Umělá inteligence I

7

Roman Barták, KTIML

roman.bartak@mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

Dnes

- Dosud popisované algoritmy nepředpokládaly přítomnost dalších agentů v prostředí, zvlášť ne agentů, kteří hrají proti nám.
- Dnes se podíváme na **hraní her**, jako na příklad **kompetitivního multi-agentního prostředí**
  - hry dvou hráčů s nulovým součtem a úplnou informací (piškvorky, šachy)
    - optimální (dokonalé) strategie (minimax, alfa-beta)
    - „nedokonalé" strategie („uříznuté" prohledávání)
  - hry s náhodným vstupem (vrhcáby – backgammon)

Hry

- **Matematická teorie her** (oblast ekonomie) vidí multi-agentní prostředí jako hru, kde má každý agent důležitý vliv na ostatní agenty.
  - je-li agentů moc, hovoříme o ekonomice (spíše než o hře)
- Umělá inteligence se zpravidla omezuje na **hry dvou hráčů**, kteří se **střídají** a mají **nulový součet** (zisk jednoho znamená ztrátu druhého).
  - **deterministické** hry vs. náhoda
  - **úplná** vs. částečná informace
  - Proč hry a UI? Protože hry jsou
    - těžké
    - dobře reprezentovatelné (a agenti mají málo akcí)
    - zábavné

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

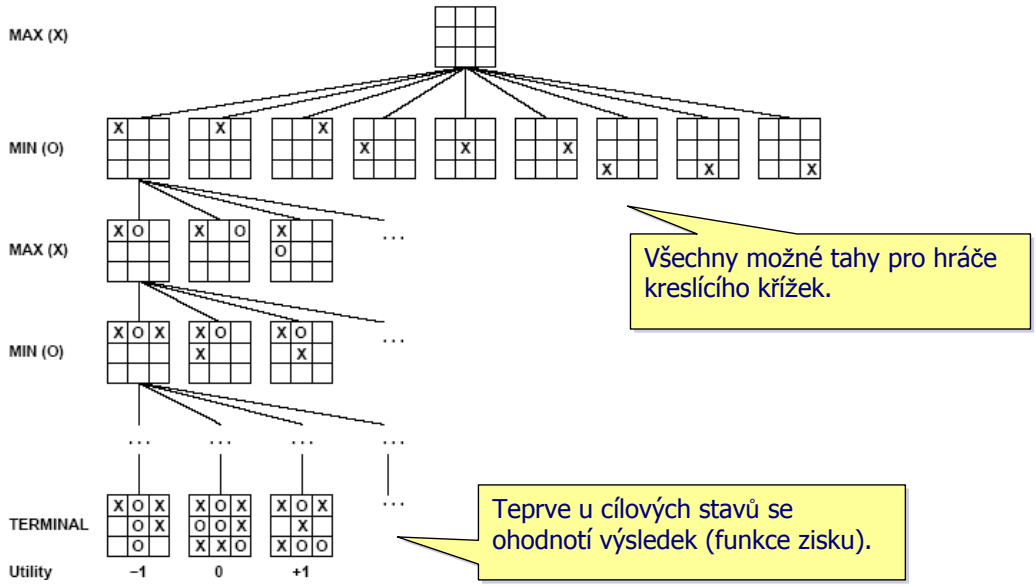
Základní nastavení

- Máme **dva hráče MAX a MIN**
  - MAX začíná a potom se hráči střídají dokud hra neskončí
  - hledáme strategii pro hráče MAX
- Opět se na hraní her podíváme jako na prohledávání:
  - **počáteční stav**: stav „desky" a kdo je na tahu
  - **funkce následníka**: seznam dvojic (tah, stav)
    - počáteční stav a funkce následníka určují **strom hry**
  - **test ukončení**: kdy hra končí (cílový stav)
  - **funkce zisku (utility)**: numerické ohodnocení koncového stavu (výhra, remíza, prohra je +1, 0, -1)
    - vysoké hodnoty jsou dobré pro MAXe, nízké naopak pro MINa

Strom hry

piškvorky

- Dva hráči střídavě kreslí křížky a kolečka dokud jeden nemá řadu tří svých symbolů nebo je vše obsazené.



Optimální strategie

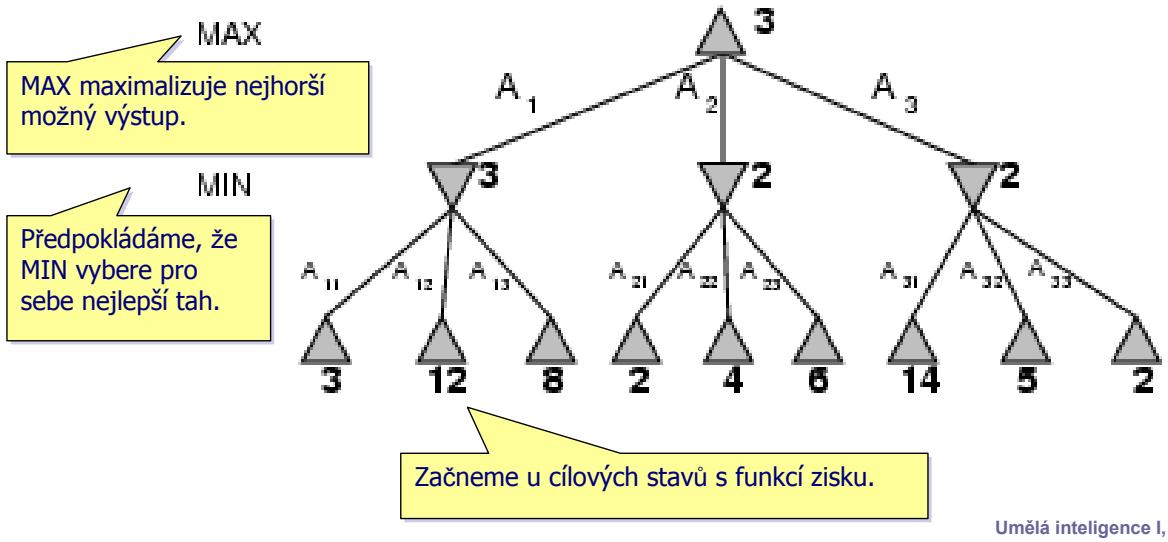
- Při klasickém prohledávání **hledáme nějakou (nejkratší) cestu do cíle**.
- U her také **hledáme cestu do koncového stavu s největším ziskem**, ale brání nám protivník MIN (určuje každý druhý krok na cestě), který hledá přesně opačné koncové stavy.
- MAX proto musí najít **podmíněnou strategii**, která určuje
  - počáteční tah
  - tah jako odpověď na každý tah MINa
  - **optimální strategie** dá takový výsledek jako libovolná jiná strategie, která je hrána proti neomylnému protivníkovi

Hodnota minimax

- Optimální strategii budeme hledat pomocí hodnoty minimax vypočtené v každém uzlu stromu hry takto:

MINIMAX-VALUE(*n*)=

UTILITY(*n*)	pokud je *n* cílový
$\max_{s \in successors(n)}$ MINIMAX-VALUE(*s*)	pokud v *n* hraje MAX
$\min_{s \in successors(n)}$ MINIMAX-VALUE(*s*)	pokud v *n* hraje MIN



Algoritmus minimax

```
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
```

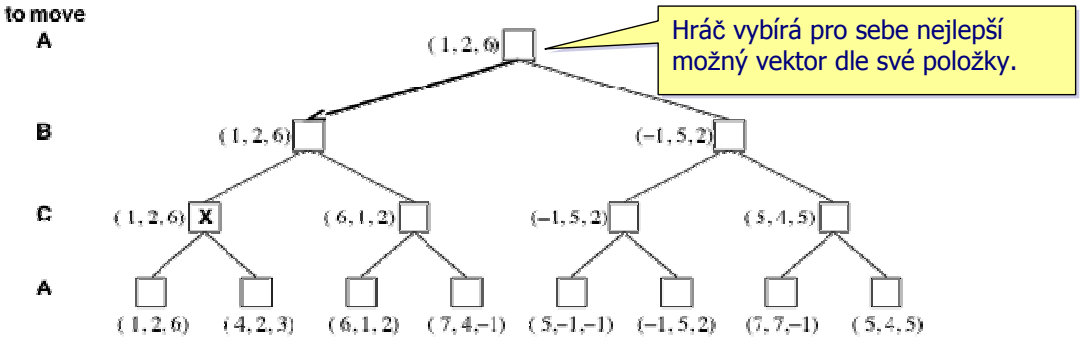
Algoritmus předpokládá, že protihráč hraje optimálně. Pokud ne, je zisk ještě větší.

- **Časová složitost $O(b^m)$**
- **Prostorová složitost $O(bm)$**

(b je počet akcí na uzel, m je délka hry)

Minimax a více hráčů

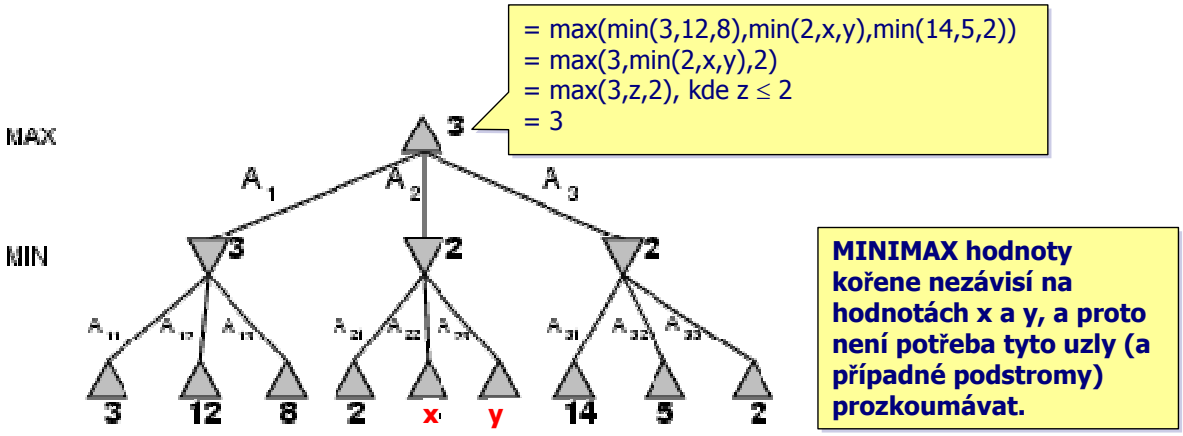
- Pro hru více hráčů použijeme **vektor hodnot** a každý hráč (na svém tahu) optimalizuje svoji položku.



- Hry více hráčů přinášejí další možnosti jako je třeba **vytváření aliancí**.
  - Aliance se zdají být přirozeným důsledkem optimálních strategií každého hráče.
  - Jsou-li A a B slabí a C silný, je pro A i B zpravidla optimální neútočit proti sobě, ale zaměřit se na C.
    Je-li potom C oslaben, aliance se často rozpadá.

Lepší minimax?

- Algoritmus minimax vždy najde optimální strategii, ale musí prozkoumat celý strom hry (všechny možné partie).
- Nešlo by dosáhnout stejného výsledku rychleji?
  - ANO!
    Nemusíme prozkoumávat všechny uzly, pokud jsou „špatné".
  - Tzv. α-β **ořezávání** eliminuje podstromy, kam se při hraní nedostaneme, protože tam nevede optimální strategie.

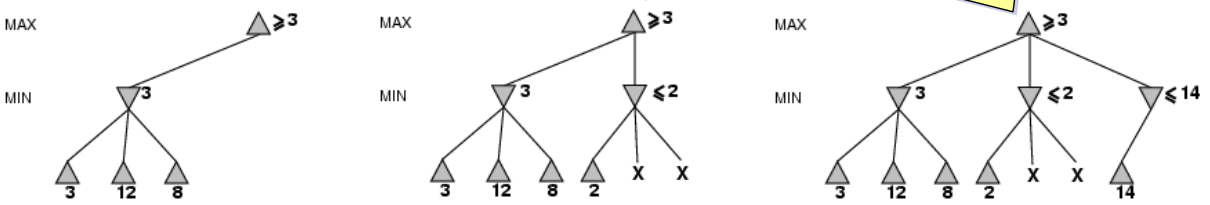


α-β ořezávání
příklad

Máme první dolní odhad MINIMAX hodnoty kořene

Ohodnocování MIN uzlu ukončíme ve chvíli, kdy dá horší (menší) MINIMAX hodnotu

U třetího MIN uzlu pořád můžeme najít lepší řešení



U MIN uzlu pořád ještě můžeme najít lepší řešení a to v rozsahu ⟨3,5⟩.

Tak to nevyšlo, optimum bylo 3.

Pokud bychom poslední blok prozkoumávali v pořadí 2,5,12, stačilo by ohodnotit 2.

Algoritmus α-β

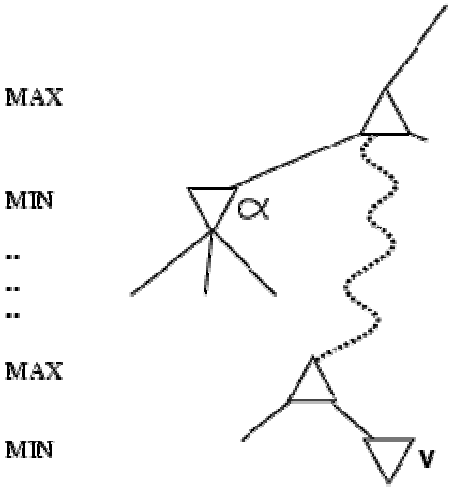
function ALPHA-BETA-SEARCH(*state*) **returns** *an action*
  inputs: *state*, current state in game
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the *action* in SUCCESSORS(*state*) with value v

---

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

function MIN-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**
    $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

Proč α-β?



- α je hodnota nejlepší volby (tj. ta největší), kterou jsme dosud nalezli pro hráče MAX
  - pokud α není horší (není menší) než v, nebude MAX nikde hrát směrem k v, proto není potřeba strom pod v prozkoumávat
- β je hodnota nejlepší volby (tj. ta nejmenší), kterou jsme dosud nalezli pro hráče MIN
  - podobně můžeme odříznout pod-stromy hráče MIN

Vlastnosti:

- Oříznutím nepřijdeme o optimální řešení.
- Při „perfektním uspořádání" srazíme časovou složitost na $O(b^{m/2})$, což dává větvící faktor $\sqrt{b}$ (minimax má b), takže lze prozkoumat uzly do dvojnásobné hlouby.

„Nedokonalé" strategie

- Metody minimax i α-β musí **prohledat strom hry až do listů pro** získání dokonalé strategie.
  - Není praktické při větší hloubce (hloubka = počet tahů pro dohrání hry).
- Můžeme jednoduše **prohledávání větve ukončit dříve** a použít heuristický odhad hodnoty MINIMAX.
  - negarantuje nalezení optimální strategie, ale
  - dokončí prohledávání v požadovaném čase
- **Realizace:**
  - test ukončení → test přerušení (cutoff)
  - funkce zisku → heuristická evaluační funkce EVAL

Evaluační funkce

- Poskytuje odhad zisku v daném podstromě (podobně jako funkce h v prohledávání).
- Kvalita výsledného algoritmu silně závisí na kvalitě heuristické evaluační funkce.

Vlastnosti:

- koncové uzly musí uspořádat stejně jako funkce zisku
- výpočet nesmí trvat moc dlouho
- pro nekoncové uzly by měla být silně svázaná se skutečnou šancí vyhrát
  - šance místo jistoty je zde z výpočtových důvodů (není čas prozkoumat vše)

- Jak takovou funkci ale najít?

Evaluační funkce

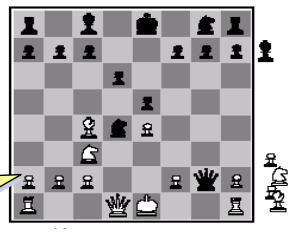
příklady

Očekávaná hodnota

- na základě vybraných vlastností, které umíme rychle rozpoznat, roztřídíme stavy do kategorií
- kategorii ohodnotíme podle podílu vítězných a prohrávajících stavů
  - EVAL = (0.72 × +1) + (0.20 × -1) + (0.08 × 0) = 0.52

„Materiální" hodnota

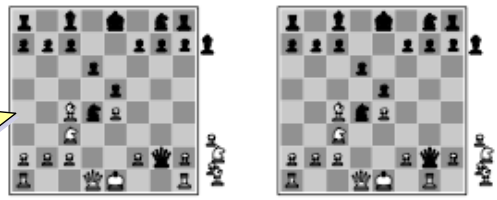
- odhadneme numerický příspěvek každé vlastnosti
  - šachy: pěšák = 1, střelec = kůň = 3, věž = 5, královna = 9
- tyto příspěvky zkombinujeme (např. vážený součet)
  - **EVAL(s) = $w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$**
  - Součet předpokládá nezávislost vlastností!
  - Možno použít nelineární kombinace.



Problémy s ořezáním

- Situace se dramaticky změní uvažováním dalšího tahu za limitní hloubkou.

Pro obě šachovnice je stejná materiální hodnota (lepší pro černou),
ale **vpravo vítězí bílý** sebráním královny bez náhrady.



  - **uklidnění** (quiescent): pokud zrovna protivník může brát, potom je odhad nestabilní a je dobré prozkoumat ještě pár tahů dopředu (například jen vybrané tahy)

- **Efekt horizontu**
  - špatnou situaci tak odsunout až za horizont, tj. není rozpoznána, ale stejně nastane.

Materiálně má více černý, ale pokud si bílý udělá z pěšáka královnu, tak **vyhraje bílý**.
Černý může opakovaně dávat věží šach, takže několik tahů to nevypadá tak špatně.

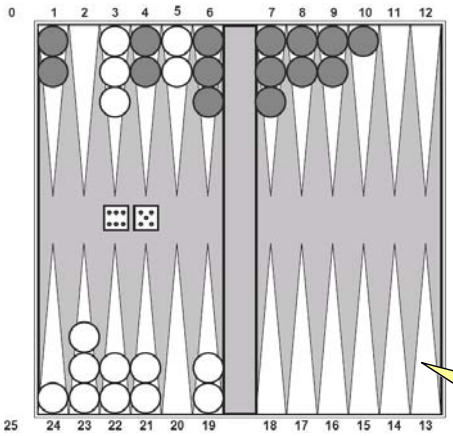
Možná vylepšení

- **Singulární prodloužení**
  - prozkoumáme sekvenci tahů, které jsou v dané pozici „jasně lepší" než ostatní
  - rychlý způsob jak prozkoumat oblast za limitní hloubkou (uklidnění je speciální případ)
- **Dopředné prořezání**
  - některé tahy v dané pozici nejsou vůbec uvažovány (lidský přístup)
  - nebezpečné, protože může minout optimální strategii
  - bezpečné, pokud jsou například tahy symetrické
- **Transpoziční tabulky**
  - podobně jako u pohledávání je možné si pamatovat jednou ohodnocené stavy, protože se do nich můžeme dostat jinou posloupností tahů

Náhoda a hry

- Reálný svět často obsahuje nepředvídatelné vnější události, které vedou k neočekávaným situacím.
- Hry simulují **nepředvídatelnost** zahrnutím prvku náhodnosti, jako je třeba hození kostkou.

Vrhcáby (backgammon)



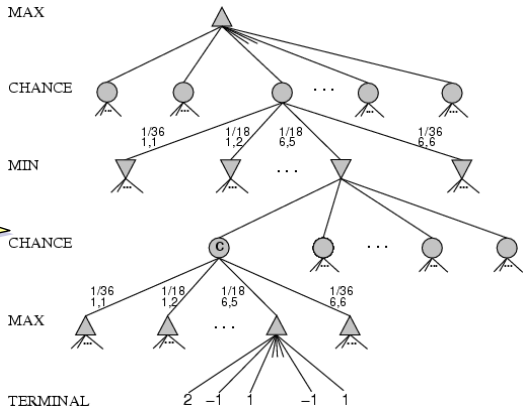
- cílem je přesunout vlastní kameny z počáteční pozice do koncové
- vítězí ten, kdo to udělá první
- hod kostkami určí, jaké tahy jsou v dané situaci povolené
  - vzdálenost skoku

Bílý může udělat čtyři legální tahy:
(5-10,5-11), (5-11,19-24), (5-10,10-16), (5-11,11-16)

Hry s náhodou

- Do stromu hry s MAX a MIN uzly přidáme **uzly náhody**, kde větve popisují všechny možné výsledky „hodu kostkou".
  - 36 výsledků pro dvě kostky, 21 po odstranění symetrií (5-6 a 6-5)
  - šance na double je 1/36, ostatní výsledky 1/18

Uzly náhody jsou vloženy do každé vrstvy resp. všude, kde je tah ovlivněn náhodou.
V této vrstvě hází MAX.



- Místo MINIMAX hodnoty počítáme **očekávanou hodnotu** (dle rozložení šancí):

EXPECTIMINIMAX-VALUE(*n*)=

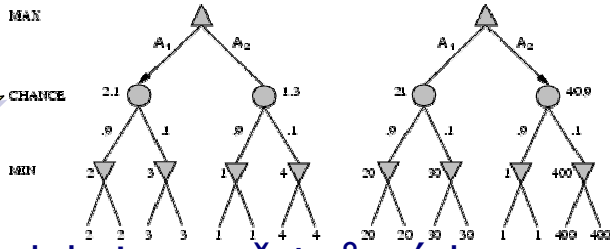
UTILITY(*n*)	pokud je *n* cílový
$\max_{s \in successors(n)}$ MINIMAX-VALUE(*s*)	pokud v *n* hraje MAX
$\min_{s \in successors(n)}$ MINIMAX-VALUE(*s*)	pokud v *n* hraje MIN
$\sum_{s \in successors(n)} P(s)$. EXPECTIMINIMAX(*s*)	pokud je *n* uzel náhody

Hry s náhodou

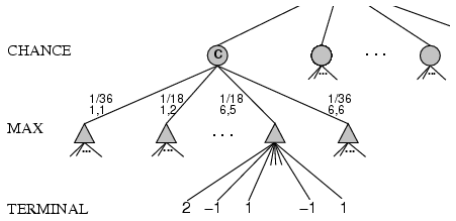
diskuse

- **Pozor na evaluační funkci** (při cutoff)
  - roli nehraje jen pořadí uzlů, ale i absolutní hodnoty
  - měly by být lineární transformací očekávaného zisku v uzlu

Vlevo vychází lépe A_1 a vpravo A_2, přestože evaluační hodnoty uspořádají listy stejně.



- **Časová složitost** $O(b^m n^m)$, kde je n počet různých náhodných výsledků
  - není realistické dostat se do větší hloubky, zvlášť pro větší náhodné větvení
- **Použití ořezání à la α-β**
  - můžeme ořezat i u uzlů náhody, pokud máme omezenou funkci zisku
  - pro výpočet očekávané hodnoty použijeme meze tam, kde hodnotu ještě nemáme



Karty

- Na první pohled vypadají karty jako hra s náhodou, ale kostky jsou zde vrženy hned na začátku!
- Zajímavé na kartách je to, že (někdy) nevidíme všechny karty protihráče (tj. máme **částečnou informaci**).

Příklad: karetní hra „větší bere" s otevřenými kartami

Situace 1: MAX: ♥6 ♦6 ♣9 8 MIN: ♥4 ♠2 ♣10 5

1. MAX dá ♣9, MIN potvrdí barvu ♣10 vítěz MIN
2. MIN dá ♠2, MAX dá ♦6 vítěz MIN
3. MAX dá ♥6, MIN potvrdí barvu ♥4 vítěz MAX
4. MIN dá ♣5, MAX potvrdí barvu ♣8 vítěz MAX

- ♣9 je pro MAXe optimální první volba

Situace 2: MAX: ♥6 ♦6 ♣9 8 MIN: ♦4 ♠2 ♣10 5

- symetrický případ, ♣9 je opět pro MAXe optimální první volba

Situace 3: MIN skryl první kartu (♥4 nebo ♦4), jaká je optimální první volba pro MAX?

- Nezávisle na ♥4 nebo ♦4 byl v předchozích situacích optimální první tah ♣9, tedy i teď je ♣9 optimální první tah.
- **Opravdu?**

Neúplná informace

Příklad: cesta k bohatství (aneb karty jinak)

- **Situace 1**: Cesta A vede k hromadě zlata a cesta B vede na rozcestí. Vydejte se doleva a narazíte na mohylu drahokamů, ale vydejte se doprava a přejede Vás autobus. Kam půjdete (když drahokamy jsou cennější než zlato)?
  - B a doleva
- **Situace 2**: Cesta A vede k hromadě zlata a cesta B vede na rozcestí. Vydejte se doprava a narazíte na mohylu drahokamů, ale vydejte se doleva a přejede Vás autobus. Kam půjdete?
  - B a doprava
- **Situace 3**: Cesta A vede k hromadě zlata a cesta B vede na rozcestí. Vydejte se správně a narazíte na mohylu drahokamů, ale vydejte se špatně a přejede Vás autobus. Kam půjdete?
  - rozumný člověk (ten kdo neriskuje ;-) jde A
- To je přesně stejný případ jako u příkladu s kartami. Na rozcestí B neznáme, která ze situací 1 a 2 nastává, stejně jako u karet nakonec také nevíme, zda MIN drží ♥4 nebo ♦4, takže je 50% šance neúspěchu.
- **Poučení:** Je potřeba uvažovat informaci, jakou budeme mít v každém stavu hry (chyba volby ♣9 je v tom, že MAX uvažoval jakoby v každém kroku viděl všechny karty).

Hry – jak si stojí UI?

- **Šachy**
  - 1997 superpočítač Deep Blue poráží Kasparova 3.5 – 2.5
  - 2002 „běžné" PC (program FRITZ) remízuje s Kramnikem
- **Dáma**
  - 1994 program Chinook mistrem světa
  - 29. 4. 2007 vyřešeno – optimální řešení je remíza
- **Go**
  - větvící faktor 361, což omezuje současné metody
  - průměrný amatérský hráč je lepší než počítač
- **Bridge**
  - 2000 program GIB dvanáctý na mistrovství světa
  - programy Jack a Wbridge5 hrají na úrovni nejlepších hráčů