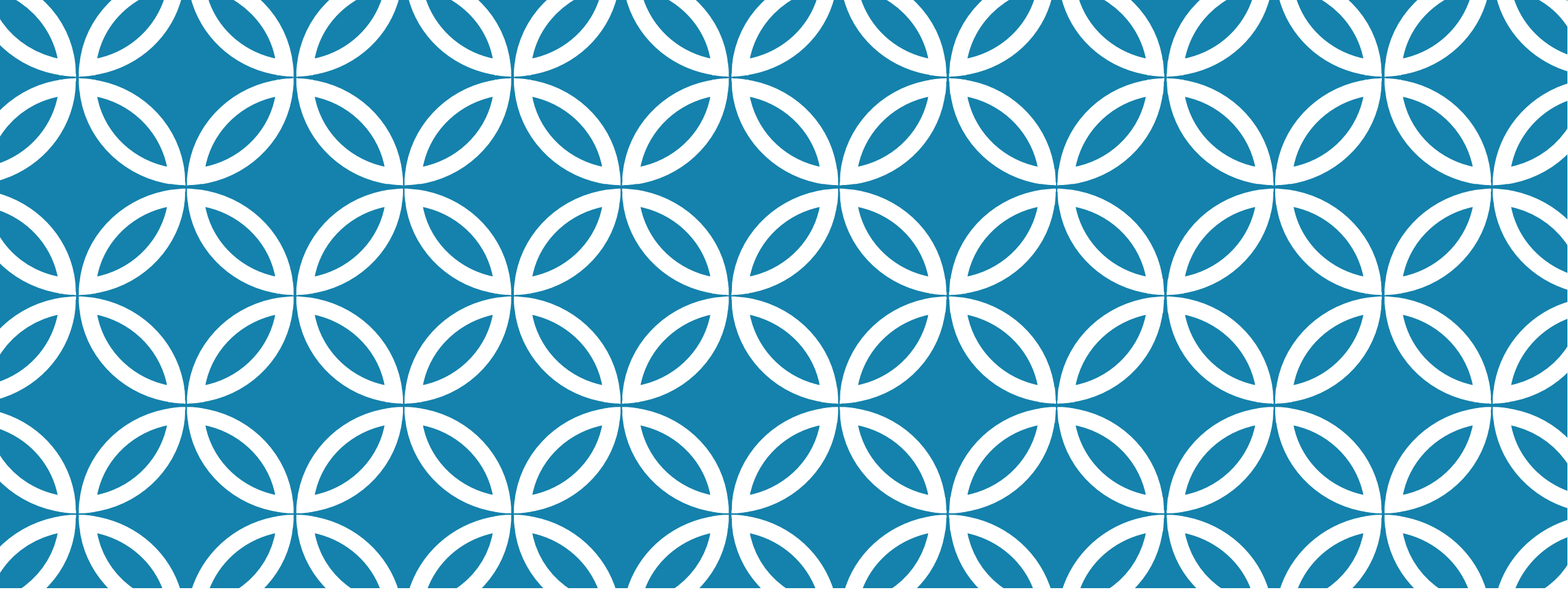


ROZHODOVACIE A OPTIMALIZAČNÉ PROBLÉMY

Marián Mach

TYPY PROBLÉMOV

Rozhodovací problém

Riešenie musí spĺňať všetky požadované podmienky (tvrdé podmienky) alebo ich minimálny počet (tvrdé a mäkké podmienky) alebo čo najväčší počet podmienok (mäkké podmienky)

Kombinovaný problém

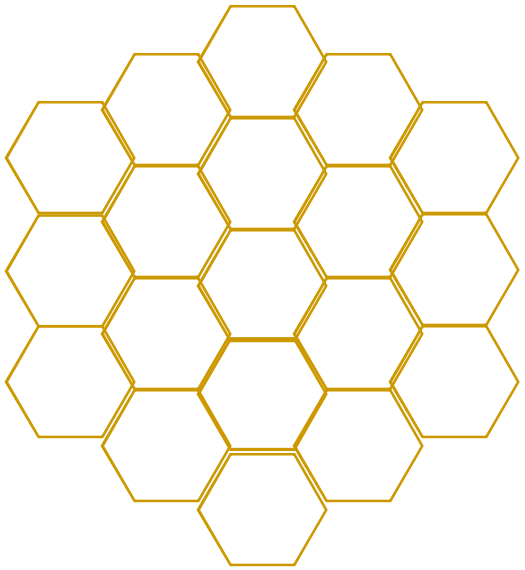
Optimalizačný problém

Riešenie musí poskytovať optimálnu (minimálnu alebo maximálnu) hodnotu cieľového kritéria

Asociovaný rozhodovací problém

Riešenie musí poskytovať hodnotu cieľového kritéria lepšiu ako zadaný prah

PRIESTOR KANDIDÁTOV



Rôzne typy úloh:

- Klasifikačná úloha
- Selekčná úloha
- Triediaca úloha

Tvar riešenia: A1, A2, A3, A4, …

- súbor hodnôt atribútov

1, 0, 0, 1, 1, 0, …, 0, 1 (dĺžka n) (2^n)

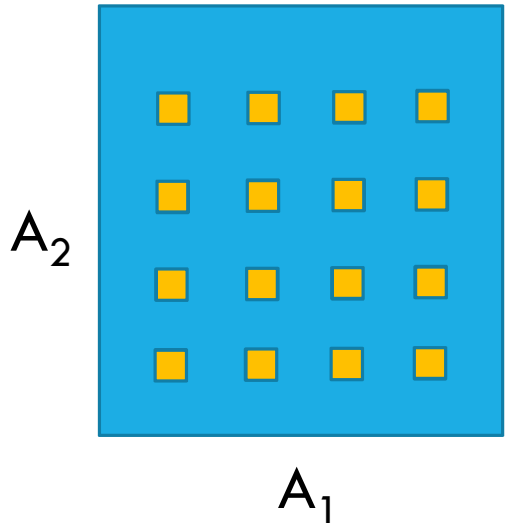
m3, m7, m8, m12, m18 (dĺžka k) (n nad k)

m9, m17, m2, …, m7 (dĺžka n) (n!)

Priestor kandidátov je usporiadaný priestor všetkých kombinácií možných hodnôt všetkých atribútov (**kandidátov** – potenciálnych riešení).

Kandidát môže byť:

- validný/nevalidný – ak sú zadané ohraničenia, ktoré musia byť splnené
- optimálny/neoptimálny – ak je zadané nejaké optimalizačné kritérium



ROZHODOVACÍ PROBLÉM AKO ÚLOHA S OHRANIČENIAMI

Množina premenných $V = \{ V_1, V_2, \ldots, V_n \}$

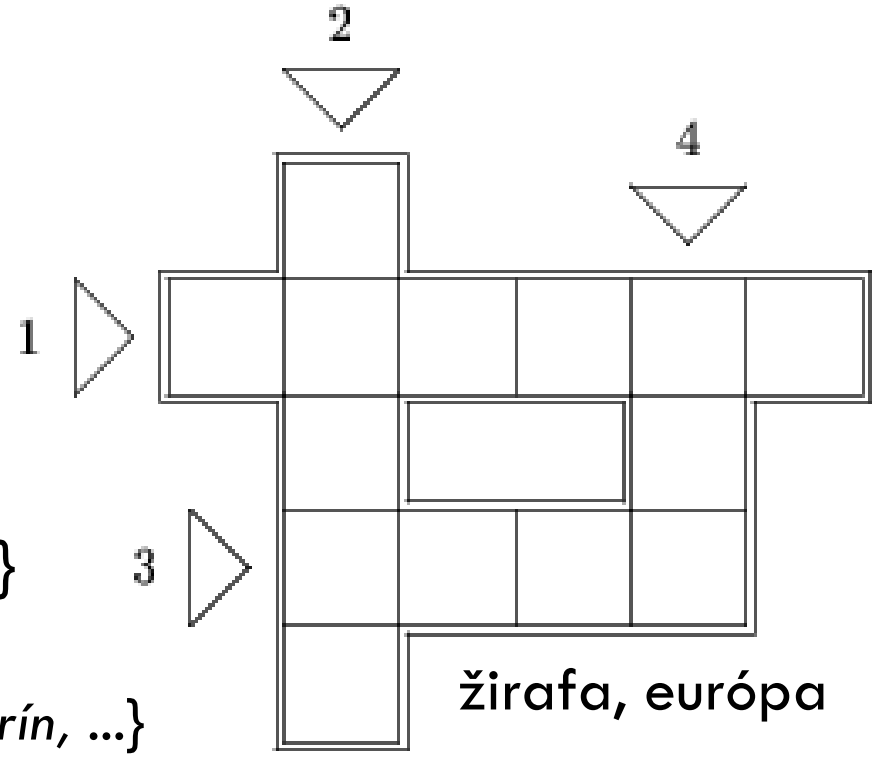
- $V = \{ V_1, V_2, V_3, V_4 \}$

Množina domén $D = \{ D_1, D_2, \ldots, D_n \}$

- $D_1 = \{$*žirafa, európa, puška, kurín, auto, íver, pór, eso*$\}$

Množina ohraničení $C = \{C^1,\ldots,C^n,C^{1,2}, \ldots, C^{n-1,n}, C^{1,2,\ldots,n}\}$

- $C^1 = \{$*žirafa, európa*$\}$
- $C^{1,2} = \{$*európa-puška, európa-kurín, európa-auto, puška-kurín, ...*$\}$
- $C^{1,2,3} = \{$*európa-puška-puška, ...*$\}$ — Vyššie árnosti zvyčajne nie sú zadané
- $C^{1,2,3,4} = \{ ? \}$ — Hľadané riešenia



žirafa, európa

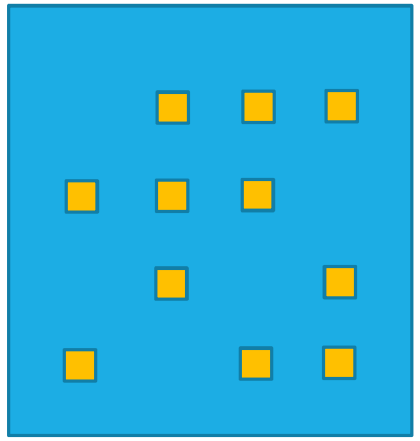
puška, kurín

auto, íver

pór, eso

*Priestor kandidátov = 8^4 alebo 8*7*6*5*

KONZISTENČNÉ ALGORITMY



Princípom je transformácia na jednoduchší problém (s prísnejšími ohraničeniami) s rovnakou množinou riešení = redukcia veľkosti priestoru kandidátov

Úprava ohraničení (ich sprísňovanie) tak, aby boli vzájomne konzistentné

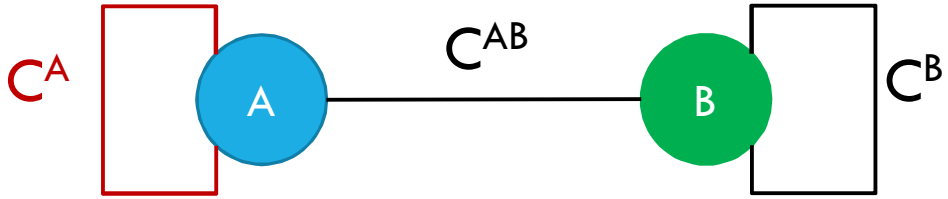
Reťazové sprísňovanie – zmena v jednom ohraničení vyvolá zmenu iného ohraničenia a na základe toho sa zmení ďalšie ohraničenie ... = **propagácia (šírenie) ohraničení**

Konzistencia (x,y) – ľubovoľná skupina x premenných je konzistentná s každou skupinou y premenných (x a y sú disjunktné)

Úplná konzistencia všetkých ohraničení (ponechanie iba validných riešení) alebo čiastočná konzistencia (sú ponechané aj niektoré nevalidné riešenia)

VNUCOVANIE KONZISTENCIE – (1,1) KONZISTENCIA

- C^2 = {*puška*, *kurín*}
- $C^{1,2}$ = {*európa-puška*, *európa-kurín*, *európa-auto*, *puška-kurín*, *puška-auto*, *kurín-auto*}
- C^1 = {~~*žirafa*~~, *európa*}



Kontrola všetkých hodnôt povolených unárnym ohraničením C^A premennej A

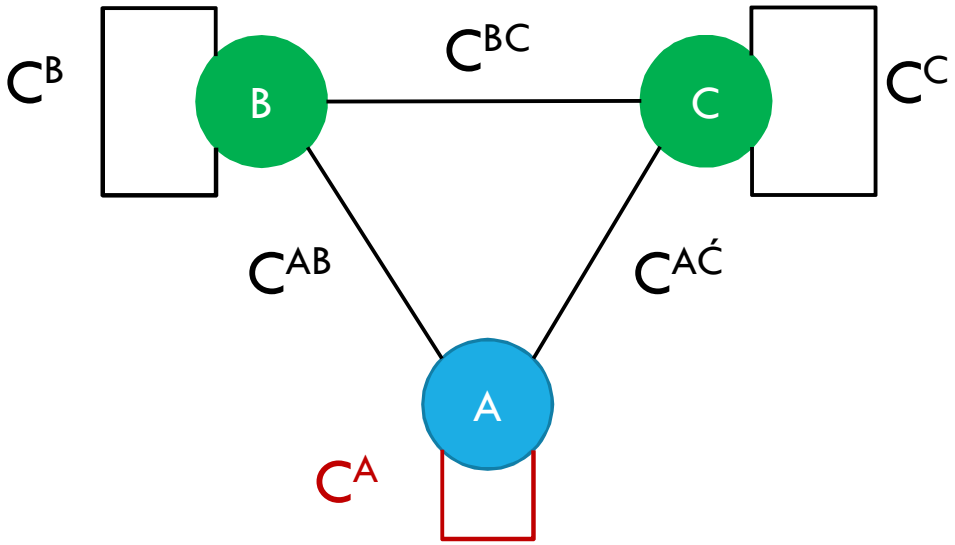
- *Pre nejakú hodnotu z C^A musí existovať hodnota premennej B, povolená jej unárnym ohraničením C^B*
- *Obe hodnoty musia byť povolené spoločným binárnym ohraničením C^{AB}*
- *Pri nesplnení nejakej podmienky, kontrolovaná hodnota sa vymaže z ohraničenia C^A premennej A*

V postavení premennej B sa postupne vystriedajú všetky premenné mimo premennej A

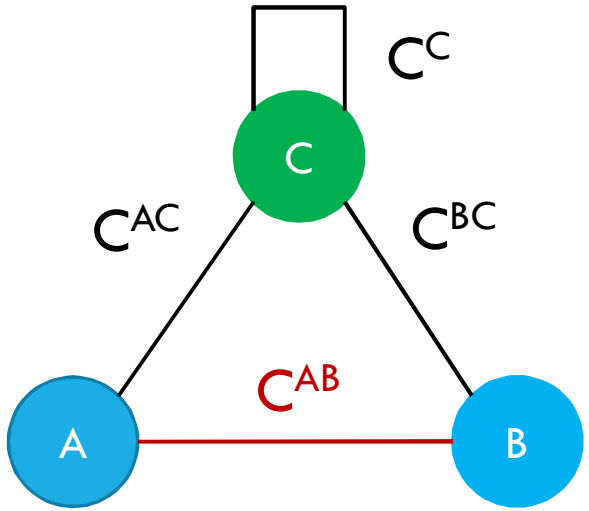
(1,1)-konzistencia = konzistencia každej premennej voči každej inej premennej

VNUCOVANIE KONZISTENCIE – INÉ KONZISTENCIE

(1,2) – 1 premenná voči 2



(2,1) – 2 premenné voči 1



PREHĽADÁVANIE DO HĹBKY + BACKTRACKING

Pokusná konštrukcia kandidátov (priradenie redukuje veľkosť priestoru kandidátov)

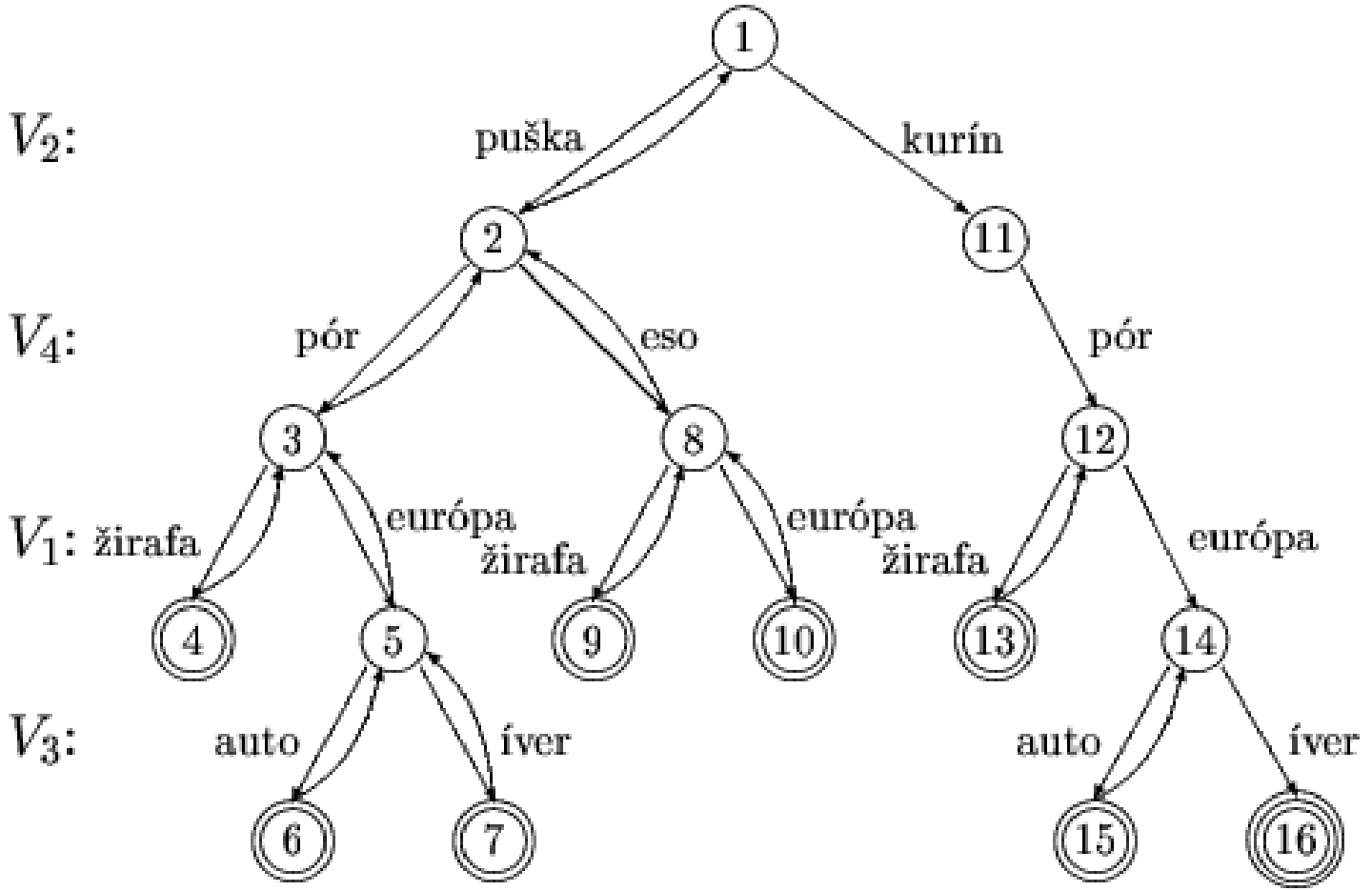
Vylepšenia:

Heuristické výbery
- usporiadania premenných a domén

Pamäťové rozšírenia
- vyhýbanie sa opakovaným neúspechom

Skokové algoritmy (backjumpingy)
- návrat k príčine neúspechu

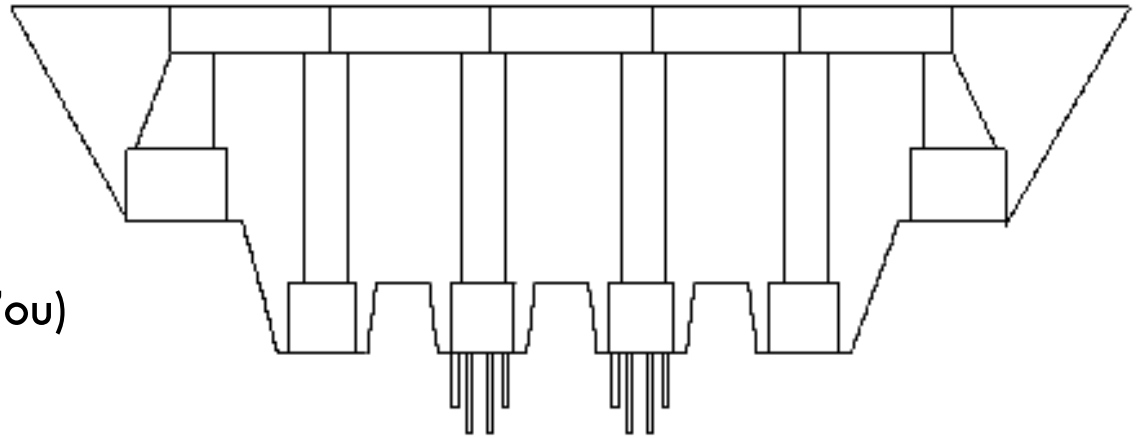
Udržiavanie konzistencie
- jedno/viackroková propagácia



PRÍKLAD VYUŽITIA - ROZVRHOVANIE

Premenné

- **Začiatok** operácie
- (**koniec/trvanie** operácie, použitý **zdroj**)



Domény

- Hodnoty **času** (diskretizácia s požadovanou presnosťou)

Ohraničenia

- **Precedenčné** – časové relácie medzi úlohami
  - A # B (# = predchádza, nasleduje po, prekrýva, obsiahnuté v, ukončuje)
  - *Pr: Výkop pre pilier predchádza betónovaniu, do štyroch dní po ukončení debnenia potrebné začať s betónovaním*
- **Disjunktné** – zdieľanie zdrojov
  - *Pr: Jedna betonárska čata môže naraz betónovať iba jeden pilier*

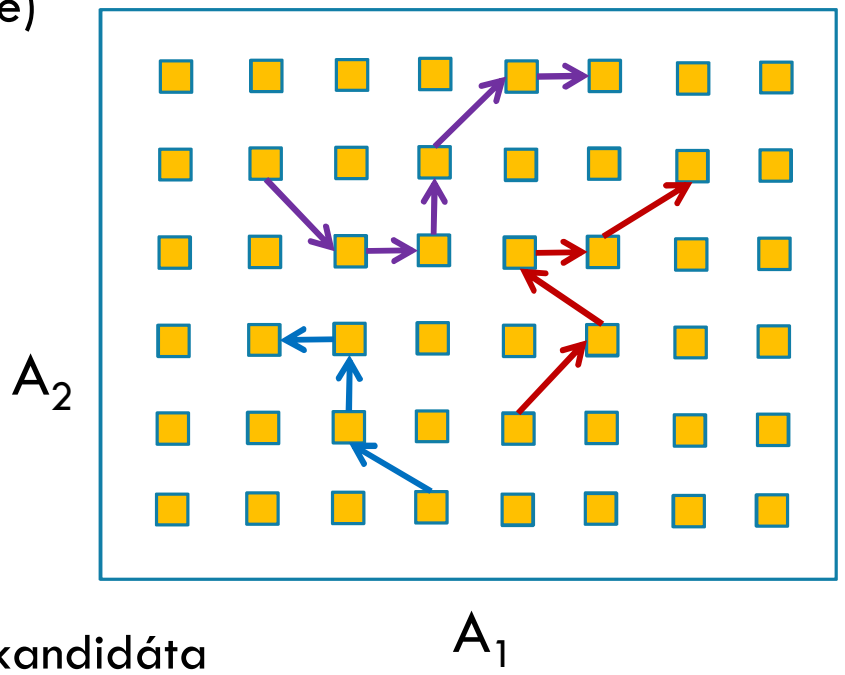
POHYB V PRIESTORE KANDIDÁTOV

Pohyb od kandidáta ku kandidátovi – trajektória v priestore kandidátov

- Nie je to systematický prechod všetkými kandidátmi (brute force)

Algoritmy:

- Individuálne – iba jedna trajektória
- Populačné – viacej trajektórií vytváraných paralelne
  - Trajektórie generované synchrónne
  - Trajektórie generované asynchrónne (rôzne rýchlosti predlžovania)



Tvorba trajektórie

- Začína zvyčajne v náhodne zvolenom kandidátovi
- Pohyb k susednému kandidátovi zmenou aktuálneho kandidáta
- Preskúmanie susedných kandidátov v blízkom okolí aktuálneho kandidáta
- Výber jedného kandidáta z okolia na základe preferencie
- Výber nasledovného (zmena aktuálneho kandidáta) môže byť deterministický alebo stochastický

HOROLEZECKÝ PREHĽADÁVACÍ ALGORITMUS

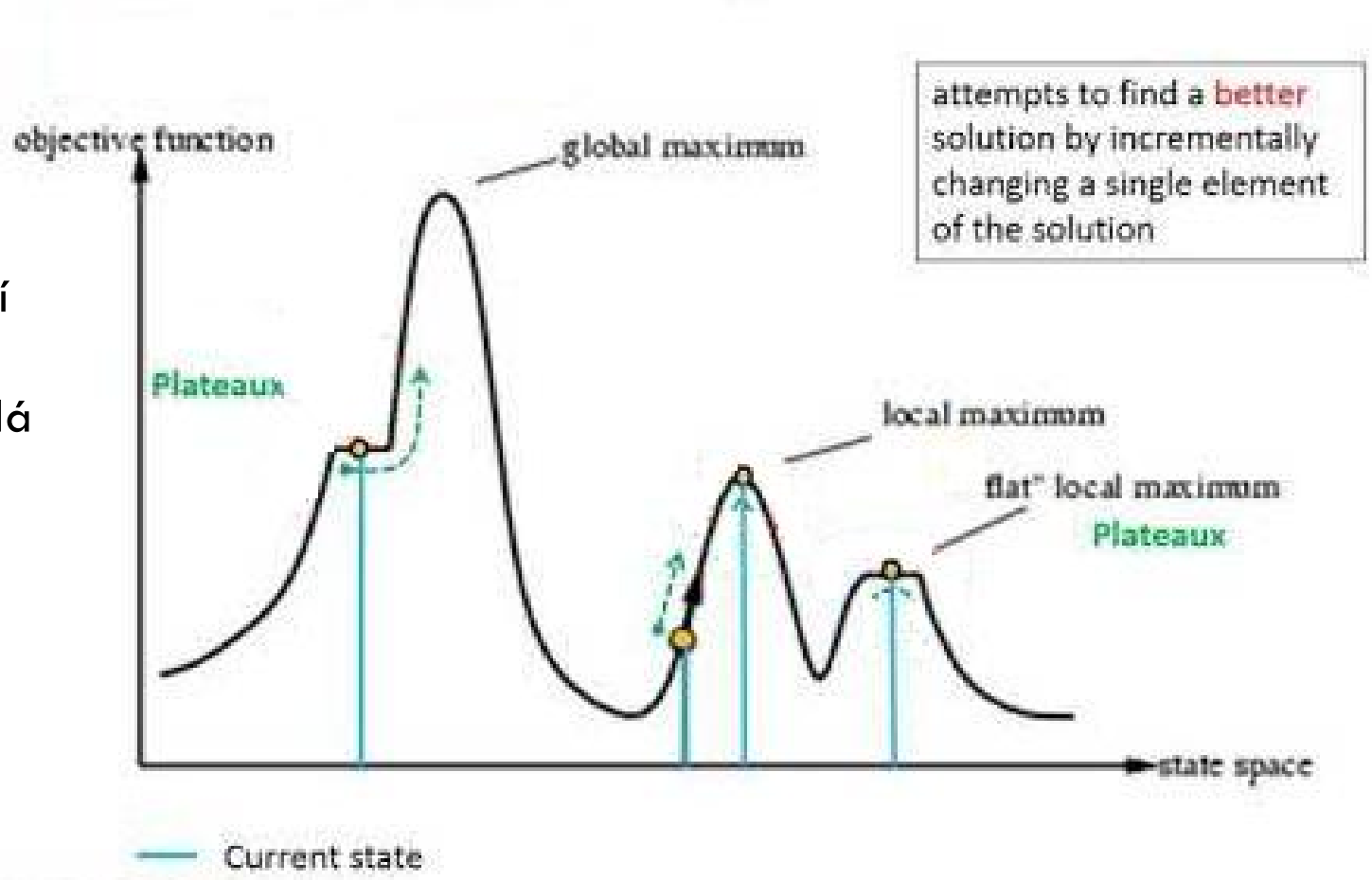
Vytváranie jednej trajektórie

Vlastnosti algoritmu

- Náhodný štart
- Deterministický výber kroku (najlepší kandidát v blízkom okolí)
- Predlžovanie trajektórie pokiaľ sa dá (koniec v lokálnom maxime)

Príklad vpravo:

- jednorozmerný priestor kandidátov (vodorovná os)
- kvalita kandidátov vyjadrená numericky (zvislá os)

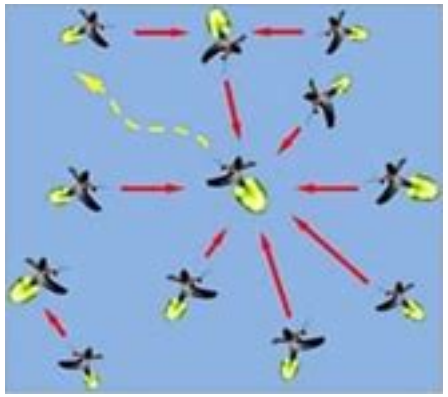


PRÍRODNE INŠPIROVANÉ ALGORITMY

Spôsob pohybu v priestore kandidátov imituje nejaký prírodný proces

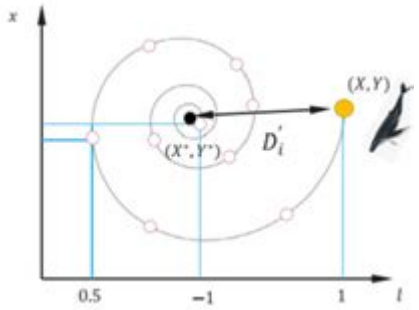
Možné imitované procesy:

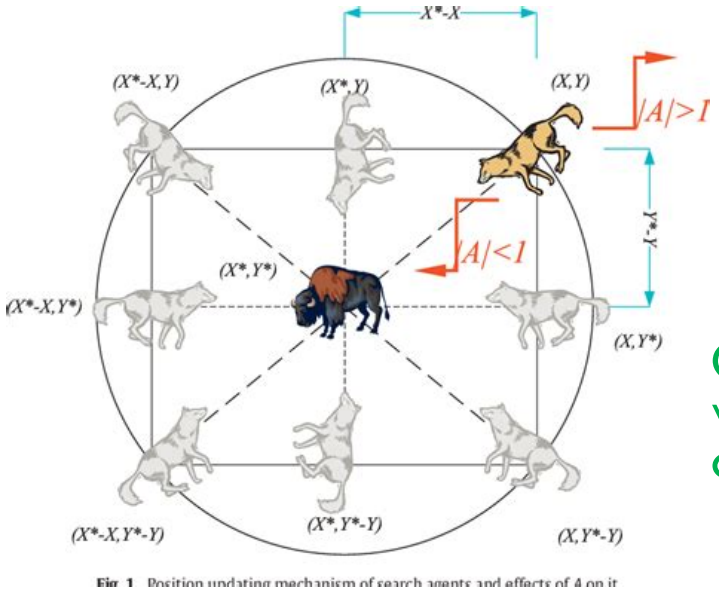
- Biologický
- Evolučný
- Fyzikálny
- Chemický
- Sociálny



Firefly algorithm

Whale optimization algorithm





Grey wolf optimizer

HEURISTICKÉ OPTIMALIZAČNÉ PROCESY (2.ROK ZS)

KKUI
Katedra kybernetiky a umelej inteligencie

- Rozhodovacia a optimalizačná úloha, vybrané kanonické typy úloh
- Perturbačný a konštrukčný prístup, konštrukčné heuristiky
- Systematické prehľadávanie, orezávacie heuristiky
- Lokálne prehľadávanie, stratégie prehľadávania, gradientové metódy.
- Únik z lokálneho extrému (variabilné, stochastické a dynamické lokálne hľadanie)
- Prehľadávanie s pamäťou, zakázané hľadanie
- Hybridné prehľadávacie algoritmy
- Algoritmy pre estimáciu distribúcie
- Evolučne orientované algoritmy (simulované žíhanie, genetický algoritmus)
- Biologicky orientované algoritmy (mravčie kolóny, PSO algoritmus, včelie algoritmy, bakteriálne algoritmy)
- Aproximačné triedy a randomizované algoritmy. Výpočtová zložitosť, základné zložitostné triedy.

VÝROKOVÁ LOGIKA

Syntax = tvorba (zložených) logických viet

- Konštanty (TRUE, FALSE)
- Symboly
- Operátory (¬, Λ, V, →, ↔, …)

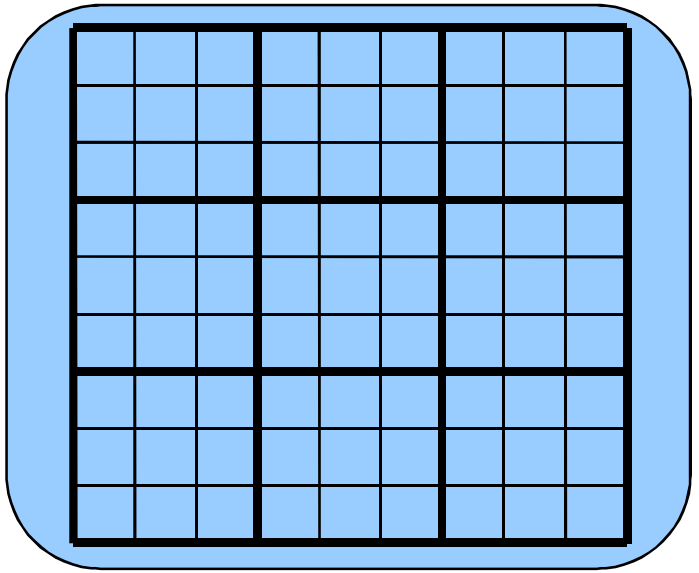
X	Y	¬ X	X ∧ Y
FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	FALSE
TRUE	TRUE	FALSE	TRUE

Sémantika = pravdivosť logických viet

- Konštanty – pevne dané (TRUE = pravda, FALSE = nepravda)
- Operátory – pravdivostné tabuľky
- Symboly – priradenie je potrebné nájsť

Úloha splniteľnosti = hľadanie takej interpretácie symbolov, aby logický model bol pravdivý (na to sa používajú logické solvery)

RIEŠENIE ÚLOH VÝROKOVOU LOGIKOU



Priestor kandidátov $= 9^{81}$

Sudoku – logický model

- Premenné: $X_{riadok,stĺpec,cifra}$ ($X_{1,1,1}$ až $X_{9,9,9}$ – 729 premenných)
- Ohraničenia:
  - V každom políčku práve jedna cifra
    - Pre každý riadok *r* a stĺpec *s* (každé políčko) platí: $=1(X_{r,s,1}, ..., X_{r,s,9})$
    - $(X_{r,s,1} \vee ... \vee X_{r,s,9}) \wedge (\sim X_{r,s,1} \vee \sim X_{r,s,2}) \wedge (\sim X_{r,s,1} \vee \sim X_{r,s,3}) \wedge (\sim X_{r,s,1} \vee \sim X_{r,s,4}) \wedge \ldots \wedge (\sim X_{r,s,8} \vee \sim X_{r,s,9})$
  - V každom štvorci každá cifra práve raz
  - V každom stĺpci každá cifra práve raz
  - V každom riadku každá cifra práve raz
    - Pre každý riadok *r* a cifru c platí: $=1(X_{r,1,c}, ..., X_{r,,9,c})$
    - $(X_{r,1,c} \vee ... \vee X_{r,9,c}) \wedge (\sim X_{r,1,c} \vee \sim X_{r,2,c}) \wedge (\sim X_{r,1,c} \vee \sim X_{r,3,c}) \wedge (\sim X_{r,1,c} \vee \sim X_{r,4,c}) \wedge \ldots \wedge (\sim X_{r,8,c} \vee \sim X_{r,9,c})$

Kardinalitné ohraničenia

SPLNITEĽNÉ MODULOVÉ TEÓRIE

Výroková logika je kombinovaná s nejakou teóriou, vznikajú nové typy logík

- VL: X ∧ Y – symbolom možno priradiť hodnoty ľubovoľne (jediným kritériom je pravdivosť celej vety)
- SMT: (C < 28) ∧ (g(f(A),f(B))=g(f(B),f(A))) – „symboly“ majú štruktúru (atómy)
  - pravdivosť prvého atómu závisí od hodnoty premennej C
  - pravdivosť druhého atómu závisí od rovnosti hodnôt A a B
  - Pravdivosť atómov nemožno nastaviť ale je daná platnosťou vzťahu, ktorý reprezentujú

Modulové teórie

- Logika rovnosti s neinterpretovanými funkciami
  - = je reflexívna (x=x), komutatívna (ak x=y potom y=x)a tranzitívna (ak x=y a y=z potom x=z)
- Lineárna aritmetika
  - ≤ a < s interpretáciou definujúcou usporiadanie na množine čísel
  - operácie nad číslami (lineárne operácie: súčet, rozdiel, súčin a podiel)

RIEŠENIE ÚLOH SPLNITEĽNÝMI MODULOVÝMI TEÓRIAMI

Plánovanie trajektórie v priestore – logický model

- Premenné: $x_1, x_2, x_3, y_1, y_2, y_3$
- Ohraničenia:
  - Trajektória nesmie ležať mimo priestor

    $(x_{min} \leq x_1) \wedge (x_1 \leq x_{max})$
  - Smer pohybu vertikálny alebo horizontálny

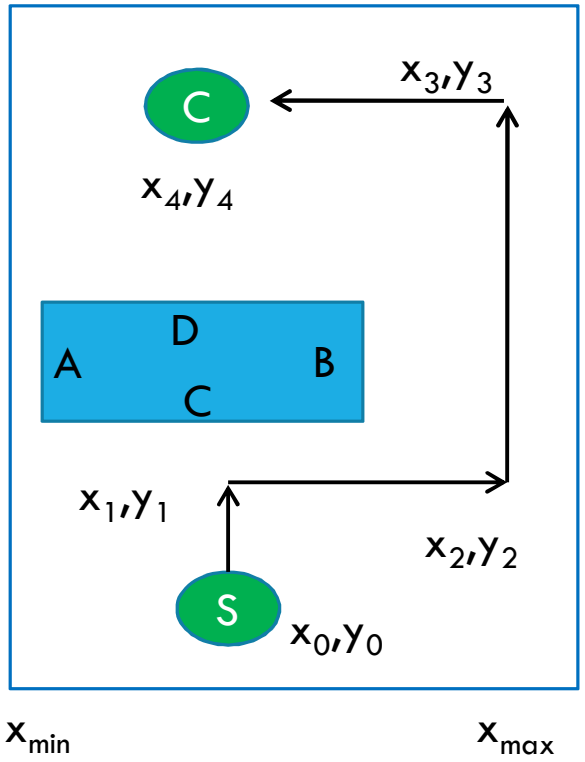
    $((x_1 = x_2) \wedge \neg(y_1 = y_2)) \vee (\neg(x_1 = x_2) \wedge (y_1 = y_2))$
  - Vyhýbanie sa prekážke

    $((x_1 \leq A) \wedge (x_2 \leq A)) \vee$ <- *segment vľavo od prekážky*

    $((B \leq x_1) \wedge (B \leq x_2)) \vee$ <- *segment vpravo od prekážky*

    $((y_1 \leq C) \wedge (y_2 \leq C)) \vee$ <- *segment pod prekážkou*

    $((D \leq y_1) \wedge (D \leq y_2))$ <- *segment nad prekážkou*



Priestor kandidátov = 6^{∞}

PROLOG

Najznámejší jazyk logického programovania, vhodný pre všeobecné použitie

Založený na formálnej logike (predikátovej logike prvého rádu)

Deklaratívny jazyk = programová logika je vyjadrená v zmysle relácií, reprezentovaných ako fakty a pravidlá

Obsahuje zabudovaný odvodzovací mechanizmus, pomocou ktorého dokazuje (ne)pravdivosť výrokov

- Každá otázka je považovaná za výrok

Facts

```
food(burger).
food(sandwich).
food(pizza).
lunch(sandwich).
dinner(pizza).
```

Rules

```
meal(X) :- food(X).
```

Queries / Goals

```
?- food(pizza).
?- meal(X), lunch(X).
?- dinner(sandwich).
```

English meanings

```
// burger is a food
// sandwich is a food
// pizza is a food
// sandwich is a lunch
// pizza is a dinner

// Every food is a meal OR Anything is a meal if it is a food

// Is pizza a food?
// Which food is meal and lunch?
// Is sandwich a dinner?
```

APLIKÁCIA LOGIKY V INTELIGENTNÝCH SYSTÉMOCH (3. ROK ZS)

KKUI
Katedra kybernetiky a umelej inteligencie

- Úvod do reprezentácie znalostí, určité a neurčité znalosti, logické prostriedky reprezentácie znalostí
- Výrokový počet – syntax a sémantika, interpretácia a vlastnosti
- Automatické dokazovanie teorém, rezolučný princíp
- Splniteľnosť, analytické tablo, DPLL algoritmus, SAT solvery
- Logické modelovanie - reprezentácia rôznych typov ohraničení, redukcia symetrie
- Splniteľné modulové teórie, diferenčná logika, lineárna aritmetika, neinterpretované funkcie, SMT solvery
- Predikátový počet – syntax a sémantika, interpretácia, rezolúcia
- Pravidlové reprezentácie, štruktúra Reteho siete, typy uzlov, dopredná inferencia, jazyk Clips
- Spätná inferencia v pravidlových systémoch, logické programovanie, jazyk Prolog
- Kvantifikované Booleove formuly, modelovanie hier
- Modálne logiky, časové logiky, lineárna časová logika

SEKVENČNÉ ROZHODOVACIE ÚLOHY

Potrebné urobiť sekvenciu rozhodnutí (zvyčajne má podobu aktivít agenta v nejakom prostredí)

Každé rozhodnutie ovplyvňuje:

- Okamžitý/odložený zisk (odmena) plynúci z daného rozhodnutia
- Zmenu situácie
- Budúce možnosti agenta pôsobenia na prostredie
- Budúce zisky z budúcich rozhodnutí

Cieľom nie je čo najväčšia okamžitá odmena, ale maximalizácia **súčtu všetkých odmien** (aktuálnej aj budúcich). Odmeny musia byť nastavené tak, že maximálna kumulatívna odmena je získateľná iba pri optimálnej sekvencii rozhodnutí (reprezentujúcej hľadané riešenie).

STAVOVÝ PRIESTOR

Graf = stavový priestor

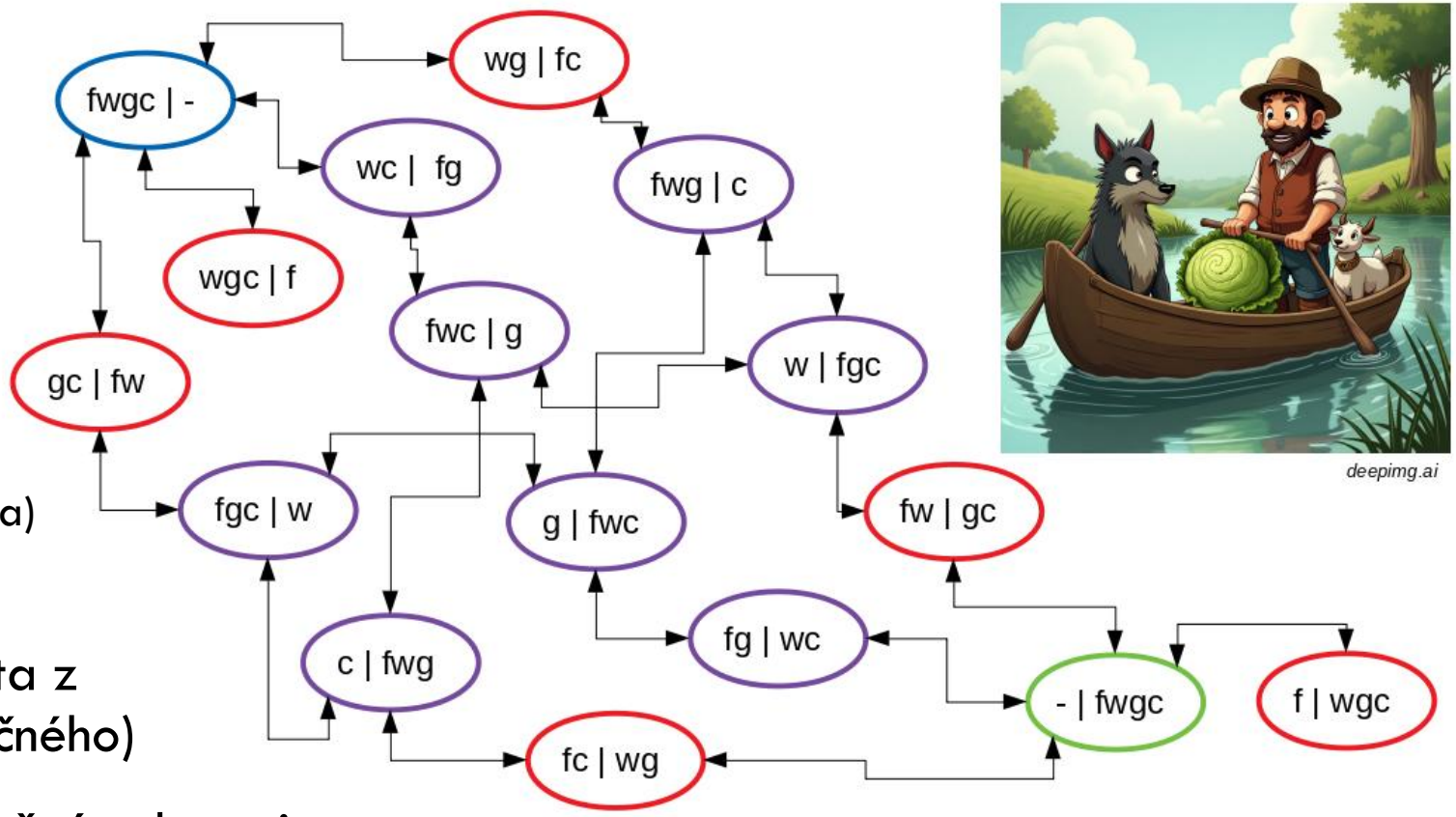
- Uzly = stavy
- Hrany = akcie

Nastavenie odmien

- Prechod do cieľového stavu (zisk)
- Prechod do zakázaného stavu (pokuta)
- Iný prechod (neutrálne, malá pokuta)

Kandidát = sekvencia akcií (cesta z počiatočného uzla do uzla konečného)

Priestor kandidátov = všetky možné sekvencie



KLASICKÉ PREHĽADÁVANIE STAVOVÉHO PRIESTORU

Prehľadávanie = budovanie stromu prehľadávania

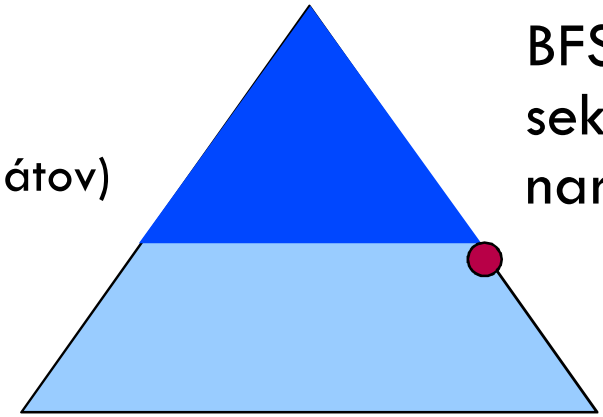
- Koreň stromu je štartovacím stavom
- Obsahuje všetky možné sekvencie v stavovom priestore (kandidátov)

Neinformované prehľadávanie (DFS, BFS)

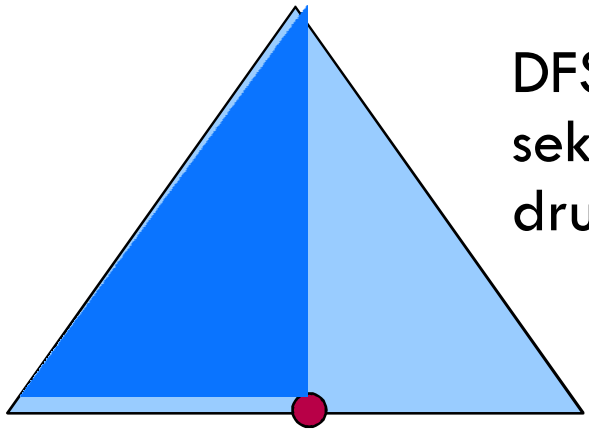
- Využíva iba štruktúru grafu (hrany a uzly)
- Systematické prechádzanie stromom dovtedy, pokiaľ
  - Dosiahne cieľový stav = úspešné hľadanie
  - Prehľadá celý strom = neúspešné hľadanie
- BFS je optimálny v zmysle dĺžky sekvencie

Informované prehľadávanie (A*)

- Využívanie dodatočnej informácie (váhy hrán)
- Má heuristickú ohodnocovaciu funkciu
- A* je optimálny v zmysle ceny sekvencie



BFS – všetky sekvencie naraz



DFS – jedna sekvencia za druhou

PREHĽADÁVANIE DO ŠÍRKY (BFS) A HĹBKY (DFS)

Neinformované (slepé) prehľadávanie do hĺbky/šírky

1. Počiatočný stav do OPEN
2. Ak je OPEN prázdny, prehľadávanie bolo neúspešné
3. Vyber z OPEN prvý stav S
4. Ak je S už v CLOSE, choď na krok 2, inak vlož ho do CLOSE
5. Nájdi všetky stavy, do ktorých sa dá dostať zo stavu S nejakou akciou
6. Ak niektorý z nájdených stavov je cieľovým, prehľadávanie bolo úspešné
7. Všetky nové stavy zaraď na začiatok/koniec OPEN
8. Choď na krok 2

OPEN – zoznam stavov, ktoré je potrebné ešte preskúmať

CLOSE – zoznam už preskúmaných stavov

HĽADANIE CESTY Z (FWGC | -) DO (- | FWGC)

1 - OPEN: **fwgc | -**
- CLOSE: **-**

2 - OPEN: **wc | fg**
- CLOSE: **fwgc | -**

3 - OPEN: **fwgc | -, fwc | g**
- CLOSE: **wc | fg, fwgc | -**

4 - OPEN: **fwc | g**
- CLOSE: **wc | fg, fwgc | -**

5 - OPEN: **c | fwg, w | fgc, wc | fg**
- CLOSE: **fwc | g, wc | fg, fwgc | -**

6 - OPEN: **fwc | g, fgc | w, w | fgc, wc | fg**
- CLOSE: **c | fwg, fwc | g, wc | fg, fwgc | -**

7 - OPEN: **fgc | w, w | fgc, wc | fg**
- CLOSE: **c | fwg, fwc | g, wc | fg, fwgc | -**

8 - OPEN: **g | fwc, c | fwg, w | fgc, wc | fg**
- CLOSE: **fgc | w, c | fwg, fwc | g, wc | fg, fwgc | -**

9 - OPEN: **fg | wc, fgc | w, fwg | c, c | fwg, w | fgc, wc | fg**
- CLOSE: **g | fwc, fgc | w, c | fwg, fwc | g, wc | fg, fwgc | -**

10 - OPEN: **fgc | w, fwg | c, c | fwg, w | fgc, wc | fg**
- CLOSE: **fg | wc, g | fwc, fgc | w, c | fwg, fwc | g , wc | fg, fwgc | -**

Ako následok prechodu z (fg | wc) do ďalších stavov bol generovaný cieľový stav (- | fwgc), algoritmus skončil úspešne.

Poz: prechody do nevalidných stavov neboli uvažované
zelené sú uzly, do ktorých sa dá dostať z modrého uzla

UČENIE POSILŇOVANÍM

Dáta, potrebné pre učenie chovania agenta (**politika**), typicky nie sú k dispozícii

„Pokus –úspech/omyl“ – učenie z úspechov (odmien) aj neúspechov (pokút)

Učenie v iteračnej štruktúre:

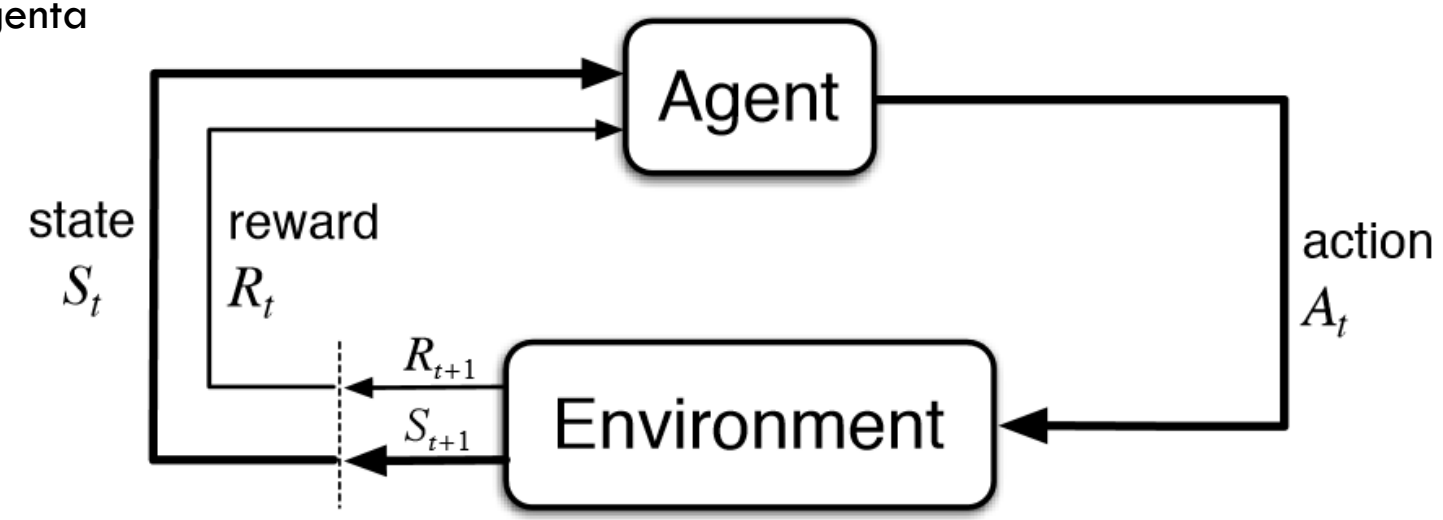
- Dáta sú získavané pôsobením agenta na prostredie (na základe nejakej **exploračnej** politiky)
- Na základe dát sa učí **cieľová** politika agenta
- Obe politiky môžu byť rovnaké/rôzne

Typy učenia:

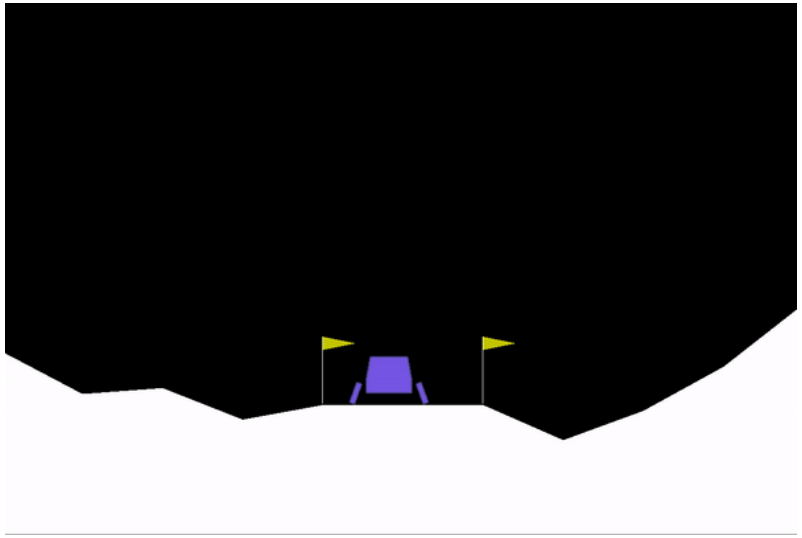
- Učenie po jednotlivých akciách
- Učenie po jednotlivých sekvenciách

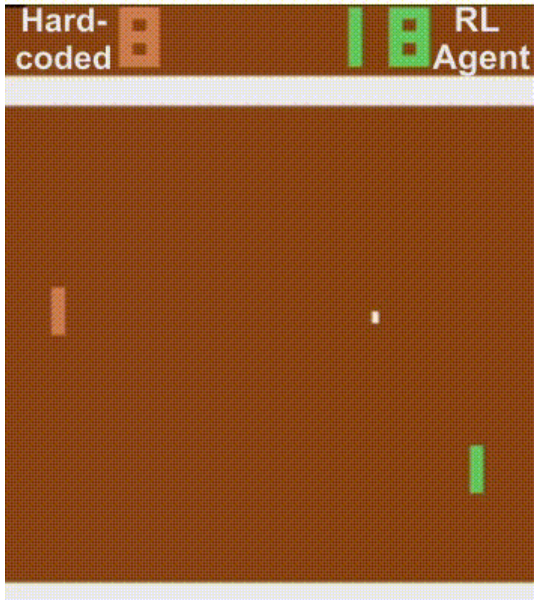
Výsledok učenia:

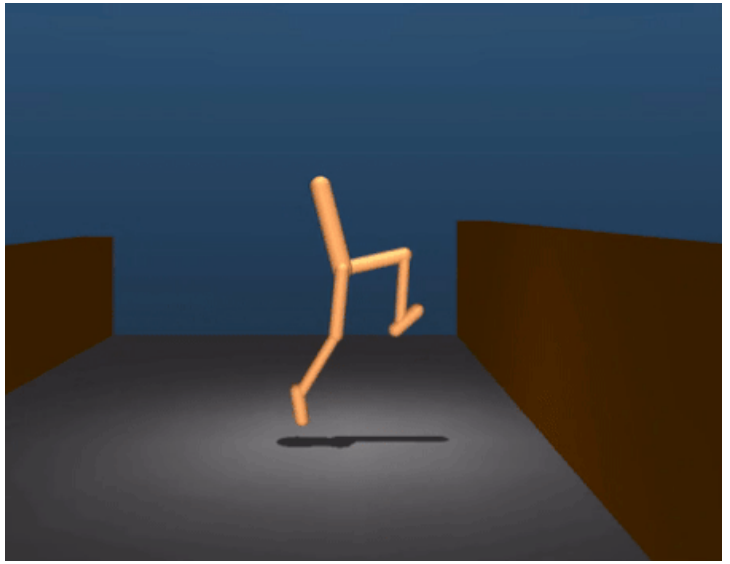
- Priamo politika
- Ohodnotenie akcií – politika je odvodená

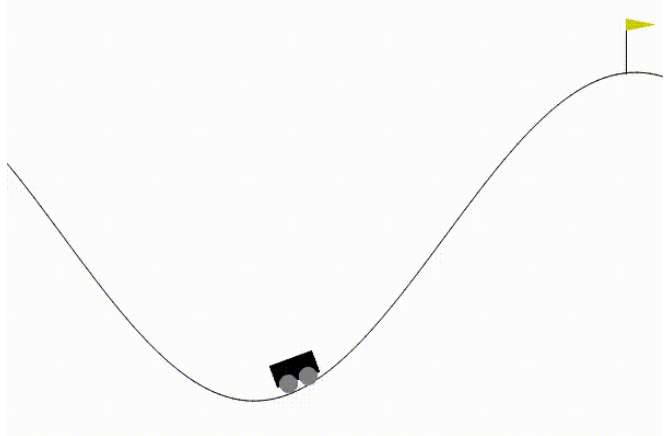


PRÍKLADY UČENIA POSILŇOVANÍM









STROJOVÉ UČENIE II (3. ROK LS)

KKUI
Katedra kybernetiky
a umelej inteligencie

- Markovovské rozhodovacie procesy
- Základné pojmy, podmienka optimality, Bellmanova rovnica, explorácia vs exploatácia, on vs off policy
- Dynamické programovanie
- Monte Carlo prístupy
- Temporal difference prístupy
- Hybridizácia základných prístupov
- Aproximácia hodnotovej funkcie, tabuľkové vs deep reprezentácie
- Policy gradient prístupy, Actor-critic
- Učenie s posilňovaním v doméne hier

Thank you for your attention!