جستجوی رقابتی

فصل ششم
سید ناصر رضوی
Email: razavi@Comp.iust.ac.ir
۱۳۸۴

مقدمه

- تصمیم های بهینه
- هرس آلفا-بتا
- تصمیم های بلادرنگ و غیر کامل

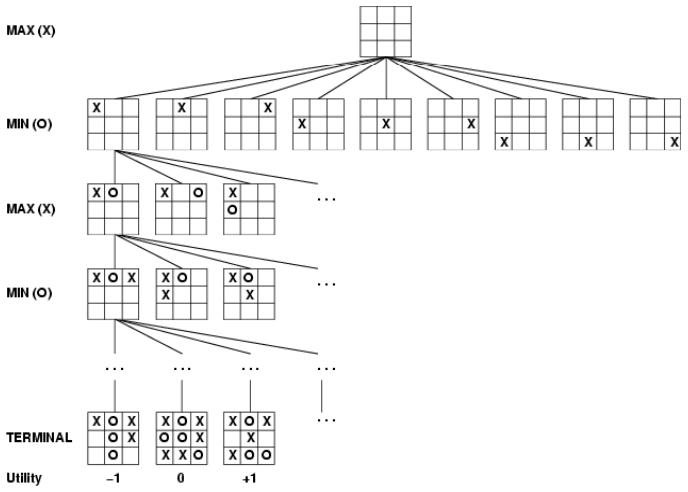
بازی در مقایسه با مسائل جستجو

- رقیب «غیر قابل پیش بینی» ← مشخص کردن یک حرکت برای هر پاسخ ممکن از طرف رقیب
- محدودیت های زمانی ← متاسفانه برای یافتن هدف باید تقریب زد

تصمیم های بهینه در بازی ها

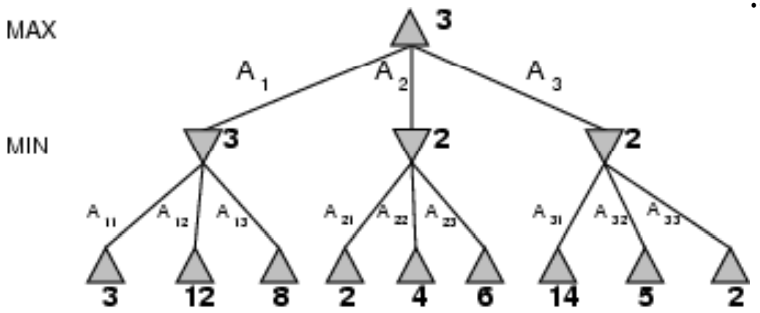
- بازی های دو نفره (MAX و MIN)
- ابتدا MAX بازی می کند، سپس MIN و ...
- بازی به عنوان یک نوع از مسأله جستجو
  - حالت اولیه: شامل موقعیت صفحه و نوبت بازیکن
  - تابع جانشین: لیستی از زوج های (*move, state*) را بر می گرداند
  - تست ترمینال: تعیین کننده پایان بازی (حالت های ترمینال)
  - تابع سودمندی: به هر حالت پایانی یک مقدار عددی می دهد
- درخت بازی: توسط حالت اولیه و حرکت های قانونی برای هر طرف مشخص می شود

درخت بازی (۲-نفره، قطعی)



Minimax

- بازی عالی در بازی های قطعی
- ایده: انتخاب حرکت به موقعیتی با بیشترین مقدار minimax
- = بهترین امتیاز قابل دستیابی در برابر بهترین بازی
- مثال:



استراتژی بهینه

- با داشتن درخت بازی، استراتژی بهینه را می توان با در نظر گرفتن مقدار minimax گره ها تعیین نمود:

$$\text{MINMAX-VALUE}(n) = \begin{cases} \text{UTILITY}(n) & \text{if } n \text{ is a terminal state} \\ \max_{s \in Successors(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MAX node} \\ \min_{s \in Successors(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MIN node} \end{cases}$$

Minimax الگوریتم

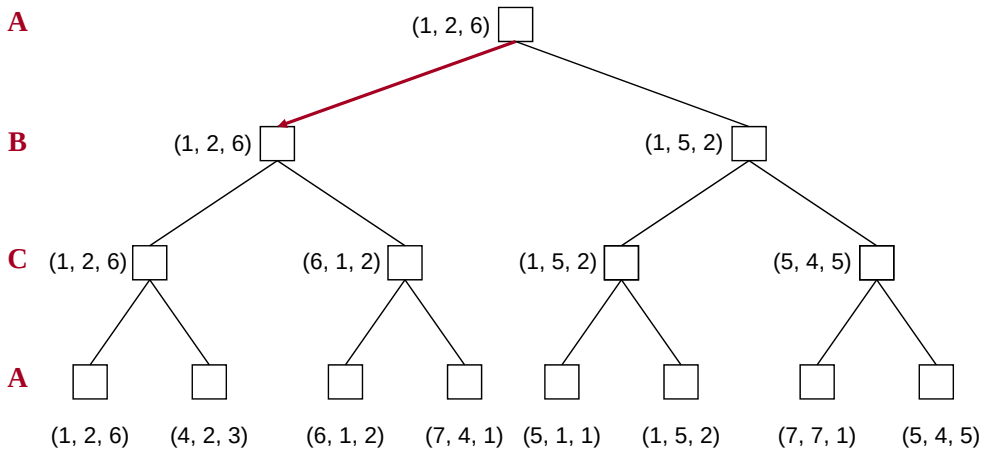
```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

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
```

خصوصیات Minimax

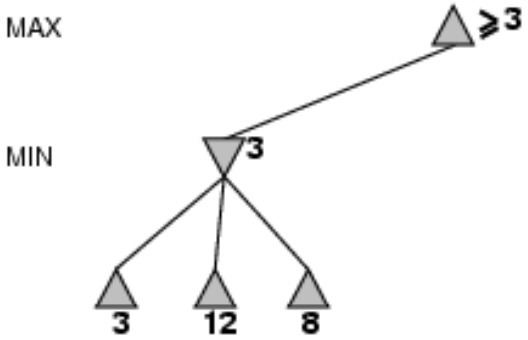
- کامل؟ بله (به شرط محدود بودن درخت)
- بهینه؟ بله (در مقابل رقیبی که بهینه بازی می کند)
- پیچیدگی زمانی؟ $O(b^m)$
- پیچیدگی حافظه؟ $O(bm)$ (کاوش اول-عمق)

- در شطرنج $b \approx 35$ و $m \approx 100$ (در بازی های معقول)
  ← راه حل دقیق کاملا مقرون به صرفه نمی باشد.

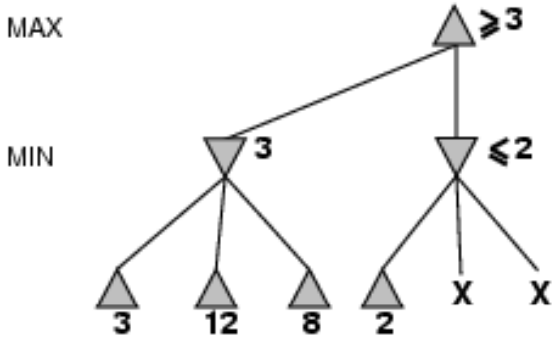
تصمیمات بهینه در بازی های چند نفره



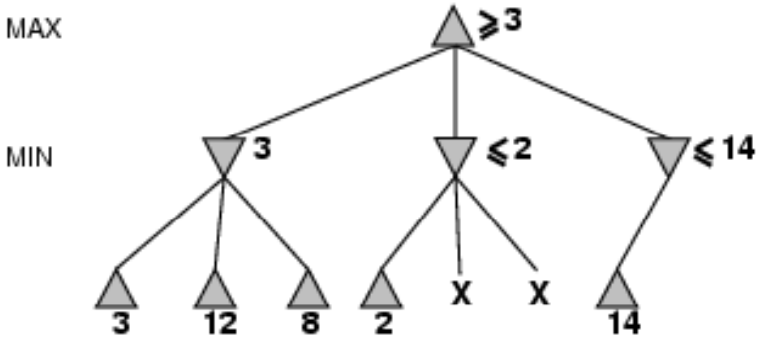
مثال هرس آلفا-بتا



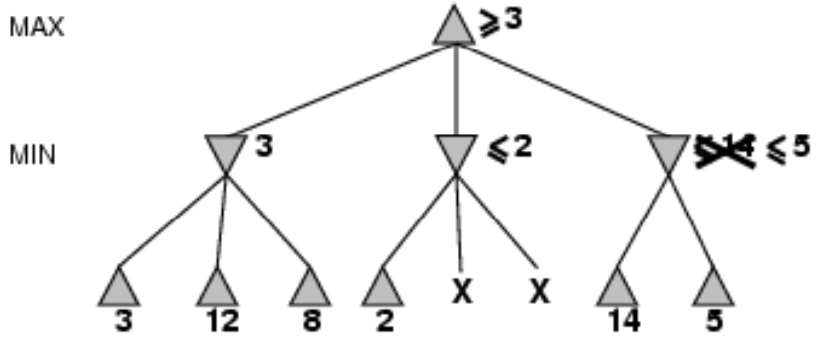
مثال هرس آلفا-بتا



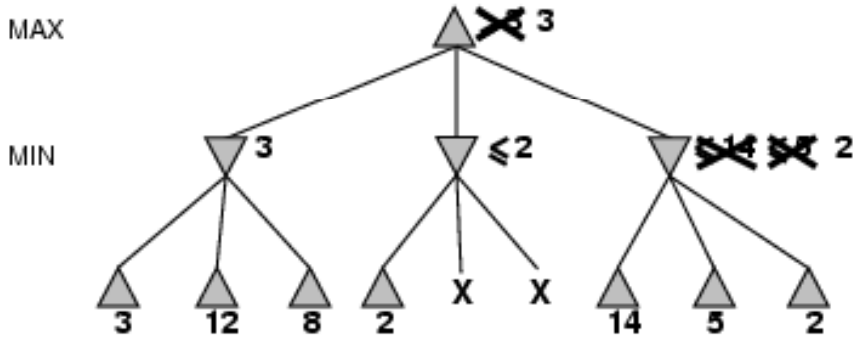
مثال هرس آلفا-بتا



مثال هرس آلفا-بتا



مثال هرس آلفا-بتا



MINIMAX-VALUE (*root*) = max (min (3, 12, 8), min (2, *x*, *y*), min (14, 5, 2))

= max (3, min (2, *x*, *y*), 2)

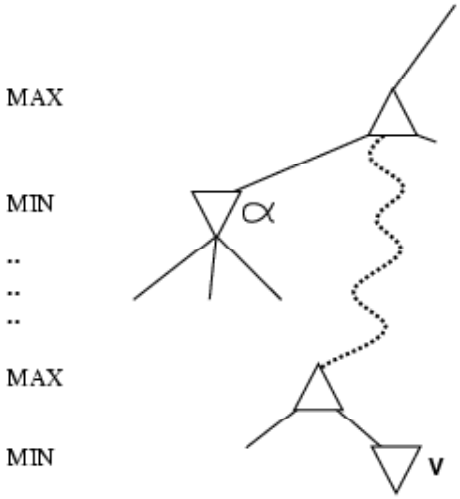
= max(3, *z*, 2) where $z \leq 2$

= 3

خواص آلفا-بتا

- هرس کردن بر روی نتیجه نهایی تاثیر ندارد
- ترتیب خوب حرکت ها میزان تاثیر الگوریتم را بهبود می بخشد
- با «بهترین ترتیب»، پیچیدگی زمانی = $O(b^{m/2})$
- ← عمق جستجوی دو برابر
- یک مثال ساده از ارزش استدلال در مورد محاسبات مرتبط (شکلی از ابر استدلال)

وجه تسمیه



- آلفا مقدار بهترین انتخاب (یعنی بالاترین مقدار) یافته شده تا کنون در هر نقطه انتخاب در طول مسیر برای *max* می باشد
- اگر v بدتر از آلفا باشد، *max* از آن اجتناب می کند

← آن شاخه را هرس می کند

- بتا نیز برای *min* مانند آلفا تعریف می شود

الگوریتم 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
```


الگوریتم آلفا-بتا

```
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
```


محدودیت های منابع

فرض کنید ۱۰۰ ثانیه زمان داریم و در هر ثانیه می توان 10^4 گره گسترش داد.

← در هر حرکت 10^6 گره

روش استاندارد:

- تست برش: مانند محدوده عمقی
- تابع ارزیابی:

= میزان تخمینی مطلوب بودن موقعیت

تابع ارزیابی

- در شطرنج، معمولا مجموع وزن دار ویژگی ها

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

- مثال: $w_1=9$ و

$$f_1(s) = (\#\text{ of white queens}) - (\#\text{ of black queens})$$

برش جستجو

- MinimaxCutoff مانند MinimaxValue می باشد جز اینکه:
  ۱. Terminal? با Cutoff? جایگزین شده و
  ۲. Utility با Eval جایگزین شده است

- آیا در عمل کار می کند؟

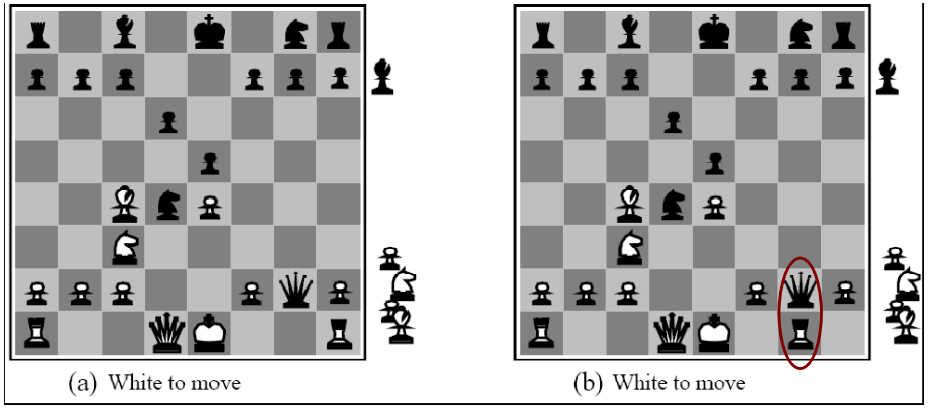
$b^m = 10^6 \rightarrow m = 4$

در شطرنج پیش بینی ۴ لایه نا امید کننده است!
- ۴ لایه = انسان مبتدی
- ۸ لایه = کامپیوترهای معمولی و انسان های حرفه ای
- ۱۲ لایه = Deep Blue و کاسپاروف

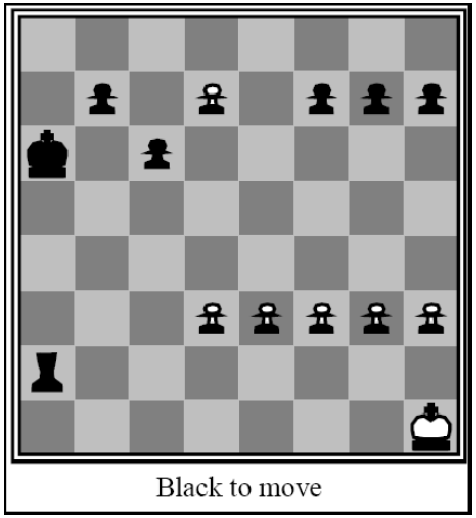
پیاده سازی

- تعیین یک محدوده عمقی مانند d
  - d باید طوری انتخاب شود که زمان لازم از قوانین بازی تجاوز نکند
- روش بهتر: استفاده از IDS تا وقتی که زمان داریم
- مشکلات:
  - خطا به دلیل تقریبی بودن تابع ارزیابی (شکل 6.8b)
- به تابع تست Cutoff پیچیده تری نیاز داریم
  - مشکل حالت های ناآرام ← جستجوی انتظار برای آرامش
  - مشکل اثر افق (شکل 6.9) ←
    - بهبود سخت افزار برای انجام جستجوهای عمیق تر
    - گسترش های انفرادی

شکل ۶-۸



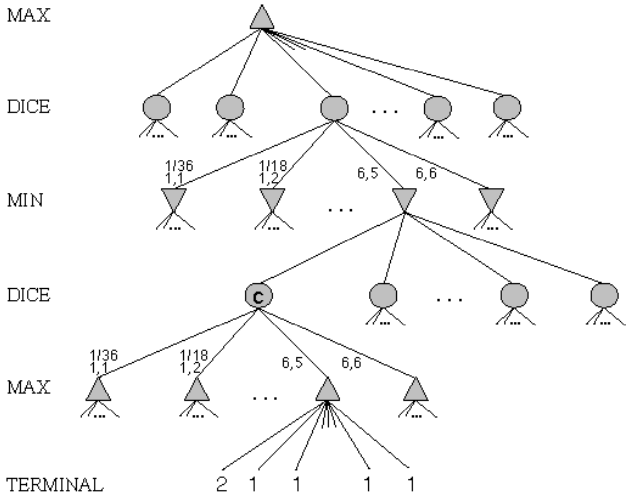
اثر افق



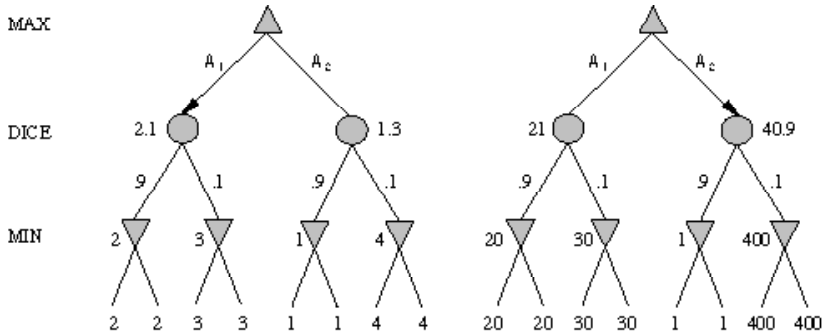
بازی های قطعی در عمل

- Chess: Deep Blue در سال ۱۹۹۷ قهرمان دنیای انسان ها (کاسپاروف) را شکست داد. Deep blue در یک ثانیه ۲۰۰ میلیون موقعیت را جستجو می کند و از توابع ارزیابی بسیار پیچیده ای استفاده می کند.
- Othello: قهرمان های انسانی از رقابت با کامپیوترها (که خیلی خوب هستند) امتناع می ورزند.
- Go: قهرمان های انسانی از رقابت با کامپیوترها (که بسیار بد هستند) امتناع می ورزند. در بازی Go فاکتور انشعاب بیشتر از ۳۰۰ می باشد، بنابراین اکثر برنامه ها برای ارائه حرکت های معقول از پایگاه های دانش الگویی استفاده می کنند.

بازی هایی که دارای عناصر شانس هستند



معنای تابع ارزیابی



- تغییری که ترتیب نسبی مقادیر را حفظ می کند، در تصمیم minimax تأثیری ندارد، اما می تواند تصمیم در مورد گره های شانس را تغییر دهد
- راه حل: تبدیلات خطی

خلاصه

- کار کردن بر روی بازی ها لذت بخش است!
- بازی ها نکات مهم متعددی را در AI به نمایش می گذارند
- عالی بودن ممکن نیست ← باید تقریب زد
- فکر کردن در مورد اینکه به چه چیزی فکر کنی ایده خوبی می باشد