

AI

---

## Kunstmatige Intelligentie (AI)

Hoofdstuk 5 van Russell/Norvig = [RN]  
Spel(l)en

voorjaar 2011 — College 6, 15 maart 2011

[www.liacs.nl/home/kosters/AI/](http://www.liacs.nl/home/kosters/AI/)

Spellen geven aanleiding tot zeer complexe zoekproblemen, bijvoorbeeld bij schaken, go, vier-op-een-rij. Extra problemen zijn contingency (onzekerheid), kansen, . . .

We hebben **evaluatie-functies** nodig om niet-eindtoestanden te beoordelen, bijvoorbeeld bij schaken: pion 1, paard/loper 3, toren 5, koningin 9, veiligheid koning, . . .

We willen **prunen**: niet alles doorrekenen. We bekijken met name het **minimax-algoritme** van Von Neumann (1928) en het  **$\alpha$ - $\beta$ -algoritme** uit 1956/58.

Leuk leesvoer: H.J. van den Herik, J.W.H.M. Uiterwijk en J. van Rijswijk, Games solved: Now and in the future, Artificial Intelligence 134 (2002) 277–311.





Marion Tinsley (1927–1995) was de beste menselijke **checkers**-speler (dammen op een schaakbord) uit de geschiedenis. In 2007 werd door Jonathan Schaeffer bewezen dat de beginspeler altijd minstens remise kan halen.

Spellen kunnen als volgt worden ingedeeld:

	deterministisch	kans
perfecte informatie	schaken, dammen, checkers, go, othello	monopoly, backgammon
onvolledige informatie	zeeslag, mastermind	bridge, poker, scrabble

En dan is er nog onderscheid in het aantal spelers.

Met name over schaken is veel gepubliceerd, waaronder allerlei anecdotes.

De ultieme uitdaging is vooralsnog het spel go.

Er zijn verschillende **strategieën** om (een benadering van) de “speltheoretische waarde” van een spel te bepalen. **Shannon** (1950) onderscheidt drie types:

**type A** reken alles tot en met zekere diepte door, en gebruik daar een evaluatie-functie

**type B** reken soms verder door (als het onrustig is: “quiescence” ); gebruik heuristische functie om dit te sturen

**type C** doelgericht menselijk zoeken

A en B zijn meer brute-force, C is meer “knowledge-based” .

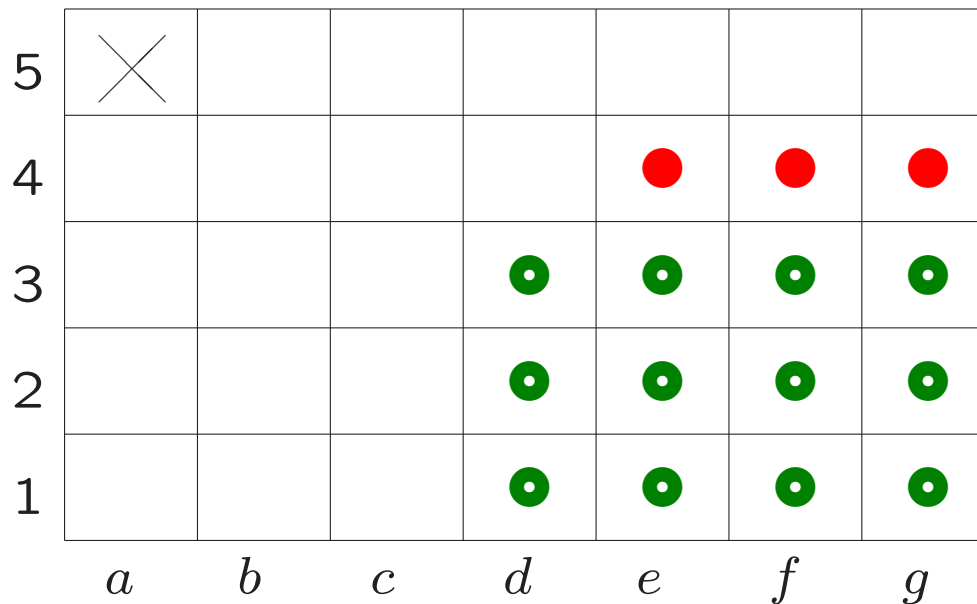
Er worden soms drie soorten oplossingen van spellen onderscheiden:

**ultra-zwak** de speltheoretische waarde van de beginstand is bekend: “je kunt vier-op-een-rij winnen”

**zwak** idem, en een optimale strategie is bekend (begin in middelste kolom, . . . , zie later)

**sterk** in elke legale positie is een optimale strategie bekend

Het spel **Chomp** wordt gespeeld met een rechthoekige reep chocola, waar de spelers om de beurt een stuk rechtsonder afhappen (een blokje en alles hier onder/rechts van). Wie het (vergiftigde) blokje linksboven eet, heeft verloren.



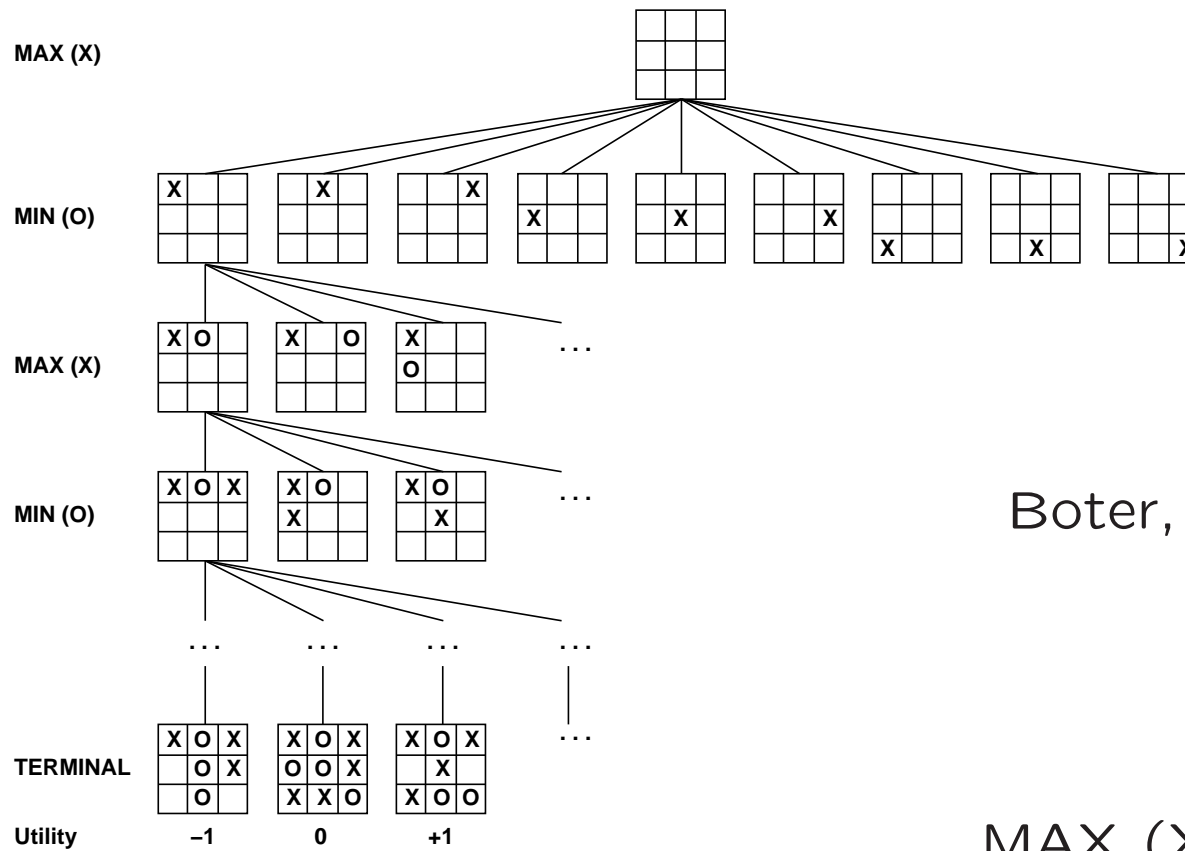
eerste hap: *d3*

tweede hap: *e4*

Bewering: de beginnende speler bij Chomp kan altijd winnen! Immers, als je door het blokje rechtsonder te nemen kunt winnen is het goed. Als dat niet zo is, heeft de tegenstander blijkbaar een voor hem winnende “tegen-hap”. Die kun je dan zelf als eerste doen, en dus daarmee winnen! Dit argument heet **strategy stealing**.

Kortom: het spel Chomp is ultra-zwak opgelost, de echte winnende zet weten we niet ...

Voor bijvoorbeeld  $2 \times 2$  en  $2 \times 3$  Chomp “wint” het blokje rechtsonder (algemener voor vierkanten: neem het vakje rechts onder het vergiftigde); bij  $3 \times 4$  Chomp het blokje uit de middelste rij, derde kolom.



Boter, kaas en eieren

twee spelers:

MAX (X) en MIN (O)

5478 toestanden

Boter, kaas en eieren is een voorbeeld van een tweepersoons deterministisch nulsom-spel met volledige informatie, waarbij de spelers om de beurt een “legale zet” doen. MAX moet een **strategie** vinden die tot een winnende eindtoestand leidt, ongeacht wat MIN doet. De strategie moet elke mogelijke zet van MIN correct beantwoorden.

Een **utility-functie** (= payoff-functie) geeft de waarde van eindtoestanden. Hier is dat:  $-1/0/1$ ; bij backgammon is dat:  $-192 \dots +192$ .

Uit symmetrie-overwegingen kunnen veel toestanden worden wegbezuinigd.

**Maxi** en **Mini** spelen het volgende eenvoudige spel: **Maxi** wijst eerst een horizontale rij aan, en daarna kiest **Mini** een verticale kolom.

3	12	8
2	4	6
14	5	2

Bijvoorbeeld: **Maxi** kiest rij 3, daarna kiest **Mini** kolom 2; dat levert einduitslag 5.

**Maxi** wil graag een zo groot mogelijk getal, **Mini** juist een zo klein mogelijk getal.

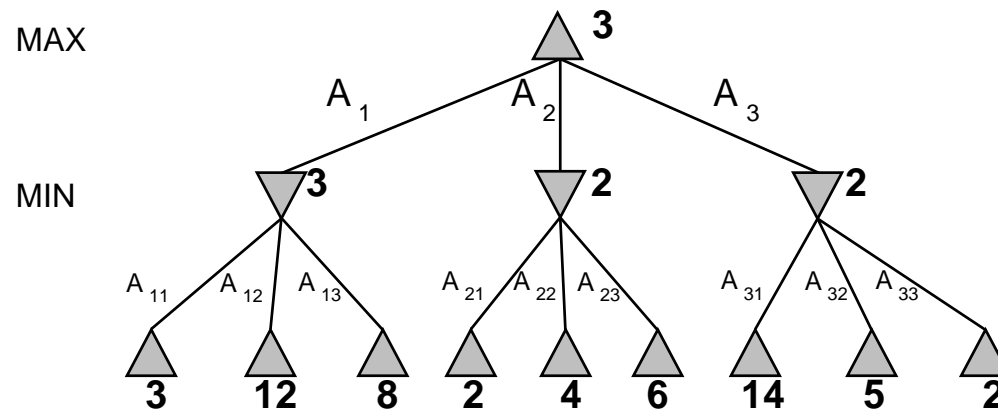
Hoe analyseren we dit spel?

Als **Maxi** rij 1 kiest, kiest **Mini** kolom 1 (levert 3); als **Maxi** rij 2 kiest, kiest **Mini** kolom 1 (levert 2); als **Maxi** rij 3 kiest, kiest **Mini** kolom 3 (levert 2). Dus kiest **Maxi** rij 1!

3	12	8
2	?	?
14	5	2

Nu merken we op dat de analyse (het **minimax-algoritme**) hetzelfde verloopt als we niet eens weten wat onder de twee vraagtekens zit. Het  **$\alpha$ - $\beta$ -algoritme** onthoudt als het ware de beste en slechtste mogelijkheden, en kijkt niet verder als dat toch nergens meer toe kan leiden (zie verderop).

In boomvorm:



Het **minimax-algoritme** is “recursief”: neem in bladeren de evaluatie-functie, in MAX-knopen het maximum van de kinderen, in MIN-knopen het minimum van de kinderen. MAX- en MIN-knopen wisselen elkaar af.

Bovenstaande boom is **één zet** (= move) diep, oftewel **twee ply**.

```
function MaxWaarde(toestand)  
  if eindtoestand then return Utility(toestand)  
  waarde  $\leftarrow -\infty$   
  for s in Opvolgers(toestand) do  
    waarde  $\leftarrow \max(\textit{waarde}, \textit{MinWaarde}(s))$   
  return waarde
```

```
function MinWaarde(toestand)  
  if eindtoestand then return Utility(toestand)  
  waarde  $\leftarrow +\infty$   
  for s in Opvolgers(toestand) do  
    waarde  $\leftarrow \min(\textit{waarde}, \textit{MaxWaarde}(s))$   
  return waarde
```

**Compleet:** als de boom eindig is (bij schaken dankzij speciale regels)

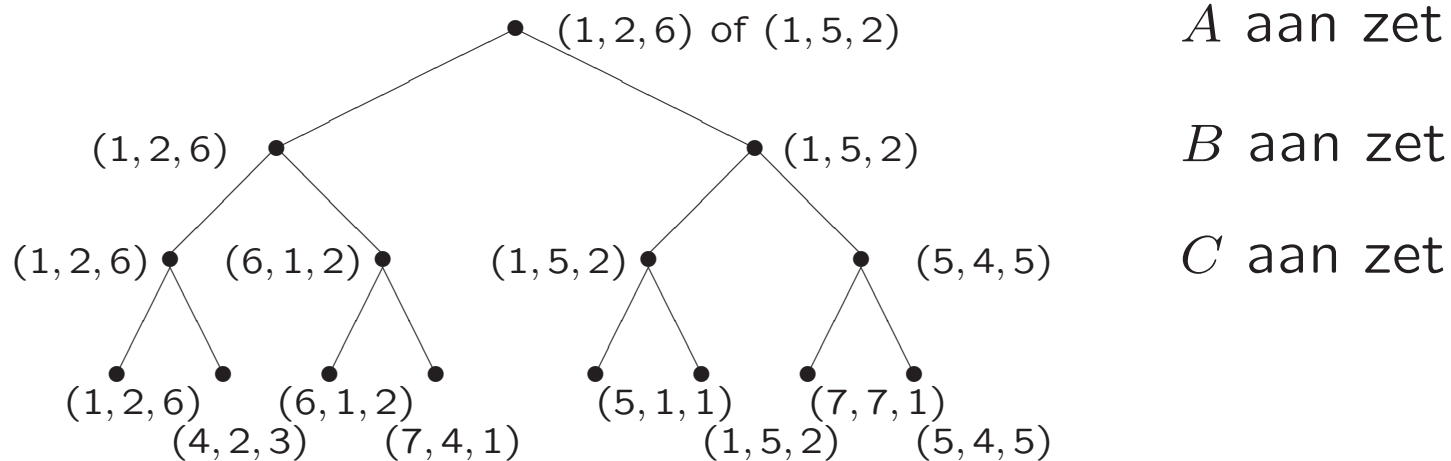
**Optimaal:** tegen een “optimale” tegenstander

**Tijdscomplexiteit:**  $O(b^m)$  ( $b$  is vertakkingsgraad,  $m$  diepte van de boom)

**Ruimtecomplexiteit:**  $O(bm)$  (bij depth-first exploratie)

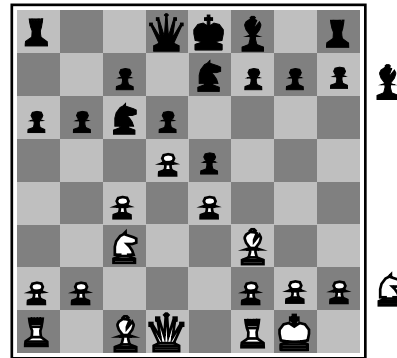
Bij schaken:  $b \approx 35$ ,  $m \approx 100$ ; een exacte oplossing is dus heeeeeeeeeel ver weg.

Met drie spelers ( $A$ ,  $B$  en  $C$ ) heb je per knoop drie evaluatiewaardes ( $A$  wil de eerste zo hoog mogelijk, etcetera):



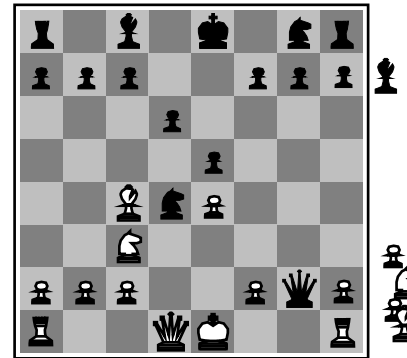
Voor twee spelers is één getal voldoende (bij een nulsomspel is de som van de twee waarden altijd gelijk).

En hoe zit het bij Diplomacy?



Black to move

White slightly better



White to move

Black winning

Voor schaken is de evaluatie-functie meestal een gewogen som van allerlei kenmerken (“features”): 9 maal (aantal witte dames – aantal zwarte dames) + ...

De functie moet eenvoudig te berekenen zijn, de kans op winnen aangeven, en kloppen met de utility-functie op eindtoestanden.

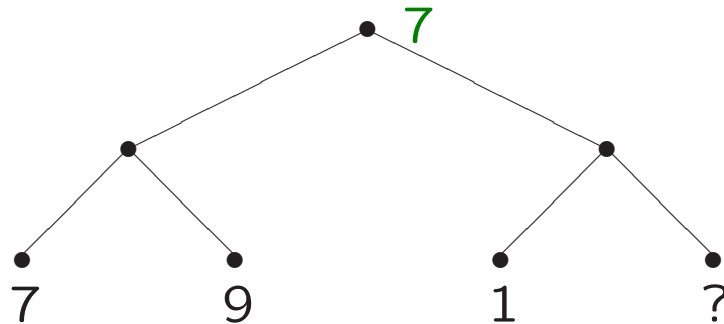
Bij **cut-off search** vervang je in het minimax-algoritme de test op eindtoestanden door een “cut-off test” — en een evaluatie-functie aldaar.

Bij schaken, met  $b = 35$  en  $b^m \approx 10^6$ , krijg je  $m = 4$ . En 4-ply vooruit kijken is hopeloos ( $\approx$  amateur). Men denkt: 8-ply  $\approx$  PC of schaakmeester, en 12-ply is wereldtop-nivo (computer of mens).

Je hebt te maken met:

- **quiescence** in een onrustige periode moet je langer doorrekenen;
- **horizon-probleem** soms wordt een noodzakelijke slechte zet uit beeld geduwd door extra (minder slechte) zetten.

Het basisidee van het  $\alpha$ - $\beta$ -algoritme is het volgende: om de minimax-waarde in de wortel van onderstaande boom te berekenen is de waarde rechts onderin niet van belang — en die subboom kun je **prunen** (snoeien). Immers, het linker kind van de wortel is 7, en het rechterkind moet dus hoger dan 7 zijn wil het nog invloed uitoefenen. Nu is het (dankzij de 1) hoogstens 1, en zijn we klaar: waarde 7.



MAX aan zet

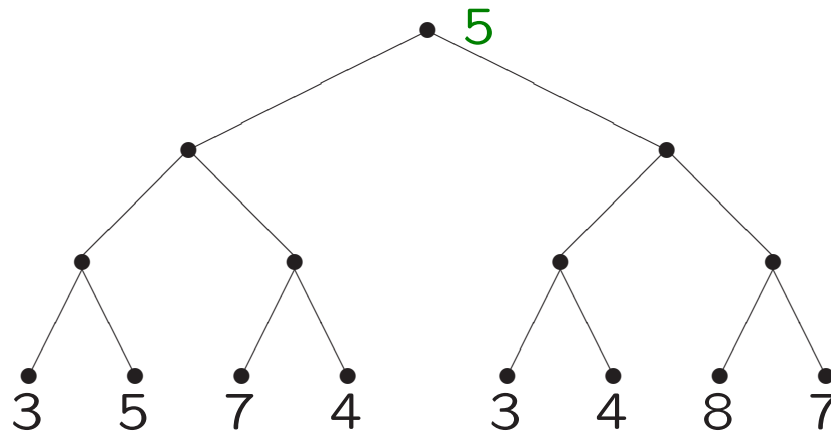
MIN aan zet

```
function MaxWaarde(toestand,  $\alpha$ ,  $\beta$ )  
  if toestand is eindtoestand then return Utility(toestand)  
    (of cut-off test, en gebruik evaluatie-functie)  
  for s in Opvolgers(toestand) do  
     $\alpha \leftarrow \max(\alpha, \text{MinWaarde}(s, \alpha, \beta))$   
    if  $\alpha \geq \beta$  then return  $\beta$   
  return  $\alpha$ 
```

Analoog de functie *MinWaarde*, zie boek (voor een iets andere formulering). Normaal geldt  $\alpha < \beta$ ;  $\alpha$  geeft de beste waarde voor MAX op het huidige pad, en  $\beta$  voor MIN.

De variabelen  $\alpha$  en  $\beta$  zijn lokaal.

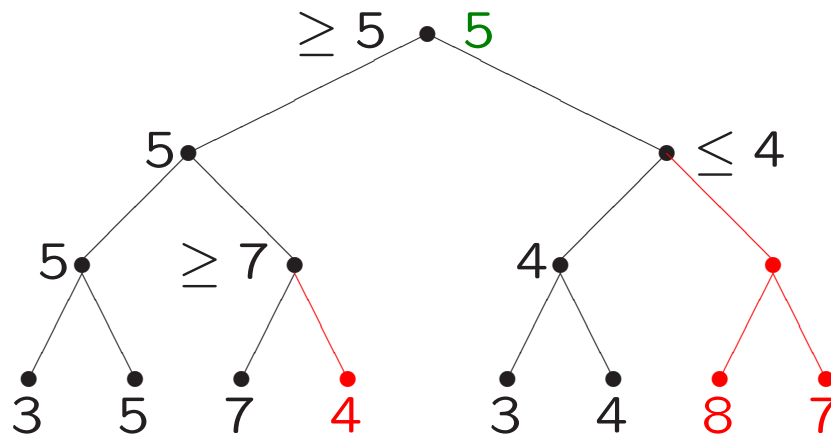
Buitenste aanroep: *MaxWaarde*(*huidigetoestand*,  $-\infty$ ,  $+\infty$ ).



MAX aan zet

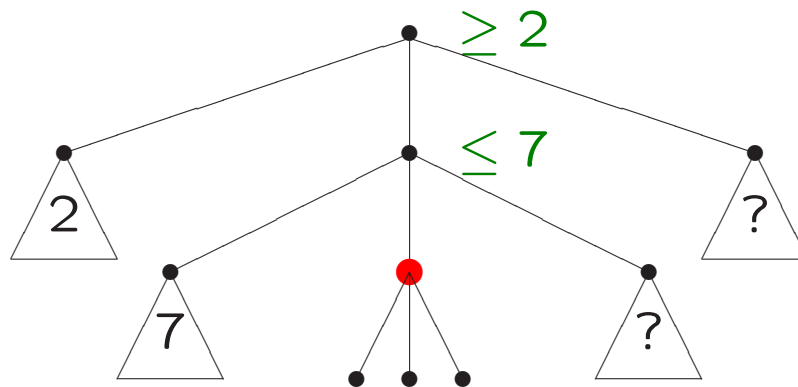
MIN aan zet

MAX aan zet



rood wordt

gepruned



MAX aan zet

MIN aan zet

MAX aan zet

Bij de MAX-knoop ● geldt dat  $\alpha = 2$  en  $\beta = 7$ . Op het pad naar die knoop kan MAX al 2 afdwingen, en MIN 7. Als een kind van ● waarde 8 heeft, hoeven de volgende kinderen niet meer bekeken te worden, en krijgt ● waarde 7 (de  $\beta$ ).

$\alpha$ – $\beta$ -pruning levert exact hetzelfde eindresultaat in de wortel als “gewoon” minimax.

De effectiviteit hangt sterk af van de volgorde waarin de kinderen (zetten) bekeken worden.

Met “perfecte ordening” bereik je tijdscomplexiteit  $O(b^{m/2})$ , dus je kunt effectief de zoekdiepte verdubbelen.

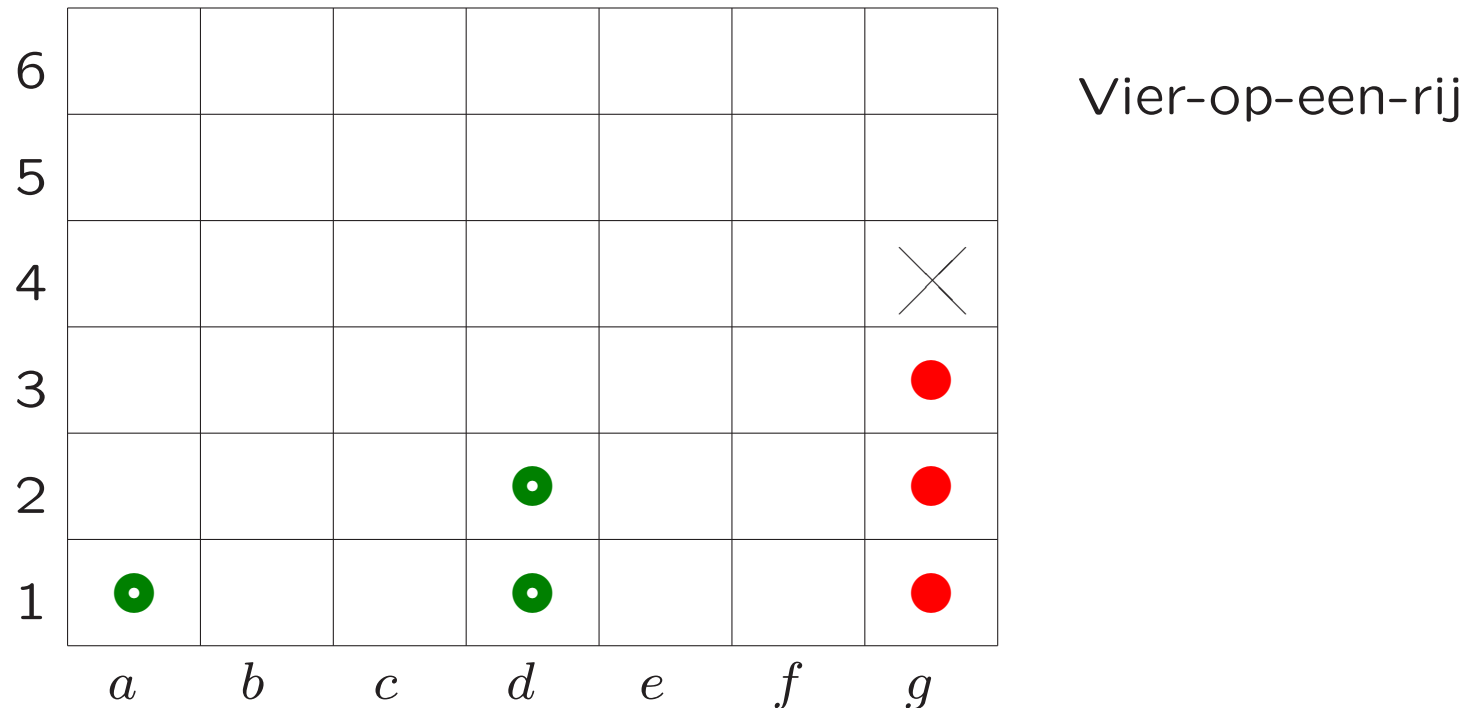
Het is dus van belang heuristieken te hebben om je zetten te ordenen. Enkele voorbeelden:

**null-move** bekijk eerst voor de tegenstander goede zetten (sla je eigen zet in gedachten even over)

**killer** als een zet ergens een snoeiing teweeg brengt, doet hij dat elders wellicht ook

**conspiracy-number**  $\approx$  aantal kinderen dat van waarde moet veranderen (samenzweren) om ouder van waarde te laten veranderen (voor stijgen MAX-knoop is maar één kind nodig, voor stijgen MIN-knoop zijn alle kinderen nodig)

**tabu-search** onthoud aantal (zeer) slechte zetten



Met **groen** aan de beurt, is *g4* zowel voor de null-move- als de killer-heuristiek de aangewezen zet.

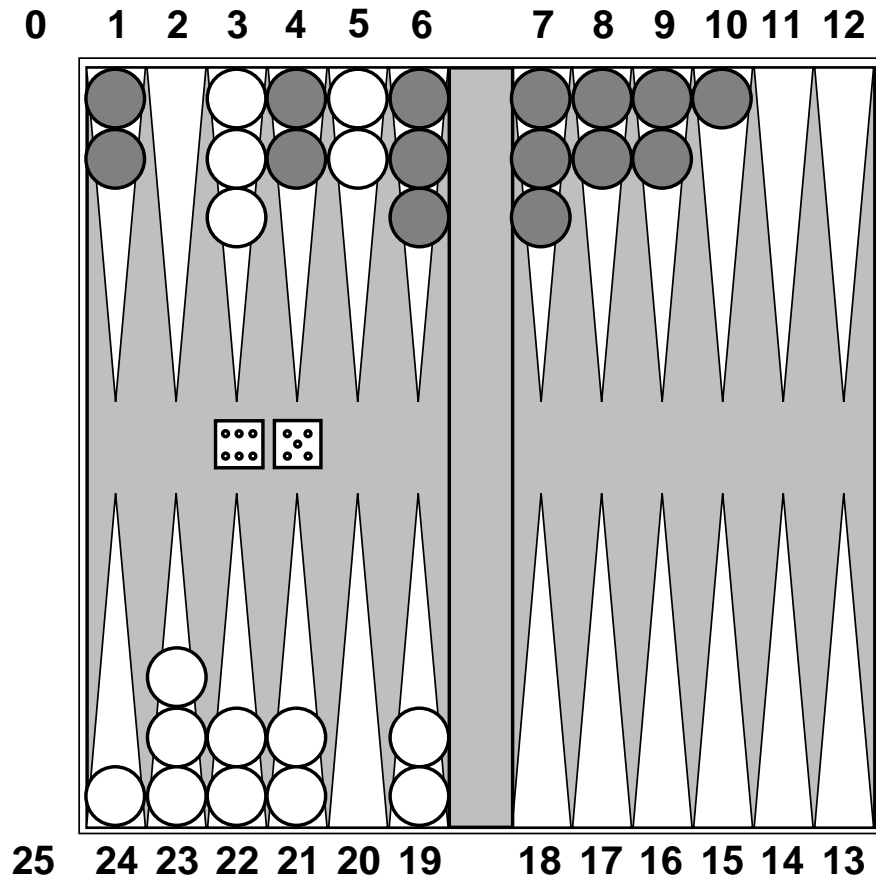
De voor **groen** winnende (begin)serie: *d1!* – *d2* – *d3!* – *d4* – *d5!* – *b1* – *b2* (een ! betekent: unieke winnende zet).



Babson task  
Yarosh, Tim Krabbé



retrograde analyse  
Smullyan

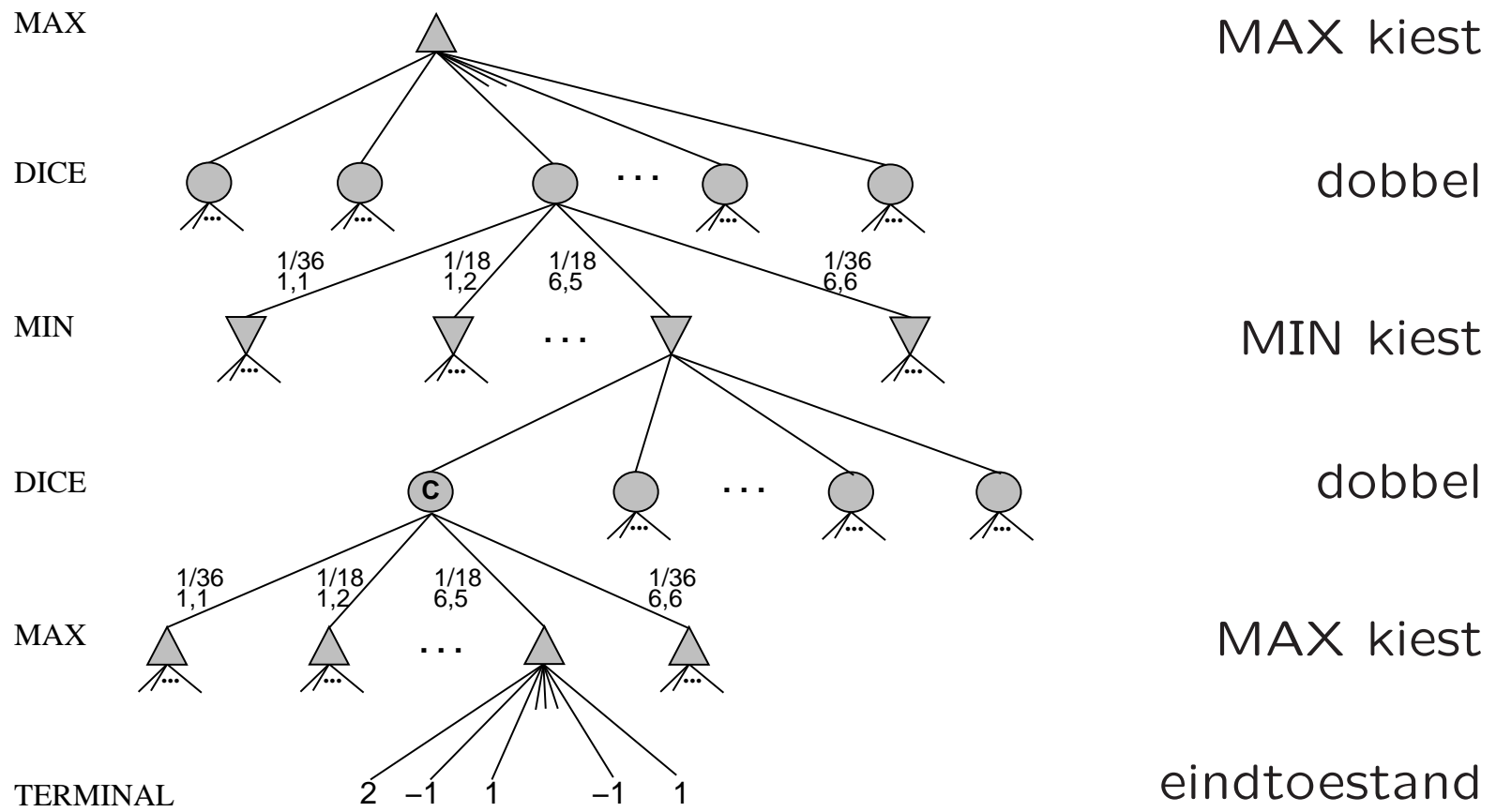


niet-deterministisch

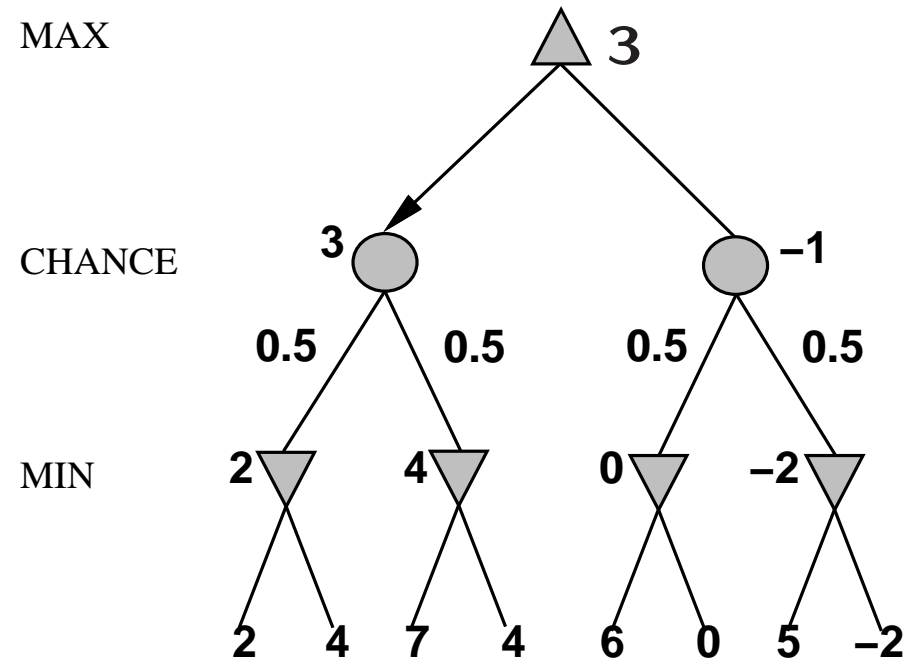
gooi 2 dobbelstenen

en kies toegestane zet

Bij backgammon krijgen we een spelboom als:



Een eenvoudig voorbeeld van een spel met een eerlijke munt:



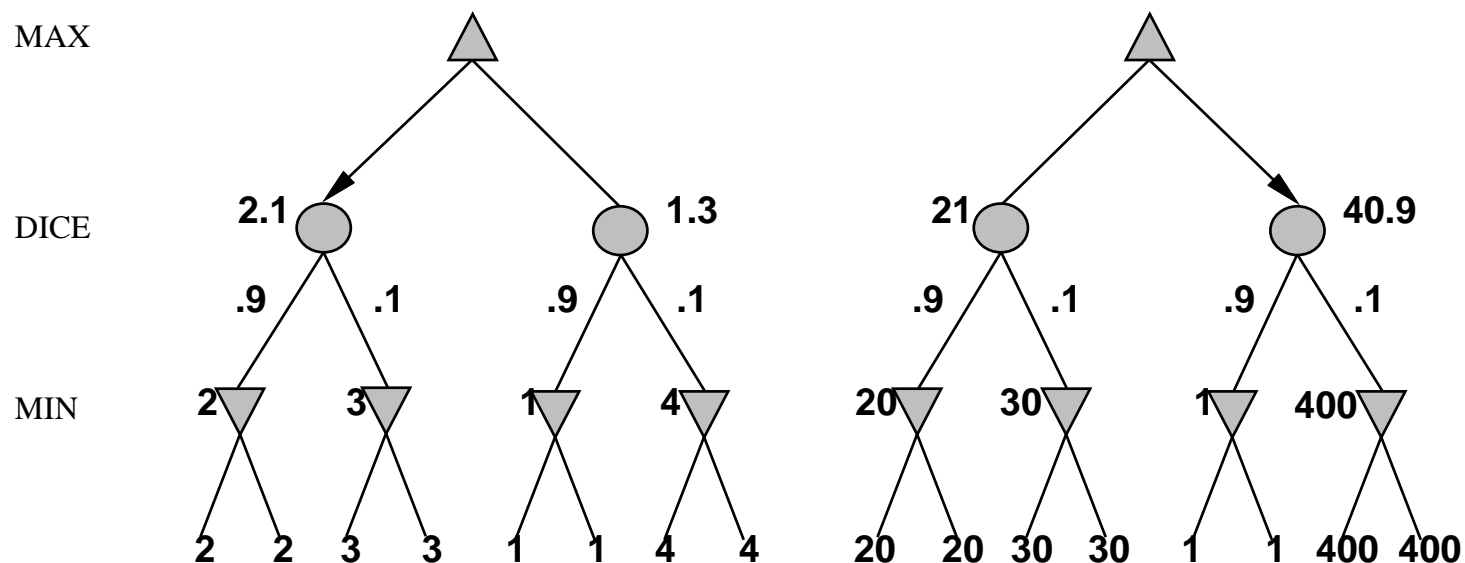
In het algemeen krijg je in een kansknoop  $n$  voor de **expecti-minimax**-waarde  $ExpectiMinimax(n)$  een formule als:

$$\sum_{s \in Opgvolgers(n)} P(s) \cdot ExpectiMinimax(s),$$

waarbij  $P(s)$  de kans op  $s$  is.

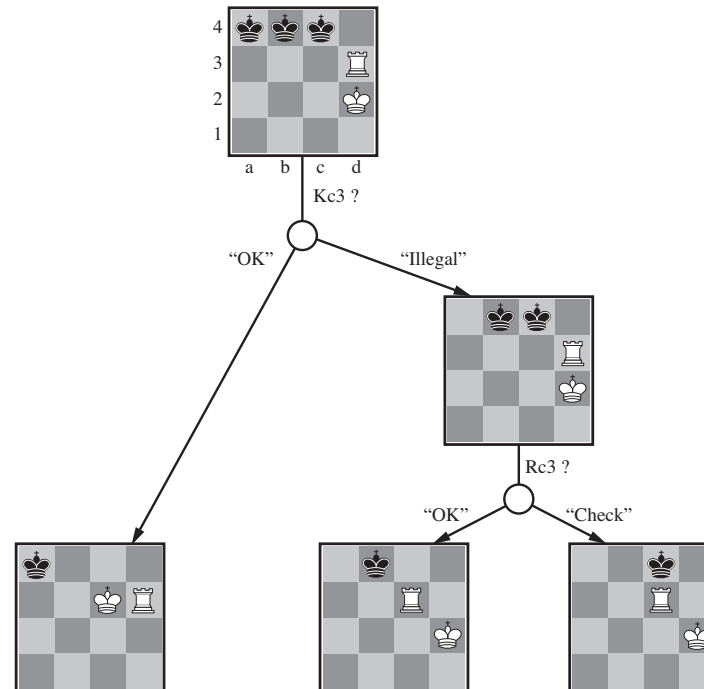
Soms is het gemiddelde “beter” dan de “exacte” minimax-waarde. Stel je voor dat je (= MAX) moet kiezen uit een MIN-knoop met kinderen 99, 1000, 1000 en 1000, en een MIN-knoop met kinderen 100, 101, 102 en 103 ...

In geval van kansknopen moet je beter op de evaluatie-functie letten!



Links, met bladeren 1, 2, 3, 4, is de “linker” zet het beste, rechts, met bladeren 1, 20, 30, 400, de “rechter” zet. De evaluatie-functie moet proportioneel zijn met de “winst” .

Een deels observeerbaar spel is **Kriegspiel**: schaak waarbij de speler alleen de eigen stukken ziet, en een scheidsrechter zetten beoordeelt (“OK”, “Verboden”, “Schaak”, ...).



Let ook op de “belief states”.

Het huiswerk voor de volgende keer (dinsdag 22 maart 2011): lees **Hoofdstuk 18.7**, p. 727–737 van [RN], over Neurale netwerken door. Kijk ook eens naar de vragen bij dit hoofdstuk.

Maak de tweede opgave: **Robocode** af; deadline 22 maart 2011.

**Werkcollege** met name over sommen: maandag 21 maart 2011, 13.45–15.30 uur, zaal **409**, zie:

`www.liacs.nl/home/kosters/AI/opgaven2.pdf`