1 Strojno ucenje

1.1 Problemski prostor, ocenjevanje znanja

1.2 Evalviranje hipotez

Pomembni kriteriji:

- **konsistentnost** hipotez z primeri (ucnimi)
- **splosnost** (tocnost za nevidene primere)
- **razumljivost** hipotez

TP=true positive, TN-true negative, FP-false positive (napaka 1. tipa), FN-false negative (napaka 2. tipa)

Klasifikacijska tocnost = $\frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{N}$

Obcutljivost/senzitivnost = $TPR = \frac{TP}{TP+FN}$

1.3 Gradnja odlocitvenih dreves

Za koliko se entropija zmanjsa po delitvi z Atributom A:

Informacijski prispevek: Gain(A) = $H(A) - I_{res}(A)$

$I_{res} = -\sum_{v_i \in A} p_{v_i} \sum_c p(c|v_i) \log_2 p(c|v_i)$

Razmerje inofrmacijskega prispevka atributa A:

IGR(A) = $\frac{\text{Gain(A)}}{\text{H(A)}}$

1.3.1 TDIDT (Top down induction decision tree) algoritem

Pozresen algoritem, ki **lokalno** izbira najbolsi atribut.

- kratkoviden algoritem

1.3.2 Binarizacija atributov

Aleternativa za resevanje problematike z vecvrednostnimi atributi: Strategije (za primer B = {Y, G, R, B}):

- $[\{Y\}, \{R, G, B\}]$ (one-vs-all)
- $[\{Y, R\}, \{G, B\}]$
- vpeljava bianrnih atributov za vsako barvo

Primer B = {Y, G, R}, konstruiramo 3 nove binarne atribute:

barva		Y	G	R
Y	$\rightarrow$	1	0	0
G		0	1	0
R		0	0	1

Prednost: manjse vejanje drevesa.

1.4 Ucenje iz sumnih podatkov (rezanje)

tocnost t...verjetnost pravilnosti klasifikacije

napaka e … $1-t$

relativna frekvenca $p = \frac{n}{N}$

m-ocena $p = \frac{n+p_a*m}{N+m}$

m… koliko zaupam apriorni verjetnosti

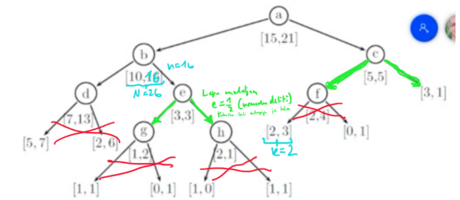
p_a apriorna verjetnost (domenski ekspert lahko pove)

Laplacova ocena verjetnosti $p = \frac{n+1}{N+k}$

k...stevilo vseh moznih razredov

1.4.1 MEP (Minimal Error Prunning)

e...staticna napaka,E...vzvratna napaka,$e \leq E \rightarrow$ rezemo poddrevo



(Laplace)

$e_L(d) = 1 - t = 1 - \frac{13+1}{20+2} = 0.363$

$E_L(d) = 12/20 \cdot e_L(d_l) + 8/20 \cdot e_L(d_d) = \frac{12}{20} \cdot (1 - \frac{7+1}{12+2}) + \frac{8}{20}(1 - \frac{13+1}{20+2})$

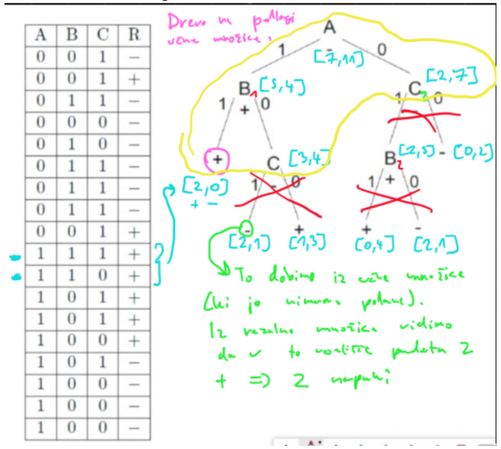
1.4.2 REP (reduced error prunning)

Dela dobro ce imamo veliko rezalno mnozico.

Obicajno uporabljamo relativno frekvenco za ocenjevanje verjetnosti.

$G(v) = \#\text{napak}_T - \#\text{napak}_v$

$G(v) \geq 0 \Rightarrow$ rezemo podrevo



A	B	C	R
0	0	1	–
0	0	1	+
0	1	1	–
0	0	0	–
0	1	0	–
0	1	1	–
0	1	1	–
0	1	1	–
0	0	1	+
1	1	1	+
1	1	0	+
1	0	1	+
1	0	1	+
1	0	0	+
1	0	1	–
1	0	0	–
1	0	0	–
1	0	0	–

$e(C) = 3$

$e_T = 2 + 3 = 5$

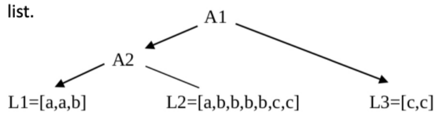
$G(C) = 5 - 3 = 2 \geq 0 \rightarrow$ rezemo

1.5 Ocenjevanje uspesnosti modelov

tocnost t … verjetnost pravilnosti klasifikacije

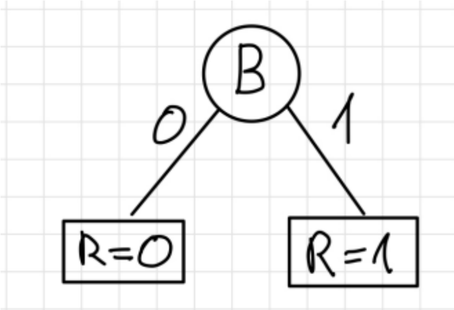
Laplacova ocena verjetnosti $p = \frac{n+1}{N+k}$

k...stevilo vseh moznih razredov



$t_{L1} = \frac{2+1}{3+3} = 0.5$, $t_{L2} = \frac{4+1}{7+3} = 0.5$, $t_{L3} = \frac{2+1}{2+3} = 0.6$

tocnost drevesa: $t_D = 3/12 \cdot 0.5 + 7/12 \cdot 0.5 + 2/12 \cdot 0.6 = 0.5167$



$e = 1 - (P(B=0)P(R=0|B=0) + P(B=1)P(R=1|B=1))$

1.6 Obravnava mankajocih atributov, navini Bayesov klasifikator

1.6.1 Naivni bayes

Ce poznamo razred, kam klasificiramo ce nepoznamo atributov:

$$\text{Klasifikator: } \text{argmax}_{c \in C} P(c) \prod_{i=1}^{n} P(x_i|c)$$

c…razred, x_i…atributi

Verjetnost::

$$P(C = c|x_1, \ldots, x_n) = \frac{P(C=c)P(X_1 = x_i|C=c)P(X_2 = x_j|C=c)\ldots}{P(X_1 = x_i)P(X_2 = x_j)\ldots}$$

Primer moski: visina ≥ 175, teza ≥ 65, spol $= M$

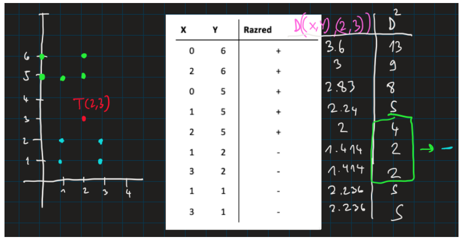
$X \backslash Y$	Razred A	Razred B
p_a	$P(A) = \frac{2}{3}$	$P(B) = \frac{1}{3}$
spol	$P(M\|A)$	$P(M\|B)$
visina	$P(V \geq 175\|A)$	$P(V \geq 175\|B)$
teza	$P(T \geq 65\|A)$	$P(T \geq 65\|B)$
$P(y)\prod_{i=1}^{n} P(x_i\|y)$	…	…

1.6.2 Nomogragmi

Ciljni razred $C = c_T$

$$X_{X_i = x_j} = \ln\left(\frac{P(X_i = x_j|C = c_T)}{P(X_i = x_j|C = \overline{c_T})}\right)$$

1.7 K-najblizjih sosedov



X	Y	Razred	$D((x,y),(2,3))$	D^2
0	6	+	3.6	13
2	6	+	3	9
0	5	+	2.83	8
1	5	+	2.24	5
2	5	+	2	4
1	2	-	1.414	2
3	2	-	1.414	2
1	1	-	2.236	5
3	1	-	2.236	5

2 Vrste ucenja

2.1 Nadzorovano ucenje (supervised learning)

Ucni primeri so podani/oznaceni kot **vrednosti vhodov in izhodov**.

$(\vec{x}_1, \vec{y}_1), (\vec{x}_2, \vec{y}_2), \ldots, (\vec{x}_N, \vec{y}_N)$

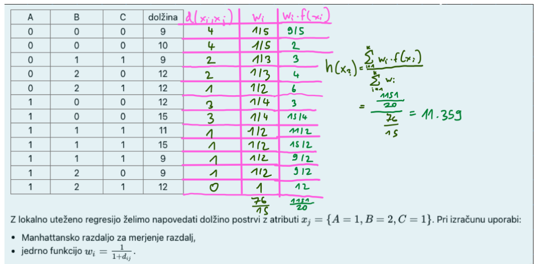
$\vec{x}_i$… atributi, $\vec{y}_i$… ciljna spremenljivka

Locimo dve vrsti problemov:

1. **Klasifikacijski problemi** - y_j diskretna
2. **Regresijski problemi** - y_j zvezna

2.1.1 Lokalno utezena regresija

$$h(\vec{x}_?) = \frac{\sum_{i=1}^{k} w_i \cdot f(\vec{x}_i)}{\sum_{i=1}^{k} w_i}, \; w_i(d)\text{...utez}$$



A	B	C	dolžina	$d(x_i,x_j)$	w_i	$w_i \cdot f(x_i)$
0	0	0	9	4	1/5	9/5
0	0	0	10	4	1/5	2
0	1	1	9	2	1/3	3
0	2	0	12	2	1/3	4
0	2	1	12	1	1/2	6
1	0	0	12	3	1/4	3
1	0	0	15	3	1/4	15/4
1	1	1	11	1	1/2	11/2
1	1	1	15	1	1/2	15/2
1	1	1	9	1	1/2	9/2
1	2	0	9	1	1/2	9/2
1	2	1	12	0	1	12
					76/15	1151/20

$h(x_?) = \frac{\sum_{i=1}^{k} w_i \cdot f(x_i)}{\sum_{i=1}^{k} w_i} = \frac{\frac{1151}{20}}{\frac{76}{15}} = 11.359$

Z lokalno uteženo regresijo želimo napovedati dolžino postrvi z atributi $x_j = \{A = 1, B = 2, C = 1\}$. Pri izračunu uporabi:

- Manhattansko razdaljo za merjenje razdalj,
- jedrno funkcijo $w_i = \frac{1}{1+d_{ij}}$.

2.1.2 Regresijska drevesa

Linearna regresija je poseben primer regresijskega drevesa.

V listih regresijskega drevesa vcasih napovemo kar povprecno vrednost.

2.2 Nenadzorovano ucenje (unsupervised learning)

Ucni primeri niso oznaceni (nimajo ciljne spremenljivke), **ucimo se vzorcev v podatkih**, (npr. grucenje)

2.2.1 Hierarhicno grucenje

Poveze po podobnosti med primeri, primer zacne kot samostojna gruca, na koncu vsi primeri pripadajo eni gruci

Dendrogram: drevo, ki predstavlja grucenje.

Single-linkage: povezava med grucami je najkrajse razdalje med primeroma iz razlicnih gruc.

Complete-linkage: povezava med grucami je najdaljsa razdalja med primeroma iz razlicnih gruc.

Average-linkage: povezava med grucami je povprecna razdalja med primeroma iz razlicnih gruc.

2.2.2 K-means

1. V prostor dodamo k centroidov, ki predstavljajo gruce.
2. Izracunamo ketri centroid je najblizji vsakemu primeru.
3. Izracunamo nove centre gruc = $\frac{1}{|G|}\sum_{i \in G} x_i$
4. Ponovimo korake 2 in 3 dokler se centri ne premaknejo.

2.3 Spodbujevalno ucenje - reinforcement learning

Inteligentni agent se uci iz zaporedja **nagrad** in **kazni**

2.4 Ocenjevanje ucenja

2.4.1 Precno preverjanje

Poseben primer **veckratnega ucenja** in testiranja

k-kratno precno preverjanje

- celo ucno mnozico razbij na k disjunktnih podmnozic
- za vsako od k podmnozic:
  - uporabi **mnozico** kot **testno mnozico**
  - uporabi **preostalih k-1** mnozic kot **ucno mnozico**
- povpreci dobljenih k ocen tocnosti v koncno oceno

Pri precnem preverjanju uporabimo vse podatke za testiranje in vse za ucenje

Metoda **leave one out** je poseben primer precnega preverjanja

Imamo dve hipotezi A in B. Izkase se, da A bolje napoveduje na ucnih podatkih B pa na testnih. Potem je B verjetno boljsa hipoteza.

3 Preiskovanje

3.1 Neinformirani preiskovalni algoritmi

3.1.1 Iskanje v sirino

3.1.2 iskanje v globino

Izboljsave:

- **Iskanje s sestopanjem**
- **depth-limited-search** (vnapej definiramo globino l (dolocimo preko domenskega znanja))

3.1.3 iterativno poglabljanje

problem gobinsko omejenega iskanja -> nastavitev meje l Mejo l postopoma povecujemo za 1, dokler ne najdemo resitve.

- **popolnost**: Da
- **optimalnost**: Da
- **casovna zahtevnost** $O(b^d)$
- **prostorska zahtevnost** $O(bd)$

Boljse od iskanja v globino/sirino

3.1.4 dvosmerno iskanje

Ideja: pognati vzporedni iskanji od zacetka do cilja in od cilja do zacetka.

Motivacija:

Implemenatcija dvosmernega iskanja

- ciljno vozlisce mora biti znano

- originalni problemski prostor preslikamo v dvosmerni prosto stanj E1, E2 dosegljiv iz E in S1,S2,S3 dosegljiv iz S (S,E) -> (S1, E1), (S1,E2), (S2, E1), (S2, E2)... Vozlisce (Si, Ei) je v dvosmernem prostur ciljo vozlisce ce velja E=S (soda dolzina na isto mesto pridemo iz obeh strani) ali S->E (liha pot sosednja)

3.1.5 Cenovno - optimalno iskanje

- posplositev iskanja v sirino (iskanje v sirino je optimalno, ce so cene vseh povezav enake 1)

- dijkstra basically (sam do zadnga noda)
- https://stackoverflow.com/a/14587449

3.2 Informirani preiskovalni algoritmi

3.2.1 Hevristicno preiskovanje

ideja: preiskovanje usmerjamo z dodatnim znanjem (ocenitven funcija za obetavnost vozlisca)

hevristika je ocenitvena funkcija za obetavnost vozlisca

- optimisticna/dopustna: $h(n) \leq h^*(n)$ (h^* je optimalna ocena)
- optimalna: $h(n) = h^*(n)$
- pesimisticna: $h(n) \geq h^*(n)$

3.2.2 Pozresno preiskovanje/ greedy best-first search

$h(n)$ hevristicna ocena

vrednotenje vozlisca $f(n) = h(n)$ hevristicna ocena ... npr manhattan distance (zracna razdalja)

- **popolnost** (ali najde vedno resitev): Ne
- **optimalnost**: Ne
- **casovna zahtevnost** $O(b^m)$, kjer je m najvecja globina drevesa

3.2.3 A*

A* is informed version of **dijkstra** (uses heuristics and pq)

Vozlisca vrednotimo: $f(n) = g(n) + h(n)$

$g(n)$ cena poti do n (znano),

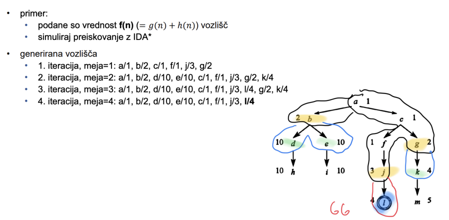
$h(n)$ cena od n do najblizjega cilja (ocena)

prioritetna vrsta (max glede na f(n)) Basically dijkstra + h(n) (A* is basically an informed variation of Dijkstra.)

- **popolnost**: Da (ce ustreza pogoju dopustnosti)
- **optimalnost**: Da (ce ustreza pogoju dopustnosti)
- **casovna zahtevnost** $O(b^m)$, kjer je m najvecja globina drevesa

3.2.4 IDA* (Iterative deepening A*)

DFS with heuristics and iterative bound (value)



Ucinkovitost

- neucinkovit ce vozlisca raznolika f(n)
- prednost: ne hrani vec vseh vozlisc kot A*
- optimalen: ce razvija v prioritetnem vrsntem redu, h(n) mora biti **monotona|konsistentna** (h(n) skos pada) (posledicno tudi dopustna)

$$h(n) \leq c(n, n') + h(n')$$

(h naslednjega vozlisca manjsi ker je blizji cilja)

- monotona → dopustna (proti primer h(n) = 0)

3.2.5 Kakovost hevristicnih funkcij

7	2	4
5		6
8	3	1

Primer igra 8 ploscic

-h_1: stevilo ploscic ki niso na pravem mestu (8)

-h_2: vsota manhattanskih razdalj ploscic do pravega mesta(3+1+2+2+2+3+3+2=18)

Kakovost h ocenimo z:

- stevilom generirarnih vozlisc

- z efektivnim faktorjem vejanja (koliko vozlisc N je algoritem generiral da je na globini d nasel resitev)

	število generiranih vozlišč			efektivni faktor vejanja		
Globina	IDS	A*(h1)	A*(h2)	IDS	A*(h1)	A*(h2)
2	10	6	3	2,45	1,79	1,79
4	112	13	12	2,87	1,48	1,45
6	680	20	18	2,73	1,34	1,30
8	6384	39	25	2,80	1,33	1,24
10	47127	93	39	2,79	1,38	1,22
12	3644035	227	73	2,78	1,42	1,24
14	?	539	113	?	1,44	1,23
16	?	1301	211	?	1,45	1,25
18	?	3056	363	?	1,46	1,26
20	?	7276	676	?	1,47	1,27
22	?	18094	1219	?	1,48	1,28
24	?	39135	1641	?	1,48	1,26

Vidimo $h_2(n) \geq h_1(n) \forall n$ pravimo h_2 **dominira** h_1

3.3 Lokalno preiskovalni algoritmi

3.3.1 Plezanje na hrib

Ne pomnemo poti do cilja, ampak samo trenutno stanje

Koristni v primerih:

- ce nas zanima samo kakovost resitve (in ne pot do cilja)

- resevanje optimizacijskih problemov (kjer je podana **kriterijska funkcija** za oceno kakovosti resitve)

Prednosti:

- majhna poraba prostora

Primer 4 kraljice na sahovnici - kriterijska funkcija: maksimiziramo - (minus) stevilo kraljic, ki se medsebojno napadajo

Tezave:

- lokalni maksimumi
- "rame, plaote" (kriterijska funkcija konstantna vrednost)
- grebeni (za plezanje navzgor je potreben sestop po pobocju grebena)

Resevanje iz lokalnih maksimumov:

- **koraki vstran**: ce ima naslednje stanje isto vrednost kriterijske funkcie, dovolimo premik v to stanje
- **stohasticno** plezanje na hrib: iz mnozice boljsih stanj, verjetnostno izberemo naslednje stanje (pri cemer upostevamo da imajo boljsa stanja vecjo verjetnost izbora)
- **nakljucni ponovni zagon**: veckrat pozeni plezanje na hrib iz nakljucnih stanj dokler ne najdes resitve

3.3.2 Simulirano ohlajanje

algoritem ki izvira iz metalurgije (ko je jeklo tekoce, so molekule v njem bolj gibljive; ko se ohlaja se strjuje in molekuele se umirjajo)

Analogija:

- generiramo nakljucne sosede trenutnega stanja

- ce najdemo **boljse stanje ga izberemo**

- ce najdemo **slabse stanje, ga izberemo z doloceno verjetnostjo**

- verjetnost izbire neoptimalnega stanja s casom pada (nizanje temperature)

3.3.3 Lokalno iskanje v snopu

Algoritem:

- v spominu hrani k aktualnih stanj namesto enega

- izberi k optimalnih stanj od sosedov aktualnih stanj

- ponavljaj do ustavitnega pogoja

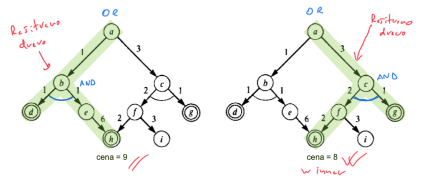
3.4 Preiskovanje grafov AND/OR, nedeterministicno okolje

Pomagajo resevati probleme z **dekompozicijo na manjse probleme** Uporabnost:

- princip deli in vladaj
- iskanje v nedeterministicnih okoljih
- igre med dvema nasprotnikoma s popolno informacijo (sah, dama)
- ekspertno resevanje problem

Primer graf dekompozicja v dva manjsa problema skozi g in f

Resitveno drevo je resitev AND/OR grafov



3.4.1 AO*

- posplositev A* na grafe AND/OR
- **popoln in optimalen** ⇔ h(n) ne precenjuje dejanske cene do cilja

$F(N)$... ocena za usmerjanje preiskovanja

$H(N)$... dinamicna hevristicna ocena

Postopek:

1. Razvij najcenejse vozlisce
   - ce list in koncno (oznaci), preveri 3. korak, nadaljuj v 1.
   - ce list in ni koncno (oznaci) vrednost vozlisca = ∞
2. Posodobi vse predhodnike
   - v AND starsih, cena starsa = ∑ sinov + povezava v
   - v OR starsih, cena starsa = min(sinovi) + povezava v
3. Koncaj ko obstaja pot od zacetnega vozlisca, po kateri v AND vozliscih po vseh sinovih prides do cilja, v OR vozliscih v vsaj enem

3.4.2 Preiskovanje v nedeterministicnem okolju:

Nedeterministican akcija - ista akcija lahko obrodi razlicna ciljna stanja

Do resitve ni vec **poti** temvec **drevesa** (uporbljamo AND/OR grafe)

Vozlsca OR **mozne akcije**, vozlisca AND **vejanja v mozna stanja**, ki so rezultat nedeterministicnih akcij

3.5 Preiskovanje brez informacij o stanju

Okolja smo razdelili na **transparent** (agent lahko zazna popolno informacija) in **netransparentna** (brez informacije o stanju)

Kej ce imamo opravka z netraspranetim okoljem?

- izvajamo preiskovanje prostora **verjetnih** stanj in ne prostora **dejanskih** stanj

- izvajamo s postokopom omejevanja moznozsti kandidatnih stanj

3.6 Igranje iger

3.6.1 Predstavitev problema

3.6.2 Algoritem MINIMAX

- m globina - b

3.6.3 Rezanje alfa-beta

4 Planiranje

plan zaporedje akcij, ki pripelje od zacetnega do koncnega stanja

4.1 Planiranje s sredstvi in cilji (STRIPS)

Agentu opisemo svet in postavimo fizikalne omejitve.

Ne zagotovalja optimalne resitve, obravnavamo le en cilj naenkrat (ko ga dosezemo, se lahko ostali izgubijo) = Sussmanova anomalija

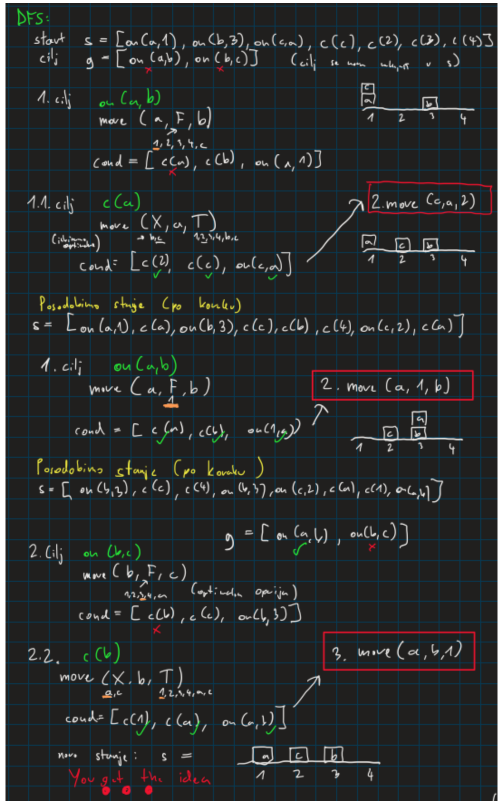
Akcija move(X, From, To)

- pogoj: cond=[clr(X), on(X,F), clr(T)] → pogoji za izvajanje akcije,
- poz. ucinki: adds=[on(X, T), clr(F)] → nova stanja,
- neg. ucinki: dels=[on(X, F), clr(T)] → izbrisana stanja,
- omejitve: constr=[F ≠ T, X≠ F, X≠ T, block(X)] → omejitve akcij (fizikalne omejitve),

Algoritem:

1. Izberi se neresen cilj iz mnozice CILJEV
2. Izberi akcijo, ki izbrani cilj doda v stanje
3. Omogoci izbrano akcijo (izpolni pogoje)
4. Izvedi akcijo (ki izopolni najvec pogojev)
5. Ce obstajajo nereseni cilji ⇒ 1.

Primer dfs, zlaganje kock



4.2 Planiranje z regresiranjem ciljev (STRIPS)

Resitev za sussmanovo anomalijo

Zacnemo v ciljih, regresiramo do zacetka ($G_i \subset S_0$):

1. $G_{i+1} = G_i \cup \text{cond}(A) - \text{adds}(A)$
2. POGOJ: $G_i \cap \text{dels}(A) = \emptyset$
3. Preveri da ni protislovja (npr. $G_{i+1} = [on(b, c), \ldots, c(c) \ldots]$)

→ zactno_stanje = [on(a,1), on(b,a), c(b), on(c,3), c(c)]
→ hocemo da zacetno_stanje ⊂ G_i

1. G_0 = [on(a,b), on(b,c)]
   - on(a,b): $A_0 = move(a, From, b)$
   - From = 1
   - POGOJ: $G_0 \cap$ dels$(A_0) = \emptyset$✓
   - G_1 = [on(a,b), on(b,c), c(a), c(b), on(a,1)]-[c(1), on(a,b)] ✓
2. G_1 =[on(b,c),c(a),c(b),on(a,1)]
   - c(a): $A_1 = move(X, a, To)$
   - X = c, To = 2
   - POGOJ: $G_1 \cap$ dels$(A_1) = \emptyset$✓
   - G_2 = [on(b,c),c(a),c(b),on(a,1),c(c),c(2),on(c,a)] -[c(a), on(c,2)] ✗(protislovje)
   - on(b,c): $A_2 = move(b, From, c)$
   - From = 3
   - POGOJ: $G_2 \cap$ dels$(A_2) = \emptyset$✓
   - G_2 = [on(b,c),c(a),c(b),on(a,1),c(c),c(b),on(b,3)] ✓
3. G_2 = ...

4.3 RAZPOREJANJE OPRAVIL (PDDL)

Razsirimo lahko notacijo (PDDL):
Akcija1 ≺ **Akcija2**: Akcija1 se mora zgoditi pred Akcijo2
Resources podajo stevila razpolozljivih resursov
DURATION opredeljuje trajanje posamezne akcije
CONSUME opredeljuje (trajno) porabo dolocene kolicine resursov
USE opredeljuje (zacasno) zasedenost kolicine resursov med izvajanjem akcije

```
Jobs (AddEngine1 ≺ AddWheels1 ≺ Inspect1,
      AddEngine2 ≺ AddWheels2 ≺ Inspect2 )
Resources (EngineHoists(1), WheelStations(1), Inspectors(2), LugNuts(500))

Action (AddEngine1 , DURATION:30,
        USE:EngineHoists(1))
Action (AddEngine2 , DURATION:60,
        USE:EngineHoists(1))
Action (AddWheels1 , DURATION:30,
        CONSUME:LugNuts(20), USE:WheelStations(1))
Action (AddWheels2 , DURATION:15,
        CONSUME:LugNuts(20), USE:WheelStations(1))
Action (Inspect i, DURATION:10,
        USE:Inspectors (1))
```

Metoda kriticne poti
kriticna pot: pot, ki je **najdaljsa** in doloca dolzino trajanja celotnega plana vsaki akciji priredimo par **[ES, LS]**

- **ES**: najbolj zgodnji mozen zacetek (Earliest Start)
  - $ES(start) = 0, \quad ES(B) = \max_{A \prec B}[ES(A) + Duration(A)]$
- **LS**: najbolj pozen mozen zacetek (Latest Start)
  - LS(Finish) = ES(Finish), $LS(A) = \min_{A \prec B}[LS(B) - Duration(A)]$

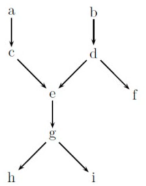
rezerva(slack)=LS-ES (**casovna rezerva**) Algoritem po **hevristiki minimum slack** → na vsaki iteraciji ima prednost akcija ki ima izpolnjene predhodnike in najnizji slack, nato posodobi [ES in LS] za celotni graf in ponovi.

5 SKLEPANJE

5.1 BAYESOVSKE MREZE

Baye. mreza = Usmerjen graf, kjer so podane zahtevane verjetnosti:

- Za vozlisca **brez starsev** verjetnosti $P(v_i)$
- Za vozlisca z **starsi** pogojne verjetnosti vseh kombinacij starsev



$p(a)$
$p(b)$
$p(c|a), p(c|\neg a)$
$p(d|b), p(d|\neg b)$
$p(e|cd), p(e|c\neg d), p(e|\neg cd), p(e|\neg c\neg d)$
$p(f|d), p(f|\neg d)$
$p(g|e), p(g|\neg e)$
$p(h|g), p(h|\neg g)$
$p(i|g), p(i|\neg g)$

Pravila verjetnostnega sklepanja:

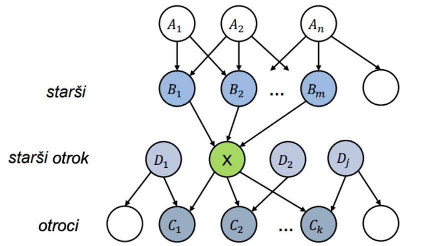
1. **Konjunkcija**: $P(X_1 X_2 \mid C) = P(X_1 \mid C) P(X_2 \mid X_1 C)$
2. **Gotov dogodek**: $P(X \mid \ldots X \ldots) = 1$
3. **Nemogoc dogodek**: $P(X \mid \ldots \overline{X} \ldots) = 0$
4. **Negacija**: $P(\overline{X} \mid C) = 1 - P(X \mid C)$
5. Ce je Y naslednik od X in je Y vsebovan v pogojnem delu: $P(X \mid YC) = P(X \mid C) \cdot \frac{P(Y|XC)}{P(Y|C)}$
6. Ce pogojni del ne vsebuje naslednika od X:

(a) ce X **nima** starsev: $P(X \mid C) = P(X)$, P(X) je podan
(b) ce **ima** X starse S: $P(X \mid C) = \sum_{S \in P_X} P(X \mid S) P(S \mid C)$

7. Iz 6b zgoraj: $P(i \mid gc) = P(i \mid g)$

5.2 OVOJNICA MARKOVA

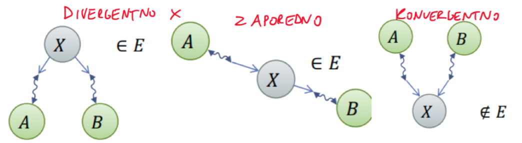
Ce so podani **starsi**, **otroci** in **starsi otrok**, je vozlisce X **neodvisno** od vseh ostalih vozlisc.



5.3 D-LOCEVANJE

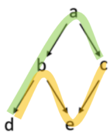
A in B v mrezi sta **neodvisni** ⇔ obstaja mnozica vozlisc E, ki d-locuje A in B, potem sledi: ($P(A|EB) = P(A|E)$)

za **vsako neusmerjeno pot P** med **A** in **B** v bayesovski mreži:
  za **vsako vozlišče X** na poti **P**:
    analiziraj pogoj za pripadnost X množici E glede na tip:
      divergentno ali **zaporedno** vozlišče: $X \in E$
      konvergentno vozlišče: X in nasledniki $\notin E$
    S_X = množice vozlišč, ki ustrezajo pogoju za X
  S_P = $\cup_X(S_X)$ // množice, ki d-ločujejo samo na poti P
                    (unija množic za vozlišča na poti)
E = $\cap_P S_P$ // množice, ki d-ločujejo v celi mreži
                    (presek množic za vse možne poti)



! pri konvergentnem izlocimo tudi vse naslednike X

Primer d-locevanje vozlisc c in d



b ∈ E: $\{\{b\}, \{a, b\}, \{b, e\}, \{a, b, e\}\}$
a ∈ E: $\{\{a\}, \{a, b\}, \{a, e\}, \{a, b, e\}\}$
na celi zeleni poti: $\{\{a\}, \{b\}, \{a, b\}, \{a, e\}, \{b, e\}, \{a, b, e\}\}$
b ∈ E: $\{\{b\}, \{a, b\}, \{b, e\}, \{a, b, e\}\}$
e ∉ E: $\{\{\}, \{a\}, \{b\}, \{a, b\}\}$
na celi oranžni poti: $\{\{\}, \{a\}, \{b\}, \{a, b\}, \{b, e\}, \{a, b, e\}\}$
$E = \{\{a\}, \{b\}, \{a, b\}, \{b, e\}, \{a, b, e\}\}$

→ P(d|ca)=P(d|a), P(d|cb)=P(d|b), P(d|cab)=P(d|ab), P(d|cbe)=P(d|be),P(d|cabe)=P(d|abe)