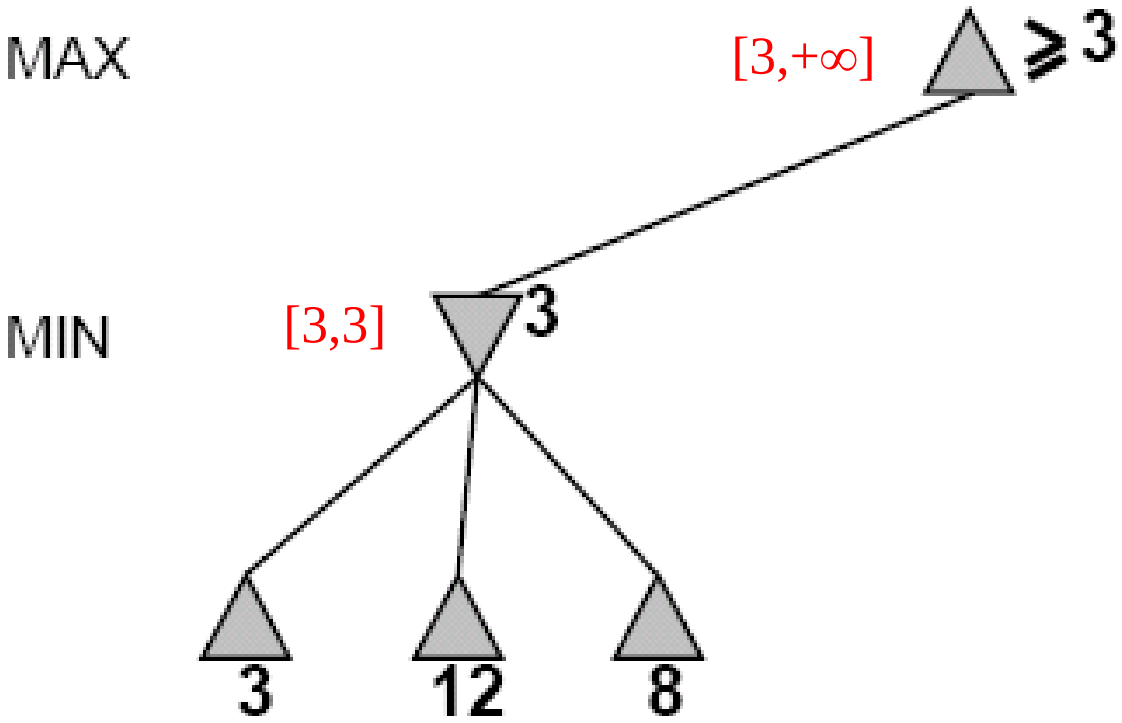
مثال: هرس آلفا–بتا



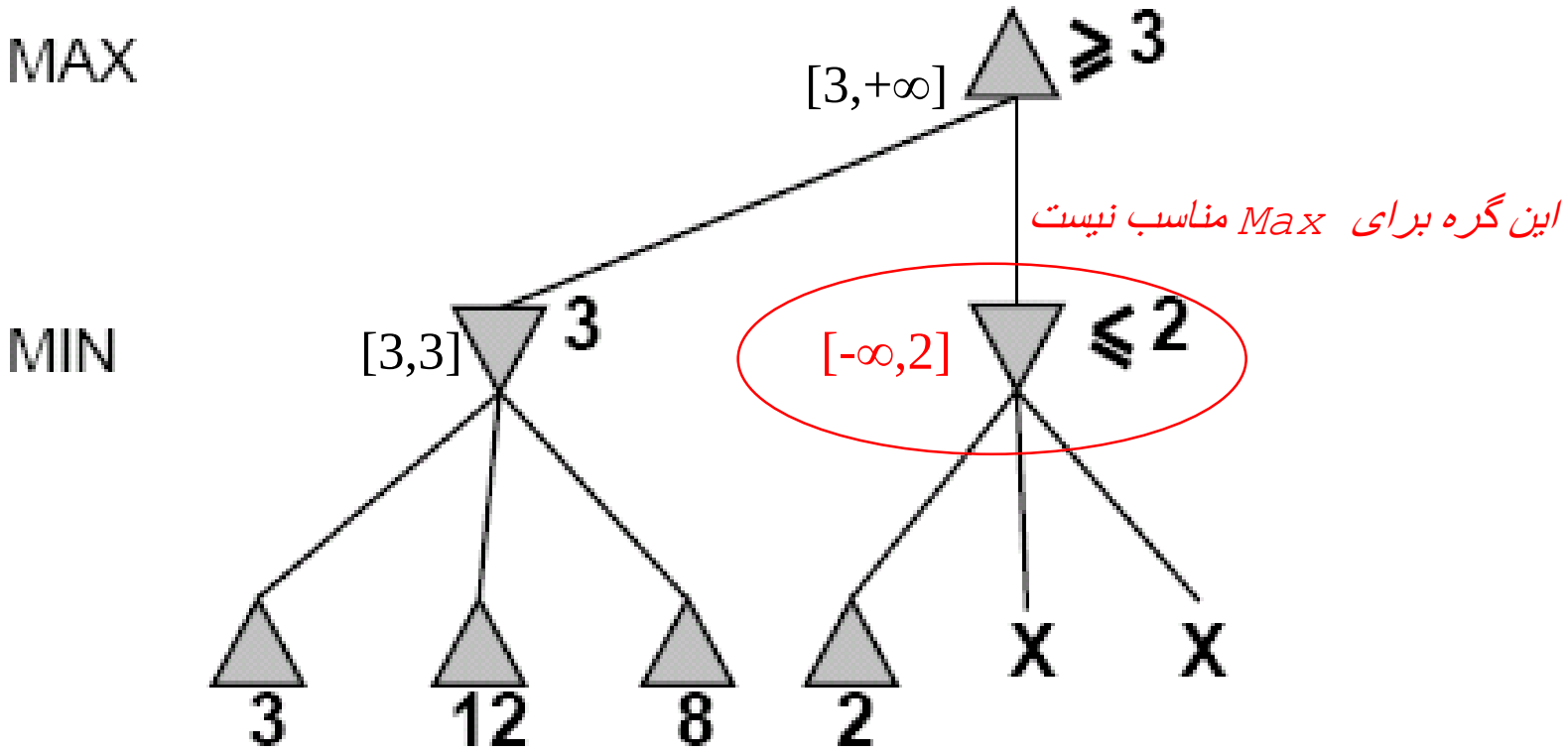
مثال: هرس آلفا–بتا



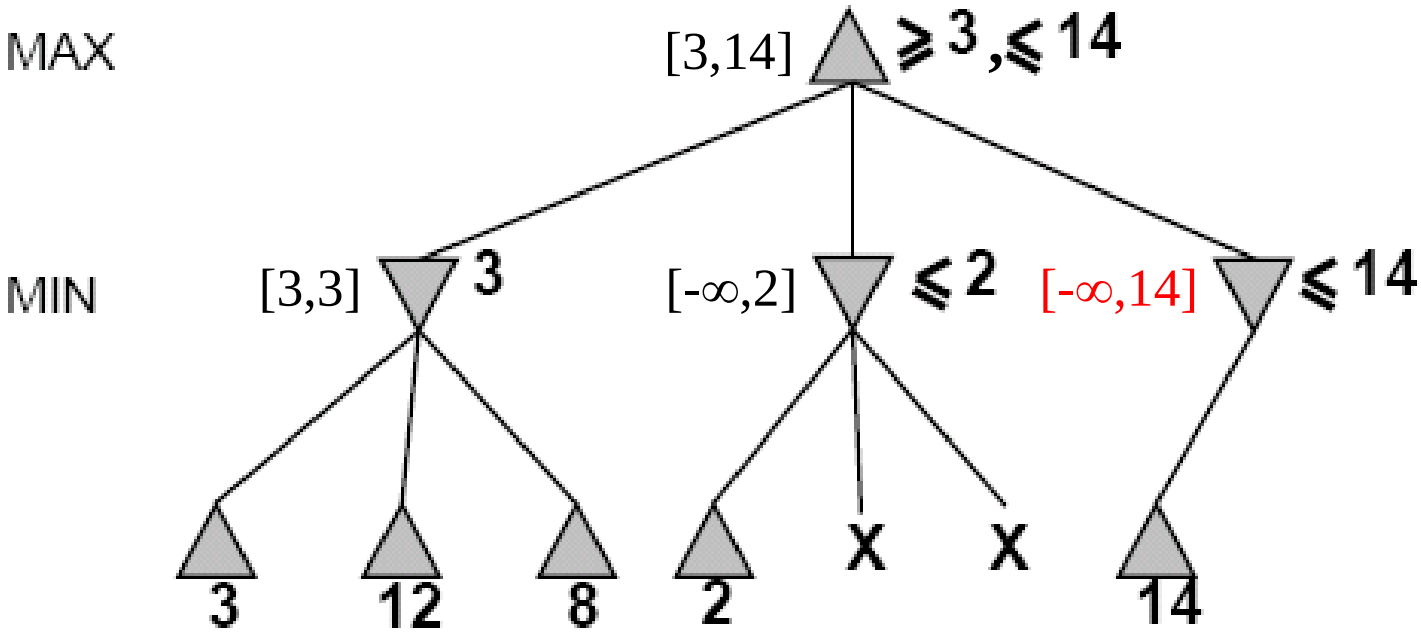
مثال: هرس آلفا–بتا



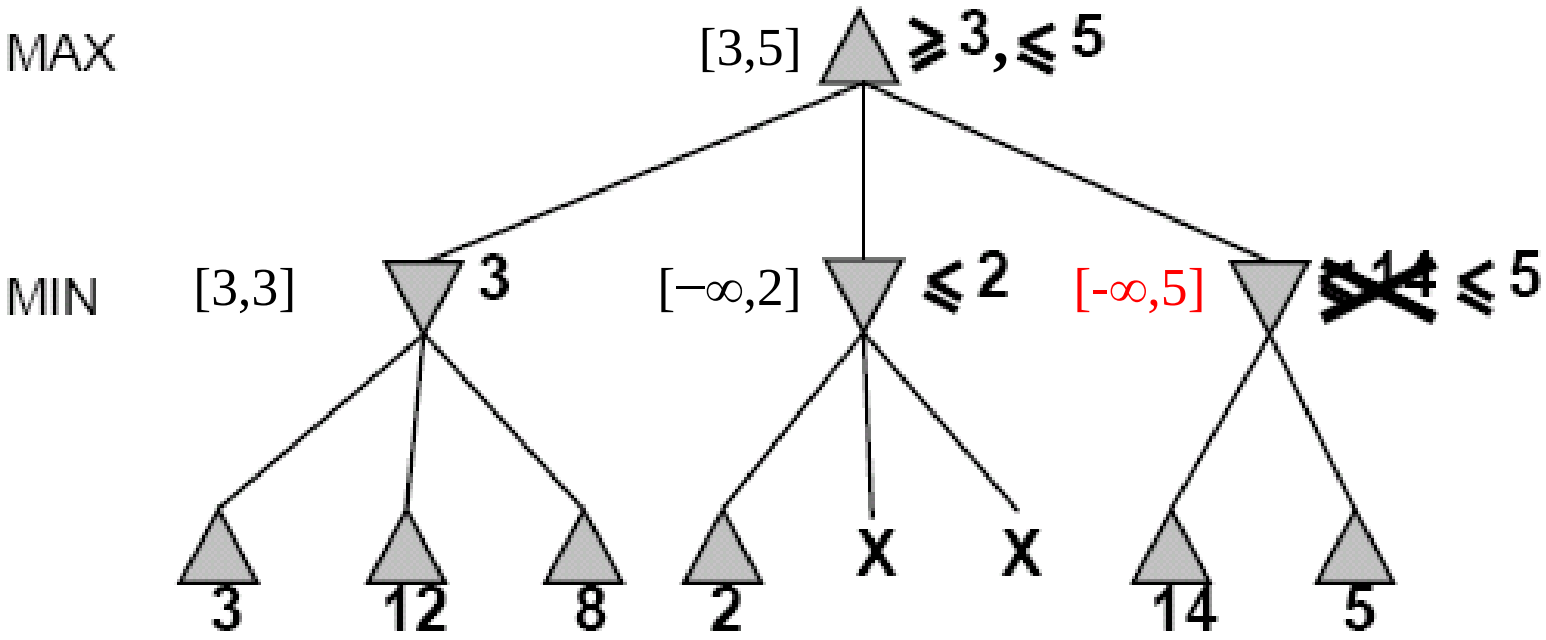
مثال: هرس آلفا–بتا



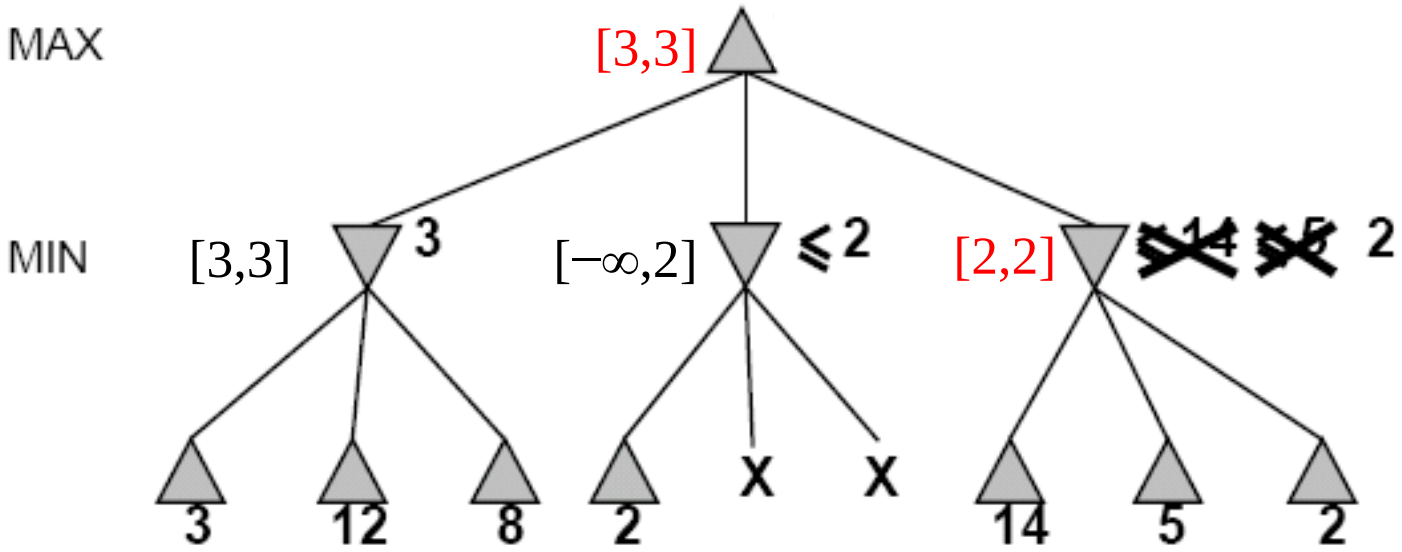
مثال: هرس آلفا–بتا



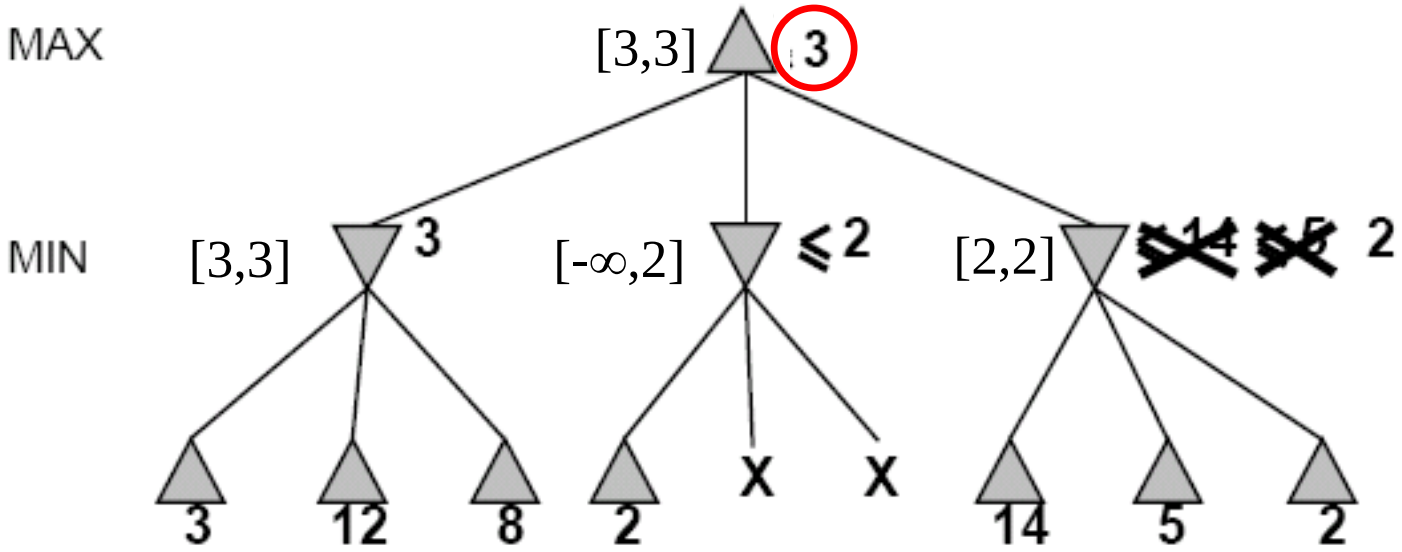
مثال: هرس آلفا–بتا



مثال: هرس آلفا–بتا



مثال: هرس آلفا–بتا



بازیهای قطعی با اطلاعات ناقص

معایب الگوریتم های پیشین

- الگوریتم minimax کل فضای جست و جوی بازی را تولید می‌کند.
- الگوریتم آلفا-بتا با وجود هرس درخت، اما کل مسیر حالتهای پایانه، حداقل برای بخشی از فضای حالت، باید جست و جو شود.
  - این عمق عملی نیست، زیرا حرکات باید در زمانی معقول انجام شود.

- **شانون(۱۹۵۰)**

برای کمتر شدن زمان جست و جو.

جایگزینی تابع سودمندی با یک تابع ارزیاب هیوریستیک (EVAL) (تخمینی از میزان سودمندی موقعیت).

بازیهای قطعی با اطلاعات ناقص

در شانون, minimax و آلفا–بتا به دو روش بطور متناوب عمل می‌کنند.

جایگزینی تابع سودمندی با تابع ارزیابی اکتشافی بنام EVAL

- تضمینی از سودمندی موقعیت ارائه می‌کند.

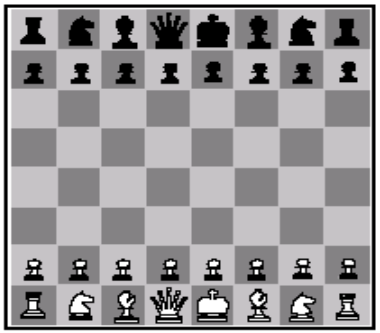
جایگزین تست پایانه با تست توقف.

- تصمیم میگیرد EVAL چه موقع اعمال شود.

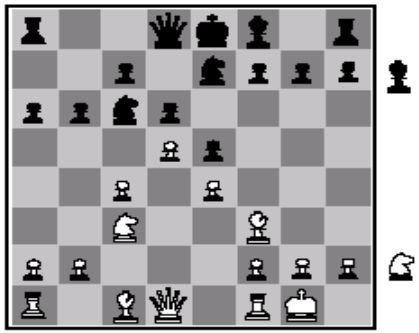
تابع ارزیابی اکتشافی EVAL

- تابع ارزیابی ، ارائه تخمینی از سودمندی مورد انتظار بازی از یک موقعیت خاص.
- توابع اکتشافی، تخمینی از فاصله تا هدف را بر می‌گرداند.
- اغلب توابع ارزیابی، **خواص** گوناگونی از حالتها را محاسبه می‌کنند.
- تابع ارزیابی نمی داند کدام حالت منجر به چه چیزی میشود، اما میتواند مقداری برگرداند که **تناسب حالتها را با هر نتیجه** نشان دهد.

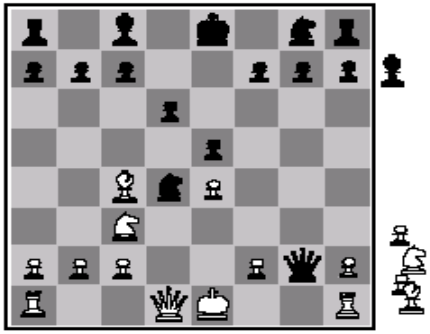
مثال: تابع EVAL



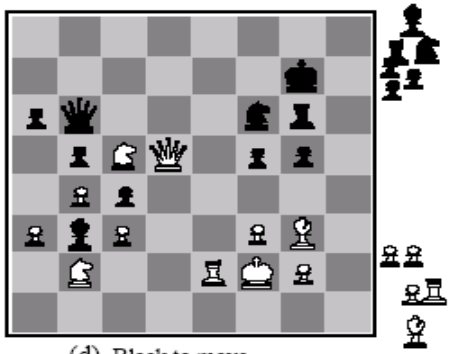
(a) White to move
Fairly even



(b) Black to move
White slightly better



(c) White to move
Black winning



(d) Black to move
White about to lose

اغلب توابع ارزیابی , مقدار عددی جداگانه ای برای هر خاصیت محاسبه، سپس آنها را ترکیب می‌کنند تا مقدار کل بدست آید.

مثال در تابع بازی شطرنج:

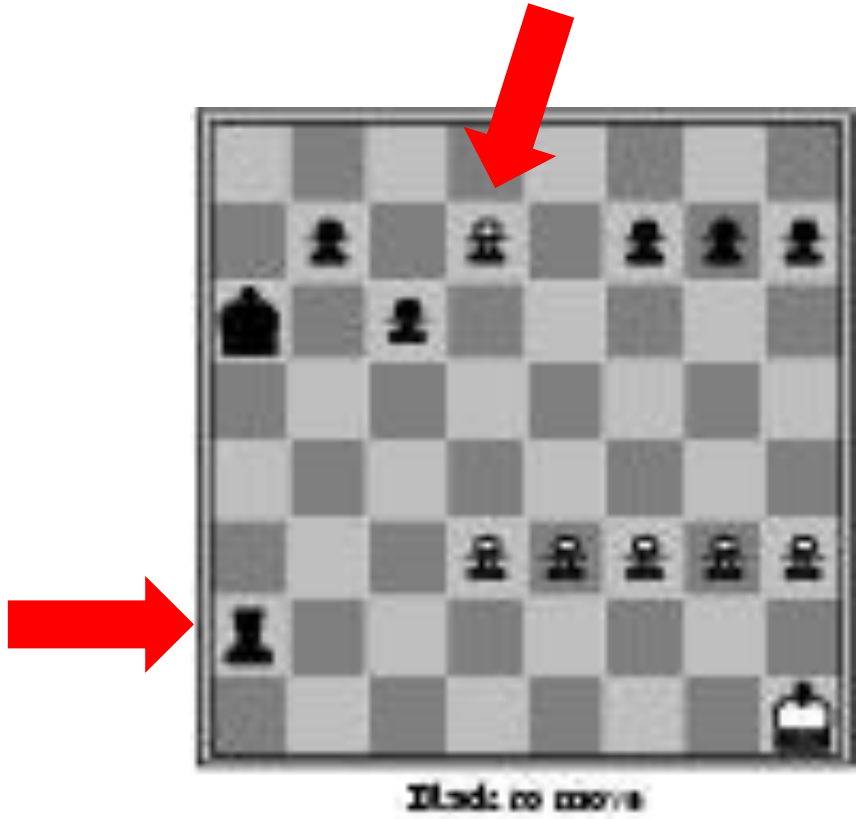
w_i تعداد هر نوع قطعه در صفحه

f_i مقادیر آن قطعات

(۱ برای پیاده، ۳ برای اسب یا فیل،۵ برای رخ و ...)

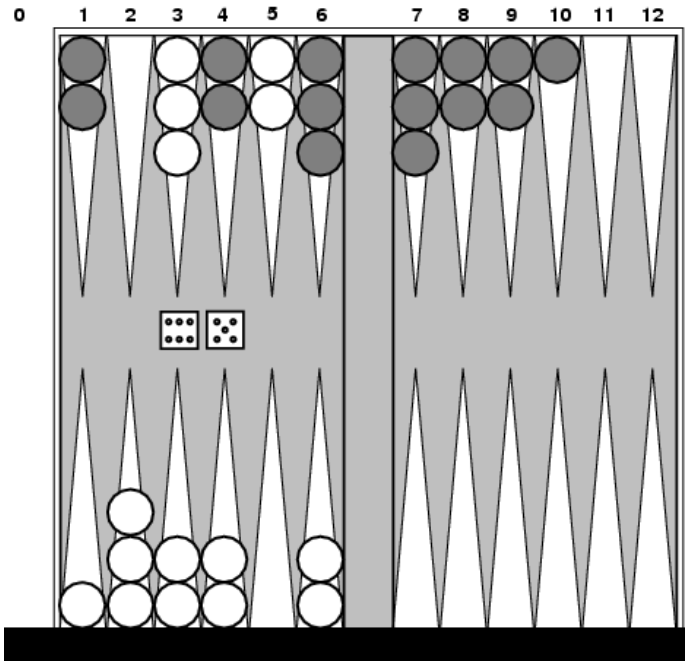
$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

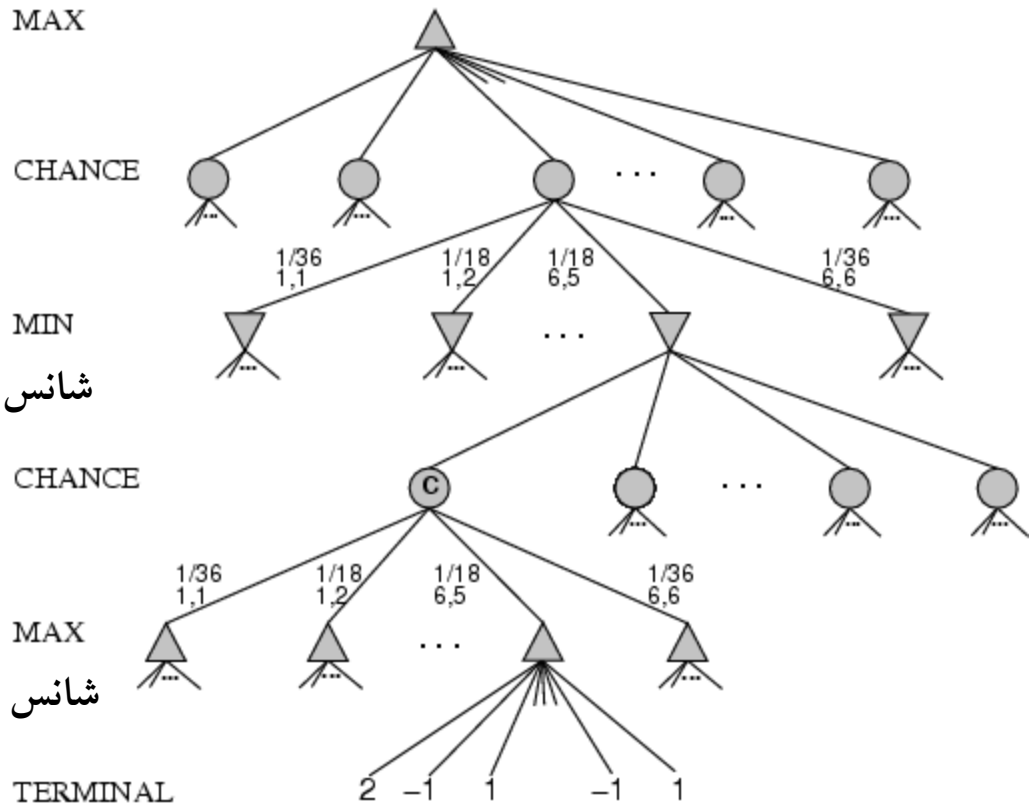
اثر افق



مثال: شکل مقابل؛
سیاه در اصل جلوست، اما اگر سفید پیاده اش را از سطر هفتم به هشتم ببرد، پیاده به وزیر تبدیل میشود و موقعیت برد برای سفید بوجود می آید

بازیهایی که حاوی عنصر شانس هستند





پایانه