

ÚVOD DO AI

Agent = něco co reaguje na prostředí

→ racionální agent volí nejlepší akci aby max. utility

$$a^* = \underset{a}{\operatorname{argmax}} \mathbb{E}[\text{utility}(o, a)]$$

observation

a | a

Druhy prostředí:

- fully / partially observable
- deterministic / stochastic - dolsí stav rávni jen na aktuálním + aktuální akci
- episodic / sequential - episody jsou nerázové
↳ akce x aktuální ef. reakce x akce
- static / dynamic
↳ nemění se prostředí agent rozmýší akci
- discrete / continuous
- single agent / multi agent
↳ co-operative x competitive

Druhy agentů

- Reflex agent: chování se nelisí podle věc

simple: $o \rightarrow a$... observation $\rightarrow a^*$

→ má transition model → poučuje si nejaky místní stav

↳ $(s_t, a_t, o_{t+1}) \rightarrow s_{t+1}$, místní stav + akce + percept \rightarrow nový stav
 $s_{t+1} \rightarrow a_{t+1}$

- Goal based agent - flexibilnější, má cíl

$(s_t, a_t, o_{t+1}) \rightarrow s_{t+1}$

$(s_{t+1}, \text{Goal}) \rightarrow a_{t+1}$

Reprezentace stavů

- atomic - nedílící, nemá místní strukturu → prohledávání stav. prost.
- factored - stav = reálný vektor → CSP, plánování
- structured - stav = mnoha objektů
a relace mezi nimi \approx graf → first-order logic

• Problem solving agent

- atomická reprezentace stavu
- $a^{\text{st}} = \text{mínima stava}$ → máme transition model
 $(\text{state}, \text{act}) \rightarrow \text{state}$
- $a^{\text{tr}} = \text{transitions mezi stavami} \sim \text{hrany}$
- chceme najít pokračování aktu, co mě dárne do jiného stavu
- ⇒ SEARCH
- předpoklady: prostředí = fully observable, discrete, stochastic, deterministic / abstract prostředí - real world prostředí je moc komplikované
 - validní ... řešení abstractu dle kritéria přesnosti na řešení původního probl.
 - useful -> využití abstractu má byt jednoduché

Strategie

- výdaje máme několik frontu
- vybíráme z nich ažel podle nějaké strategie
 - ↳ expandujeme ho = sousedy (neznámé všechny) dárne dle fázy

Tree search

- neřešíme si novější, do fronty díváme si
- z grafu generuje nějaký až exponenciálně mohoucí strom

Graph search

- pamatuje si novější vzdálosti

Implementation

- DFS, BFS, Dijkstra (fancy BFS)
- iterative deepening - pamatuje si jen aktuálně nejlepších vrcholů
 - ↳ postupně rozšiřuje search rodu
- Dijkstrou - expanduje užel co je nejblíže ke startu - $g(n)$
 - ↳ výpočtuji $g(n)$ souseďů aktuálně nejlepšího vrcholu
- Best first search - mám heuristiku h , co udává odlišnou vzdáenosť do cíle
 - ↳ vybírá vrchol s co nejméně h
- A^* ⇒ výbírá min $f(n) := g(n) + h(n)$

- Heuristiky
- aby heuristiky fungovaly a dolo se A^* dokáro, je funguje, mimo jiné je optimální

Def: Heuristika $h: V \rightarrow \mathbb{R}$ je

1) přípusťná $\equiv \forall m: 0 \leq h(m) \leq$ délka nejkratší cesty z m do cíle

2) monotoní \equiv pro \forall souseda m' méně m platí

$$h(m) \leq c(m, m') + h(m') \dots \Delta\text{-nearest}$$

Monotoní heuristika je přípusťná

Rk: necht m_1, m_2, \dots, m_ℓ je optimální cesta z m_1 do cíle m_ℓ

$$H_i: h(m_i) - h(m_{i+1}) \leq c(m_i, m_{i+1})$$

$$\begin{aligned} \Rightarrow h(m_1) &= h(m_1) - h(m_2) + h(m_2) - h(m_3) + \dots + h(m_{\ell-1}) - h(m_\ell) + h(m_\ell) \\ &\leq c(m_1, m_2) + c(m_2, m_3) + \dots + c(m_{\ell-1}, m_\ell) \end{aligned}$$

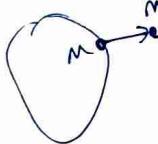
↳ délka nejkratší cesty

Pro monotoní heuristiku jsou hodnoty $f(m)$ některé na každé cestě

$$\hookrightarrow f(m) = h(m) + g(m) \quad \hookrightarrow g(m') = g(m) + c(m, m')$$

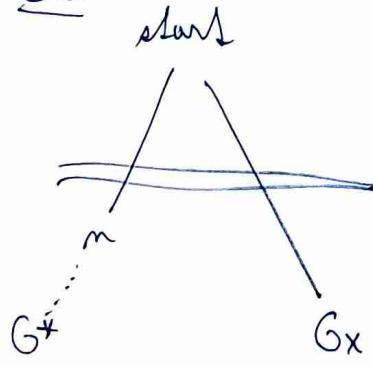
\rightarrow expandoval jsem m , následně m' , kdežto $f(m') \geq f(m)$

$$\begin{aligned} m' \quad f(m') &= g(m') + h(m') = g(m) + c(m, m') + h(m') \geq g(m) + h(m) = f(m) \end{aligned}$$



Věta: Pokud je $h(m)$ přípusťná, pak je tree search verze A^* optimální.
↳ neboli: první expandovaný vý je optimální

Rk:



* h je přípusťná

Pro spor necht algoritmus expanduje jenom G_x a nem je optimální

\rightarrow necht je optimální G^* s cenou C^*

$$\Rightarrow f(G_x) = g(G_x) + h(G_x) = g(G_x) > C^*$$

\rightarrow necht m je vrchol $\&$ fronty na optimální cestě

$$\hookrightarrow \text{pak } f(m) = g(m) + h(m) \leq C^*$$

$$\Rightarrow \text{dopravně } f(m) < f(G_x) \Rightarrow m \text{ se expanduje před } G_x$$

Věta: Pokud je $h(n)$ přípusťná, pak je Graph search mere A* optimální.

Pr:

- moving problem G-S: najdu nejkratší cestu do městka n, až
ži najdu lepsi, protože jsem už v něm vzdálil

→ ab pro monotonii jsou body $f(n)$ nelesající a A* expanduje
nesel s nejmenším $f(n)$

⇒ měříme si délky cestami do n vždy zpočtuje n
Kombinování herciků

Inverze: Pokud jsou h_1, h_2 přípusťné / monotonie, tak

$$\text{i)} \lambda h_1 + (1-\lambda) h_2, \quad \lambda \in [0, 1]$$

$$\text{ii)} \max\{h_1, h_2\}$$

jsou taky přípusťné

afinní kombinace $\leq \max$ $\Rightarrow \max$ je pro A* lepsi

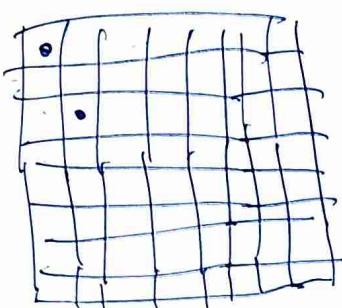
Grádkame, že \max dominuje afin.

• A* prelomá řešení vždy i.e. $f(n) < c^*$, tedy $h(n) < c^* - g(n)$

⇒ pokud A* expanduje vrchol s max. h(n), musí být iš. afin.

• Constraint Satisfaction Problems - CSP

- 8-Queen problem



proměnné: $r_1, \dots, r_8 \in [8]$

r_i = pozice královny v i -tém sloupci

\Rightarrow domény: $\{1, 2, \dots, 8\}$

\Rightarrow funkce: královny se nesmějí sbíhat

$\hookrightarrow f_i \neq j : r(i) \neq r(j) \wedge |i-j| \neq |r(i) - r(j)|$

Obecně:

- konečné mnoho proměnných

- domény = konečné množiny hodnot pro \forall proměnnou

- konečné mnoho funkcií

\hookrightarrow funkcionální \wedge relace mezi funkciemi proměnných

\hookrightarrow axioma funkcií = fóbi o vlivení jedné proměnné na ostatní

Příklad:

• Sudkov: $\forall i, j \in [9] : x_{ij} \in \{1, 2, \dots, 9\}$

funkce: ∇ řádek, ∇ sloupec, ∇ : all-different

• Barvení vrcholů grafu \wedge barvami

proměnné: vrcholy G

domény: $[k]$

funkce: $\nabla u, v \in E : C(u) \neq C(v)$

Řešení:

• Backtracking = tree/graph search s DFS strategií

—

1. přírodní hodnota zvolené (dozd nedohodnocené) proměnné

2. kontroluj funkcií ma již ohodnocených proměnných

\rightarrow pokud je vše splněno \rightarrow aktuální proměnné

\rightarrow final závěr získan vrchol

\rightarrow pokud některá hodnota neplatí \rightarrow mot se k minulé proměnné

Forward checking

- náleží si „projektávání“ hodnoty, které jsou relaxány
- když dám někam drážku, tak si projektaím ohrožené polohy
- ⇒ projektované hodnoty může nekonávat

Arc-Consistency

- paralelne řešení binární podmínky

⇒ tří podmínek \sim arc \sim grafu podmínek

Def: Arc (V_i, V_j) je arc-consistent $\equiv \forall x \in D_i \exists y \in D_j \exists$

$\begin{cases} \text{je } (V_i, V_j) \text{ p. kons.} \\ \text{není vlastní, je } (V_j, V_i) \text{ je} \end{cases}$

(x, y) splň řešení podmínky

Def: CSP je arc-consistent $\equiv \forall$ arc je consistent všem směrech

Příklad:

$$A = \{1, 2, 3\}$$

$$B = \{1, 2, \underline{3}\}$$

$$C = \{1, 2, \underline{3}\}$$

$$A > B$$

$$B = C$$

Agenda

- 1 $A > B$
- 2 $B < A$
- 3 $B = C \checkmark$
- 4 $C = B$
- 5 $A > B$
- 6 $B = C$

Arcs

$$A > B$$

$$B < A$$

$$B = C$$

$$C = B$$

Algoritmus AC-3

1. udelej z tří bin. podmínky dva Arcs: $A = B \Rightarrow A = B \& B = A$
 2. přidej řešeny Arcs do Agenda
 3. Ofočuj, dokud nebude Agenda prázdná
 - rem Arc (V_i, V_j) z agenda
 - pro tři hodnoty V_i musí být nějaká hodnota V_j
 - odstraní neobsahující hodnoty z V_i
 - pokud se doména V_i změnila, přidej do agenda řešeny (V_k, V_i)
- ↳ potom následně můžeme přidat další funkce

• Maintaining arc consistency (look ahead)

1. užívej problem Arc-consistency

2. backtracking

- vždy, když proměnné přidáme hodnotu \Rightarrow obnov konzistence

• Silnější konzistence

- ac-consistency je lokální

sudoku:

		6
3	9	
2	1	8

$$X_{1,1} = \{4, 7\}$$

$$X_{1,2} = \{4, 7\}$$

$$X_{2,3} = \{4, 7, 5\}$$

$$X_{1,1} \neq X_{1,2}$$

$$\times \quad \times$$

\rightarrow je to arc-consistent

\hookrightarrow 5 méně odstranit

\Rightarrow lze mítis k -consistency

\hookrightarrow pro konzistentní příjem kardigib $k+1$ proměnných
vytvoříme konzistentní hodnotu v kard. dolší ($k+1$) proměnné

Věta: Pokud je CSP i -consistent pro $\forall i=1, \dots, n$ a máme
právě m proměnných, potom lze vyřešit bez backtrackingu.

Důkaz: DFS vždy najde hodnotu konzistentní s ohodnocením jednotlivých

\hookrightarrow bohužel, časová komplexitá k -consistency je eksponentiální $\propto k^m$

• Global constraints

- typický pro vícenásobné global constraints \sim podproblemy s konkrétní strukturou

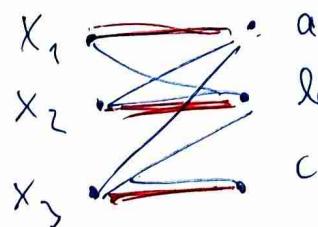
\Rightarrow all-different(X_1, X_2, \dots, X_k)

\downarrow
dají se vyřešit efektivně

$$X_1 = \{a, b\}$$

$$X_2 = \{a, b\}$$

$$X_3 = \{c, d\}$$



\Rightarrow maximální párování

\rightarrow řešení hranu nemá vzhledem max. párování

\Rightarrow odstranit hodnotu

Variable ordering

- jak vybrat pořadí vybrání proměných?

→ fail-first principle ... verne si, co nejsvíce počde & fail

- dom heuristic: nejménší doména

- deg heuristic: proximální ručičková nebo nejvíce funkcionál

Value ordering

→ succeed-first principle → negativní heuristiky

SAT - solvency

→ 8-Queens polynomický množstvovat formou CNF - bodové pravidlo

Algoritmus DPLL - splňte hledaný CNF

1) jednotlivá propagace x_1 : $x_1 \wedge (x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_2 \vee x_2)$
⇒ $(\underline{x_2} \vee \neg x_3) \wedge (x_3 \vee \underline{x_2})$

2) existuje výsledek x_2 : ⇒ koncový ✓

→ pokud nemá cistý výsledek ⇒ branching

Vybějení

- analýza konfliktů - když je nějaký konflikt druhým konfliktom

- variable (value) ordering

- random restarts

- clever indexing - efektivní identifikace jednotlivých literálů

- ↳ watched literals

- z 2 hlavních sledují 2 literaly

- dokud oba neohodnotíme, tak může mít jednotlivá

- ↳ pokud nejaky ohodnotíme ⇒ výběr doba

- clause learning - ne vždy využívám, že můžu kombinovat hodnoty

- druhomady nevyužije ⇒ užívám novou elementu, když ji užívám

Knowledge-based agents - Wumpus

- negativ formalizace

TELL ... problem do knowledge-base

ASK ... dofar = inference

- je dané faktické výročí?

↳ řešit je výročí a vložit do modelu KB \Rightarrow ANO

↳ konečně: IDK

Wumpus může: březa slibuje den, stenky slibují výpadek

$$\begin{array}{l} P_{i,j} = \text{přít} \\ W_{i,j} = \text{Wumpus} \\ B_{i,j} = \text{březa} \\ S_{i,j} = \text{stenky} \end{array} \quad \left. \begin{array}{l} \neg P_{1,1} \\ \neg W_{1,1} \\ \neg B_{1,1} \\ \neg S_{1,1} \end{array} \right\} \rightarrow \text{racinám kávu}$$

\rightarrow forwardin'

\rightarrow model světa

$$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$$

$$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$$

\rightarrow je genom 1 wumpus

$$\rightarrow \text{alegorie 1: } W_{1,1} \vee W_{1,2} \vee \dots \vee W_{n,n}$$

$$\rightarrow \text{max. 1: prototyp } x_1, x_2, y_1, y_2: \neg W_{x_1,y_1} \vee \neg W_{x_2,y_2}$$

\rightarrow určitá modelová sentense d je $M(\alpha)$

\rightarrow $KD \models \alpha \dots M(KD) \subseteq M(\alpha) \rightarrow \alpha$ je dosledek KB

$\models KD \models \alpha \Leftrightarrow KD \wedge \neg \alpha$ je neplnitelná

$$\rightarrow \text{redukce: redukční pravidlo: } \frac{x \vee y \quad z \vee \neg y}{x \vee z}$$

\rightarrow řešit jsem odvozil \square , že $KD \models \alpha$

\rightarrow řešit měří rozhodnutí dle kterého je $KD \not\models \alpha$

Hornské členě = nejvýše 1 pozici literálů

↳ rovnaté inferece : $C \wedge (B \Rightarrow A) \wedge (\neg D \Rightarrow B) \rightarrow \text{prolog}$

→ existuje lineární algoritmus \Rightarrow LI-resolučna

$$\frac{A \quad A \Rightarrow B}{B}$$

2) Backward chaining - rozdíl query na sub-queries

- goal-driven metoda

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$A \wedge D \Rightarrow L$$

$$A$$

$$B$$

ASK: plati Q ?

→ maximální split P

→ maximální split $L \wedge M$

→ maximální split L

→ $A \wedge B \Rightarrow L \quad \checkmark$

→ maximální split M

→ $B \wedge L \Rightarrow M \quad \checkmark$

plati

2) Forward Chaining

→ pro + kláreni si pamatuj n # neplatných předpokladů

↳ smíření, kdyží odvozí nový fakt (vložit před. vložit)

Fakta: $A, B, L, \neg R, P \quad Q$

$$A \wedge P = L$$

$$A \wedge B = L$$

$$\neg R \wedge L \Rightarrow M$$

$$L \wedge R \Rightarrow P$$

$$P \Rightarrow Q$$

data-driven

Reasoning dle této formou logické inference

! Neutro přístup lze uplatnit jen na statistické prostředí

"Autonómické" plánovanie

→ so kdejším se postarát miestu v čase?

Fluent = pomernej ahojovaný časom

↪ $L_{x,y}^1$ = agent je v čase t na pozicii (x,y)

Observational model

→ spojuje observation s modelom sveta

$L_{x,y}^1 \Rightarrow (\text{Breeze}^1 \Leftrightarrow B_{x,y})$

↪ rieka, ada v čase t číhá breza

$\text{Safe}_{x,y}^1 \Leftrightarrow (\neg \text{Pit}_{x,y} \wedge \neg (\text{W}_{x,y} \wedge \text{Wumpus Alive}^1))$

Transition model

→ popisuje ako aké vznikajú svet

• effect axioms - vznikajú, co aké zmieny

$L_{x,y}^1 \wedge \text{North}^1 \wedge \text{Forward}^1 \Rightarrow L_{x,y+1}^{1+1} \wedge \neg L_{x,y}^1$

• frame axioms - vznikajú, co sa nemieni

$\text{Forward}^1 \Rightarrow (\text{Have Arrow}^{1+1} \Leftrightarrow \text{Have Arrow}^1)$

axiomy rávne
→ je jich pôsoba strošne moc ⇒ neefektívne

• successor-state axioms

→ pro každý fluent F definujeme pravidlo F^{1+1} formou

fluentu a aké v čase t

$F^{1+1} \Leftrightarrow (\text{A}\&(\text{CoZpísobi}F)^1 \vee (F^1 \wedge \neg(\text{A}\&(\text{CoZpísobi}(\text{Not}F))))$

$\text{Have Arrow} \Leftrightarrow \text{Pick Up Arrow}^1 \vee (\text{Have Arrow}^1 \wedge \neg \text{Shoot}^1)$

• precondition axioms - kdy ke skúfku: $\text{Shoot}^1 \Rightarrow \text{Have Arrow}^1$

• action-exclusion axioms: $\forall i, \forall i \neq j : \neg (A_i^1 \wedge A_j^1)$

↪ vznikne poriadok súčasné násťreky aké - vtedy by maliť súčetne efekty/poriadky

Hybridní plánování

- logickou odvodíme pravidly o světě
 \hookrightarrow na různé místy mohou se rozcházet, co bude dělat
- pro plánování stavy popisujeme A^*

SAT Plan

\rightarrow problém rozdělujeme do SATu

- Init⁰ ... počáteční stav světa

- Transition¹, ... Transition^T \rightarrow axiomy popisující akce $\left\{ \begin{array}{l} \text{successor-state} \\ \text{transition} \end{array} \right.$

- Goal¹ ... cíle, aby cíl plnil v case A.

$\hookrightarrow (HaveGold^1 \wedge ClimbedOn^1)$

\rightarrow řešeního dle SATu

- získal jazyk model \Rightarrow existuje řešení, model ho kóduje

- získal model než, neexistuje plán dleky A

\rightarrow SAT Plan málo rekonstruovat $t = 1, \dots T$

Automatizace plánování

- plán lze majit prohledáváním (A^*), ale mám hodně stava

\hookrightarrow potřeboval bych hodně delšího hermítka - jde ale jen ely dobré

\rightarrow ne myrokuji logické postupujem na základě blokov formulí

\hookrightarrow chci tedy byť logický 1. následn. \Rightarrow schéma akionu

Situací k volitelnosti

stavy popisujeme pomocí predikátu: at(Robot, Location)

- potom se provádí sítě relací mezi různými situacemi

\Rightarrow (fluents): at(Robot, Location, s) \hookrightarrow konkrétní situace

- zde se nemíme (rigidní predikáty): connected(loc1, loc2)

\rightarrow pro t akci possibility axiom: $\Phi(s) \Rightarrow Possible(s, a)$, Φ je formule

\rightarrow pro t fluent successor-state axiom: některá rada plati v delších stavech

$Poss(a, s) \Rightarrow F(s') \Leftrightarrow (a \models s' F) \vee (F(s) \wedge F \text{ is not made False by } a)$

→ plánování → situačním reakcím

↳ ($\exists s$) Goal(s)

↳ ($\exists s$): HaveGold(s) \wedge ClimbedOnt(a, s)

• Klasické plánování

stav = vektor proměnných --- factored representation
akční schéma popisuje jak agent může měnit svět

⇒ problém: stav je hodně i pro malé problémy

→ plánování aby bylo snadné řešit problémy

→ stav je vždy minimálnější rámec

- fluents - neměnící se stav

- rigid atoms - neměnící se

Konvence closed world assumption = atom $a \notin S \Rightarrow a \text{ negated} \vee \perp$

Df: Stav s splňuje vklad $g \equiv g^+ \subseteq S \wedge g^- \cap S = \emptyset$

↓
positive atoms

↓
atoms, or from
negation $\neg g$

Akční schéma (operator)

Load (car, bot, location)

preconditions:

empty(car)

at(car, location)

at(bot, location)

effects:

not empty(car)

not at(bot, place)

at(bot, car)

⇒ operátor má jméno, parametry, triedočky a efekty

⇒ akce je instance operátora - za proměnné desadíme konstanty

Domain model = fópis operátoru

Planning problem obsahuje

- Domain model
- fórmulem stav, jake objekty (konstanty) ne sviše existují
- cíl

Plán je sekvence akcí

→ rady kde jsou akce řazeny?

Df: Akce a je aplikovatelná na stav s

$$= \text{precond}^+(a) \subseteq s \wedge \text{precond}^-(a) \cap s = \emptyset$$

Výsledek aplikování akce a na stav s je

$$\gamma(s, a) := (s \setminus \text{effects}^-(a)) \cup \text{effects}^+(a)$$

Akce a je relevantní pro cíl g \equiv

i) akce působí cíli: $g \cap \text{effects}(a) \neq \emptyset$

ii) efekty akce nejsou v konfliktu s cílem:

$$g \cap \text{effects}^+(a) = \emptyset$$

$$g \cap \text{effects}^-(a) = \emptyset$$

Regressní možnosti pro akce a relevantní k cíli a je

$$\gamma^{-1}(g, a) := (g \setminus \text{effects}(a)) \cup \text{preconditions}(a)$$

Plán $\pi = \langle a_1, a_2, \dots, a_n \rangle$ k problemu P $\equiv \gamma^*(s_0, \pi)$ splňuje cíl

• Forward-search - upravuje stav

- začíná se stavem $s = s_0$, provádí aplikaci akce, ježich je předvolba
↳ na splnění, následně $s \leftarrow \gamma(s, a)$

• Backward-search - upravuje cíl

- začíná se zadáním cílem a potom aplikací akce, pro které je jasné
definování $\gamma^{-1}(g, a) \Rightarrow$ pro udělení $g \leftarrow \gamma^{-1}(g, a)$

→ menší branching factor, ale je těžší organizovat heuristiky

Plánovací heuristiky

- chame pripomienky heuristiky = dohľad na # akcií do cieľa
 \Rightarrow nejednoznačne rešenie soho problemu

- 1) ignorujú preťaženosť akcií \Rightarrow takto je problem ťažšie riešiť
- 2) ignorujú negatívny efekt akcií
 (nejmenší možný aktív, čo vytvára podmienky cieľa)

Hierarchické plánovanie - rozloženie problemu na pod-problemy

Príklad: Hranisko v ráre

objekty: tyč, disk, stôl

predikaty: menín(d₁, d₂)
 top(1, tyč, disk)
 below(d₁, d₂)

init: at(1, *), menín(...), top(1, nejmenší disk)
 below(...), below(nejväčší disk, stôl)

goal: top(1, stôl), top(2, stôl)

akcie: move(from, to, disk, cilový disk, pred)

prediktory: top(from, disk)
 below(disk, pred)
 top(1, cil, cilový disk)
 menín(disk, cilový disk)

efekty: top(from, pred), not top(from, disk)
 top(1, disk), not top(1, cil)
 below(disk, cil), not below(disk, pred)

Plan-Space Planning

open-goal = aktív, čo ještě nesplní, riešiť

\Rightarrow nájsť aktív, čo ho splní - jiní prediktory - noví open goals

koncová výzva = jedna aktív, čo vás ťažšie řešiť

\Rightarrow riešiť, že niektoré aktív sa musí udeliť pred. jinou aktívou

\Rightarrow start = aktív bez ťažšie řešiť, mostrov ťažšie řešiť / cilový aktív = aktív s ťažšie řešiť aktívou

Probabilistic reasoning

- prostředí může být číslované - prowarovatelné / nedetermin.
- logical agent
 - může pracovat s belief states = mísit možného stavu světa
 - ⇒ contingency plans - handle every possible eventuality
↳ velké, komplexní
 - vše je buď true / false
- probabilistic agent - slouží dle výroku $\in [0, 1]$
 - možné světy $\omega \in \Omega$, $0 \leq P(\omega) \leq 1$ → diskretní
 - $\sum_{\omega \in \Omega} P(\omega) = 1$ → pravděpodobnost
 - jazyk (events)

$$P(A) = \sum_{\omega \in A} P(\omega) \quad \rightarrow \text{psáme } P(A \cap B) := P(A, B)$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$\rightarrow \text{product rule: } P(A \cap B) = P(A|B) \cdot P(B)$$

→ marginalizace:

$$P(Y) = \sum_{Z \in \mathcal{Z}} P(Y, Z) = \sum_{Z \in \mathcal{Z}} P(Y|Z) \cdot P(Z)$$

- hidden prob P
odpojí se z výpočtu
- Y je proměnná → vlastní dílčí vektor pro výpočet
možné hodnoty Y
- normalizace → jde o dležitých \approx fázích v množině
- $$P(Y | E=e) = P(Y, E=e) / P(E=e) = \alpha \cdot P(Y, E=e)$$
- ↳ málo hodná proměnná pro evidence, e je cožsem viděl
- nyní, že na horizontu má vystát suma všech vektorů 1
- $$\sum_{y \in \text{Im}(Y)} P(Y=y | E=e) = 1$$
- ⇒ tedy $P(E=e)$ ignoruje a na horizontu provede normalizaci
- hidden random vars
- $$P(Y | E=e) = \alpha P(Y, E=e) = \alpha \sum_h P(Y, E=e, h) \rightarrow \text{často mi tak usnadní výpočet}$$

→ merávalos

$$x \perp Y \Rightarrow P(X|Y) = P(X), \quad P(X, Y) = P(X)P(Y)$$

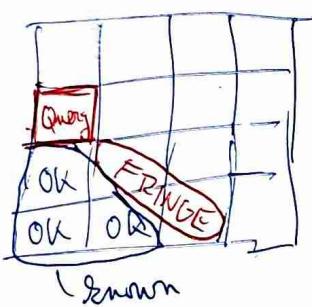
$$x \perp c \mid Y \Rightarrow P(X|Y, C) = P(X|C)$$

$$P(X, Y|C) = P(X|C) \cdot P(Y|C)$$

reduzible
zu Tabelle

→ male Adultery (♂ dimensi messi)

Wurzfus:



$$P_{i,j} = \text{pit at } (i,j) - \text{pro } f_{i,j}$$

$$p_{i,j} = \text{breere at } (i,j) - \text{pro naastvriene politie}$$

$$\text{known} = \neg f_{1,1} \wedge \neg f_{1,2} \wedge \neg f_{2,1}$$

$$\text{b} = \neg b_{1,1} \wedge \neg b_{1,2} \wedge \neg b_{2,1}$$

$$\Rightarrow P[P_{1,3} | \text{known}, \text{b}] = ?$$

$$\rightarrow \text{primarie: } P = \alpha \cdot \sum_{\text{unknown}} P(P_{1,3} | \text{unknown}, \text{known}, \text{b})$$

$\hookrightarrow P_{i,j}$ erne P_{1,3} a known $\Rightarrow 2^2$ cílení s mít

$\Rightarrow \text{unknown} = \text{Query} \cup \text{Fringe} \dots \text{Pits}$

$$P = \alpha \sum_{\text{unknown}} P(P_{1,3}, \text{unknown}, \text{known}, \text{b}) \quad \stackrel{\text{Bayes}}{\hookrightarrow} \quad P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

$$= \alpha \sum_{\text{unknown}} P(b | P_{1,3}, \text{known}, \text{unknown}) \cdot P(P_{1,3}, \text{known}, \text{unknown})$$

$$= \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b | P_{1,3}, \& \text{fringe}, \text{other}) \cdot P(P_{1,3}, \&, \text{fringe}, \text{other})$$

$$= \alpha \sum_{\text{fringe}} \sum_{\text{other}} P(b | P_{1,3}, \& \text{fringe}) \cdot P(P_{1,3}, \&, f, o)$$

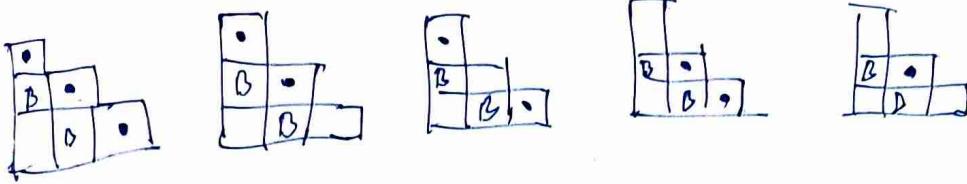
$$= \alpha \sum_{\text{fringe}} P(b | P_{1,3}, \& f) \sum_{\text{other}} P(P_{1,3}, \&, f, o)$$

$$= \alpha \sum_{\text{fringe}} P(b | P_{1,3}, \& f) \cdot \sum_{\text{other}} P(P_{1,3}) \cdot P(\&) \cdot P(f) \cdot P(o)$$

$$= \alpha \cdot P(\&) \cdot P(P_{1,3}) \cdot \sum_{\text{fringe}} P(b | P_{1,3}, \& f) \cdot P(f) \cdot \sum_{\text{other}} P(o) = 1$$

$$= \alpha \cdot P(P_{1,3}) \cdot \sum_{\text{fringe}} P(b | P_{1,3}, \&, f) \cdot P(f) \rightarrow |\text{fringe}| = 2 \Rightarrow 2^2 \text{ mohli}$$

$p = 0.2$... prob of pit



$$P(P_{1,2} | \text{Zum}, \delta) = \alpha \langle 0.2(0.04 + 0.2 \cdot 0.8 + 0.8 \cdot 0.2), 0.8(0.04 + 0.2 \cdot 0.8) \rangle \\ = \langle 0.31, 0.69 \rangle$$

\downarrow zum \downarrow δ

Bayesovo pravidlo: $P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)} = \alpha \cdot P(X|Y) \cdot P(Y)$

Nášom Bayesovský model

$$P(\text{Rúčka} | \text{Rúčka}) = P(\text{Rúčka} | \text{Rúčka}) \cdot P(\text{Rúčka}) / P(\text{Rúčka})$$

$$\rightarrow P(\text{Cause}, \text{Effect}_1, \dots, \text{Effect}_n) = \\ = P(\text{Cause}) \cdot P(E_1, \dots, E_n | \text{Cause}) = \\ = P(\text{Cause}) \cdot \prod_i P(E_i | \text{Cause})$$

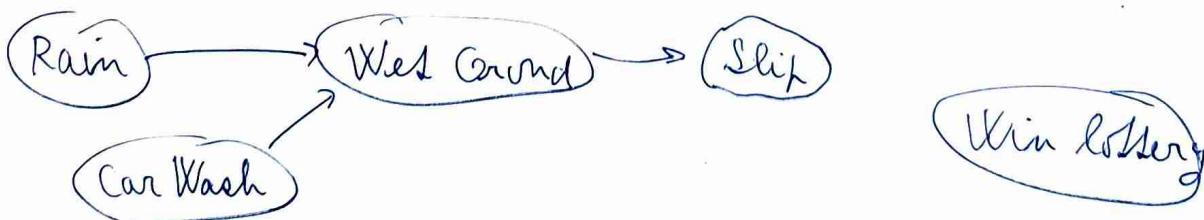
\hookrightarrow pretože vypočítame nezávislosť effectov na jednotlivých príčinach

Bayesovi súťaži

- reprezentácia vztahu medzi výskumom a rezultátmi meri prameňmi

- výsledky \sim prameňmi
- predchádzajúci =: parents

\rightarrow kaviarek vtedy X má poslúžiť distribuciou $P(X | \text{Parents}(x))$

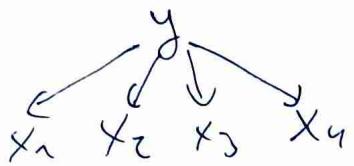


$$P(\text{Rain}, \text{Wet}, \text{Car}, \text{Slip}, \text{Win}) = P(R) \cdot P(C) \cdot P(W|R, C) \cdot P(S|W) \cdot P(L)$$

\rightarrow full joint probability distribution

$$\Rightarrow P(X_1, X_2, \dots, X_n) = \prod_i P(X_i | \text{Parents}(X_i))$$

Pričíp Bayes



$$P(y, x_1, x_2, x_3, x_4) = P(y) \cdot \prod_i P(x_i | y)$$

↳ naivé Bayes

Konstrukcia Bayesovské siťe

→ daktineme množinu náhodných premených X_1, \dots, X_m

↳ chceme vyzrieť D.S. aby $P(X_1, \dots, X_m) = \prod_i P(X_i | \text{Parents}(X_i))$

⇒ usporiadáme premené ako X_1, \dots, X_m

↳ ideálne aby príčiny boli siedmokrát nezávislé

→ Hraný vyzriebenie Axiom:

X_1 nehráde musí pôsobiť

$X_i \dots \in$ množina $\{X_1, \dots, X_{i-1}\}$ vyzriebene vyzierajúci podmienku aby $P(X_i | \text{Parents}(X_i)) = P(X_i | \{X_1, \dots, X_{i-1}\})$

→ prečo to funguje?

$$P(X_1, \dots, X_m) = P(X_m | X_1, \dots, X_{m-1}) \cdot P(X_1, \dots, X_{m-1})$$

$$= P(X_m | X_1, \dots, X_{m-1}) \cdot P(X_{m-1} | X_1, \dots, X_{m-2}) \cdot P(X_1, \dots, X_{m-2})$$

$$= \prod_i P(X_i | X_1, \dots, X_{i-1}) = \prod_i P(X_i | \text{Parents}(X_i))$$

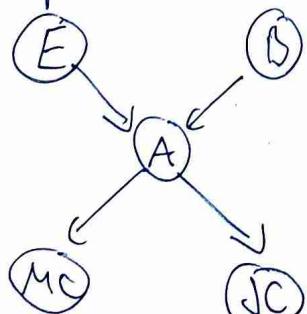
Pričíp

- alarm proti kremičkam, následne racionálne pri konferencií

↳ John volá Edgára slyši alarm, ale občas si to opláče s telefonom

↳ Mary —————— n ——————, ak občas ho nevyslúší

→ pričípová D.S.



Pričíp: MC, JC, A, B, E



... súvisi s tým reprez.

$$P(MC | JC) \neq P(MC | \bar{JC})$$

$$\dots P(A | MC) \neq P(A | \bar{MC}), \text{ pretože } \bar{JC}$$

$$\dots P(D | A) = P(D | A, JC)$$

$$\dots P(E | A, B) \neq P(E | A, \bar{B}) = 1$$

Inference v Bayesovské síti

stejně formule

$$P(b|j,m) = \alpha \cdot P(b,j,m) = \alpha \cdot \sum_e \sum_a P(b,j,m, e, a)$$

→ ale pokud ten \prod je nefaktoriální $\prod_i P(x_i, \text{Parents}(x_i))$

→ když když jsou vytvořil srovnatelnou síť, tak asi nejdřív něčeho $P(A|SC, MC)$

→ nebo rozumějící síti bych dostal

$$\begin{aligned} P(b|j,m) &= \alpha \sum_e \sum_a P(e) P(b) P(a|e, b) \cdot P(m|a) \cdot P(j|a) \\ &= \alpha P(b) \cdot \sum_e P(e) \sum_a P(a|e, b) \cdot P(m|a) \cdot P(j|a) \end{aligned}$$

→ některé říci budu fiktivní výčet

→ něčeho $P(m|a)$ je stejně, když máme e a nejsou s něčím spojeny

→ má tostromovou strukturu

⇒ dynamické programování

Eliminace proměnných

$$P(B|j,m) = \alpha P(B) \cdot \sum_e P(e) \sum_a P(a|e, B) \cdot P(m|a) \cdot P(j|a)$$

$$= \alpha f_1(B) \cdot \sum_e f_2(e) \cdot \sum_a f_3(A, B, e) \cdot f_4(A) \cdot f_5(A) \quad \begin{matrix} m, j \\ \text{jsou} \\ \text{konstanty} \end{matrix}$$

→ pro B už máme nebo síti kolmou formulované počí: CPT

→ f_1, \dots, f_5 = factory, vyhodnocují když se počítá oblast

$f_4(A) \cdot f_5(A) \dots$ využívají k výpočtu pro abecedních $A = \text{True}$ a $A = \text{False}$

$f_3(A, B, e) \cdot f'(A) \rightarrow$ vznikne nová oblast velikosti $f'(A, B, e)$
 ↳ když vložíme výpočetní hodnotu počí A upraví

složitější situace

A	B	$f_1(A, B)$	B	C	$f_2(B, C)$
T	T	0.3	T	T	0.2
T	F	0.7	T	F	0.8
F	T	0.9	F	T	0.6
F	F	0.1	F	F	0.4

A	B	C	$f'(A, B, C)$	$\sum f(A, B, C)$
T	T	T	0.3 \cdot 0.2	
T	T	F	0.3 \cdot 0.8	
T	F	T	0.7 \cdot 0.6	
T	F	F	0.7 \cdot 0.4	
:	:	:		

→ příklad se zde
 složitější pro všechny
 hodnoty A
 a doslouží $f'(B, C)$

Monte-Carlo method

- sam plýtěme, odkud řek
- kde v instance málohyd působí
- jak to generovat?
 - ↳ sítí topologicky nejdále a nejdále
 - ⇒ místní řek počítat ty dodatečné řeky distribuce
- ale my mísíme nějaký evidence a

• Rejection sampling: vzdály nekonistentní s e ignorují

$$\Rightarrow P(X|e) = \frac{\# \text{ řek plati } X \text{ i.e.}}{\# \text{ řek plati } e}$$

• Likelihood weighting

→ generujeme jen vzdály konistentní s e

⇒ hledáme působení cívané a evidence zafixují

→ ale když by mifigovalo

$$\hookrightarrow \text{chci vzhledem } P(X|e) = \frac{P(X, e)}{P(e)}$$

↪ vlastně bych rád řek $P(e) = 1$

⇒ když nejdou soubory řek, řek mu působení na řeku

$$w(z, e) := \prod_i P(e_i | \text{Parents}(e_i))$$

$$\Rightarrow P(X|e) \approx \frac{1}{\# \text{ řek plati } X \text{ i.e.}} \cdot w(X, e)$$

Decision making

- Transition model → define $P(X_s | X_{0:s-1})$
 - Markov assumption: $P(X_s | X_{0:s-1}) = P(X_s | X_{s-1})$
 - forward pass for 'walking a path step by step' → 'stochastic' model
 - Sensor (observation) model → $P(E_s | X_{0:s}, E_{1:s-1})$
 - Markov assumption: $P(E_s | X_{0:s}, E_{1:s-1}) = P(E_s | X_s)$
 - Use so called forward Bayes rule since $X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow \dots$
- $$P(X_{0:s}) = P(X_0) \prod_{t=1}^s P(X_t | X_{t-1})$$
- $$P(X_{0:s}, E_{1:s}) = P(X_0) \prod_{t=1}^s P(X_t | X_{t-1}) \cdot P(E_t | X_t)$$

Inference tasks

1) Filtrering: Where am i now?

$$P(X_s | \mathcal{E}_{1:s}) =: f_{1:s}$$

$$f_{1:0} = P(X_0)$$

$$\begin{aligned} f_{1:s+1} &= P(X_{s+1} | \mathcal{E}_{1:s+1}) = P(X_{s+1} | e_{1:s}, e_{s+1}) \\ &= \alpha \cdot P(e_{s+1} | X_{s+1}, \mathcal{C}_{1:s}) \cdot P(X_{s+1} | \mathcal{C}_{1:s}) \quad \dots \text{Bayes rule} \\ &= \alpha \cdot P(e_{s+1} | X_{s+1}) \cdot P(X_{s+1} | \mathcal{C}_{1:s}) \quad \dots \text{Markov assumption} \\ &= \alpha \cdot P(\mathcal{C}_{s+1} | X_{s+1}) \cdot \sum_{X_s} P(X_s | \mathcal{C}_{1:s}) \cdot P(X_{s+1} | e_{1:s}, X_s) \\ &= \alpha \cdot P(\mathcal{C}_{s+1} | X_{s+1}) \cdot \sum_{X_s} f_{1:s} \cdot P(X_{s+1} | X_s) \quad \dots \text{Markov assumption} \end{aligned}$$

2) Prediction:

$$P(X_{s+\varepsilon} | \mathcal{E}_{1:s}) = ?$$

$$P(X_s | \mathcal{E}_{1:s}) = f_{1:s}, \quad P(X_{s+\varepsilon+1} | \mathcal{E}_{1:s}) = \sum_{X_{s+\varepsilon}} P(X_{s+\varepsilon} | \mathcal{E}_{1:s}) \cdot P(X_{s+\varepsilon+1} | X_{s+\varepsilon})$$

3) Smoothing \rightarrow where was I in the past?

$$P(X_\ell | e_{1:s}), \quad 0 \leq \ell < s.$$

$$\rightarrow P(X_\ell | e_{1:s}) = P(X_\ell | e_{1:\ell}, e_{\ell+1:s})$$

$$= \alpha P(e_{\ell+1:s} | X_\ell, e_{1:\ell}) \cdot P(X_\ell, e_{1:\ell}) \quad \leftarrow \text{Bayes Rule}$$

$$= \alpha \underbrace{P(e_{\ell+1:s} | X_\ell)}_{b_{\ell+1:s}} \cdot \underbrace{f_{1:s}}_{\text{Markov assumption}} \quad \text{-- Markov assumption}$$

$$\rightarrow b_{\ell+1:s} = P(e_{\ell+1:s} | X_\ell) = \sum_{X_{\ell+1}} P(X_{\ell+1} | X_\ell) \cdot P(e_{\ell+1:s} | X_\ell, X_{\ell+1})$$

$$= \sum_{X_{\ell+1}} P(X_{\ell+1} | X_\ell) \cdot P(e_{\ell+1:s} | X_{\ell+1}) \quad \text{-- Markov}$$

$$= \sum_{X_{\ell+1}} P(X_{\ell+1} | X_\ell) \cdot P(e_{\ell+1}, e_{\ell+2:s} | X_{\ell+1})$$

$$= \sum_{X_{\ell+1}} P(X_{\ell+1} | X_\ell) \cdot P(e_{\ell+1} | X_{\ell+1}) \cdot P(e_{\ell+2:s} | X_{\ell+1})$$

\hookrightarrow meránulos \rightarrow je videt $\in \mathbb{B-S}$.

$$= \sum_{X_{\ell+1}} P(X_{\ell+1} | X_\ell) \cdot P(e_{\ell+1} | X_{\ell+1}) \cdot \underbrace{b_{\ell+2:s}}$$

$$\rightarrow b_{s+1:s} = P(e_{s+1:s} | X_s) = P(\text{nic} | X_s) = 1$$

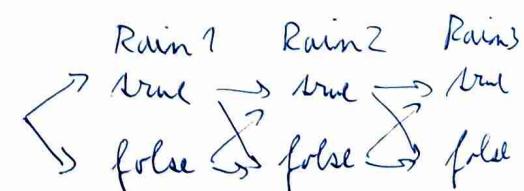
\hookrightarrow base-case

4) Most likely explanation

\rightarrow chia mojist sekvenci sahnu litera' nejsprv vygenerovala pravou

$$\underset{X_{1:s}}{\operatorname{argmax}} P(X_{1:s} | e_{1:s})$$

\rightarrow doda' sekvence $X_{1:s}$ je cesta grafem

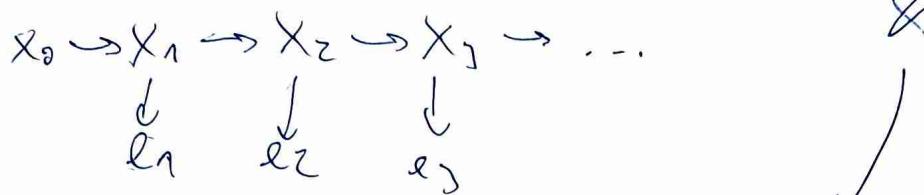


$$\max_{X_{1:s}} P(X_1, \dots, X_s, X_{s+1} | e_{1:s+1}) =$$

$$= \alpha \cdot P(e_{s+1} | X_{s+1}) \cdot \max_{X_s} \left\{ P(X_{s+1} | X_s) \cdot \max_{X_{1:s-1}} P(X_1, \dots, X_s | e_{1:s}) \right\}$$

Hidden markov models

→ řešení pro mís model plní Markovské předpoklady, kde x_t je HMM



→ jediná proměnná x , kterou můžeme

→ počítat jedinou proměnnou E

↳ tento jednoduchý model lze reprezentovat následně

• Transition model je matici $\in \mathbb{R}^{S \times S}$, kde $\text{Im}(X) = \{1, \dots, S\}$

$$T_{(i,j)} = P(x_s=j | x_{s-1}=i)$$

• Sensor model

$$O_{s(i,j)} = P(E_s = e_s | x_s = i) \quad \rightarrow \text{diagonální matici}$$

→ filtering a smoothing, resp. decoding

• f = forward message propagation

• b = backward message propagation

se dojde implementovat formou maticového nastavení

→ například lokalizace robota podle číselného sekvence

↳ mám evidence $e_1, \dots, e_m \rightarrow$ čai X_s

Dynamické Bayesovské sítě

– reprezentuje fkt. v rámci

– správnějsi replikované množiny šířek

– jediná proměnná má rozhil lodi ve stejném / předchází časovém intervalu
 ↳ markov assumption

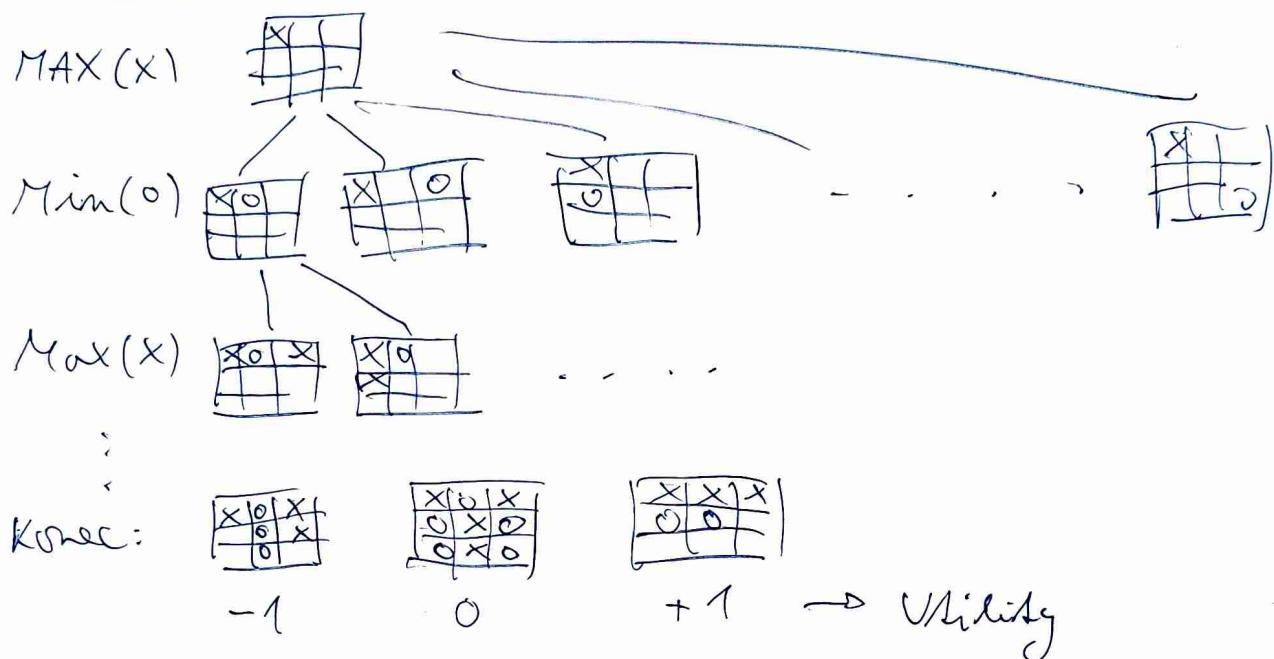
→ HMM je speciál case DSS, ale DSS lze reprezentovat jako HMM

↳ ale je to exponenciálně mnohem výši

Hry a multi-agentní systémy

- nejvíce nějsí hry: deterministické, dva bráni, střídání kohu, zero-sum, plné pozorovatelné
 ↳ Šachy, Go, Piatrovky...

• Minimax

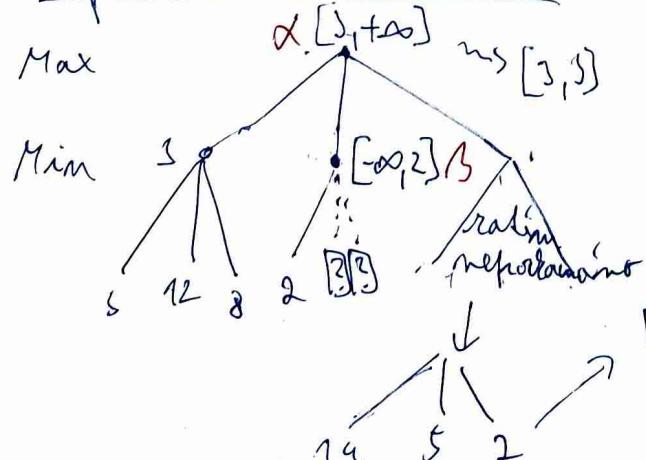


Def: Pro stav n definujeme $\text{Minimax Value}(n)$ jako

- $\text{utility}(n)$... když n je terminální stav
- $\max_{\text{sefornici}(n)} \text{Minimax Value}(s)$... Max je na vrchu
- $\min_{\text{sefornici}(n)} \text{Minimax Value}(s)$... Min je na vrchu

→ v kořeni je min / max, podle toho co je akt hry

• Alfa-Beta projevování



α = nejlepší hodnota pro MAX

β = nejlepší pro MIN

↳ max vrátí hodnotu ne větší

↳ řada myšlenkování potomku je zrušena

↳ když pochádza fronta se nejlepší hodnotou

↳ moží nejméní větší hodnotu

↳ heuristikou

Evaluacion function

- reálné využití stromu až mnoho razy generují
- výkonná prohledávání v mějoté houbce
 - a optimuje utility funkci evaluation function = heuristika
- majdu mějoté rozmí features (Sachy)
 - ↳ # pěšáků
 - ↳ # figur celkem
 - ↳ is queen alive
 - ↳ is back / garde
- k vidění příkladům načtu → EVAL je nejdále lin. kombinace

Stochastické hry

- random element - hrazení kostkou
- formální Expected Minimal Value = EMV(n)
 - 1, : Utility(n) ... n je terminální
 - 2, $\sum_{s \in \text{postomi}(n)} P(s) \cdot \text{EMV}(s)$... n je random-nal
 - 3, $\max_s \text{EMV}(s)$... Max hraje
 - 4, $\min_s \text{EMV}(s)$... Min hraje

• Single - move games

- všichni hráči procházejí a mají jen 1 krok

→ payoff function dává hráči utilitu podle toho jak hráč vypadá

→ Two-finger Moran

hráči Odd a Even mítou 1 nebo 2 prsty

$$\text{pravidlo: } \begin{cases} O=1 \\ E=1 \end{cases} \quad \begin{cases} V(O) = -2 \\ V(E) = 2 \end{cases}$$

$$\begin{cases} O=2 \\ E=1 \end{cases} \quad \begin{cases} V(O) = 3 \\ V(E) = -3 \end{cases}$$

→ policy = strategie

- pure - deterministická
- mixed - randomizovaná

→ prisoners dilemma

Alice a Bob chycení, můžou svědčit, jdou do vězení

		A: T	A: $\neg T$
		A = -5 B = -5	A = -10 B = 0
		A = 0 B = -10	A = -1 B = -1
B: T			
B: $\neg T$			

Racionální strategie je Testify - je to dominant pure strat.

Def: Strategie S pro hráče f dominantní strategie S' = výsledek S je pro f lepší než výsledek S' pro každého můžou volbu strategie oslovitelného hráče

Def: Nash equilibrium: ráčig hráč si neponáje, pokud nemají strategii, ještěkdy ji oplatí nezměnit

Def: Výsledek hry je Pareto dominated jiném výsledku, jestliže by ho všichni hráči preferovali.

Def: Outcome je Pareto optimal, když jež některé

→ prisnes dilema vrahů, když má domíne strategie

≈ Nash eq. (Testify, Testify), která je Pareto dominated výsledkem (refuse, refuse)

• Maximin technique - rovnava mezi pure strategii (optimální)

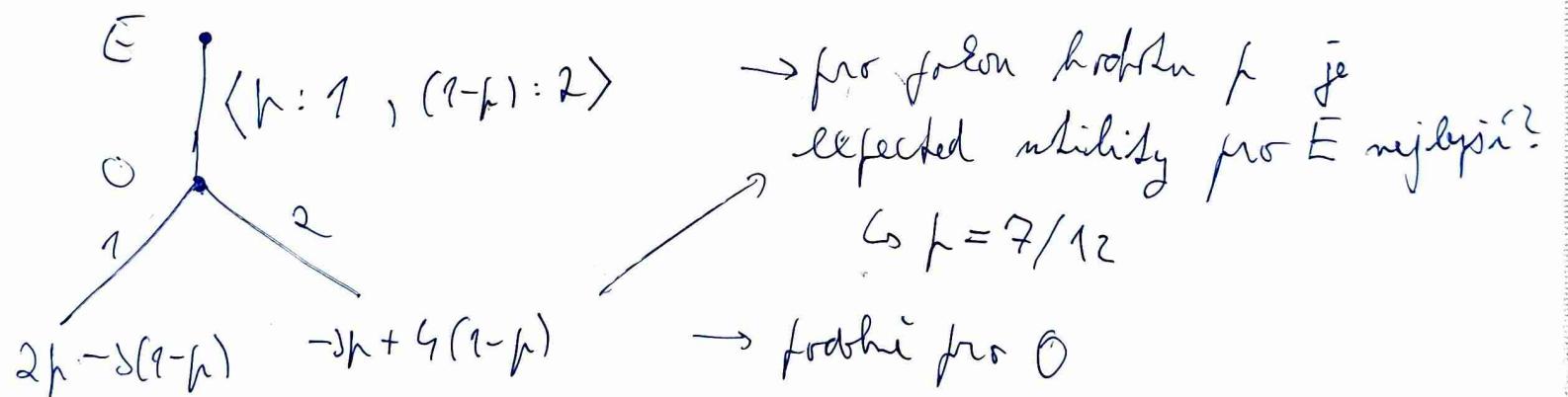
→ hledáme se hraček, když je Turn-Taking game

⇒ druhý hraček má výběr

→ první hraček hraje své mixed strategii

↳ druhý může as well své faire

→ posl. 1 prstou = μ



→ pro další hračku μ je
očekávaný výsledek pro E nejlepší?

$$\hookrightarrow \mu = 7/12$$

→ probabilita pro 0

⇒ optimální strategie pro oba hračky $[7/12: 1, 5/12: 2]$

Repeated games

- hráči mohou riešiť stejný problem a mať historiu od predch. obehov hráčov
- payoff sa sčítá
- strategie proverbal players dilemma
 - hraje 100 her
 - vlastné ríkani len na si poslednú heru
 - ⇒ rovnakože je pravidlo Testify ⇒ všichni 100 her riešia
- fct, že hráč dobieha hráč = 0.9a.
 - ⇒ viera vo všetky hráče je fct 100, ale nst. sure
 - perpetual punishment: refuse, akend druhý hráč nedá testify
 - po 100 her Testify
 - tit-for-tat
 - hraje refuse a po 100 her má možnosť opísť ďalšiu fct
 - ↳ však funguje prekročenie dobieha

Theorie her analýzujú chváli a agresiu

Inverenčné hery sa súvisí s definíciou postrieb, aby celkový payoff pre všetky agresy bol čo najlepší

↳ Mechanism design

↳ písané = autue

Mechanism design

Aneb funguje tak, že rády Bidder má své vlastní hodnoty pro daný předmět, a ten představuje mít common value (je něco mezi nimi)

→ nížší individual bid

higher value / revenue

↳ nejvyšší bid dostane item

Good mechanisms

- maximizuje očekávanou revizi pro prodajce
- maximizuje globální výhodu - vyhovuje tomu, že si lidé si nemají nejvíc

1) Ascending-bid auction

- ráčíme s minimem bid_{min}

→ pokud se někdo již schytá rozložit → posledním biderem b_{min+d}

→ pokud je, dokud ne někdo nedá víc

• simple dominant strategy = bid iff price $\leq v_{xi}$

• odradí ostali / pokud sam je Elon Musk

2) descending-bid auction

- ráčíme výšky a snížíme, dokud se někdo nekonfí

• rychlejší → prodaj → rychlejší → rychlejší

3) sealed-bid auction

- nížší fiktivní bid v obálce → nejvyšší vyhrají

• něma dominant strategy

4) sealed-bid second-price auction

- jako sealed-bid, ale vyhřeje rozhodně druhou nejvyšší cenu

⇒ dominant strategy = bid v_i

Tragedy of Commons

- common goals: pollution
- same but 'roflask' - to za akce to zlepšit'
- nebo -5 na zdraví občanů a -1 pro svobodu
- dominant je mi meditací
- aktivní houť pro mědy
- náleží nás mědy Vicker-Claire-Groves mechanismus

Decision Trees

- Syg machine learning: výzva vektorem na vrchol funkce hledání
- je to strom, ve vnitřních větvích jsou testy, přesnější
je vše odřídi → kroužky ~ odřídi.
- listy = return values

Hypothesis space = možna shromažďovat jen konzistentní se
vstupními daty → chceme řešit nejednotlivé

Jak vytvořit strom?

↳ Okamžitý běh

- výděl a peníž
- vzdálené nejdůležitější atribut a tím dle dr. Zájmu
- atributy: výběrám restauraci → hungry?, Type rest.?, Day of Week?
- odřídi: jít / nejít

- máme možné exempláře ex.

↳ hungry lude asi hodně: Not hungry → No
Hungry → spis Yes

- ↳ day of week: asi půl na půl, moc info mi to neřídí

- formálně to provádí stříška s výpočtem entropie

→ Zde je nám nejdůležitější atribut, zaté examples rozdělím podle hodnot toho atributu

Hungry

Not Hungry

→ Sedí se rekurencií zavolám na týkající dve bramádky dok, ale ignoruju atribut Hungry

→ at some point mi nebude mít rády využívat atribut nebo všechny examples srovnou do stejné bramádky → list

• Logická klasifikace

- bee hr rozdělení formule

Type(x_1 , French).

→ atributy jsou predikáty

$\neg \text{Hungry}(x_1) \wedge \text{Fri/Sat}(x_1) \wedge \dots$

- rozlišování jsou svý predikáty

fm: Will Work (r) $\Leftrightarrow \text{Hungry}(r) \vee (\neg \text{Hungry}(r) \wedge \text{Type}(r, \text{French})) \vee \dots$

→ Hypothéza je schváhlitelná formulka pro hardon, záruk

→ vnitřne, ře existuje nějaká hypotéza konzistentní se všemi předlohy

Jak bych inconsistent?

- false negative - říčím NE ale je ANO
- false positive - říčím ANO ale je NE

→ Jak získej hypotézu?

1) Current-best-Hypothesis

- formuloju si ten ? hypotézu a vylepšuj ji postupně example

- potom example je

• Konsistencie + hypotéza → mi není

• false negative - hypotéza je moc přísná → generalizace
⇒ obecné počínky (x), nebo specifické možnosti (v)

• false positive ⇒ specializuj: píčidlo (x), nebo obecné (v)

2) Least commitment search

→ nemám jen 1 hypoten, ale všechny, co jsou konsistentní s všemi všechny examples

Version space

→ jak efektivně reprezentovat version space?

→ máme boundary sets

• G-set = most general boundary

↳ můžeme říct, že daným jen True = vše projde

→ pro k následující example:

• false positive pro $h \in G$

⇒ mohu mít $h \in G$ za všechny, iimmediate generalizace h'

• false negative pro $h \in G$

⇒ mohu mít $h \in G$ - je méně specifické

• S-set = most specific boundary

↳ můžeme říct, že

následující example

• false positive ⇒ mohu mít $h \in S$

• false negative ⇒ mohu mít $h \in S$ za generalizace h'

→ všechno „merí“ G a S je konsistentní se všemi examples
→ (ne lineární)

↳ můžem mít několik nesplňujících mě hypothesis space

↳ abych mohl mít co fj. immediate generalizace / generalizaci

Statistické mély

- bodové odhady - metoda momentů
 $\hat{\theta} = \text{most likely explanation}$

↳ představují maximální nezávislost a stejnou distribuci

↳ využívají funkci pravd. → rozdíl mezi funkční

$$L(x_1, \dots, x_n; \theta) = P(X_1=x_1, \dots, X_n=x_n; \theta)$$

→ logaritmickým m.

$$\ell(\bar{x}, \theta) := \log L(\bar{x}, \theta) = \prod_i f_i(x_i)$$

diskrétní ↳ spojité

$$\ell(\bar{x}, \theta) := \log L(\bar{x}, \theta)$$

→ najde maximum pomocí derivace podle θ

Expectation maximization alg.

→ máme: několik hidden variable, která obsahují data, ale neznáme jejich hodnoty

1) pustíme, že všechny parametry funkce modelu jsou funkce

2) inferujeme expected hodnoty skrytých proměnných, abych "adoplnil" data - máme Bayesovskou sítě

3) počítáme se, jestli to sedí s modelem

↳ je to most-likely explanation?

→ updatuje parametry (funkce pro všechny)

Decision Theory

$R_a(s)$ = reward za akci a v reakciu.

↳ n deterministický postup

→ Nedeterministický, faktičky obzehovat

$R(a)$ → mimoždu vlivem důraznější akce na

Def: Expected utility akce a) je funkce jen racionálního
dostatečného postupu e (vztahy mezi sítom) je

$$e = e_1, e_2, \dots, e_s$$

$$EV[a|e] := \sum_s P[R(a)=s|a,e] \cdot U(s)$$

↳ utility skala

Maximum EU principle: akce $= \arg \max_a EV(a|e)$

Utility theory

→ nej výše obecně myslitelná utility skala, je pro agenta
leží v řadě, jestli se mu někdo líbí stav A nerozdíl stav B

↳ použití selection

→ chove myšlenku utility funkce, aby

$$U(A) < U(B) \equiv A < B$$

$$U(A) = U(B) \equiv A \sim B$$

→ nejlepší stav $S_{\max} \Rightarrow U=1$ } $U \in [0, 1]$
nejhorší stav $S_{\min} \Rightarrow U=0$

↳ Jakou hodnotu S?

→ zastavíme se agenta, jestli da' pohled na S nebo

loterie s pravd. p meri S_{\max} a S_{\min} : $\langle p: S_{\max}, (1-p): S_{\min} \rangle$

→ binárním myšlením najde 'soltore' p, tedy
pro agenta $S \sim$ loterie (p)

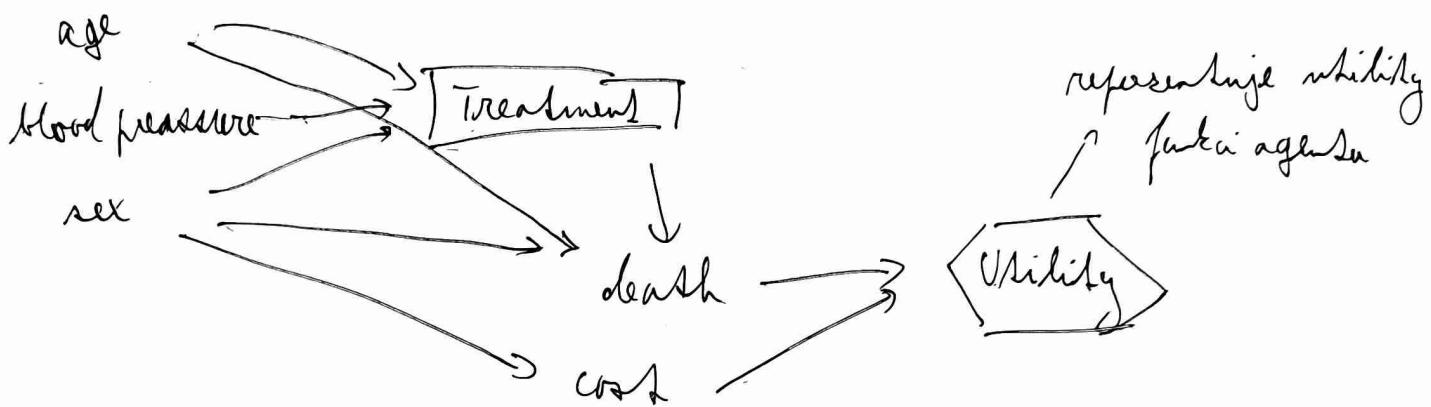
$$\Rightarrow U(S) := p$$

Decision networks

→ Bayesovské sítě, do kterých formou větu pro mimořádné funkce
dáme i decision nodes

↳ decision node je samozřejmě nějakou akci

→ utility nodes → reprezentují utility pro agenta



proces:

1. nastavim hodnoty funkcií cestou z evidence

2. pro každou hodnotu decision větu

a) nastav hodnotu větu

b) spočítaj postupně při ročku/předku utility node
↳ normální bayesovská inference

c) urči utility pro koho a kai

3) urči aktuální utility

Sequential decision problems

- agent se má plni rozhodnutí a nejde o nějakou funkci

→ Saturn: robit robot, \uparrow NORTH, \downarrow WEST, \leftarrow SOUTH, \rightarrow EAST

↳ funkce je nebezpečné, využití jednotlivé funkce

↳ využití, kde jsou a čas dojít druhé na co nejdříve

Def: Následující rozhodovací proces

$$P_a(s, s'), R(s)$$

$$\Rightarrow celková recompence R(s_0, s_1, \dots) = \sum_{i=0}^{\infty} \gamma^i R(s_i)$$

→ vlastní politika $\pi: S \rightarrow A$

→ očekávaná recompence $V^\pi(s) := \mathbb{E}[R^\pi | s_0 = s]$, $a_i \sim \pi(s_i)$

$$\begin{aligned} \text{Bellmanova rovnice:} \\ V^\pi(s) &= \mathbb{E}_{a, s_1} \left[r_0 + \sum_{i=1}^{\infty} \gamma^i R(s_i) \mid s_0 = s \right] \\ &= \mathbb{E}_{a, s_1} \left[r_0 + \gamma V^\pi(s_1) \mid s_0 = s \right] \end{aligned}$$

$$\Rightarrow V^\pi(s) \approx r_0 + \gamma \cdot \max_a \mathbb{E}_{s'} [V^\pi(s')]$$

$$V(s) = r_0 + \gamma \cdot \max_a \sum_{s'} P_a(s, s') \cdot V(s')$$

↳ pro kriticky optimální recompence / politiku

↳ kde má kdo svou roli

$$\pi^* = \arg \max_a \sum_{s'} P_a(s, s') \cdot V^{\pi^*}(s')$$

• Value iteration

→ máme nové fce hrdý stav, jejíž řešení je optimální maticová V

→ ale řešení nejsou lineární :

1. V zkráceního maticové

2. iterativní formou vzhledem k součtu vlastních vektorů

$$V(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} P_a(s, s') \cdot V(s')$$

→ po delším, obecně se do nějaké rovnice mísí

→ pokud je maximální změna hodnoty v matici $< \epsilon \Rightarrow$ konec

• Policy iteration

④ je možné mít optimal policy a řešit V nejoptimální

1. V spíš maticové, π maticové

2. iterativní

$V \leftarrow$ násobek policy (dole)

→ pokud pro nějaký stav $s \in S$ existuje alespoň $a \in A$ t.ž.

$$EV(s|a) > EV(s|\pi(s)), \text{ tak } \pi(s) \leftarrow a$$

→ pokud se policy nemění → konec

Vyrobek policy (π): vráží V^{π_i}

$$V^{\pi_i}(s) = R(s) + \gamma \cdot \sum_{s'} P(s'|s, \pi_i(s)) \cdot V^{\pi_i}(s')$$

→ když se dátové řešení formou Gaussovy

→ metr pro velké prostupy approximace formou value iteration

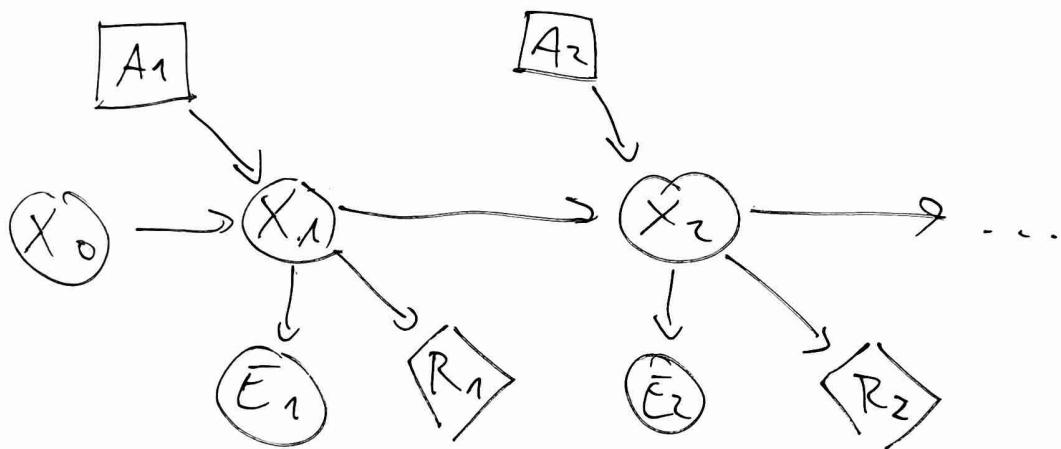
• Partially observable MDP

- problem fully observable

→ need more main sensor model $P(s|e)$

→ receive, etc. gene → belief states = post' distribution p(s |
reaching current' state)

→ resulting se as dynamic decision network



je vlastná má sit' byť? je tam diskont' faktor γ

sobie budoucnost nem' sú členiteľné

→ rozhoduchi' faktor sú, tie súc' filtering rozhoduchi' tie
fakt. a - riešenie nijaké' poslonyfakti' alej', výberu sú nezávisí'