Igranje igara

Poglavlje 5

Prema slajdovima Stuarta Russella (hvala)!

Sažetak

♢ Igre

♢ Savršena (perfektna) igra
  - minimax odluke
  - α–β podrezivanje

♢ Ograničeni resursi i približna procjena (evaluacija)

♢ Slučajne igre

♢ Igre s nepotpunom informacijom

Igre vs. problemi pretraživanja

“Nepredvidivi” protivnik $\Longrightarrow$ rješenje je strategija:
— specificira potez za svaki mogući odgovor protivnika

Vremenska granica $\Longrightarrow$ ne očekujemo naći cilj, moramo aproksimirati

Povijest napada na problem:

- Računalo razmatra moguće načine igre (Babbage, 1846)
- Algoritam za savršenu (perfektnu) igru (Zermelo, 1912; Von Neumann, 1944)
- Konačni horizont (dubina), približna procjena (Zuse, 1945; Wiener, 1948; Shannon, 1950)
- Prvi program za šah (Turing, 1951)
- Strojno učenje za poboljšanje procjene (Samuel, 1952–57)
- Podrezivanje za produbljenje pretrage (McCarthy, 1956)

Vrste igara — podjela

	deterministic	chance
perfect information	**chess, checkers, go, othello**	**backgammon monopoly**
imperfect information	**battleships, blind tictactoe**	**bridge, poker, scrabble nuclear war**

Nepotpuna informacija = dio poteza **nije** vidljiv drugom igraču

Slučajnost = bacanje kocke, izvlačenje karata, . . .

Primjer igre = križić–kružić

engl. = tic–tac-toe

Imamo **dva** igrača = imena su

- MAX (prvi) — stavlja križić (X)
- MIN (drugi) — stavlja kružić (O)

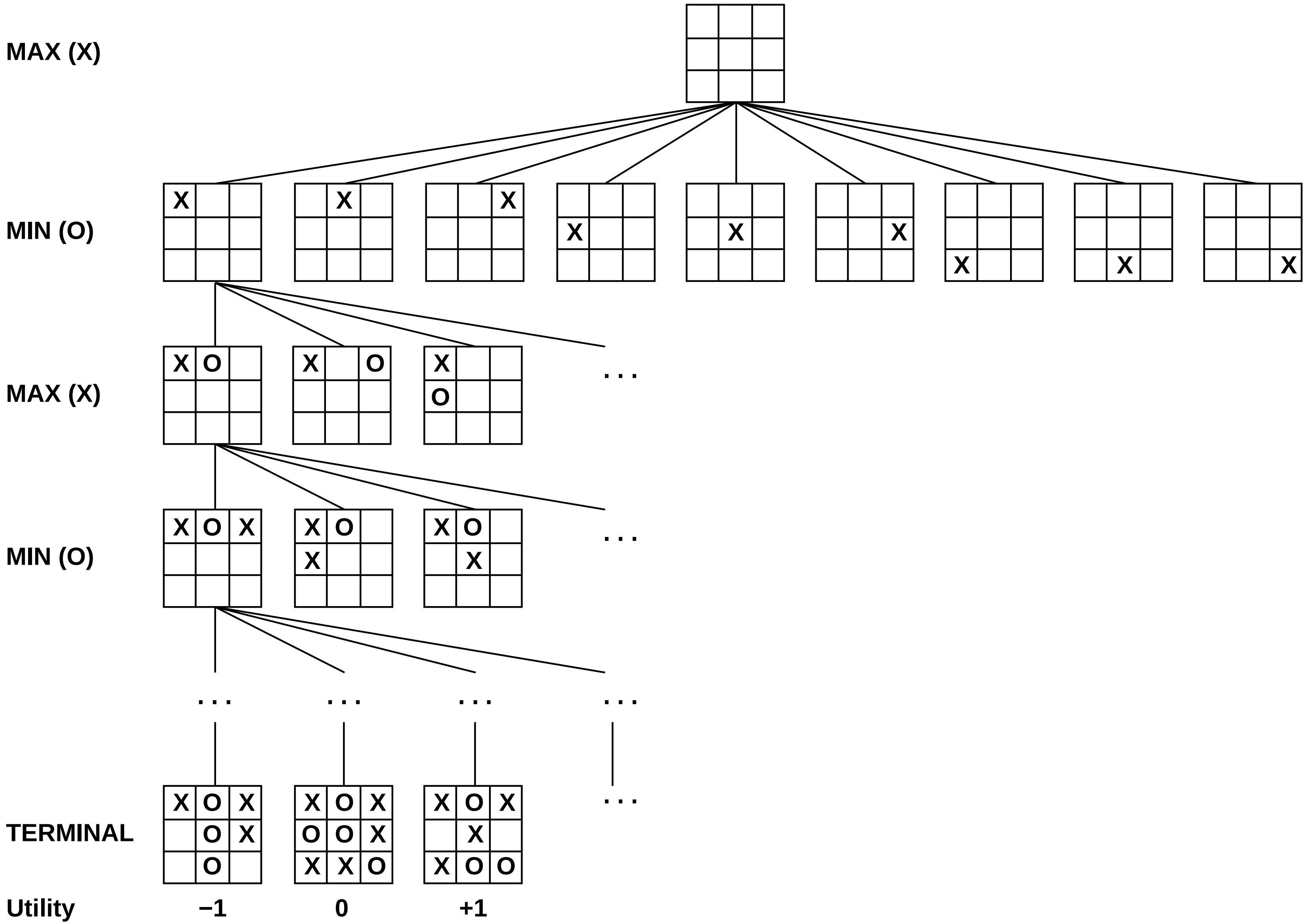
Igra se **naizmjence** — jedan, pa drugi, ...

Cilj (pobjeda/poraz): 3 ista znaka u redu (vodoravno, okomito, koso)

Funkcija korisnosti (dobitka) za **prvog** igrača (engl. utility function):

- pobjeda — dobitak $= +1$
- poraz — dobitak $= -1$
- neriješeno — dobitak $= 0$

Stablo igre (2 igrača, deterministička, naizmjence)



Stablo igre (2 igrača, deterministička, naizmjence)

Manje stablo — korištenjem simetrije ploče . . .

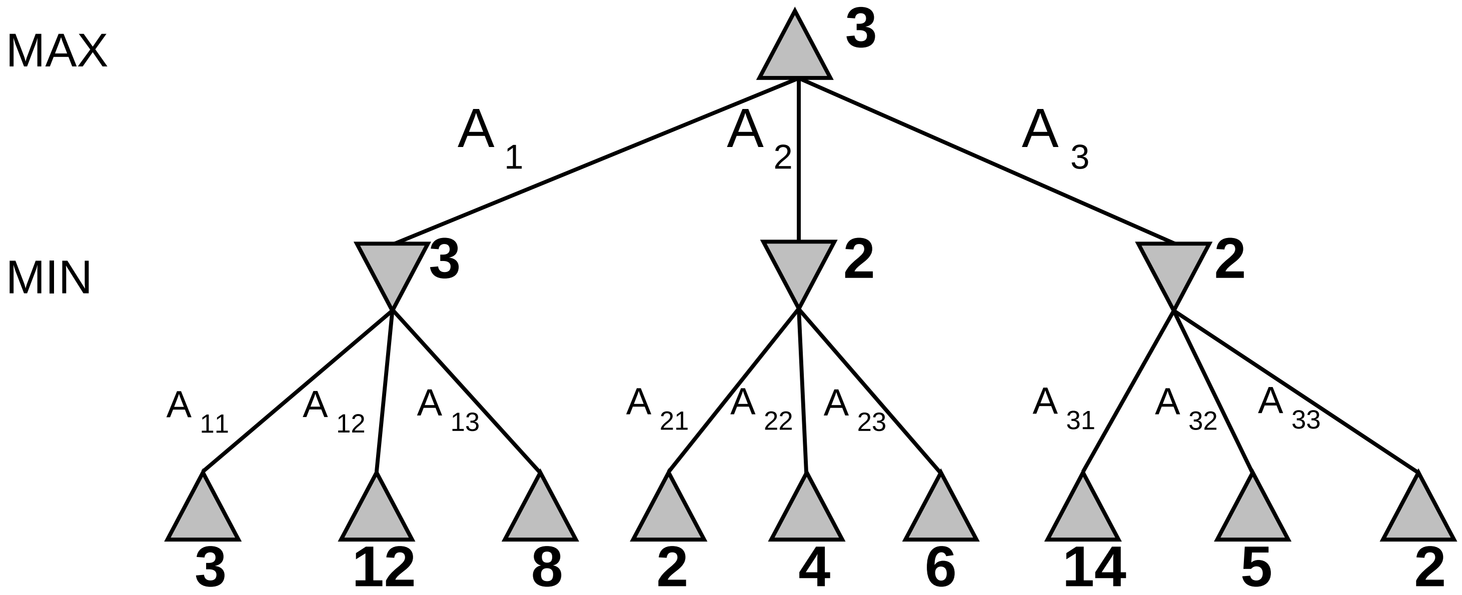
(v. blok_3)

Minimax strategija

"Savršena" igra za determinističke igre s potpunom informacijom

Ideja: izaberi potez na poziciju s najvećom minimax vrijednošću
= najbolji dostižni dobitak protiv najbolje igre

Na pr., 2-kružna igra (jedan potez = dva kruga, prema igračima):



Minimax strategija — iz perspektive prvog igrača

minimax vrijednost čvora = **dobitak** za prvog igrača (MAX), kad je u tom stanju = **gubitak** za drugog igrača (MIN).

Kad MAX bira potez

- želi ići u stanje s **maksimalnom** vrijednošću

Kad MIN bira potez

- želi ići u stanje s **minimalnom** vrijednošću = njegov maksimalni "dobitak"

U **završnom** stanju igre, vrijednost = dobitak (za MAX).

Kod **više** od 2 igrača — onaj tko je **na potezu**, maksimizira **svoj** dobitak iz sljedećih stanja (imamo vektor dobitaka po igračima).

Minimax algoritam (za 2 igrača)

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

    return the *a* in ACTIONS(*state*) maximizing MIN-VALUE(RESULT(a, *state*))

---

function MAX-VALUE(*state*) **returns** *a utility value*
    if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
    $v \leftarrow -\infty$
    for a, s in SUCCESSORS(*state*) **do** $v \leftarrow$ MAX(v, MIN-VALUE(s))
    return v

---

function MIN-VALUE(*state*) **returns** *a utility value*
    if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
    $v \leftarrow \infty$
    for a, s in SUCCESSORS(*state*) **do** $v \leftarrow$ MIN(v, MAX-VALUE(s))
    return v

Svojstva minimax algoritma

<u>Potpun</u>??

Svojstva minimax algoritma

Potpun?? **Da**, ako je stablo konačno (šah ima posebna pravila za to).

Napomena: konačna strategija može postojati čak i u beskonačnom stablu!

Optimalan??

Svojstva minimax algoritma

Potpun?? **Da**, ako je stablo konačno (šah ima posebna pravila za to)

Optimalan?? **Da**, protiv optimalnog protivnika. U suprotnom??

Vremenska složenost??

Svojstva minimax algoritma

Potpun?? **Da**, ako je stablo konačno (šah ima posebna pravila za to)

Optimalan?? **Da**, protiv optimalnog protivnika. U suprotnom??

Vremenska složenost?? $O(b^m)$

Prostorna složenost??

Svojstva minimax algoritma

Potpun?? **Da**, ako je stablo konačno (šah ima posebna pravila za to)

Optimalan?? **Da**, protiv optimalnog protivnika. U suprotnom??

Vremenska složenost?? $O(b^m)$

Prostorna složenost?? $O(bm)$ (istraživanje u dubinu — DFS)

Za šah, $b \approx 35$ (prosjek), $m \approx 100$ — za "razumne" igre (do mata)
$\Longrightarrow$ egzaktno rješenje je potpuno neizvedivo (nemoguće)

Ali, treba li istražiti **svaki** put do konačnih stanja?

Svojstva minimax algoritma

Potpun?? **Da**, ako je stablo konačno (šah ima posebna pravila za to)

Optimalan?? **Da**, protiv optimalnog protivnika. U suprotnom??

Vremenska složenost?? $O(b^m)$

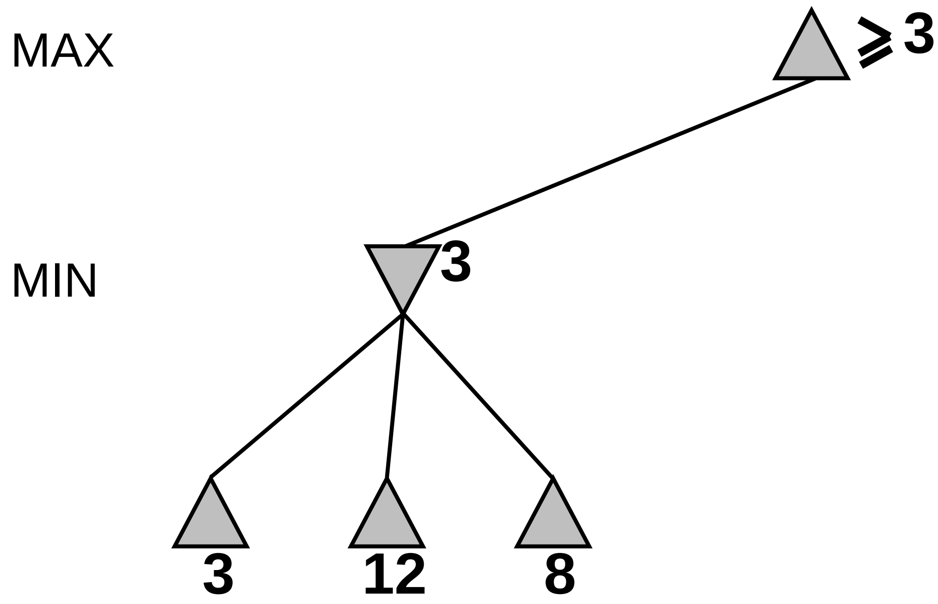
Prostorna složenost?? $O(bm)$ (istraživanje u dubinu — DFS)

Za šah, $b \approx 35$ (prosjek), $m \approx 100$ — za "razumne" igre (do mata)
$\Longrightarrow$ egzaktno rješenje je potpuno neizvedivo (nemoguće)

Ali, treba li istražiti **svaki** put do konačnih stanja?

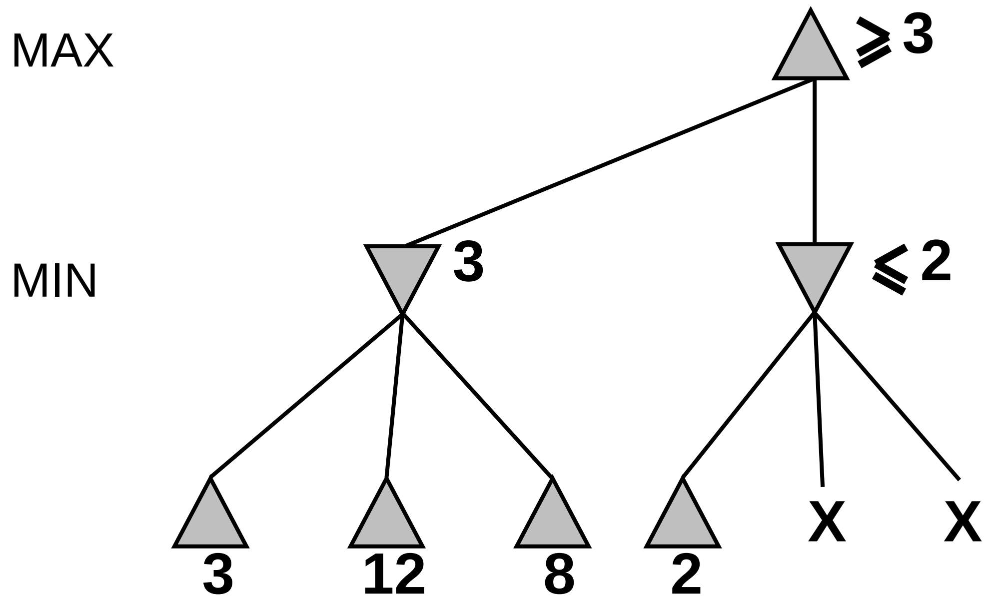
Srećom, ne baš — traženje se može drastično **skratiti**
tzv. podrezivanjem grana u stablu (kao kod pravih stabala).

Primjer α–β podrezivanja



Zaključak nakon **prve** faze do dna. Sva stanja moramo **pogledati**.

Primjer α–β podrezivanja

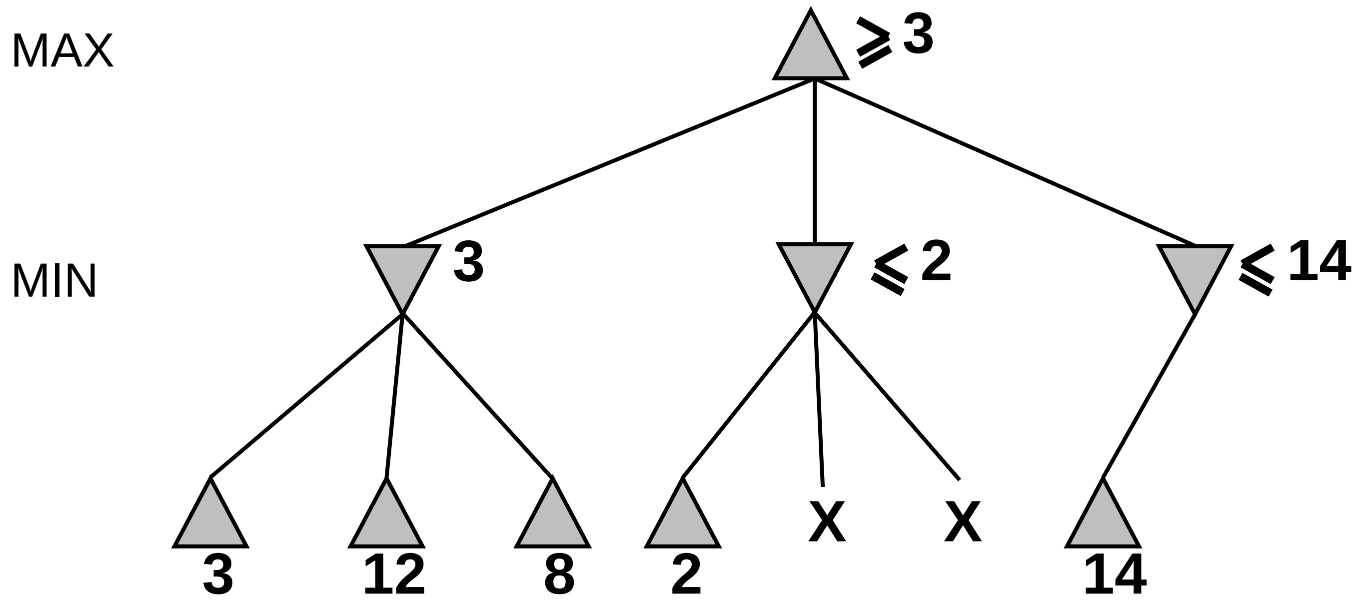


Zaključak nakon **druge** faze do dna.

Nakon prvog stanja, X-ove **ne gledamo**.

Razlog: MIN je najviše 2, pa **ne može** povećati MAX!

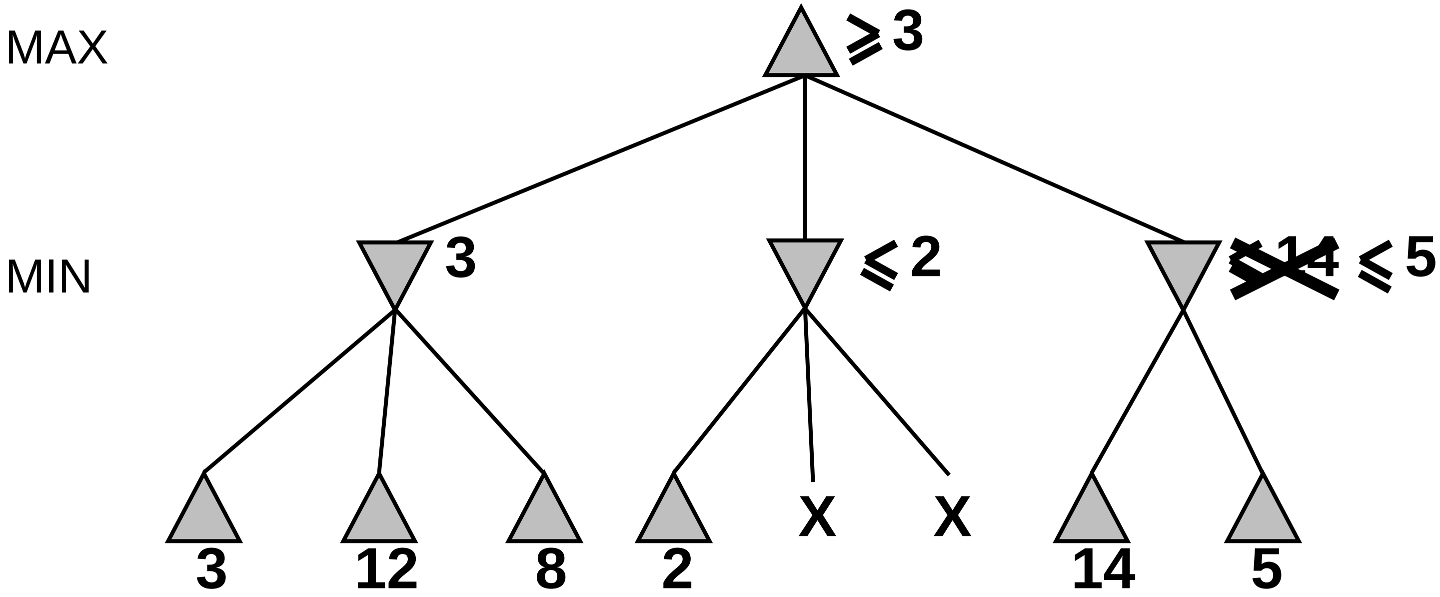
Primjer α–β podrezivanja



Zaključak u **trećoj** fazi do dna.

Prvo stanje **može** povećati MAX — moramo gledati **dalje**!

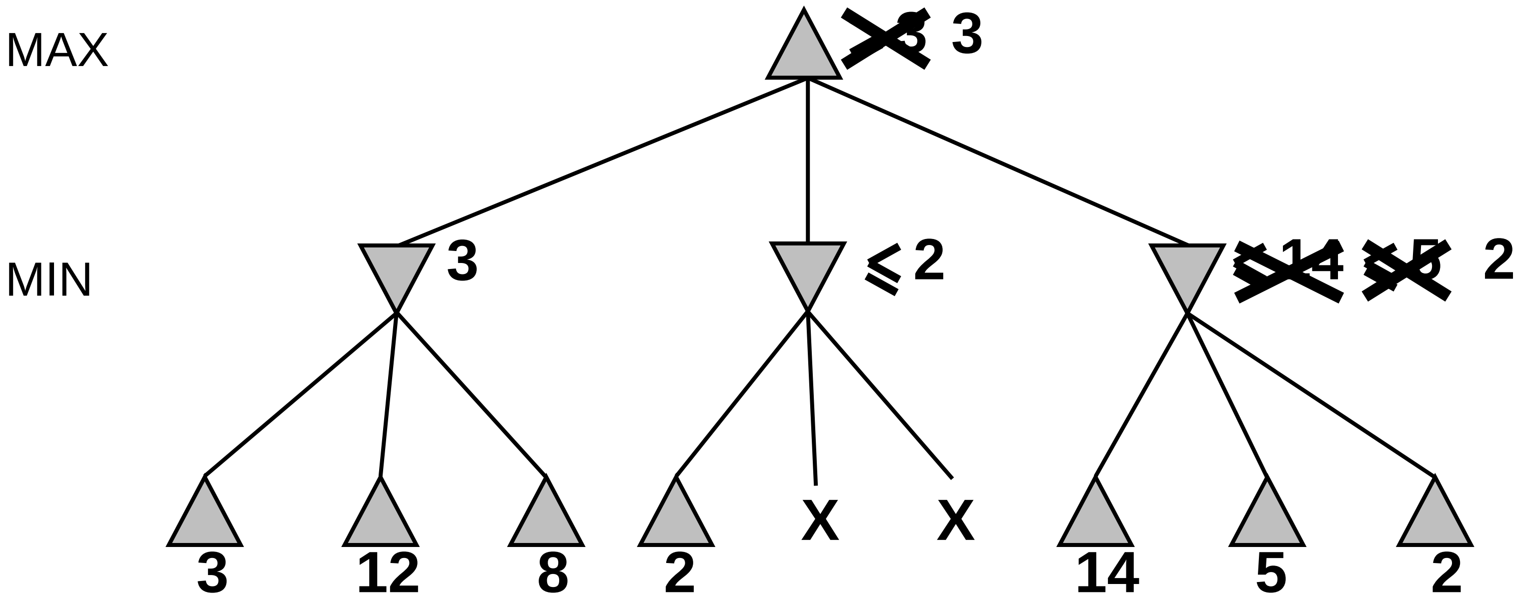
Primjer α–β podrezivanja



Zaključak u **trećoj** fazi do dna.

Drugo stanje **može** povećati MAX — moramo gledati **dalje**!

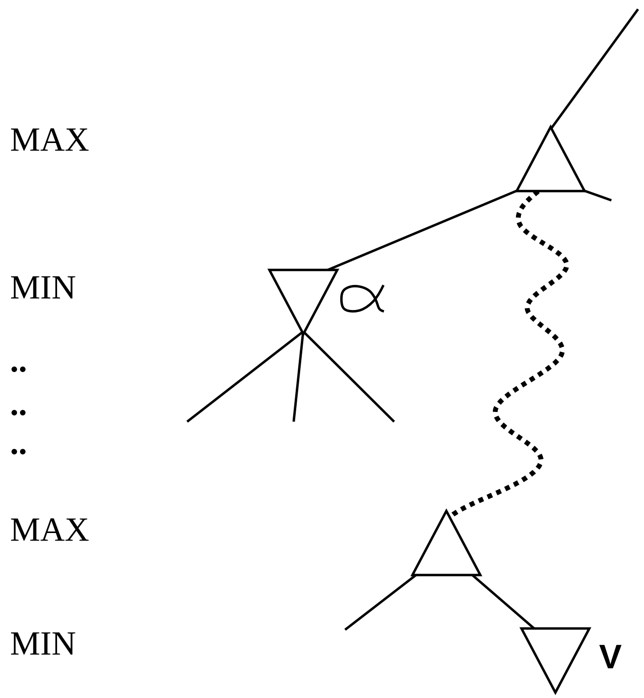
Primjer α–β podrezivanja



Konačni zaključak u **trećoj** fazi do dna.

Da smo **prvo** pogledali stanje (2) — ostala smo mogli **(p)odrezati**!

Zašto se podrezivanje zove α–β?



α je **najbolja** (najveća) vrijednost (za MAX) koju smo našli do sada, na bilo kojem putu, u točki izbora za MAX.

Ako je V lošije od α, MAX će to izbjeći $\Longrightarrow$ podrezati tu granu!

Zašto se podrezivanje zove α–β?

Sasvim analogno, ako dobijemo β i V u nekim točkama za MAX ...

β je **najbolja** (najmanja) vrijednost (za MIN) koju smo našli do sada, na bilo kojem putu, u točki izbora za MIN.

Ako je V lošije od β, MIN će to izbjeći $\Longrightarrow$ podrezati tu granu!

Realizacija podrezivanja za oba igrača:

- U svakoj točki izbora pamtimo **par** (α, β)
- Inicijalizacija je $(-\infty, +\infty)$, jer α **raste**, a β **pada**

Algoritam α–β podrezivanja

function ALPHA-BETA-DECISION(*state*) **returns** an action
  return the *a* in ACTIONS(*state*) maximizing MIN-VALUE(RESULT(*a*, *state*))

---

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*)
  $v \leftarrow -\infty$
  for *a*, *s* in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

---

function MIN-VALUE(*state*, α, β) **returns** *a utility value*
  same as MAX-VALUE but with roles of α, β reversed

Svojstva α–β podrezivanja

Podrezivanje **nema** utjecaja na konačni rezulatat (ostaje isti)

Dobar poredak poteza (u pretrazi) **poboljšava** efikasnost podrezivanja

Uz "idealni poredak", vremenska složenost $= O(b^{m/2})$
$\Longrightarrow$ **udvostručuje** rješivu dubinu (za isto vrijeme)

Jednostavan primjer doprinosa (vrijednosti) razmišljanja o tome
koja računanja su relevantna (bitna) za rezultat
(oblik meta-razmišljanja)

Nažalost, u šahu, čak i "pola" dubine daje 35^{50} — još uvijek nemoguće!

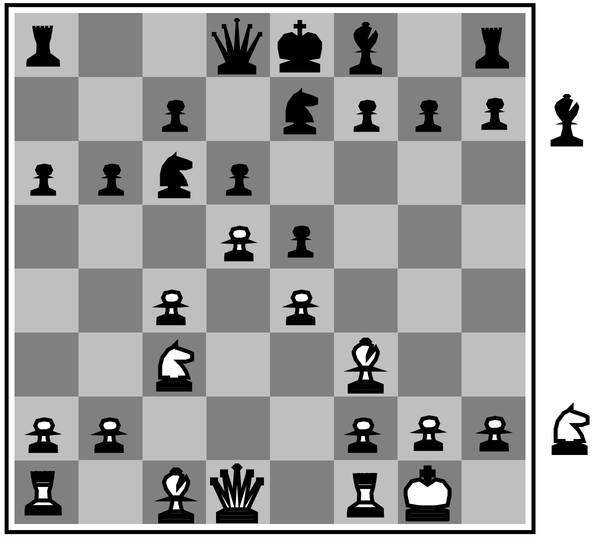
Ograničeni resursi — prostor, vrijeme

Standardni pristup:

- Koristimo CUTOFF-TEST umjesto TERMINAL-TEST
  tj., ograničenje na **dubinu**,
  uz "mirno" ili "tiho" (engl. quiescence) traženje
  $=$ ne očekuju se nagle promjene dobitka
- Koristimo EVAL umjesto UTILITY
  tj., evaluacijsku funkciju ili funkciju procjene
  koja procjenjuje "poželjenost" pozicije

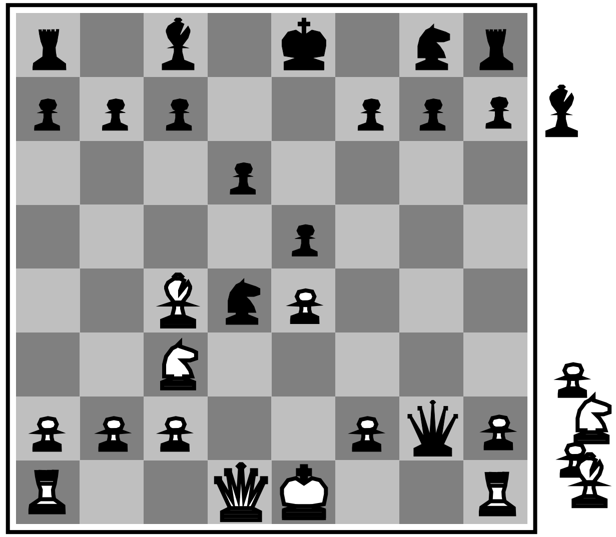
Uzmimo da imamo 100 sekundi i istražujemo 10^4 čvorova po sekundi:
$\Longrightarrow$ 10^6 čvorova po potezu $\approx 35^{8/2} = (\sqrt{35})^8$, ili $b = \sqrt{35} \approx 6$
$\Longrightarrow$ α–β podrezivanje stiže do dubine **8**
$\Longrightarrow$ sasvim dobar šah program!

Evaluacijske funkcije (procjene)



Black to move

White slightly better



White to move

Black winning

Za šah, obično je f = linearna težinska suma pojedinih obilježja pozicije

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

Na pr., $w_1 = 9$, uz

$f_1(s)$ = (broj bijelih kraljica) – (broj crnih kraljica), itd.

Obični Minimax algoritam — ponavljanje

Skraćeni zapis:

- $M(s) = \textsc{Minimax}(s)$ — minimax vrijednost čvora (stanja) s
- $\mathrm{succ}(s)$ = funkcija sljedbenika = svi valjani potezi u stanju s
- $\mathrm{terminal}(s) = \textsc{Terminal-Test}(s)$ = provjera završnog stanja

Minimax vrijednost čvora (stanja) s je

$$\textsc{Minimax}(s) = M(s) =$$

$$\begin{cases} \textsc{Utility}(s) & \text{ako je } \mathrm{terminal}(s) \\ \max \{\, M(t) \mid t \in \mathrm{succ}(s) \,\} & \text{ako je } s = \mathrm{MAX} \text{ čvor} \\ \min \{\, M(t) \mid t \in \mathrm{succ}(s) \,\} & \text{ako je } s = \mathrm{MIN} \text{ čvor} \end{cases}$$

Rekurzija u **dubinu**, postavljanje vrijednosti pri **povratku**!
prostorna složenost $O(m)$, ali vremenska je $O(b^m)$.

Problemi s dubinom traženja

U stvarnim igrama (šah i sl.),

- potrebna **dubina** m za nalaženje **optimalne** strategije (odn. prvog poteza za MAX) je **prevelika** za raspoloživo **vrijeme**

$\Longrightarrow$ Moramo postaviti ograničenje na **dubinu** pretrage:

- **najviše** do dubine $d_{\max}$,
- **ispod** te dubine — procjena dobitka na temelju heurističke funkcije $h = \text{Eval}$.

Posljedice:

- svaki igrač donosi **nesavršene** odluke (poteze)
  (inače je igra **nezanimljiva** — sve se **zna** od prvog poteza!)
- nakon **svakog** (nesavršenog) poteza protivnika, igrač treba **iznova** napraviti procjenu — za **sljedeći** potez.

Heuristički Minimax algoritam (za 2 igrača)

Vodimo računa o dubini, **početna** dubina je $d = 0$. Oznake:

- $H(s,d) = \text{H–MINIMAX}(s,d)$ = heuristička minimax vrijednost čvora (stanja) s na dubini d
- $h(s) = \text{EVAL}(s)$ = procjena dobitka za MAX u stanju s
- $\text{cut}(s,d) = \text{CUTOFF-TEST}(s,d)$ = provjera prekida spuštanja

Heuristička minimax vrijednost čvora (stanja) s na dubini d je

$$\text{H–MINIMAX}(s,d) = H(s,d) =$$

$$\begin{cases} h(s) & \text{ako je } \text{cut}(s,d) \\ \max\{\, H(t,d+1) \mid t \in \text{succ}(s) \,\} & \text{ako je } s = \text{MAX čvor} \\ \min\{\, H(t,d+1) \mid t \in \text{succ}(s) \,\} & \text{ako je } s = \text{MIN čvor} \end{cases}$$

Uloga funkcije prekida

Usporedba s običnim minimax algoritmom:

$$\mathrm{terminal}(s) = \textsc{Terminal-Test}(s)$$

prelazi u

$$\mathrm{cut}(s,d) = \textsc{Cutoff-Test}(s,d)$$

Drugim riječima, funkcija cut

- pretvara **neterminalne** čvorove (stanja) igre
- u **terminalne** listove stabla pretrage — kao da su završna stanja,

a dobitak se **procjenjuje** na temelju evaluacijske funkcije h.

Najjednostavnije "rezanje":

$$\mathrm{cut}(s,d) = \mathsf{true} \iff d \geq d_{\max} \ \text{ili} \ \mathrm{terminal}(s) = \mathsf{true}$$

Bolje: **iterativno** spuštanje u dubinu (IDS), do **isteka** vremena.

Dobre heurističke funkcije

Očito: **loša** procjena može dovesti do (nepotrebnog) **poraza**.

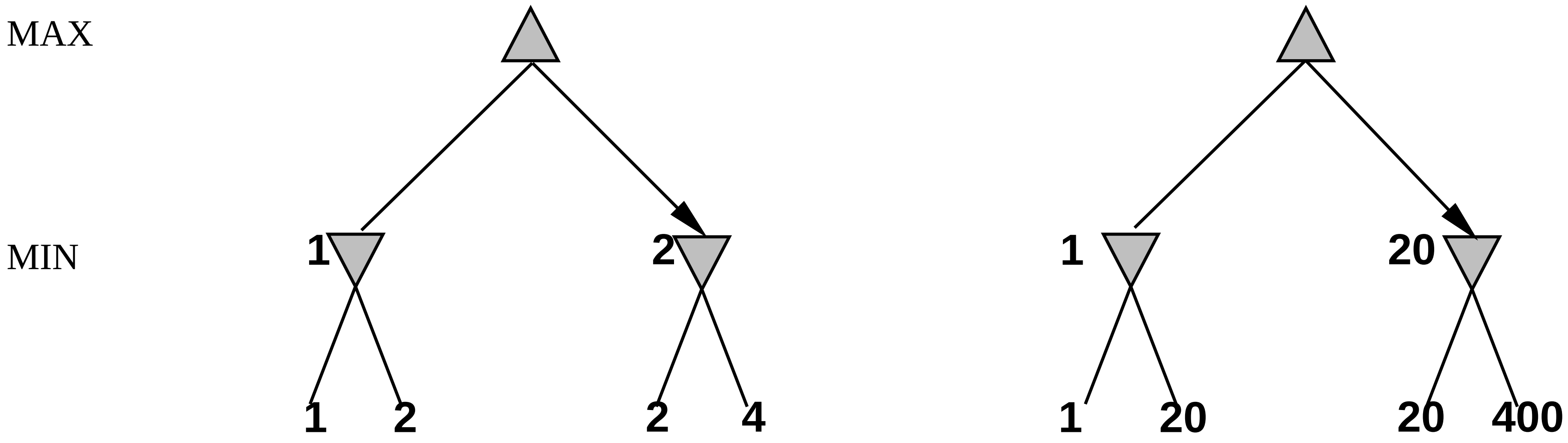
Bitno:

- za **terminalna** stanja, procjena h mora dati **isti poredak** kao i pravi dobitak = UTILITY funkcija.

Ovo odgovara "dopustivosti" heuristike!

Primjer: Stvarne vrijednosti dobitka nisu bitne

Invarijantnost "odluka" — poretka terminalnih stanja:



Ponašanje ostaje isto (čuva se)

za bilo koju **monotonu** transformaciju funkcije $h = \textsc{Eval}$.

Dakle, samo **poredak** je bitan:

dobitak u determinističkim igrama ima ulogu funkcije **poredavanja** prema korisnosti.

Dobre heurističke funkcije (nastavak)

Za **neterminalna** stanja

- računanje mora biti dovoljno **brzo** — to je poanta
- procjena mora biti **visoko korelirana** sa stvarnom "šansom" za pobjedu, odn. stvarnim dobitkom

  (procjena je "nagađanje" = "šansa", iako je igra deterministička).

Zato su stvarne procjene sastavljene iz puno dijelova.

Lakša varijanta = linearna težinska suma pojedinih "lokalnih" heuristika (pojedinih obilježja pozicije)

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

Iza ovog se "skriva" nezavisnost pojedinih doprinosa f_i.

Teža varijanta = nelinearna kombinacija (zavisna obilježja).

Heurističko α–β podrezivanje

Dodatno skraćenje pretrage

heuristička modifikacija α–β podrezivanja

Potrebne promjene u osnovnom algoritmu — isto kao za minimax:

U funkcijama Max-Value i Min-Value treba

- dodati dubinu d kao argument
- umjesto testa terminalnog stanja

  if Terminal-Test(*state*) **then return** Utility(*state*)

- treba staviti provjeru prekida

  if Cutoff-Test(*state*, *d*) **then return** Eval(*state*)

Napomena: igrači mogu koristiti **različite** heurističke funkcije h_1 i h_2.

Primjeri za determinističke igre

v. FER, UI–4, od str. 12 nadalje (do str. 21)

Determinističke igre u praksi

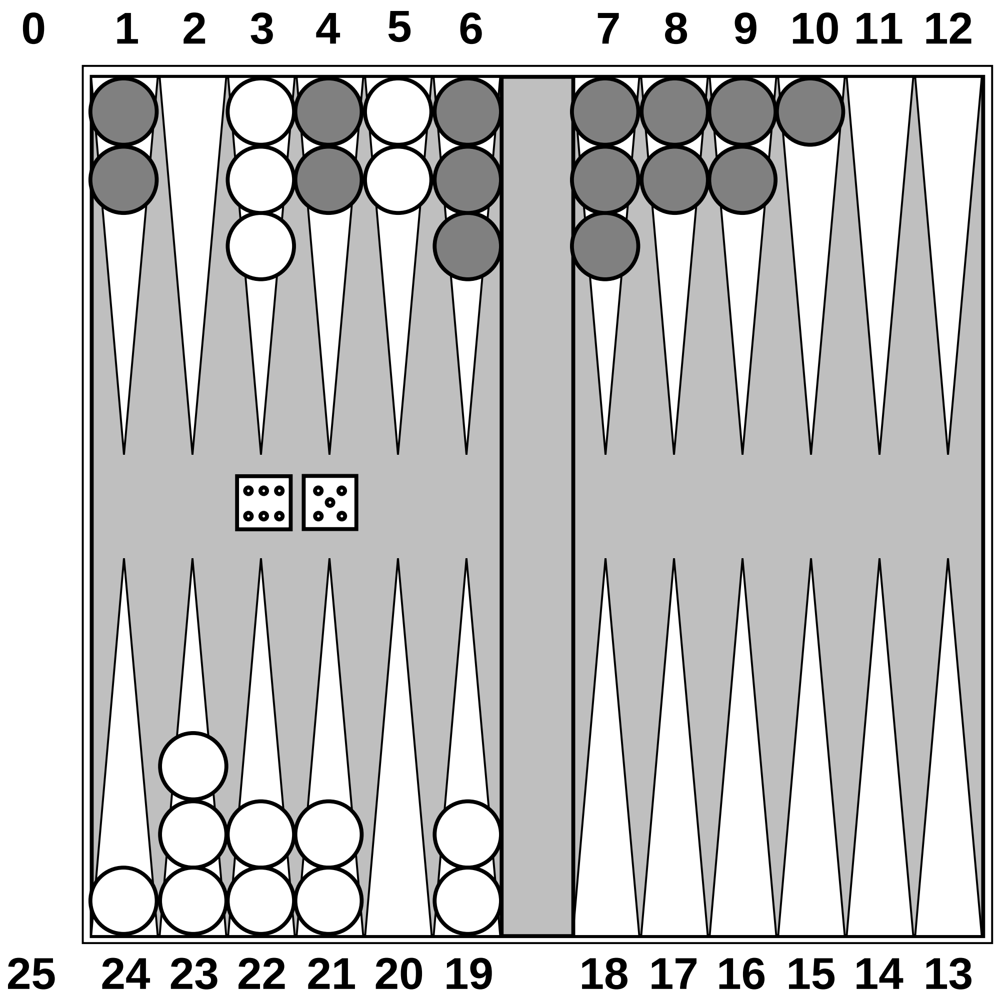
Dama: Program CHINOOK završio je 1994.g. dugogodišnju vladavinu ljudskog svjetskog prvaka Mariona Tinsleyja. Igra savršeno — koristeći α-β pretragu i bazu "završnica" (malo figura na ploči, oko $39 \cdot 10^{12}$ pozicija).

Šah: IBM-ov DEEP BLUE pobijedio je svjetskog prvaka Garija Kasparova u meču od 6 partija 1997. godine. Pretražuje oko 200 miliona pozicija u sekundi, ima vrlo složenu evaluacijsku funkciju i koristi (tajne) metode za produbljivanje pretrage i do 40 "krugova" duboko. Moderni programi imaju veći "rejting" od ljudi.

Othello ili Reversi: manji prostor pretrage, obično 5–15 dozvoljenih poteza. Današnji programi su bitno bolji od najboljih ljudskih igrača.

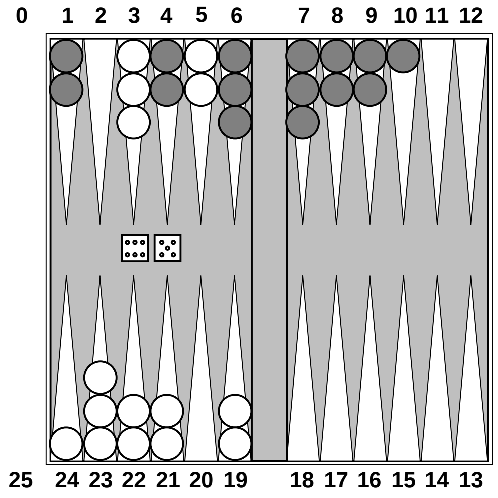
Go: Faktor grananja je ogroman, $b > 300$. Moderni programi su na razini majstora, još uvijek gube protiv najboljih ljudskih igrača.

Nedeterminističke igre: backgammon



Bacaju se dvije kocke — element slučajnosti (ili sreće/nesreće)

Backgammon — kratki opis



Cilj igre: maknuti sve svoje figure izvan ploče.

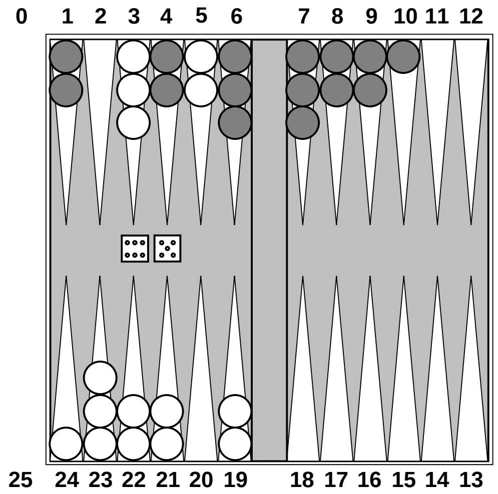
Potezi:

bijeli igra u smjeru kazaljke na satu — prema 25,
crni igra obratnim smjerom — prema 0.

Figura se može maknuti na bilo koju poziciju, osim ako tamo već postoje bar dvije protivnikove figure.

Ako postoji samo jedna, ona je “zarobljena” i mora krenuti iznova.

Backgammon — kratki opis (nastavak)



U ovoj poziciji, bijeli je bacio 6-5 – to su zadane duljine poteza.

Mora izabrati između 4 dozvoljena poteza:

(5-10, 5-11), (5-11, 19-24), (5-10, 10-16) i (5-11, 11-16)

Oznaka (5-11, 11-16) znači

makni jednu figuru s pozicije 5 na 11 — za 6 mjesta,

onda makni (tu istu) figuru s 11 na 16 — za 5 mjesta.

Backgammon — modeliranje stabla igre

Bijeli, naravno, zna svoje dozvoljene poteze (grananje). Međutim,

- on ne zna što će crni baciti, tj. što će biti dozvoljeni potezi crnog!

Za konstrukciju stabla igre:

Slučajnost se tretira kao novi igrač = CHANCE

Oznaka = kružni čvorovi, između MAX i MIN čvorova.

Grananje u CHANCE čvoru

— predstavlja sve moguće ishode bacanja para kocaka,

— svaka grana je označena ishodom i vjerojatnošću tog ishoda.

Backgammon — grananje u CHANCE čvoru

Konkretno,

par kocaka možemo baciti na 36 načina i svi su jednako vjerojatni.

Ishod bacanja je par (x, y), za $x, y \in \{1, \ldots, 6\}$.

Vjerojatnost svakog ishoda je $P(x, y) = 1/36$.

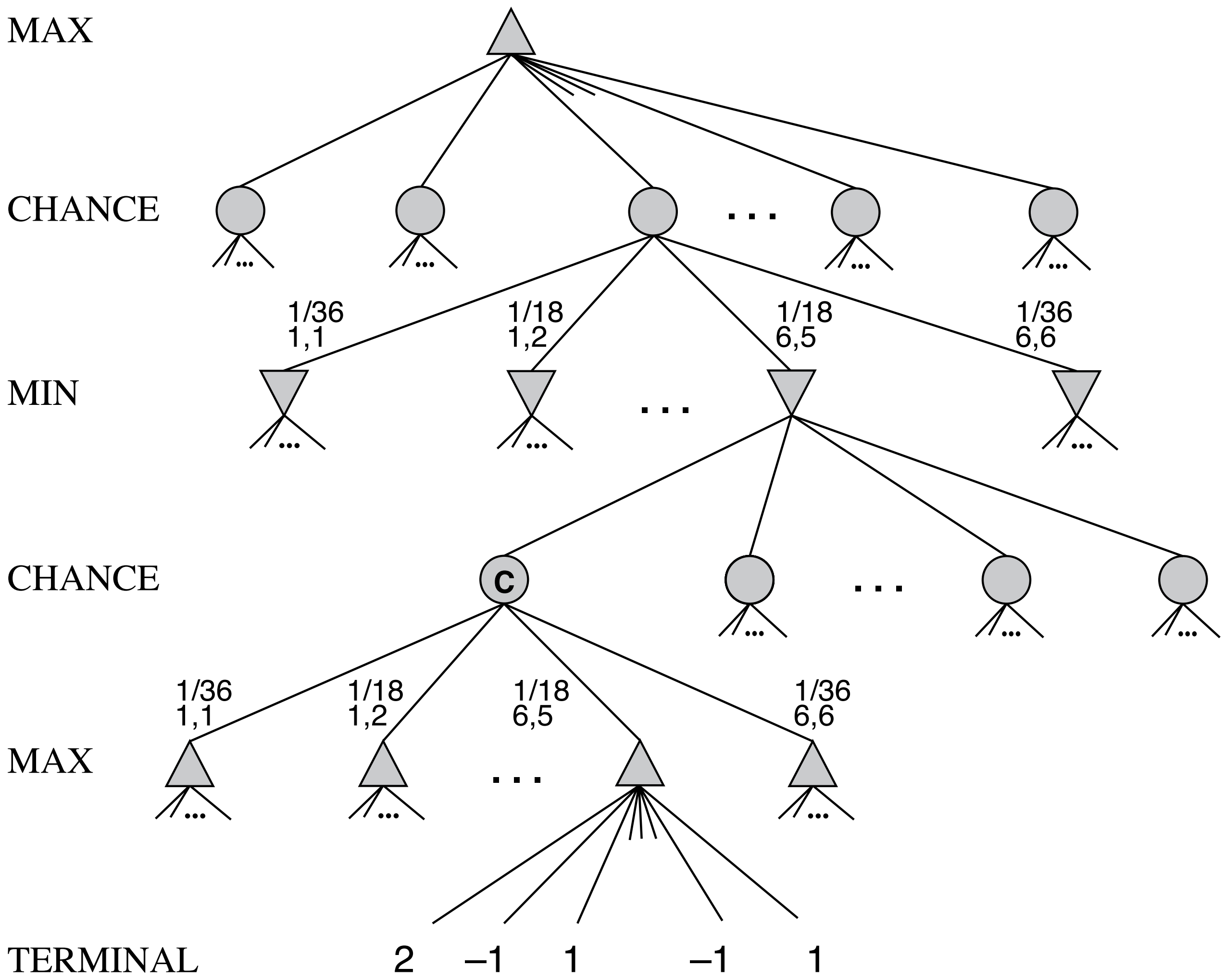
No, u igri je 6-5 isto što i 5-6 (samo brojevi su važni, ne i poredak).

Kad to uvažimo, imamo "samo" 21 mogući ishod

- 6 duplih kocaka (1-1 do 6-6),
  svaki ima vjerojatnost $P(\text{x-x}) = 1/36$
- 15 različitih kocaka (1-2 do 5-6),
  svaki ima vjerojatnost $P(\text{x-y}) = 1/18 \quad (\text{x} < \text{y})$.

Dakle, faktor grananja u svakom CHANCE čvoru je $n = 21$.

Shema stabla igre za poziciju u backgammonu



Izbor poteza — prema očekivanom dobitku

Kako odlučujemo?

Naime, pozicije nemaju precizno definirane minimax vrijednosti!

Umjesto toga, jedino razumno što možemo napraviti u CHANCE čvoru

$=$ izračunati **očekivanu** vrijednost pozicije,

$=$ srednja vrijednost po svim mogućim ishodima u tom čvoru.

To radimo za svaki CHANCE čvor posebno.

U svim ostalim čvorovima postupamo kao i ranije

- u terminalnim čvorovima — vrijednost $=$ dobitak
- u MAX i MIN čvorovima (za koje se zna ishod bacanja tik ispred) radimo isto šo i prije — optimiziramo očekivani dobitak (max/min).

Ovo je generalizacija MINIMAX algoritma za detereminističke igre na nedeterminističke igre sa slučajnim čvorovima $=$ EXPECTIMINIMAX.

Expectiminimax algoritam

Skraćeni zapis:

- $EM(s) = \textsc{Expectiminimax}(s)$ — očekivana minimax vrijednost čvora (stanja) s

$$\textsc{Expectiminimax}(s) = EM(s) =$$

$$\begin{cases} \textsc{Utility}(s) & \text{ako je } \mathrm{terminal}(s) \\ \max\{\, EM(t) \mid t \in \mathrm{succ}(s) \,\} & \text{ako je } s = \mathrm{MAX} \text{ čvor} \\ \min\{\, EM(t) \mid t \in \mathrm{succ}(s) \,\} & \text{ako je } s = \mathrm{MIN} \text{ čvor} \\ \mathrm{avg}\{\, EM(t) \mid t \in \mathrm{succ}(s) \,\} & \text{ako je } s = \mathrm{CHANCE} \text{ čvor} \end{cases}$$

Srednja vrijednost avg je

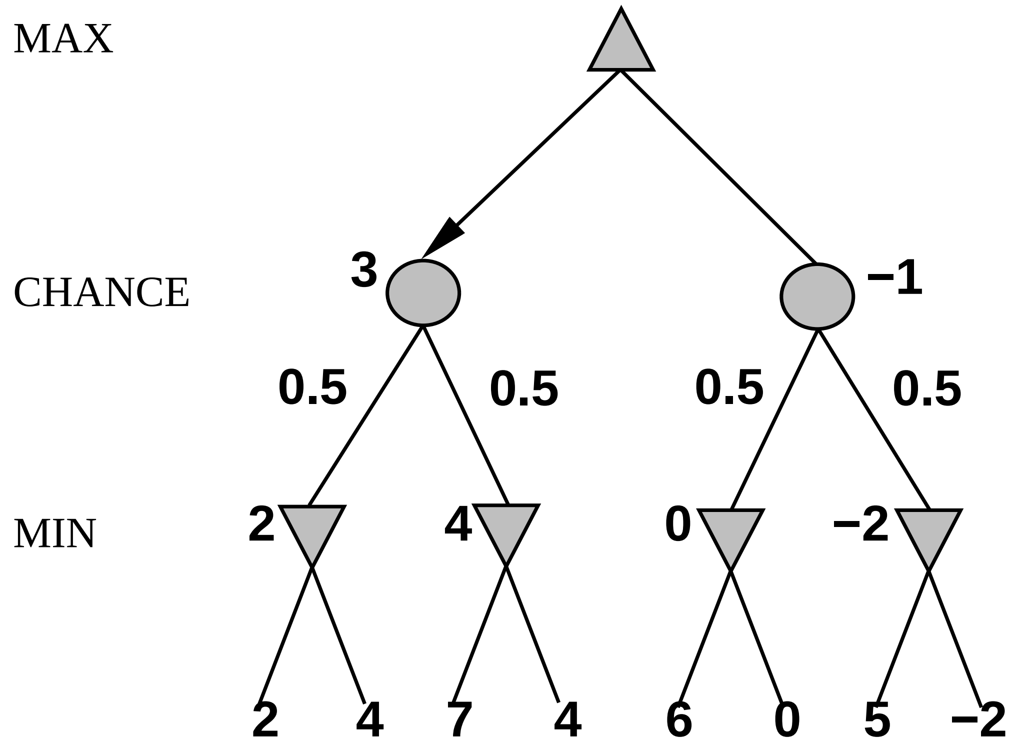
$$\Sigma_{t \in \mathrm{succ}(s)}\, P(t) \cdot EM(t)$$

Ovdje t znači ishod "slučajnosti" u CHANCE stanju s.

Nedeterminističke igre — primjer

U nedeterminističkim igrama, slučajnost se dobiva
bacanjem kocke (ili više njih), miješanjem karata, i sl.

Jednostavni primjer — bacanjem novčića (pola–pola):



Nedeterminističke igre u praksi — složenost

Vremenska složenost: Osim "običnog" grananja igre b, expectiminimax dodatno pretražuje i sve moguće ishode slučajnosti.

Ako je $n =$ broj mogućih ishoda, onda je potrebno vrijeme $O(b^m n^m)$.

Čak i uz ograničenje dubine pretrage na vrlo mali $d_{\max}$, dodatno trajanje (obzirom na obični minimax)

$\Longrightarrow$ nerealno je "daleko gledati unaprijed" u većini igara.

Primjer: u backgammon igri je $n = 21$. Faktor grananja b je obično oko 20, ali može narasti na 4000 za bacanje "duplih" kocaka.

Za dubinu traženja $d = 4$, uz $b = 20$, dobivamo (tri bacanja kocke)

$$20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$$

Stvarno, možemo pretražiti najviše tri kruga!

Nedeterminističke igre u praksi — heuristike

Dakle, primjena dobrih heuristika za procjenu je nužna!

Napomena: Program TDGAMMON koristi pretragu samo do dubine 2
+ jako dobru funkciju procjene EVAL
$\approx$ razina svjetskog prvaka

Loša stvar: kako dubina raste, vjerojatnost stizanja do danog čvora drastično pada
$\Longrightarrow$ korist od "gledanja unaprijed" se smanjuje
α–β podrezivanje je puno manje efikasno.

Dobre heurističke funkcije — promjena!

Što su ovdje “dobre” ili “dopustive” heurističke funkcije?

Princip je isti kao i prije:

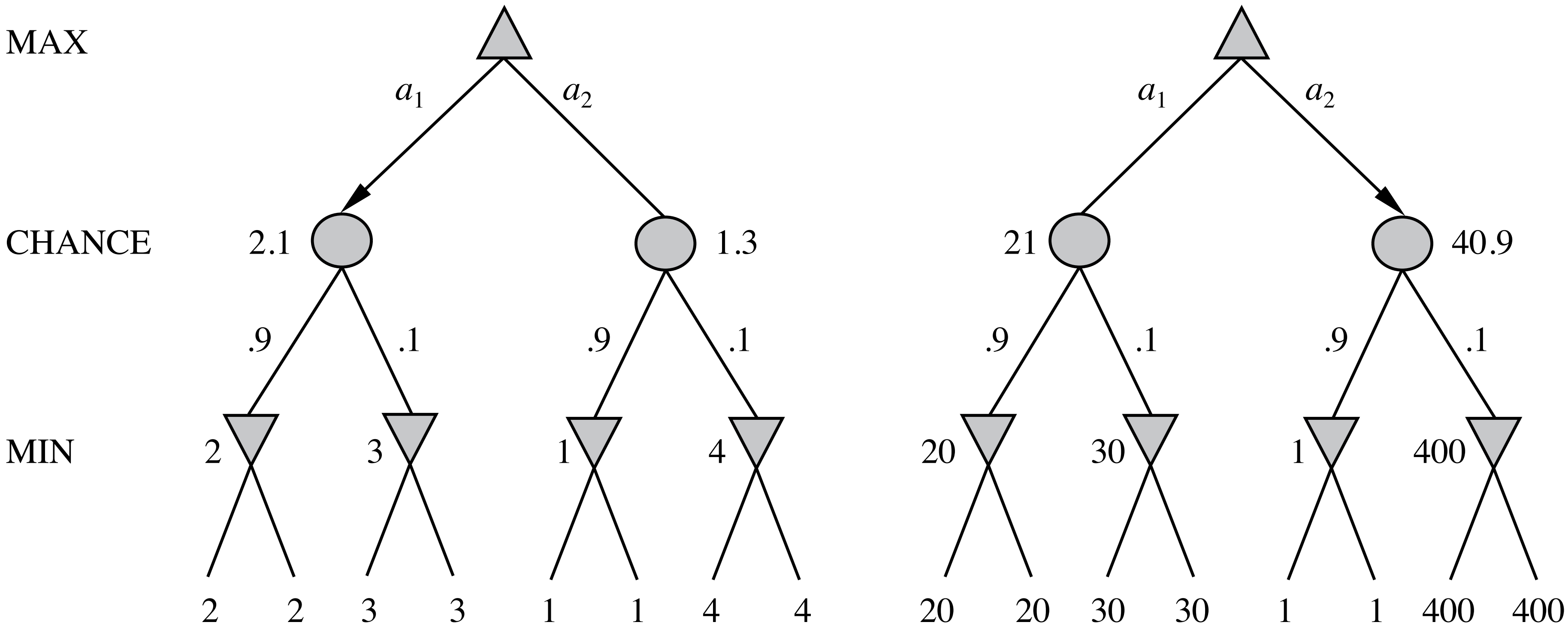
- za **terminalna** stanja, procjena h mora dati **isti poredak** kao i pravi očekivani dobitak — iz UTILITY funkcije.

Drugim riječima: Invarijantnost “odluka” — poretka terminalnih stanja.

Međutim, zbog računanja očekivanja u CHANCE čvorovima,

- stvarne vrijednosti funkcije $h = \text{EVAL}$ postaju bitne!

Primjer: Stvarne vrijednosti dobitka jesu bitne



Na lijevoj slici, MAX treba izabrati lijevu granu, a na desnoj slici treba izabrati desnu granu.

Razlog: drastično povećanje dobitka u najdesnijem završnom čvoru ($4 \mapsto 400$) mijenja odnos grana u CHANCE čvorovima!

Dobre heurističke funkcije — zaključak

Ponašanje ostaje isto (čuva se) samo za

> **pozitivnu linearnu** transformaciju funkcije očekivanog dobitka dane pozicije.

Stoga EVAL mora biti proporcionalna očekivanom dobitku!

Igre s nepotpunom informacijom — ukratko

Primjeri:

ratovi i ratne igre — ne znamo protivnikov raspored snaga ...
kartaške igre — ne znamo protivnikove početne karte,
uz dodatak slučajnosti ako se karte "vuku" i kasnije.

Kartaške igre, potapanje brodova:

obično možemo izračunati vjerojatnost za svaku moguću podjelu (početni raspored karata, brodova).

Kao da imamo jedno veliko "bacanje kocaka" na početku igre.*

Ideja:

izračunaj minimax vrijednost svake akcije u svakoj podjeli,
onda izberi akciju najvećim očekivanjem preko svih podjela.*

Poseban slučaj: ako je akcija optimalna za sve podjele $\Longrightarrow$ optimalna.*

Primjer zdravog razuma za *

Put A vodi prema maloj hrpi zlata

Put B vodi prema razdvajanju staza

  uzmi lijevu stazu i naći ćeš gomilu dragulja;

  uzmi desnu stazu i pregazit će te autobus.

Zaključak: idi B, samo uzmi lijevo!

Primjer zdravog razuma za *

Put A vodi prema maloj hrpi zlata
Put B vodi prema razdvajanju staza
    uzmi lijevu stazu i naći ćeš gomilu dragulja;
    uzmi desnu stazu i pregazit će te autobus.

Put A vodi prema maloj hrpi zlata
Put B vodi prema razdvajanju staza (suprotan izbor)
    uzmi lijevu stazu i pregazit će te autobus;
    uzmi desnu stazu i naći ćeš gomilu dragulja.

Zaključak: opet idi B, samo sad uzmi desno!

Primjer zdravog razuma za *

Put A vodi prema maloj hrpi zlata
Put B vodi prema razdvajanju staza
    uzmi lijevu stazu i naći ćeš gomilu dragulja;
    uzmi desnu stazu i pregazit će te autobus.

Put A vodi prema maloj hrpi zlata
Put B vodi prema razdvajanju staza (suprotan izbor)
    uzmi lijevu stazu i pregazit će te autobus;
    uzmi desnu stazu i naći ćeš gomilu dragulja.

Put A vodi prema maloj hrpi zlata
Put B vodi prema razdvajanju staza (pogađanje)
    izaberi korektno i naći ćeš gomilu dragulja;
    izaberi pogrešno i pregazit će te autobus.

Ovo je srednja vrijednost prethodnih slučajeva!   Opet idi B???

Ispravna analiza

Objašnjenje za *:

Intuicija da je globalna vrijednost neke akcije
$=$ prosjek njezinih vrijednosti preko svih stvarnih stanja
je **POGREŠNA**!

Može se koristi za aproksimaciju, na smanjenom broju stanja (uzorak).

Uz djelomično opažanje/informacije, vrijednost akcije ovisi o
stanju informiranosti ili uvjerenja u kojem je agent tog trena.

Može se generirati i pretraživati stablo stanja informiranosti.

To dovodi do razumnih ponašanja, poput
- ♢ Akcija za prikupljanje informacija (igra potapanja brodova)
- ♢ Signaliziranja nekom od partnera (kartaške igre)
- ♢ Slučajnih akcija za prikrivanje informacija protivniku

Sažetak

Igre su zabavne za rad na njima (i opasne!)

One ilustriraju nekoliko važnih stvari o UI:

◇ savršenost je nedostižna $\Longrightarrow$ moramo aproksimirati

◇ dobra ideja = razmisliti o čemu treba razmišljati (voditi računa)

◇ nesigurnost/neodređenost ograničava dodjelu vrijednosti stanjima
= baza za podrezivanje u nedeterminističkim igrama

◇ optimalne odluke ovise o stanju informiranosti,
a ne o stvarnom stanju

Igre su Umjetnoj inteligenciji isto što i utrke za projektiranje automobila.