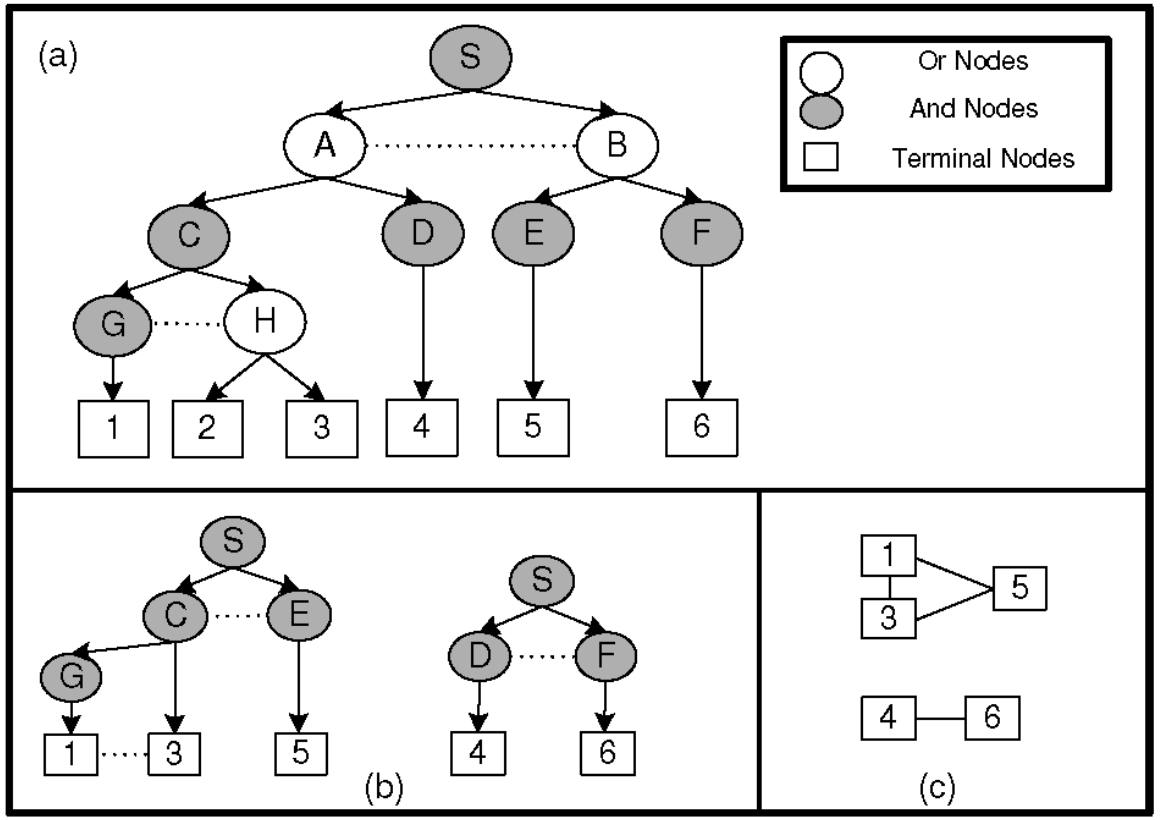
Спротивставено пребарување

Мултиагенти кои се натпреваруваат меѓусебно

- Соработка (collaboration) наспроти заемен натпревар (competitiveness)
- Игри меѓу двајца или повеќе натпреварувачи
- Целта е спротивставена: секој од играчите го игра својот најсилен потег, кој за противникот е најнеповолен

Помошна алатка: И/ИЛИ графови

- Решението се темели на декомпозиција на проблемот во помали потпроблеми



Основниот услов во еден И/ИЛИ граф

- Ако патот стигнал до ИЛИ јазел, тогаш може да продолжи по БИЛО кое од неговите деца
- Ако патот стигнал до И јазел, тогаш мора да помине низ сите негови деца
- Практична препорака: на една длабочина да има само И или само ИЛИ јазли

И/ИЛИ графовите се насочени кон целта

- Целта во ИЛИ јазелот е да дојде до точната цел во барем еден од јазлите кои потекнуваат од него.
- Целта во И јазелот е да дојде до точната цел во сите јазли кои потекнуваат од него.
- Притоа, се проверува дали листовите ја исполнуваат целта и тогаш означувањето се врши од долу кон врвот (bottom-up)

Како се применуваат овие дрва за игрите

- Заемно спротивставени двајца противници:
  - сите сопствени јазли се ИЛИ јазли
  - сите јазли на противникот се И јазли
- Заемно спротивставени повеќе противници:
  - сите сопствени јазли се ИЛИ јазли
  - сите јазли на противниците се И јазли
  - за секој противник се резервира по една длабочина од дрвото

Играње карти

- Кога се играат карти, еден потег најчесто има три или четири потега
- Сопствените јазли се ИЛИ јазли, бидејќи секој може да одбере еден потег
- Јазлите на противниците се И јазли, бидејќи за секој од нив треба да се даде одговор
- Целта на секој од играчите е да го одигра својот најсилен потег
- Најсилниот потег за еден играч е најнеповолниот потег за неговите противници
- Minimax, Neg(a)max

Кои типови игри се решаваат т.е. се сведуваат на И/ИЛИ дрва

- Х-0: b = 9, m = 9, вк = 9!
- шах $b_{ср} = 35$, $m_{ср} = 2*50$, вк = 35^{100} (околу 10^{154})
- реверси b < 15, m = 60, вк << 15^{60}
- кокарача (connect 4, four in a row, 4 wins, vier gewinnt) b < 7, m = 42, вк << 7^{42}
- дама, го, игри со карти, јамб, ...
- Кај игрите за двајца, еден потег има два полупотега
- Кај игрите меѓу n противници, потегот има n меѓупотези

Која е суштината на Minimax?

- Применлива е за детерминистичките игри
- Секој од противниците го одбира потегот што за него има максимална вредност
- Се базира врз функција на проценка f (или Eval)
- $f = Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$

  w_i: тежинска функција со која им се дава јачина на избраните критериуми за вреднување на потегот s

  f_i: критериум врз основа на кој се проценува еден потег
- Функцијата може да биде:
  - Статичка: тежините се постојани
  - Динамичка: тежините се менуваат

Алгоритамот Minimax

- За игри со двајца играчи
- Претпоставува дека противникот игра совесно и еднакво добро како нас.

Za $s \in$ Sledni_potezi(n)

Minimax(n)=

$$\begin{cases} \text{Utility(n), ако n е терминална состојба} \\ \text{Max Minimax(s), ако n е MAX јазел} \\ \text{Min Minimax(s), ако n е MIN јазел} \end{cases}$$

Критериуми за вреднување

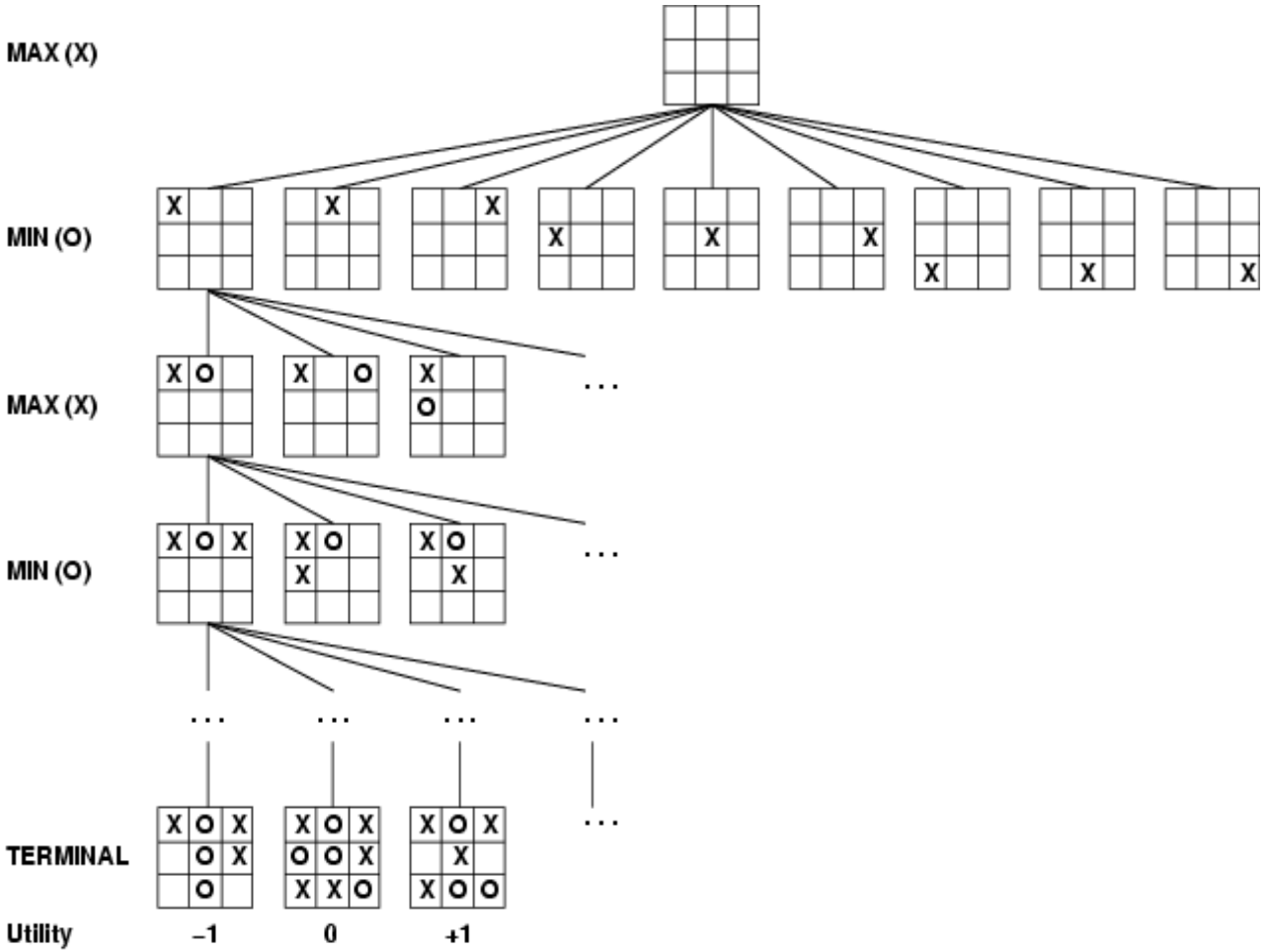
- X-0:
  - дали се спречуваат три исти вредности на противникот по хоризонтала, вертикала, дијагонала
  - дали се овозможуваат три исти сопствени вредности по хоризонтала, вертикала, дијагонала
  - колкава е мобилноста (сопствена, на противникот)

- Реверси:
  - зафаќање агол
  - колку пулови на противникот ќе се превртат
  - колкава е мобилноста (сопствена, на противникот)
  - дали ќе се овозможи противникот да нема право на сопствен потег
  - ...

Вреднување на потезите

- На листот во кој победуваме ние, му се доделува вредноста +∞ (или 32767)
- На листот во кој победува противникот, му се доделува вредноста -∞ (или -32768)
- На листот кој означува реми, му се доделува вредноста 0
- Означувањето се врши откако ќе се развие **целосното дрво**, од листовите кон врвот
- Ова е теоретски можно за некои игри. Кај таквите игри човекот не може да победи.
- Практично, кај другите поинтересни игри, дрвото е преголемо, па затоа, означувањето се базира на дел од дрвото.

Еве што значи тоа за играта X-0



Пример за игра меѓу тројца играчи

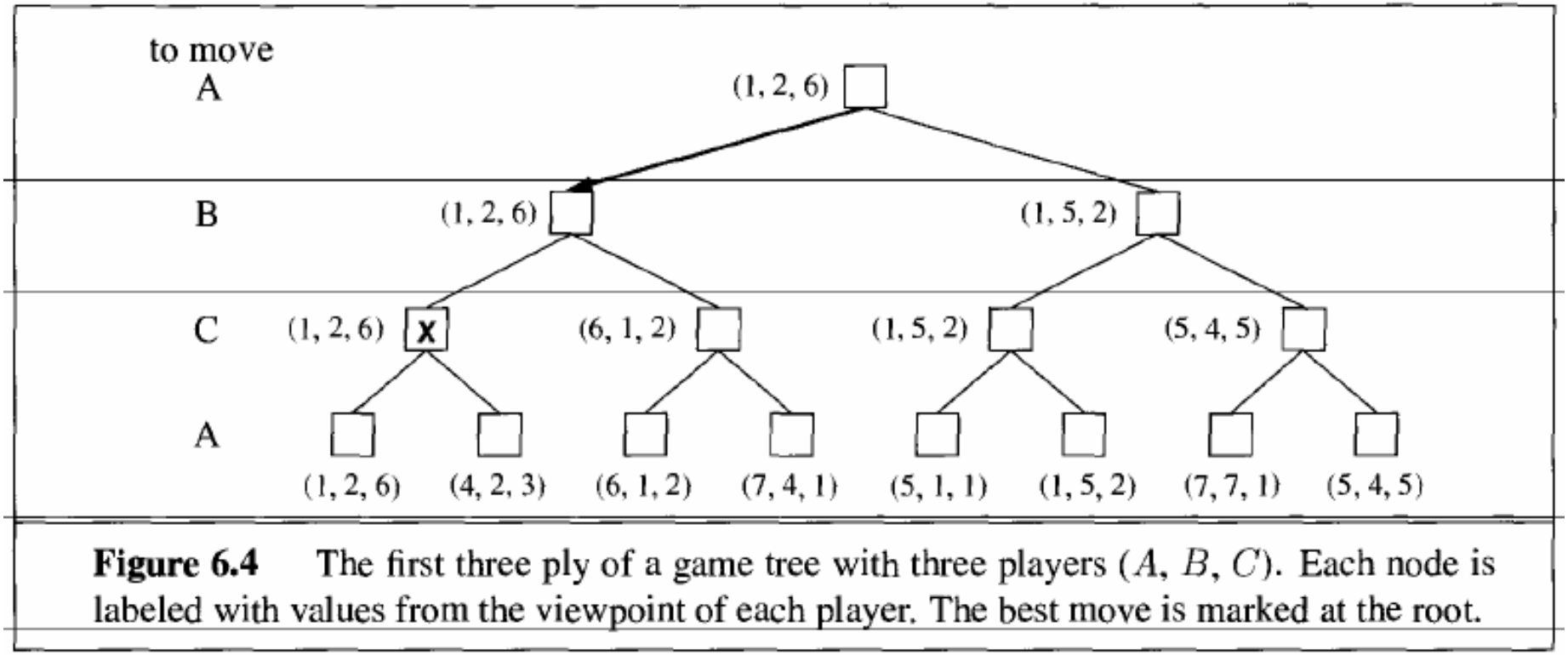
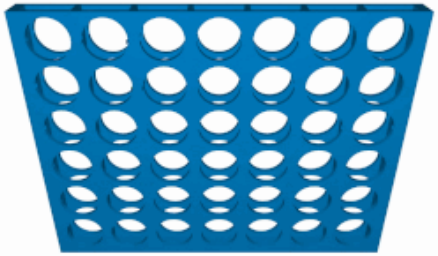


Figure 6.4 The first three ply of a game tree with three players (A, B, C). Each node is labeled with values from the viewpoint of each player. The best move is marked at the root.

- Utility функцијата враќа вектор
- Секој си бира максимум за себе
- Како да се моделираат сојузи?

Забелешка: Копија на една игра за тројца, Расел и Норвиг, глава 6, стр. 6

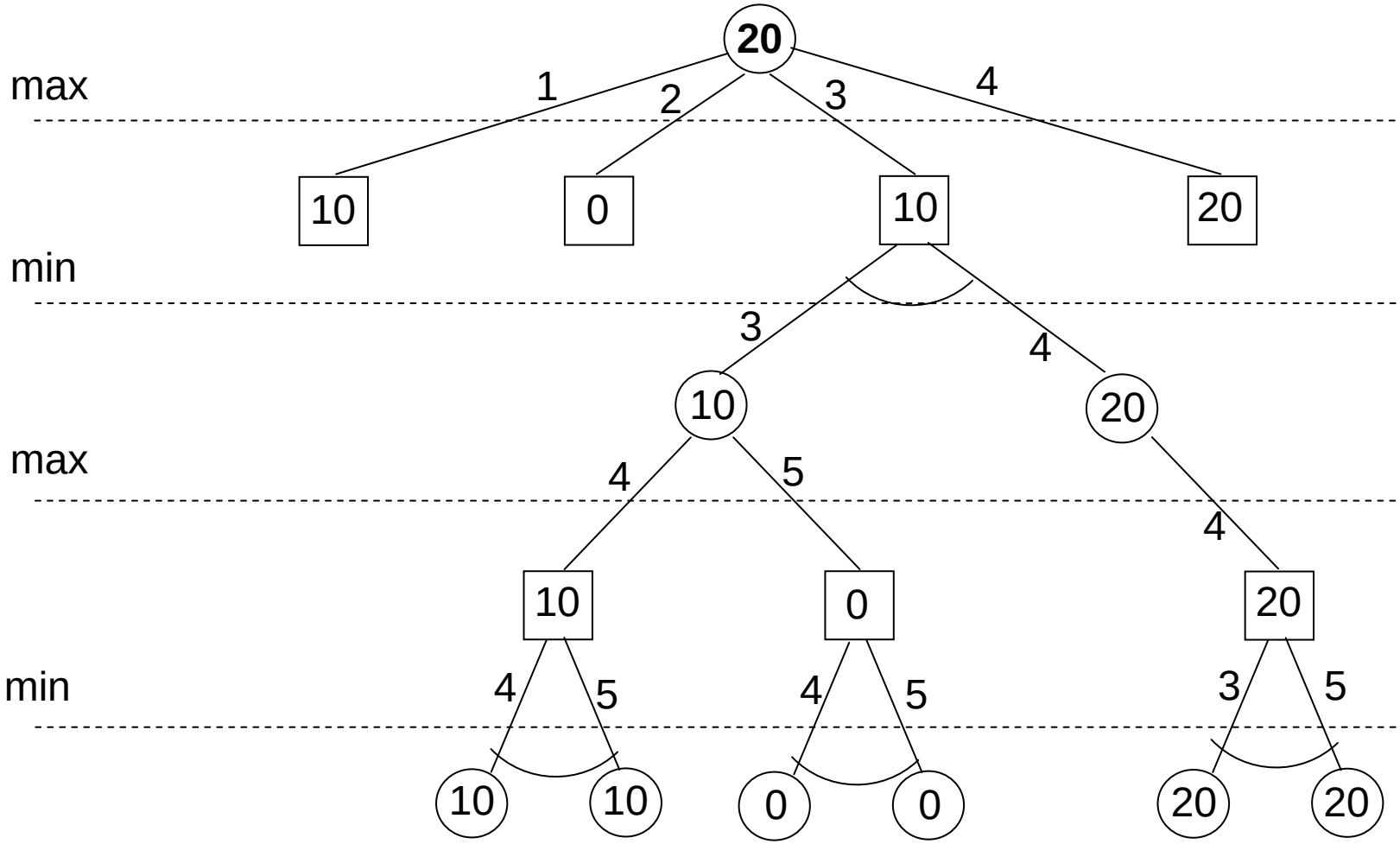
Minimax на дело: “4 добиваат”



- Два пула со иста боја:
  - по хоризонтала или по вертикала: 10 / -10 поени
  - по дијагонала: 20 / -20 поени
- Три пула со иста боја:
  - по хоризонтала или по вертикала: 100 / -100 поени
  - по дијагонала: 200 / -200 поени
- Четири сопствени пулови: ∞
- Четири противнички пулови: -∞

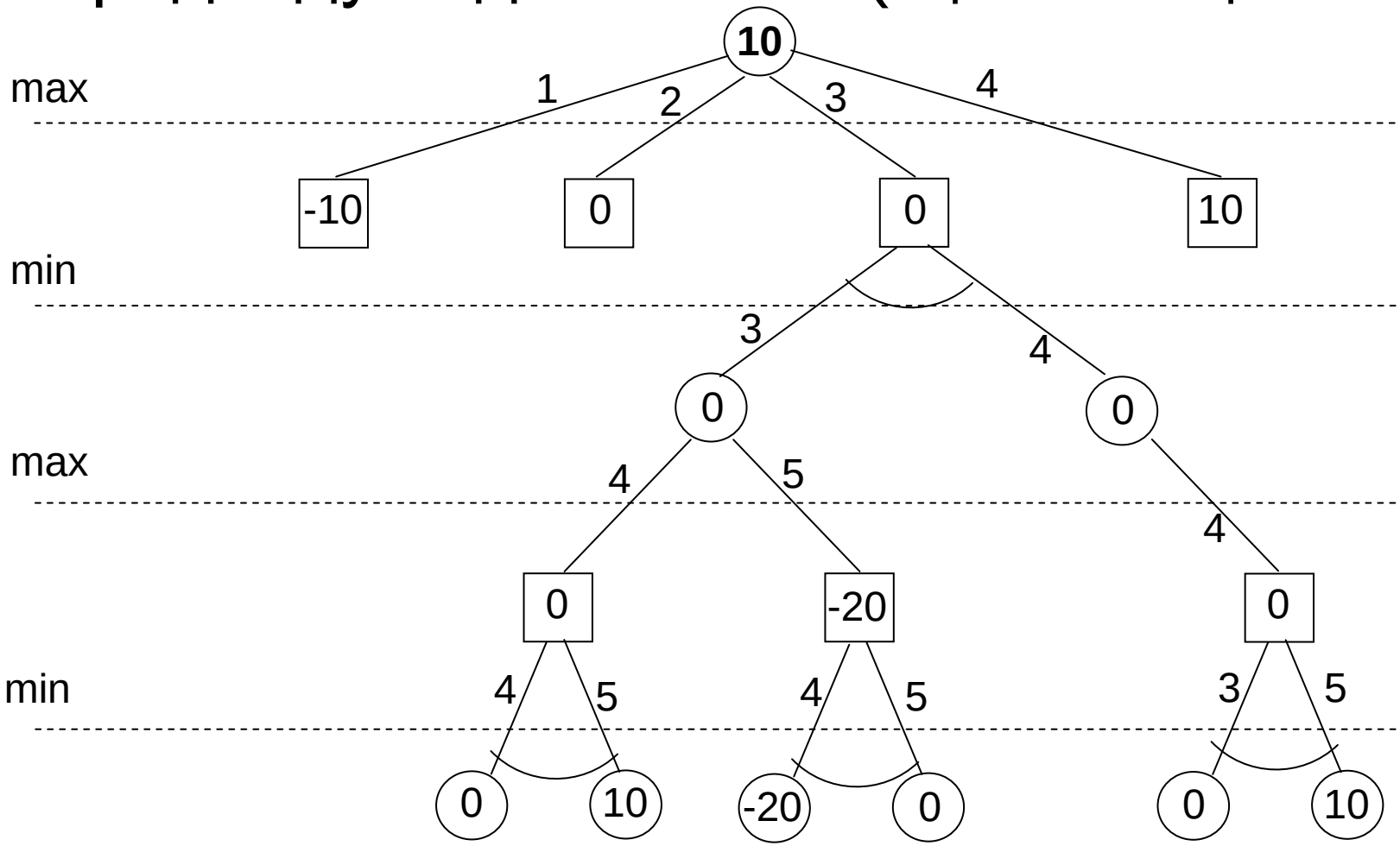
1. Функцијата се базира **само** на сопствените пулови
2. Функцијата ги зема предвид и пуловите на противникот

За кој потег би се одлучил првиот ако предвидува два потега (вер. 1: само сопствените)?



1. Функцијата се базира **само** на сопствените пулови

За кој потег би се одлучил првиот ако предвидува два потега (вер. 2. и на противникот)?



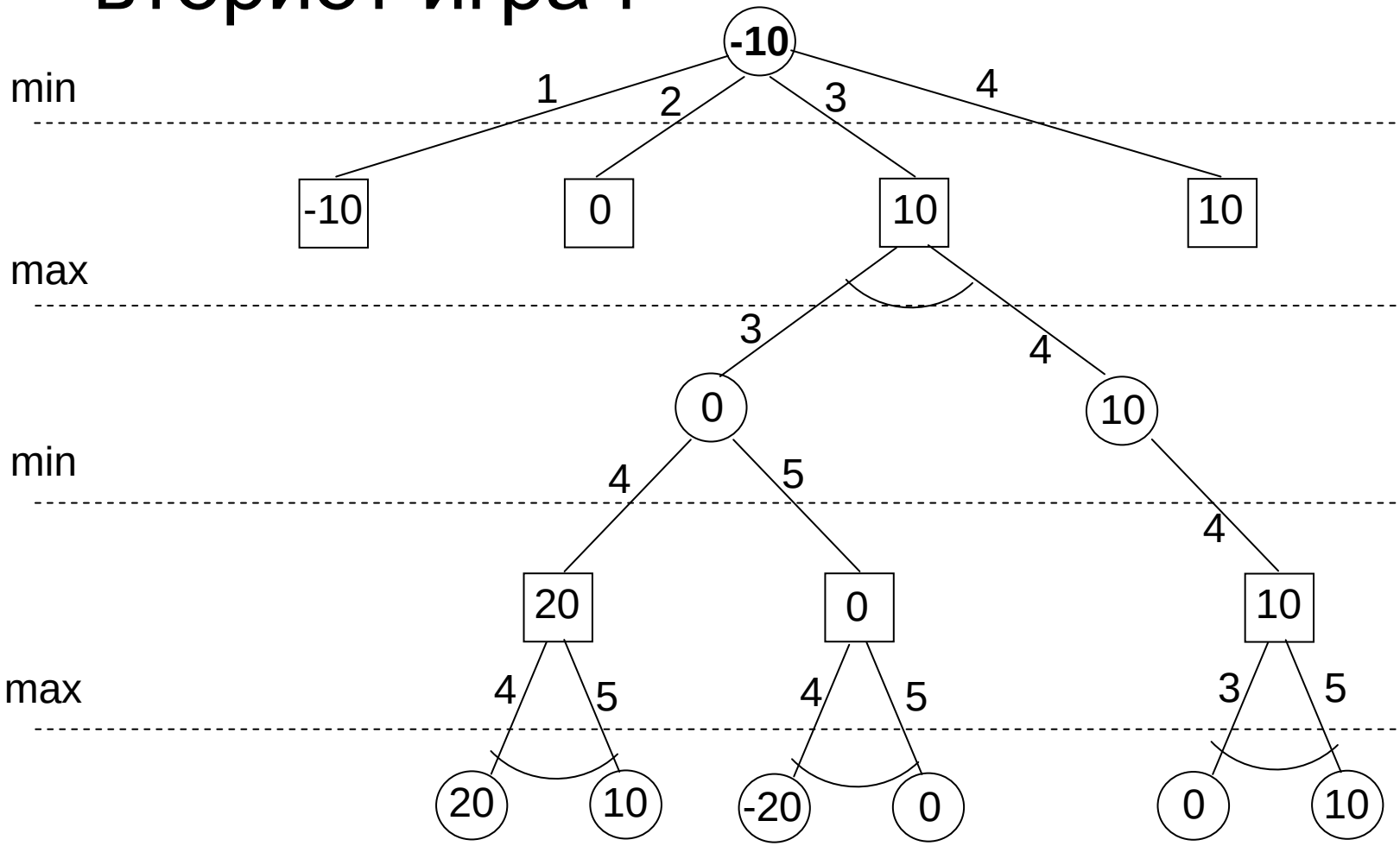
2. Функцијата ги зема предвид и пуловите на противникот

Што значи тоа за неговиот противник?

- Сето она што за првиот е min, за вториот е max и обратно

- Ако во играта се повеќе противници, тогаш се минимизираат противничките потези во два или повеќе делумни потези

Дрвото на “4 добиваат” од аспект на вториот играч



Minimax алгоритам, според Расел и Норвиг

```
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
```

Neg(a)max

- Варијанта на Minimax
- Вредноста на противникот е негативна вредност на вредноста на играчот
- Се базира врз правилото:

  min(a,b) = -max(-a,-b)
- Пребарувањето е олеснето, бидејќи се користи само еден критериум за избор на најповолниот потег

Својства на двата алгоритма

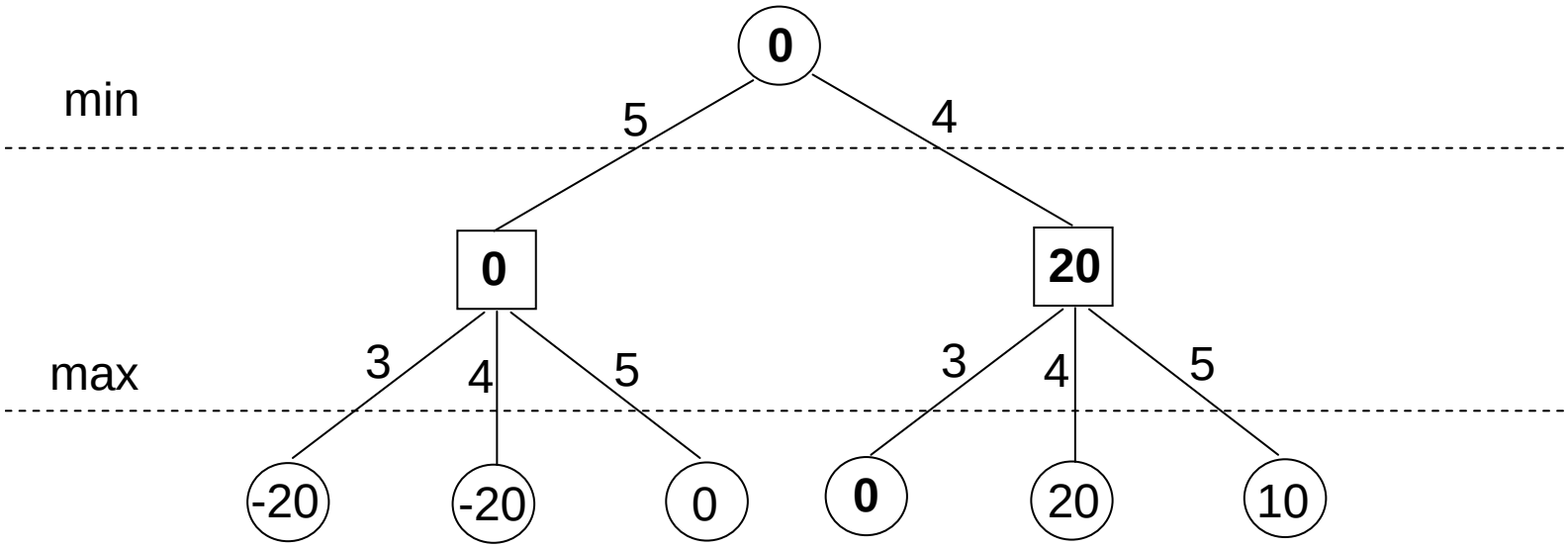
- Комплетни, под услов играта да има конечно дрво
- Оптимални, при што целта е да се обезбеди макар реми
- Временска сложеност (комплексност): $O(b^m)$
- Просторна сложеност (комплексност): O(bm)

- Огромната комплексност се решава со кастрење (pruning)

Алфа-бета кастрење (Alpha – beta pruning)

- Експоненцијалната временска комплексност не може да се елиминира, но
- може да се отстрани оној дел од дрвото што делува дека не влијае врз финалната одлука.
- Ако во дрвото не се проценети некои јазли и притоа веќе е најден јазел што во однос на Minimax критериумот има подобра вредност, тогаш тие нема да влијаат на финалната одлука, па според тоа може да се отстранат.

Што може да се поткастри тука?



Minmax(A) = min {max{-20,-20,0}, max{0,x,y}} = min {**0**, max{**0**,x,y}} = 0

Внимание: редоследот на потезите е променет. Зошто?

Зошто алфа-бета поткастрување

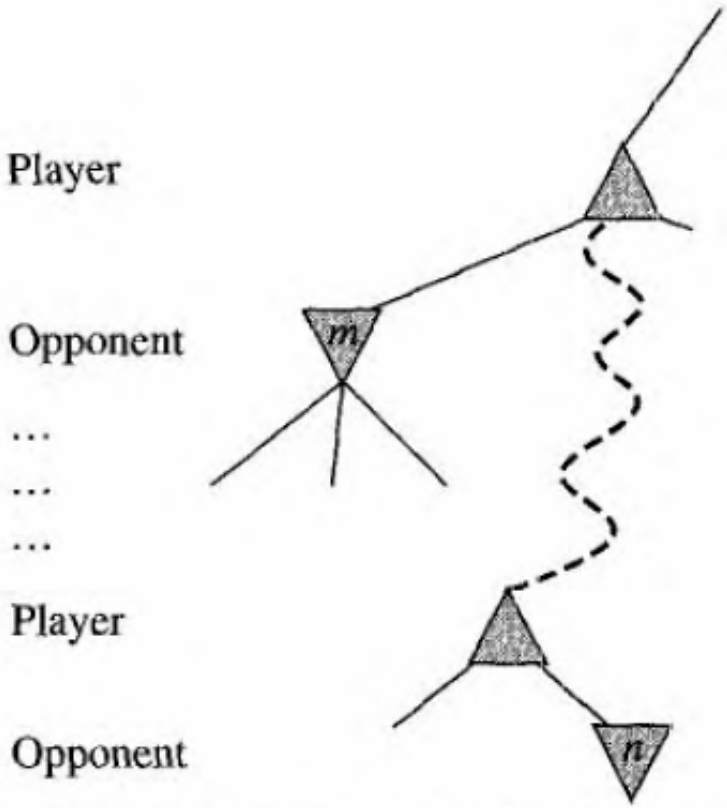


Figure 6.6 Alpha–beta pruning: the general case. If m is **better** than n for Player, we will never get to n in play.

α-β алгоритамот, според Расел и Норвиг

function ALPHA-BETA-SEARCH(*state*) **returns** *an action*
  inputs: *state*, current state in game
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE($state, \alpha, \beta$) **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

Min во α-β алгоритамот, според Расел и Норвиг

```
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
```

Што се во суштина α и β?

- α : најголемата вредност што е најдена кога се вреднува со Max
- β : најмалата вредност што е најдена кога се вреднува со Min
- Со нив се подредуваат деловите од дрвото што ќе се скастрат
- Зависи од редоследот по кој се разложуваат јазлите (имено затоа, во примерот, потезите беа обратно дадени)
- Временска комплексност
  - кај оптимално подредено дрво: $O(b^{m/2})$
  - кај дрво во кое јазлите се избираат по случаен избор: $O(b^{3m/4})$
- Просторна комплексност: длабочината на пребарувањето се удвојува, т.е изнесува d

Динамичка функција за вреднување на потезите

- Ако играта постојано се игра со иста стратегија, тогаш важи дека:

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

- Ако при играњето се процени дека некои од критериумите противникот ги игра почесто, а други поретко, тогаш во векторот на сите тежини се воведува промена, на пример:

  $\Delta W_t = c(v_{t+1} - v_t) * gradient(v_t, W)$

- Променливите тежини се резултат на **учење**.

Како се реализира учењето?

- Генетски алгоритми
- Невронски мрежи
- Автомати што учат
- ...

Успеси кај детерминистичките игри

- Chinook 1994: програма што игра дама (checkers); базирана на библиотека однапред генерирани завршници, 444 милијарди позиции
- Deep Blue 1997: базирана на прецизна проценка во која се развиваат до 40 потези, пребарува 200 милиони позиции во секунда
- Othello (Reversi): несовладлива за човекот ☹
- Go: несовладлива за компјутерот ☺

- Ограничување на современите компјутери: развива во просек 10000 чекори во секунда

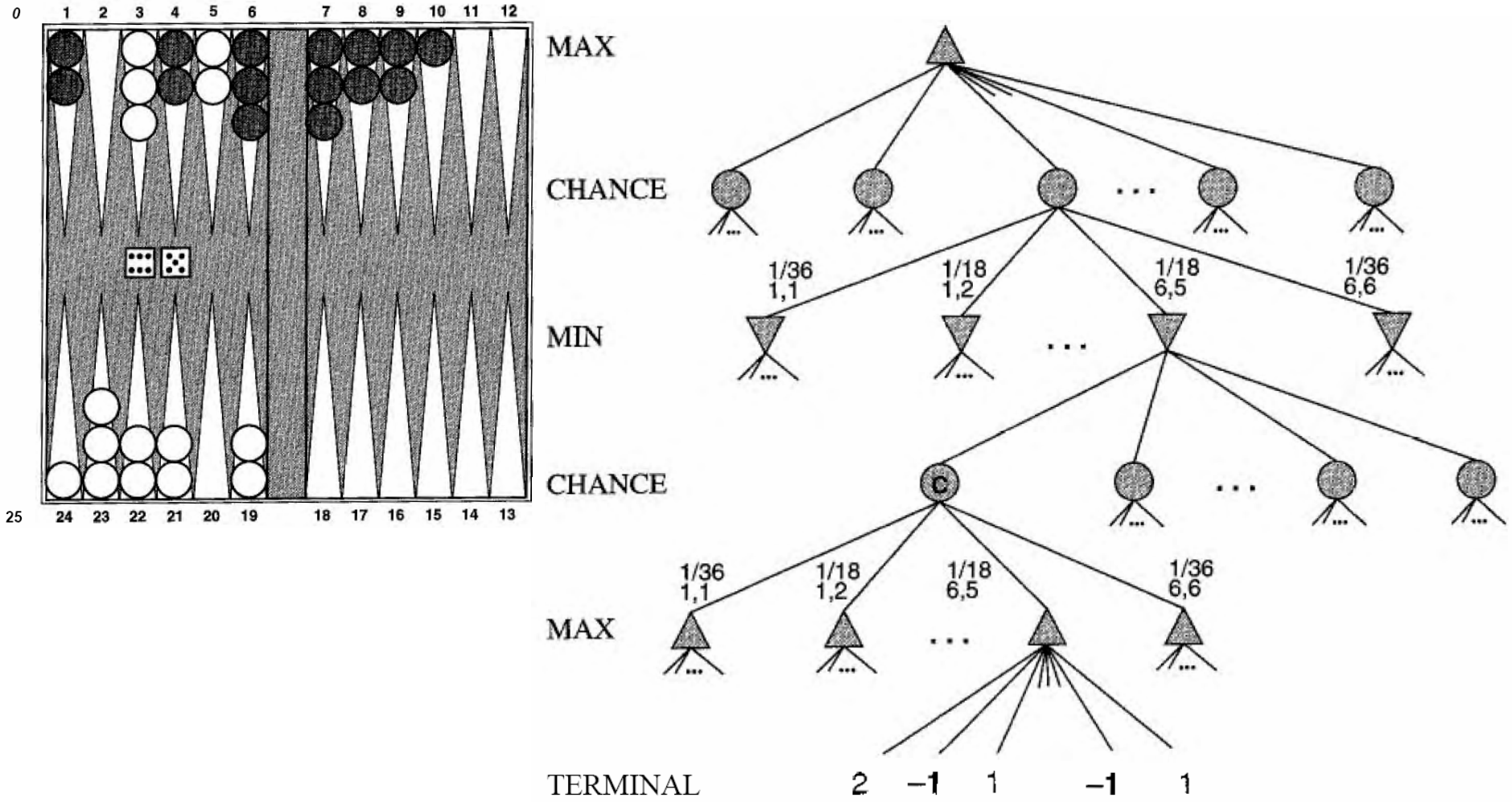
Како се играат игрите во кои има елемент на среќа?

- Соодветна на игри со коцки
- Стратегијата останува непроменета, но при дефиниција на функцијата на проценка, се воведува и факторот на среќа.
- Тоа значи дека меѓу јазлите Min и Max се вметнува елементот на несигурнсот, а потоа се одредува која е просечната очекувана вредност.

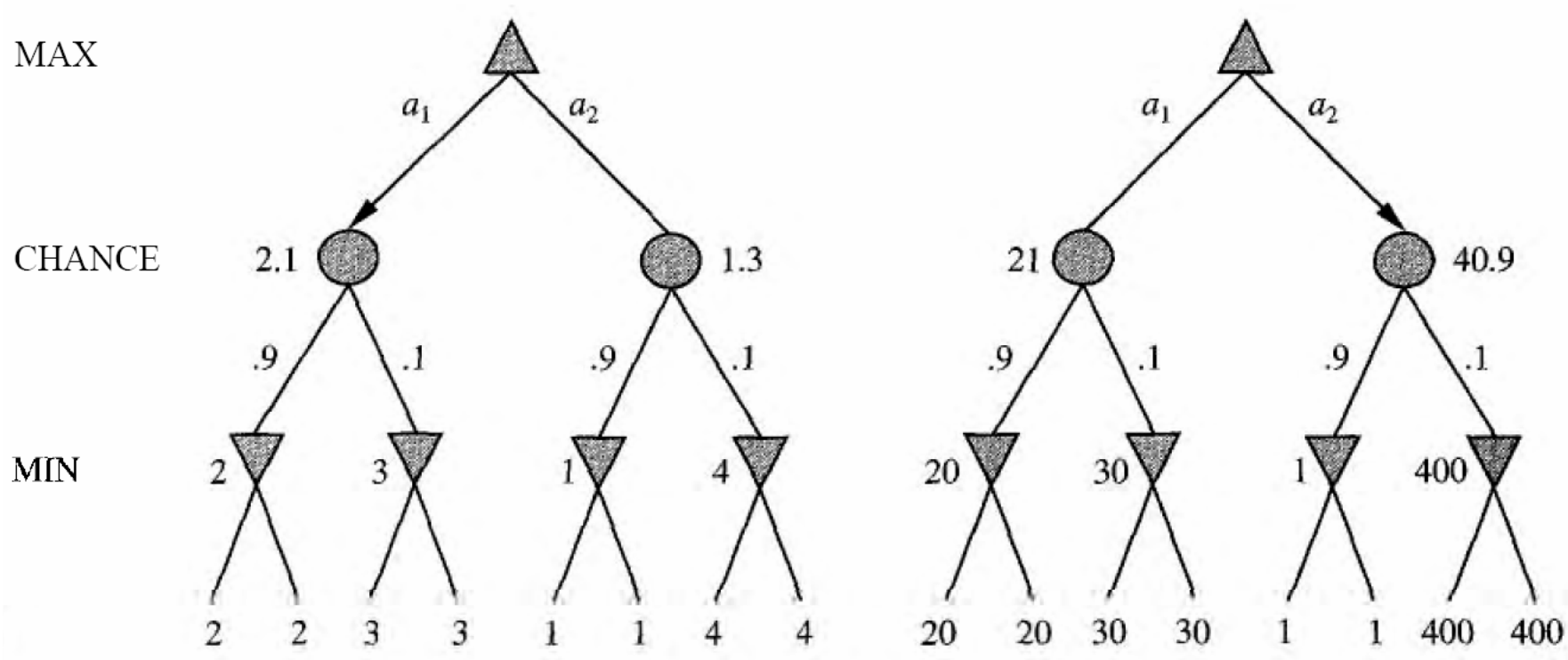
Како се играат игрите во кои има несигурна информација?

- Соодветна за игрите со карти
- Се воведува расудување во врска со тековната и следните состојби на секој од играчите за која се верува дека е најочекувана
- Тоа значи дека јазелот ја вклучува просечната веројатност дека ќе се случи имено таа ситуација

Игри кои вклучуваат среќа

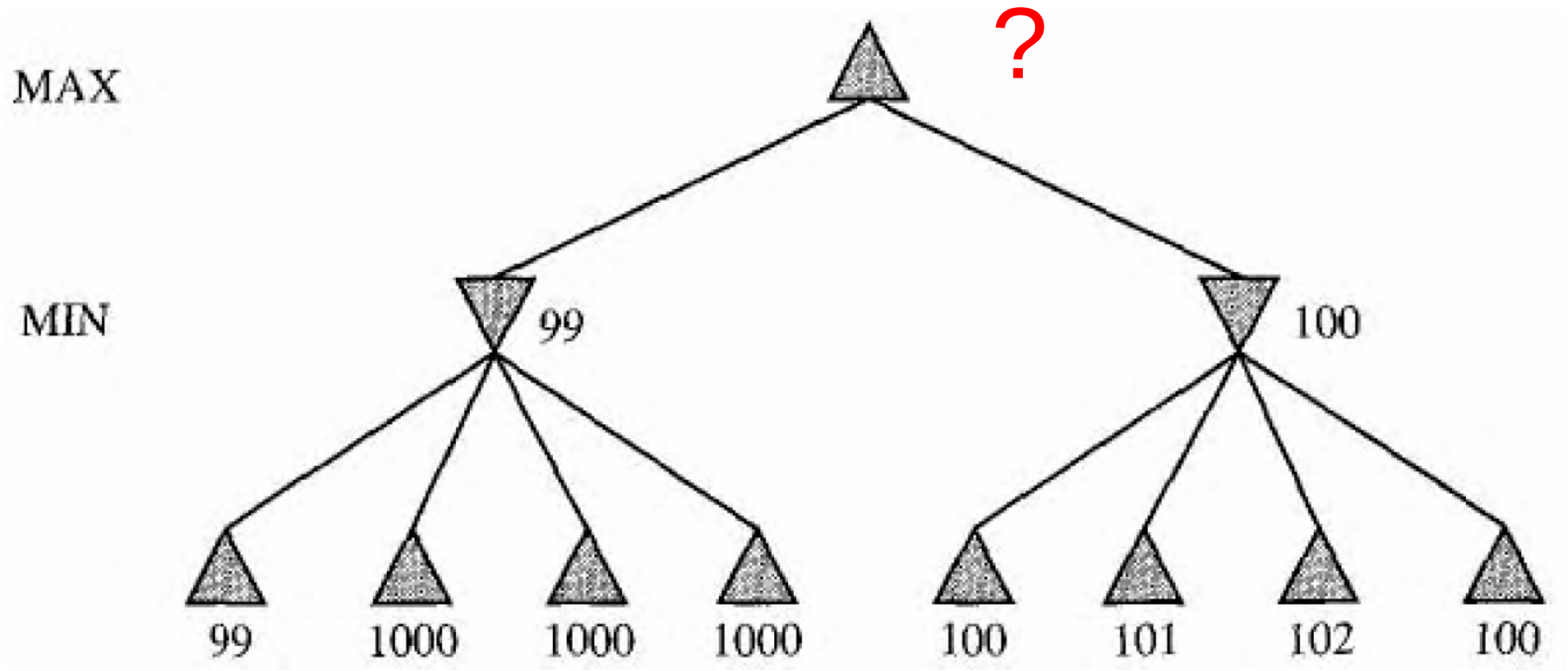


Кај игри со среќа се битни и вредностите на ф-јата за проценка



А не само подреденоста на вредностите

Но и кај Минимакс



И овде може да вклучиме веројатност на избор и да се надеваме на грешка на противникот

Очекувана вредност на minimax кај игри кои вклучуваат и среќа

за $s \in$ Sledni_potezi(n)

Expectiminimax(n)=

$$\begin{cases} \text{Utility(n), ако n е терминална состојба} \\ \max \text{ Expectiminimax(s), ако n е MAX јазел} \\ \min \text{ Expectiminimax(s), ако n е MIN јазел} \\ \sum P(s) \bullet \text{Expectiminimax(s), ако n е случаен јазел} \end{cases}$$

Кај недетерминистички системи*

- $Result_i(A)$ – можни резултатни состојби од акцијата A извршена во состојбата S
- $P(Result_i(A)|S, Do(A))$ – веројатност за секоја од резултатните состојби од акцијата A извршена во состојбата S
- $EU(A|S)$ – Очекувана полезност на акцијата A во состојбата S

$$EU(A \mid S) = \sum_i P(Result_i(A) \mid S, Do(A)) \cdot U(Result_i(A))$$

Како да се дефинира Utility(n)?

- Одредувањето на ф-јата на полезност преставува сложен проблем дури и за нешто општо прифатено како универзална вредност, како што се парите

Пример со коцкањето*

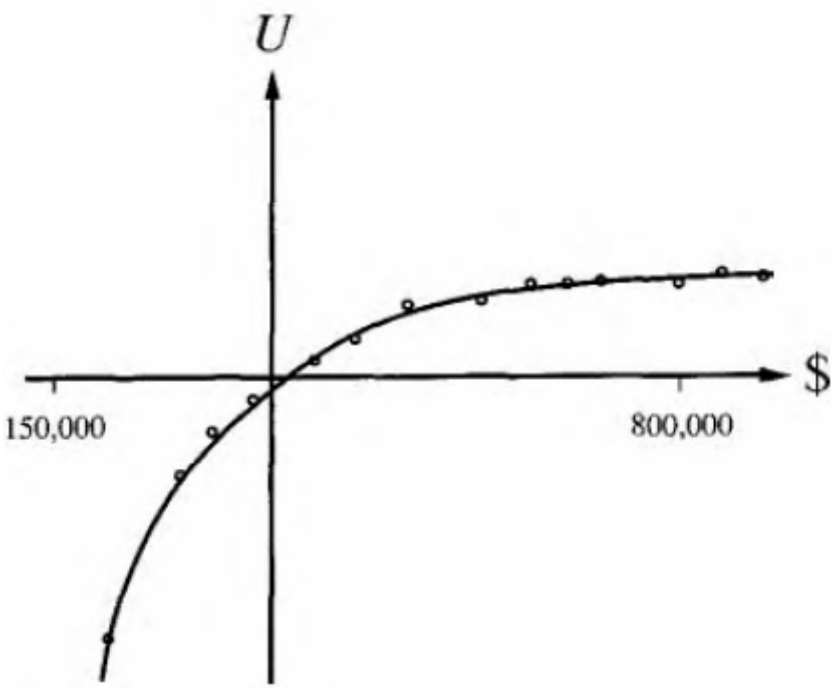
- Замислете ако ви се понуди: да фрлате паричка и ако погодите дали е „петка“ или „глава“ да добиете 3 милиони евра, а ако не погодите нема да добиете ништо. Или ако изберете да не фрлате паричка ќе добиете 1 милион евра.
  Што ќе изберете?

- Или формално (ако k е вашата досегашна заштеда):

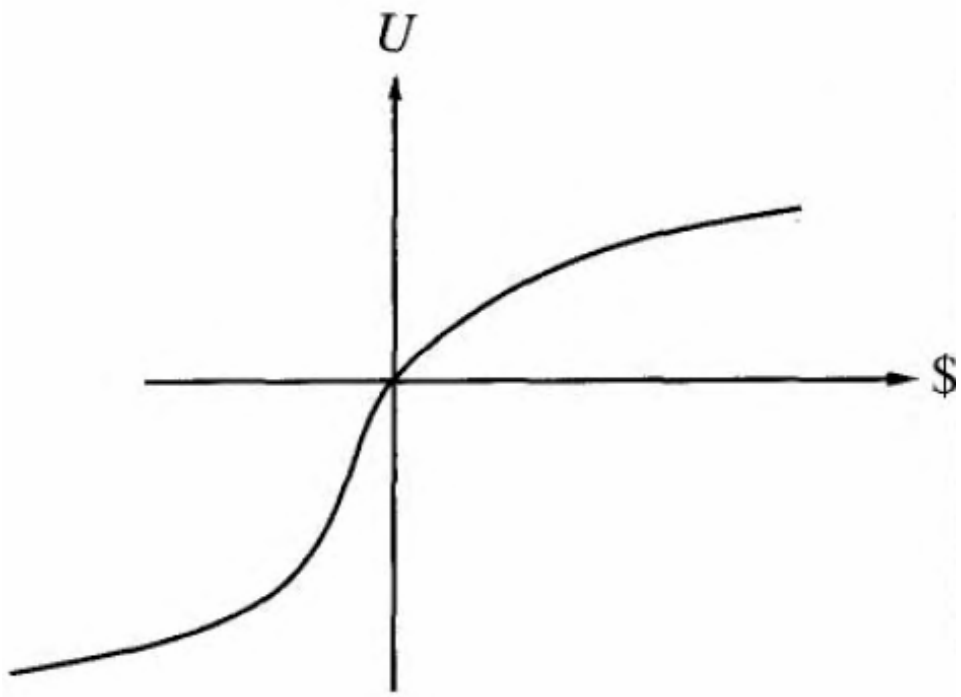
  $EU(Prifati)=\frac{1}{2}U(S_k)+\frac{1}{2}U(S_{k+3.000.000})$
  $EU(Odbij)=U(S_{k+1.000.000})$

- Зависи дали $k\approx 0$ или $k=500.000.000$ или уште повеќе ако е $k=5.000.000.000$

Општа вредност (полезност) на парите*



За некој поединец
во ограничен опсег



Општо за сите
за цел опсег

Примена во системи за одлучување за кредитирање, инвестирање, осигурување и сл.

Користена литература

- Artificial Intelligence, A Modern Approach 2nd edition, Russel and Norvig
- Artificial Intelligence, A New Synthesis, Nils J. Nilsson

Прашања?