به نام خدا

هوش مصنوعی

فصل ششم

جستوجوی خصمانه

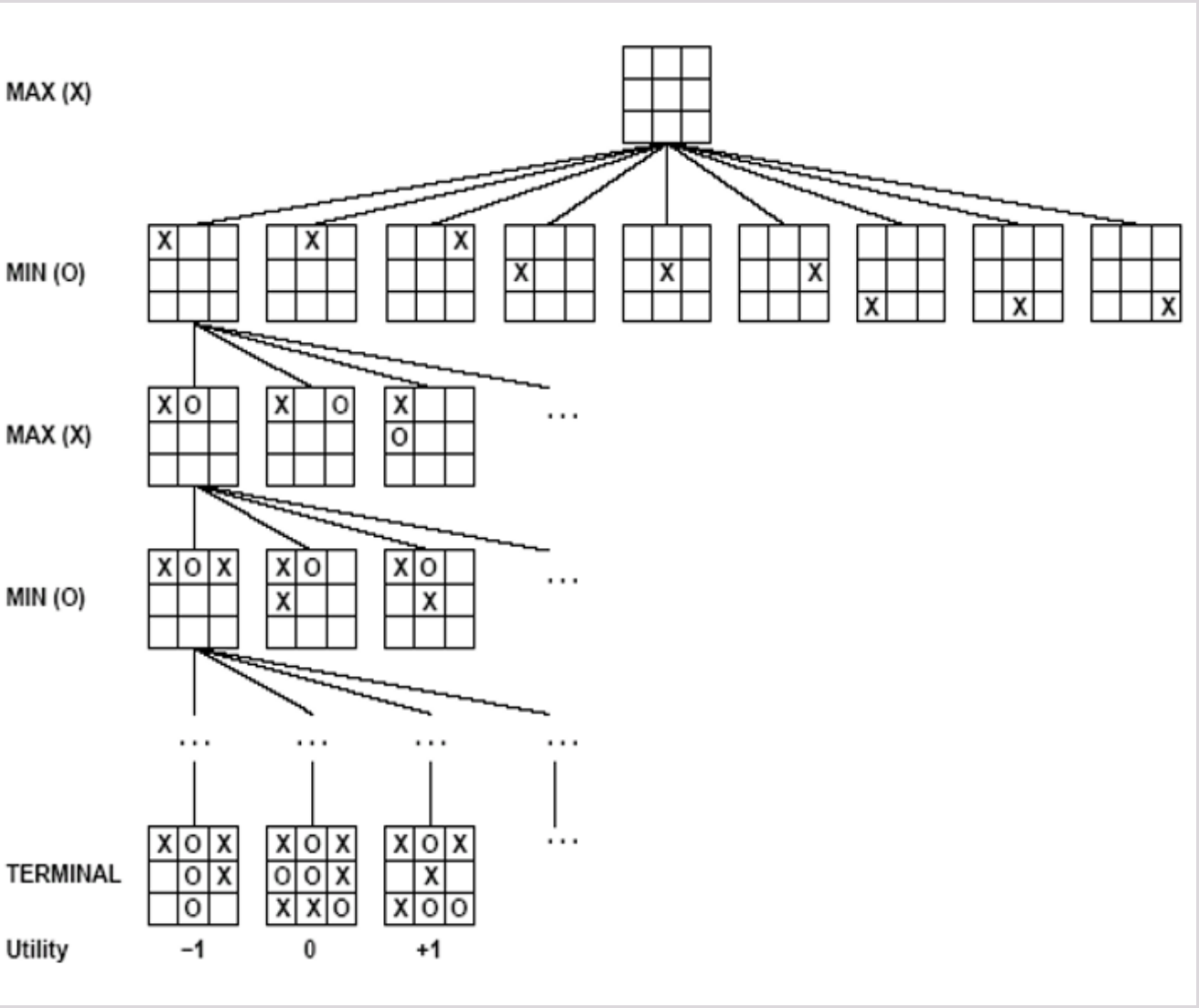
بازی ها

- محیط های رقابتی، که در آنها اهداف عامل ها با یکدیگر در تعارض هستند، منجر به پیدایش مسئله های خصمانه می شود که به عنوان بازی ها شناخته شده اند .
- هر محیط چند عاملی را بهعنوان یک بازی درنظر میگیرد، به شرطی که اثر هر عامل بر روی عاملهای دیگر، ارزشمند باشد
- . این قضیه مستقل از رقابتی یا همکار بودن عاملها است
- در این فصل محیط به معنای محیطهای قطعی و کاملاً قابل مشاهده است که در آن، دو عامل قرار دارند که فعالیتهای آنها باید به نوبت انجام شوند و مقادیر سودمندی در انتهای بازی، مساوی و متضاد یکدیگر هستند.
- تضاد بین توابع سودمندی وضعیت خصمانه را بهوجود میآورد.

بازی های تصمیم گیری بهینه

- بازی های max و min :
  - اول MAXحرکت میکند و سپس به نوبت بازی میکنند تا بازی تمام شود.
  - حالت اولیه، شامل موقعیت صفحه و شناسههایی است که بازیکنان باید حرکت دهند.
  - تابع پسین، لیستی از جفتهای (حالت و حرکت) را برمیگرداند که هر کدام حاوی یک حرکت معتبر و حالت حاصل است.
  - آزمون پایانه، تعیین میکند بازی کی تمام میشود. حالتها در پایان بازی به نام حالتهای پایانه شناخته .
  - تابع سودمندی، (تابع پاداش یا تابع هدف)، برای هر حالت پایانه یک مقدار عددی را ارائه میکند.

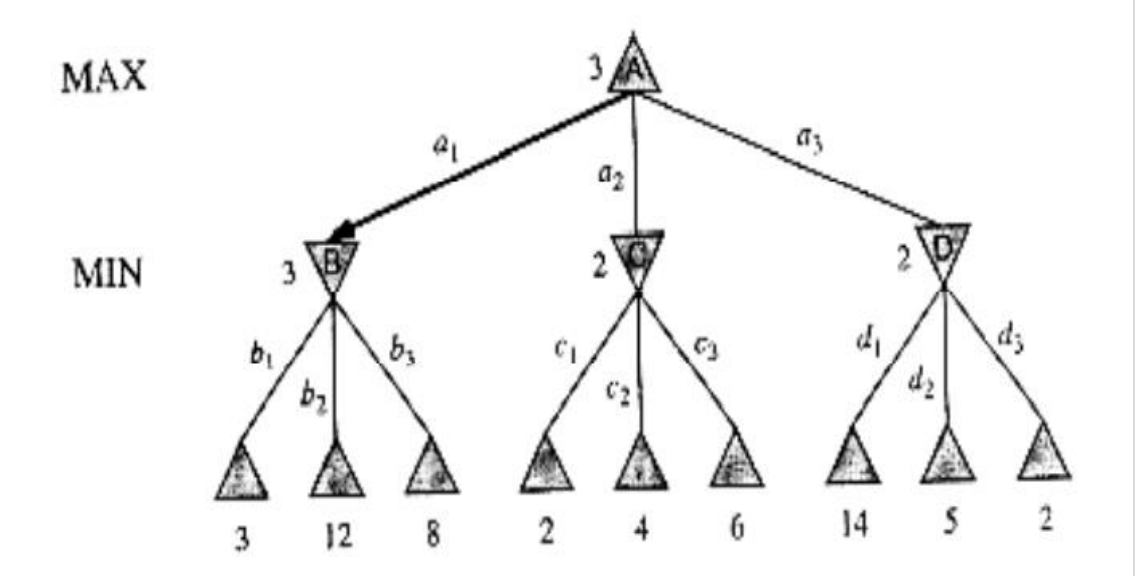
- حالت اولیه و حرکات معتبر برای هر بازیکن، درخت بازی را برای آن بازی ایجاد میکند.



بخشی از درخت بازی را برای دوز بازی نشان میدهد. در حالت اولیه، MAXدارای ۹ حرکت است. MAXعلامت Xو MINعلامت Oرا به نوبت در صف قرار میدهند تا به گره های برگ متناظر با حالت های پایانه برسیم، به طوریکه یک بازیکن سه مهره در هر سطر، ستون یا قطر داشته باشد یا مربع ها پر شوند. عدد موجود در هر گره برگ مقدار سودمندی حالت پایانه را از دیدگاه MAX نشان میدهد. مقادیر بزرگ به نفع MAXو به ضرر MINهستند. کار MAXاستفاده از درخت جستوجو برای تعیین بهترین حرکت است.

بازیکن : انتخاب بهترین حالت

حریف : انتخاب بهترین موقعیت برای خودش و یا بدترین وضعیت برای بازیکن .



- ✓ درخت بازی با دو ply
- ✓ گره های ▲ گره های max هستند . گره های ▽ گره های min هستند .
- ✓ گره های پایانه مقادیر سودمندی را برای max نشان می دهند .
- ✓ بهترین حرکت max در ریشه a1 است . زیرا منجر به جانشینی با بیشترین مقدار minimax می شود .
- ✓ بهترین پاسخ min حرکت b1 می باشد . زیرا منجر به جانشینی با کم ترین مقدار minimax می شود.

بازی های تصمیم گیری بهینه

- مقادیر minimaxهر حالت جانشین را با استفاده از محاسبه بازگشتی بهدست میآورد.
- بازگشتی، به طرف برگهای درخت حرکت میکند و سپس وقتی بازگشتی برمی گردد، مقادیر minimaxذخیره میشود.
- الگوریتم minimax،درخت بازی را به روش عمقی اکتشاف میکند.
- اگر حداکثر عمق درخت mباشد و در هر نقطه bحرکت معتبر وجود داشته باشد، پیچیدگی زمانی $O(b^m)$ است. پیچیدگی فضا برای الگوریتمی که تمام پسین ها را فوراً تولید میکند برابر با( )bmOو برای الگوریتمی که هر بار یک پسین را تولید میکند $O(bm)$ است.

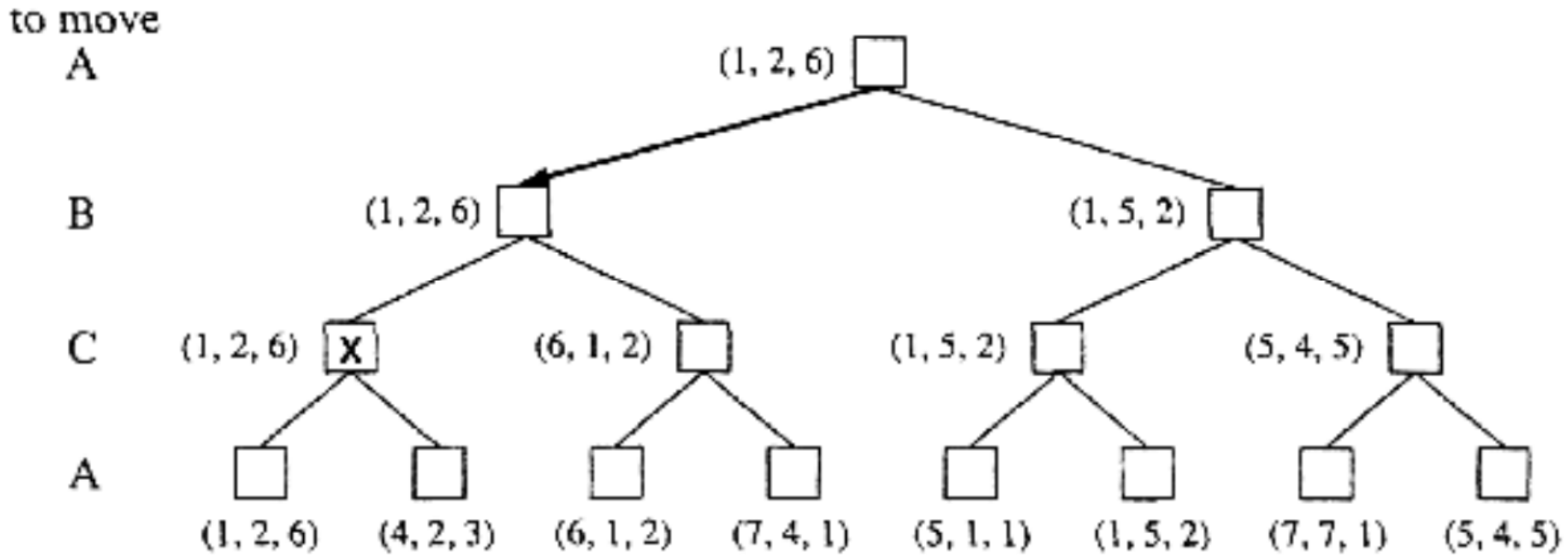
```
function MINIMAX-DECISION(state) returns an action
  inputs: state, current state in game

  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
```

بازی های چندنفره :

- اولاً، لازم است برای هر گره، بهجای یک مقدار یک بردار را درنظر میگیریم.
- برای حالتهای پایانه، این بردار از نظر هر بازیکن، سودمندی هر گره را مشخص میکند
- تفاوت اساسی که بازیهای چند نفره با دو نفره دارد، تشکیل اتحادهای رسمی و غیررسمی بین بازیکنان است.
- ممکن است این اتحادها در بازیهای چند نفره رفتاری بهینه برای بازیکنان به شمار رود. به عنوان مثال اگر A و B ضعیف و C در وضعیتی قویتر از آنها باشد، همکاری و اتحاد بین A و B برای تضعیف C، رفتاری بهینهتر نسبت به تلاشهای جداگانه هر یک از آنها میباشد. در ادامهی بازی این اتحاد میتواند حفظ شود و یا از بین برود.



شکل ۶-۴ سه سطح درخت بازی با سه بازیکن A، B و C. هر گره از دیدگاه هر بازیکن با برچسب‌هایی مشخص شده است. بهترین حرکت در ریشه علامت‌گذاری شده است.

هرس کردن آلفا بتا:

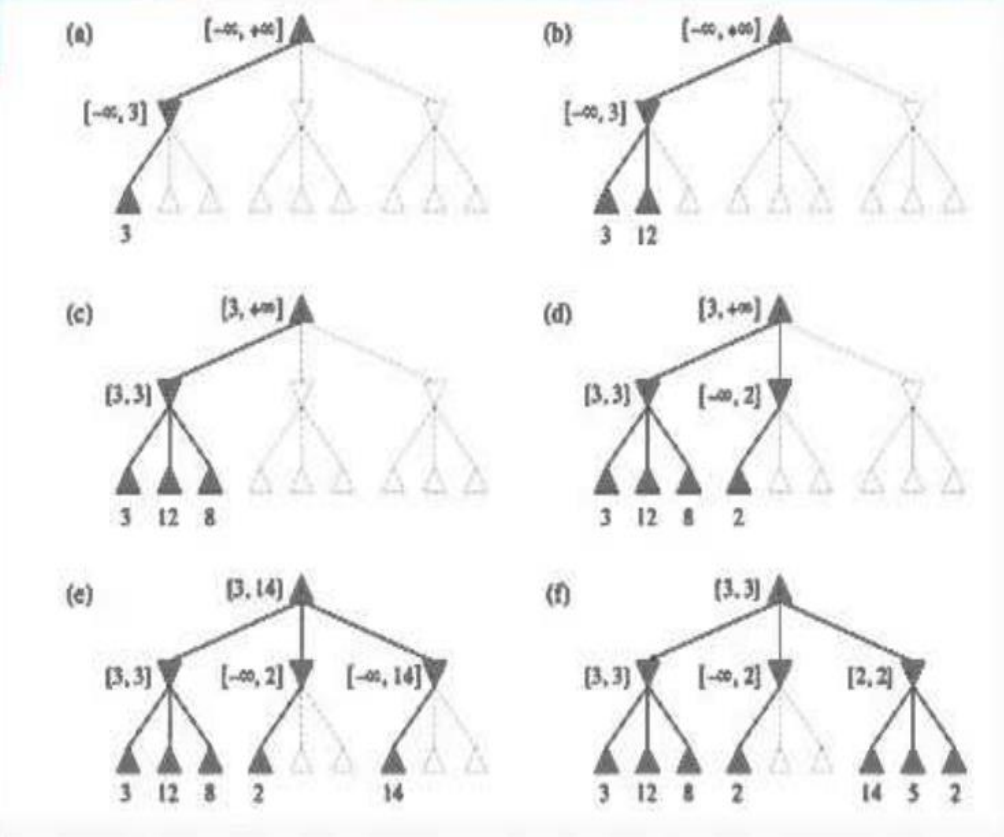
- در الگوریتم minimax تعداد حالتهای بازی برحسب تعداد حرکتها به صورت نمایی رشد میکند. با ایده ی هرس کردن می توان این هزینه ی نمایی را به نصف کاهش داد.
- وقتی این تکنیک به درخت minimax اعمال میشود، همان حرکتی را برم یگرداند که minimax برخواهد گرداند، اما انشعابهایی که در تصمیم نهایی تأثیر ندارند، حذف خواهد شد.

- آلفا = مقدار بهترین انتخابی است که تاکنون در هر نقطه انتخاب در مسیر مربوط به MAX پیدا شده است.
- بتا = مقدار بهترین انتخابی است که تاکنون در هر نقطه انتخاب در مسیر مربوط به MIN پیدا شده است.

❖ . (الف) اولین برگ زیر Bدارای مقدار ۳ است. لذا مقدار Bکه یک گره MINاست، حداکثر ۳ است. (ب) برگ دوم زیر Bدارای مقدار ۱۲ است. MINاین حرکت را انجام نمی دهد. لذا مقدار Bهنوز حداکثر ۳ است. (پ) مقدار برگ سوم زیر Bبرابر با ۸ است. تمام جانشین های Bرا دیدیم و لذا مقدار Bدقیقاً ۳ است. اکنون، میتوانیم نتیجه بگیریم که مقدار ریشه حداقل ۳ است، زیرا MAXدارای انتخابی به ارزش ۳ در ریشه است.

❖ (ت) اولین برگ زیر Cدارای مقدار ۲ است. لذا، Cکه یک گره MINاست، حداکثر دارای مقدار ۲ است. می دانیم که ارزش Bبرابر با ۳ است. لذا MAXهرگز Cرا انتخاب نخواهد کرد. لذا، هیچ نقطه دیگری از Cانتخاب نمیشود. این نمونهای از هرس کردن آلفا ـ بتا است.

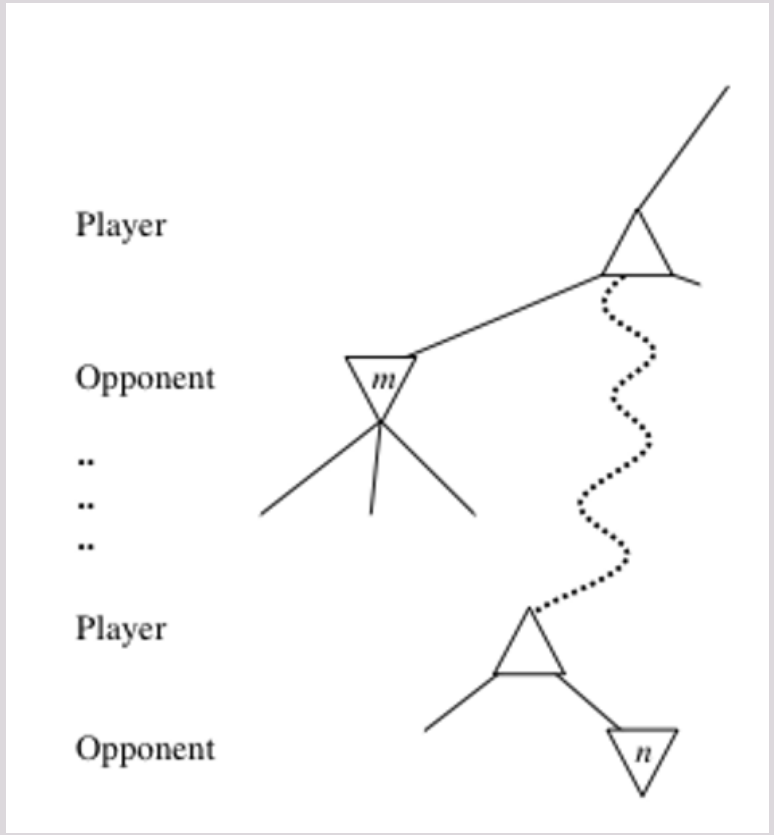
❖ (ث) اولین گره زیر Dدارای مقدار ۱۴ است. لذا ارزش Dحداکثر ۱۴ است. این، هنوز بیشتر از بهترین انتخاب (MAX 3یعنی) است. لذا باید جانشین های Dرا بررسی کنیم. همچنین توجه کنید که قیدهایی بر روی تمام جانشینهای ریشه داریم، لذا مقدار ریشه نیز حداکثر ۱۴ است. (ج) ارزش جانشین دوم Dبرابر با ۵ است. باز هم باید اکتشاف را ادامه دهیم. ارزش جانشین سوم ۲ است و درنتیجه ارزش Dدقیقاً ۲ است. تصمیم MAXدر ریشه، انتقال به Bاست که مقدار ۳ را بهدست میدهد.



$$\begin{aligned} \text{MINIMAX-VALUE}_{(\text{root})} &= \max(\min(3,12,8),\min(2,x,y),\min(14,5,2)) \\ &= \max(3,\min(2,x,y),2) \\ &= \max(3,z,2) \quad \text{where } z \le 2 \\ &= 3 \end{aligned}$$

هرس کردن آلفا بتا:

- اصل کلی این است ، گره nرا درجایی از درخت درنظر بگیرید به طوری که بازیکن بتواند به آن گره برود. اگر بازیکن، در گره والد nیا هر نقطه ای در سطح بالاتر، انتخاب بهتری مثل mدارد، آنگاه در بازی واقعی هرگز به nنخواهیم رسید.
- جستوجوی آلفا ـ بتا مقادیر aو bرا با هرس کردن انشعابهای باقیمانده در یک گروه به هنگامسازی میکند. این کار به محض این که مشخص شد مقدار گره فعلی بدتر از مقدار aیا bمربوط به MAXیا MINاست ، انجام میگیرد.
- اثربخشی هرس کردن آلفا ـ بتا به ترتیب بررسی پسینها بستگی دارد.
- نتیجه میگیریم که بهتر است ابتدا پسینهایی بررسی شوند که ممکن است بهترین باشند
- تعداد گره هایی که باید بررسی شوند به $O(b^{d/2})$ تقلیل میابد
- فاکتور انشعاب مؤثر به جای b برابر با جذرb خواهد بود



- 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
- n هیچوقت در بازی واقعی قابل دسترس نخواهد بود
- در نتیجه n هرس میشود

الگوریتم هرس کردن آلفا بتا:

```
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game

  v ← MAX-VALUE(state, −∞, +∞)
  return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s, α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

تصمیمات بی درنگ:

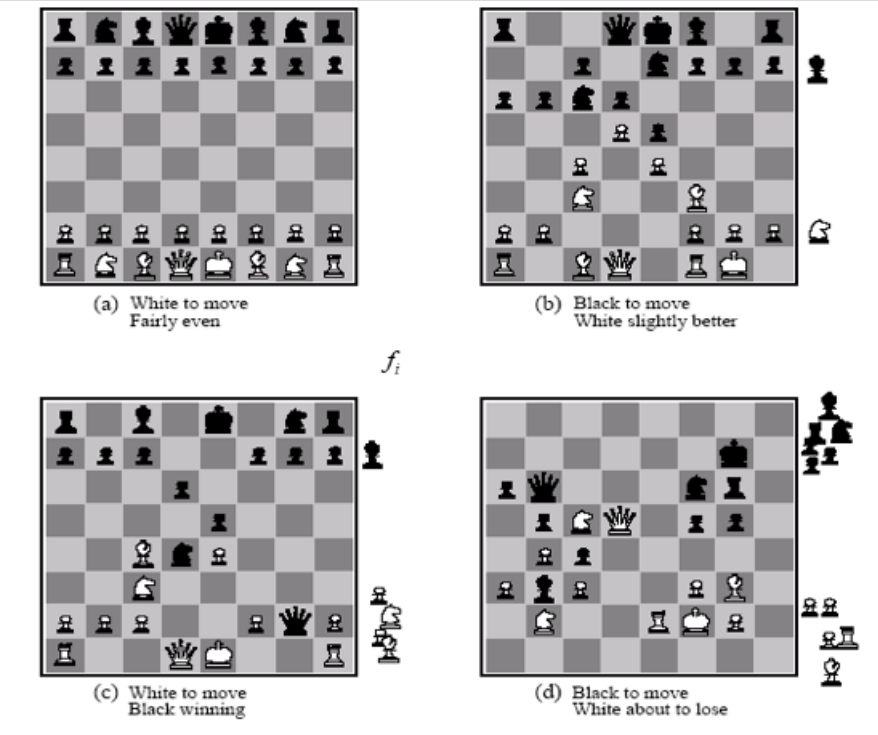
- الگوریتم minimax،کل فضای جستوجوی بازی را تولید میکند، درحالیکه الگوریتم آلفا ـ بتا اجازه میدهد بخش زیادی از آن هرس شود. اما در آلفا ـ بتا کل مسیر حالتهای پایانه، حداقل برای بخشی از فضای حالت، باید جستوجو شود. این عمق، عملی نیست زیرا حرکات باید در یک زمان معقول انجام شوند

- شانون (۱۹۵۰) پیشنهاد کرد :

- به جای اینکه برنامه ها جستوجو را زودتر خاتمه دهند و سپس تابع ارزیابی اکتشافی را به حالت های جستوجو اعمال کنند، بهتر است از گرههای غیرپایانه به گرههای پایانه بپردازند.

- به عبارت دیگر minimaxو آلفا ـ بتا به دو روش، بهطور متناوب عمل کنند: تابع سودمندی با تابع ارزیابی اکتشافی به نام EVALجایگزین شود تا تخمینی از سودمندی موقعیت را ارائه کند و تست پایانه با تست توقف جایگزین شود که تصمی م میگیرد EVALکی اعمال گردد.

تابع ارزیابی اکتشافی EVAL:



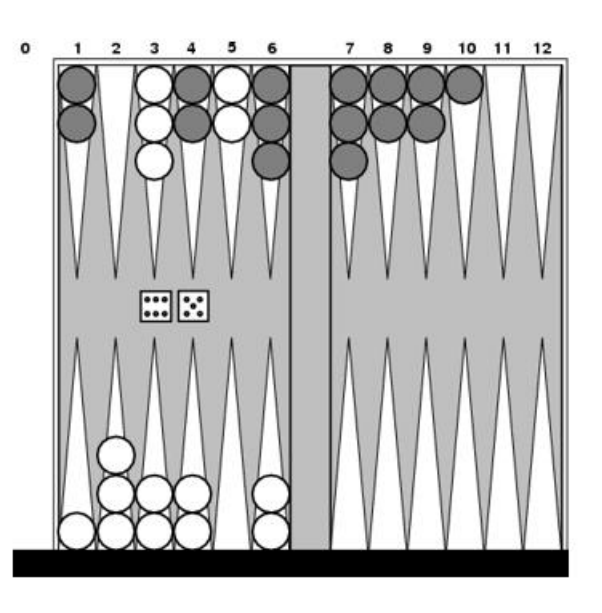
$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

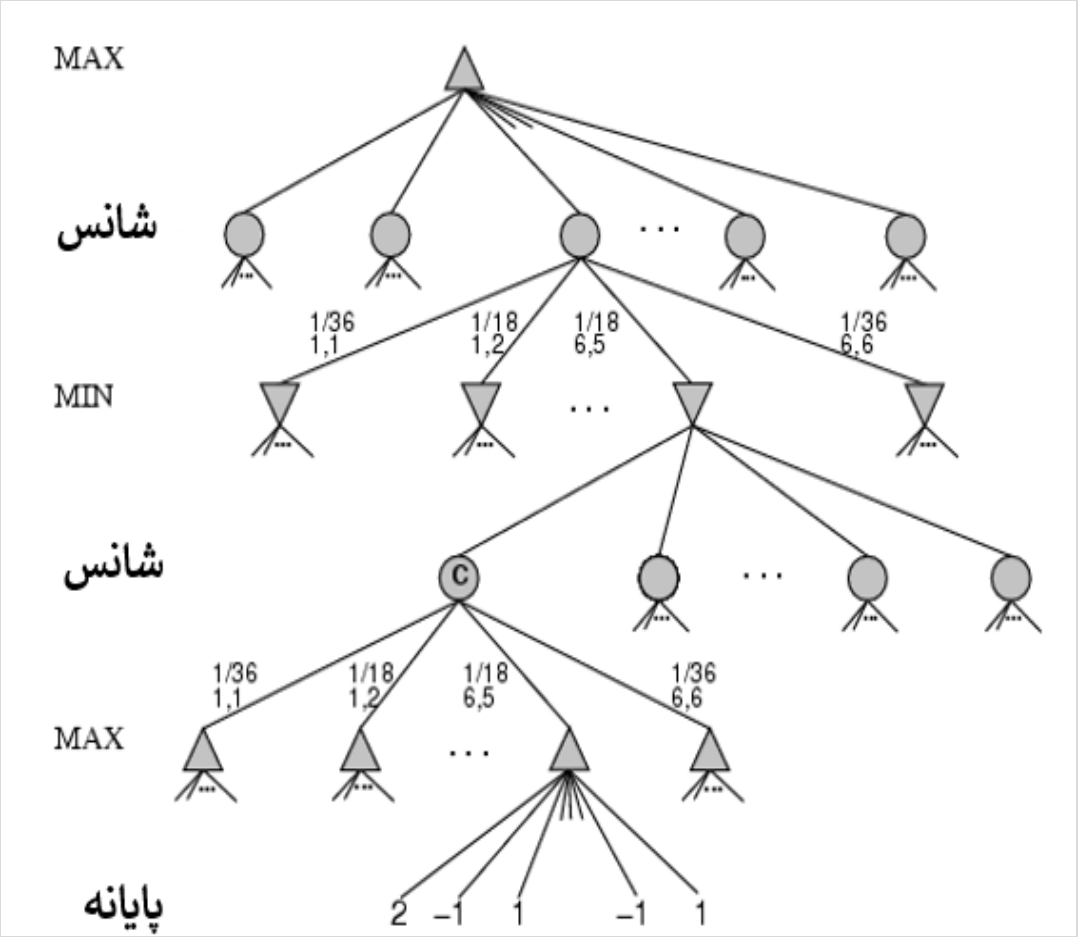
- ✓ اغلب توابع ارزیابی, مقدار عددی جداگانه ای برای هر خاصیت محاسبه، سپس آنها را ترکیب میکنند تا مقدار کل بدست آید
- ✓ مثال در تابع بازی شطرنج :
  - ✓ F(i) تعداد هرنوع قطعه در صفحه
  - ✓ W(i) مقادیر آن قطعات(۱ برای پیاده ، ۳ برای اسب یا فیل ، ۵ برای رخ و ....)

اثر افق:

- برطرف کردن اثر افق دشوارتر است . این اثر وقتی بهوجود میآید که برنامه با حرکتی از رقیب مواجه میشود که منجر به لطمه شدید اجتنابناپذیر میگردد.
- سیاه در اصل جلو است، اما اگر سفید بتواند پیاده اش را از سطر هفتم به هشتم ببرد، پیاده به وزیر تبدیل میشود و موقعیت برد را برای سفید بهوجود میآورد.

بازی هایی که حاوی عنصر شانس هستند:

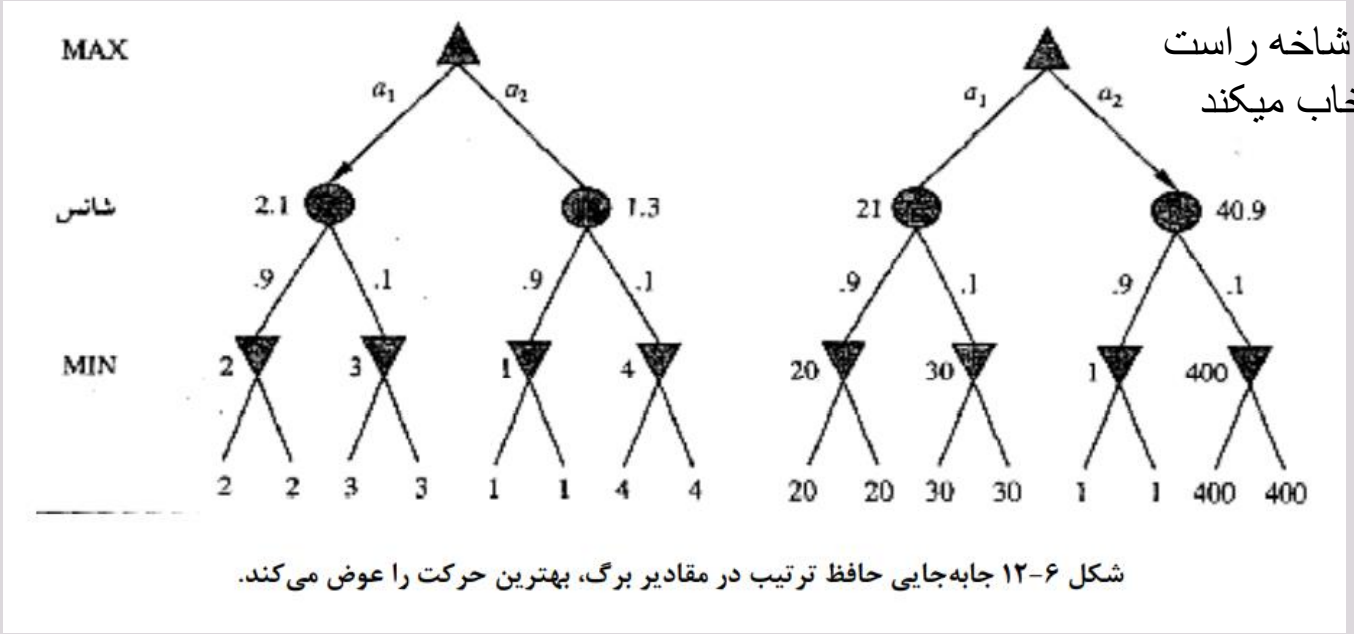




اهمیت تابع ارزیابی:

Max شاخه چپ
را انتخاب میکند

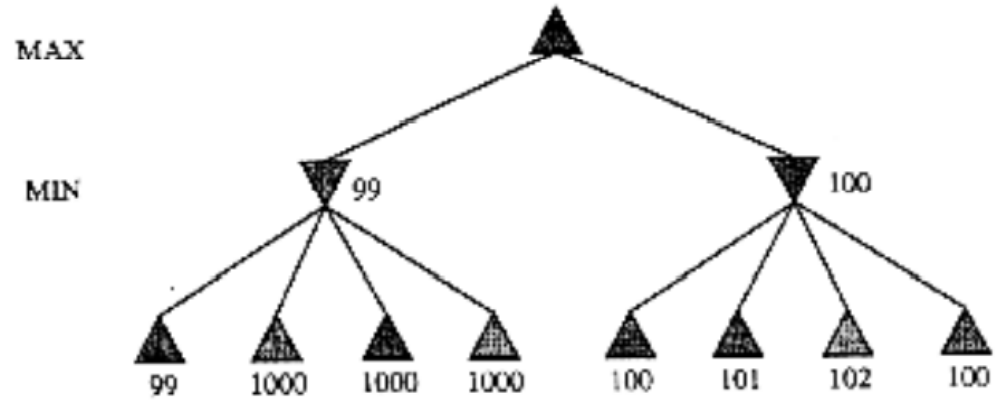
Max شاخه راست
را انتخاب میکند



شکل ۶-۱۲ جابه‌جایی حافظ ترتیب در مقادیر برگ، بهترین حرکت را عوض می‌کند.

برای اجتناب از این حساسیت، تابع ارزیابی باید یک جابجایی خطی مثبت از احتمال برنده شدن از یک موقعیت باشد

جستوجو خصمانه:



شکل ۶-۱۳ درخت بازی با دو ply که ممکن است minimax برای آن مناسب نباشد.

- در درخت فوق minimax انشعاب راست را پیشنهاد می کند.
- این انتخاب بر این فرض استوار است که تمام گره هایی با مقادیر ۱۰۰، ۱۰۱، ۱۰۲، ۱۰۰ واقعا بهتر از گره ای با برچسب ۹۹ هستند اما گره ای با برچسب ۹۹، همزادهایی با برچسب ۱۰۰۰ داردو ممکن است نتیجه بگیریم که مقدار آن بیشتر است.
- یک روش حل این مشکل این است که ارزیابی ، توزیع احتمال را بر روی مقادیر ممکن برگرداند.

پایان فصل ششم