ALGORITMY UMĚLÉ INTELIGENCE

Algoritmy umělé inteligence

Seznámíme se s následujícími pojmy:

- **stavovýprostor, operátor/akce, graf úlohy, cesta řešení**
- **informované a neinformované algoritmy hledání, algoritmus iterativního prohlubování, hodnotící a heuristická funkce, paprskové prohledávání, algoritmus A*, algoritmus IDA***
- **přípustné hledání, konzistentní heuristická funkce**
- **obousměrné hledání, směrování vln**

Skripta kap. 10, str. 164 – 171

Umělá inteligence

UI (AI) - součást informatiky s průniky mimo obor

Stručná historie UI

- 1943-56 začátky (modelování neuronů a sítí na počítači)
- 1952-69 velká očekávání (GPS, Lisp, microworlds)
- 1966-74 vystřízlivění
- 1969-79 znalostní systémy (zpracování nejistoty, plánování)
- 1980-88 UI se stává průmyslem (5. generace, star wars)
- od 1986 - znovuobjevení neuronových sítí
- současnost - mobilní agenti

? Co je to UI ?

? Etické, politické, sociální otázky ?

Cena nadace Vize 2000 za rok 2002 - Joseph Weizenbaum - Eliza

Řešení problémů hledáním

- **UI v širším pohledu** - viz předmět 33ZUI
- **Implementační nástroje UI** - viz předmět 36JUI, v X36TIN se věnujeme jen části navazující na **grafovou tématiku**.

"Problem Solving"

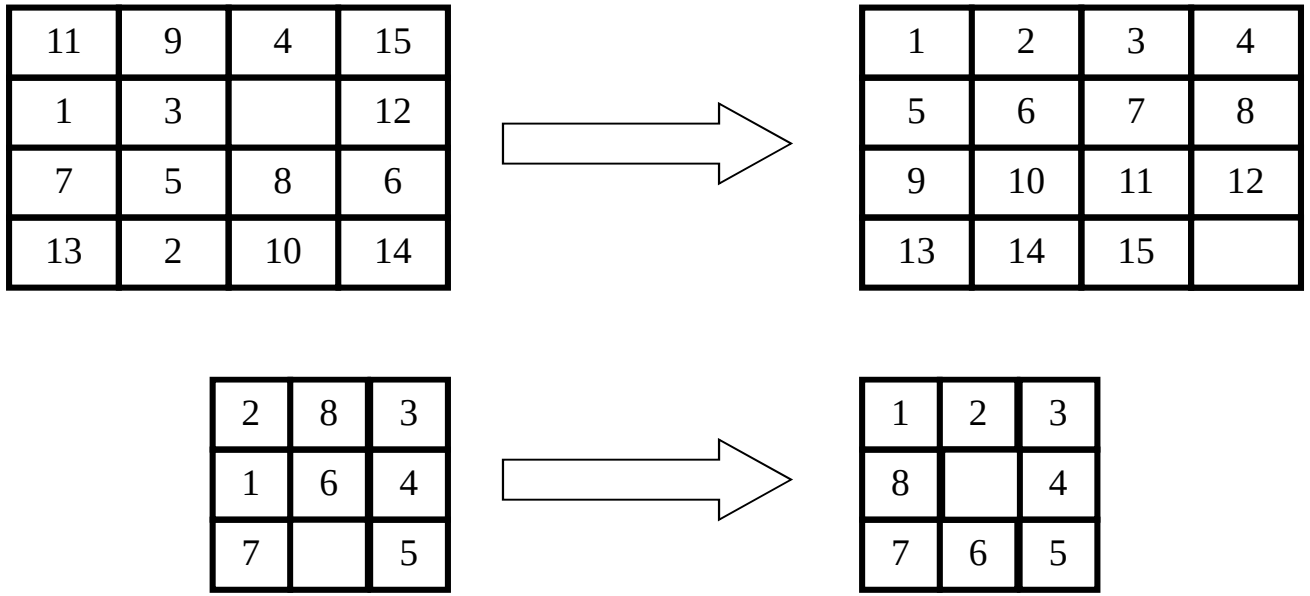
Problém (specifikace problému):
souhrn informací, podle nichž je možno rozhodovat se, co dělat

počáteční stav	**operátor/akce**	**stavový prostor**
cílový stav	**cesta řešení**	**cena cesty**
cena hledání	**graf problému**	

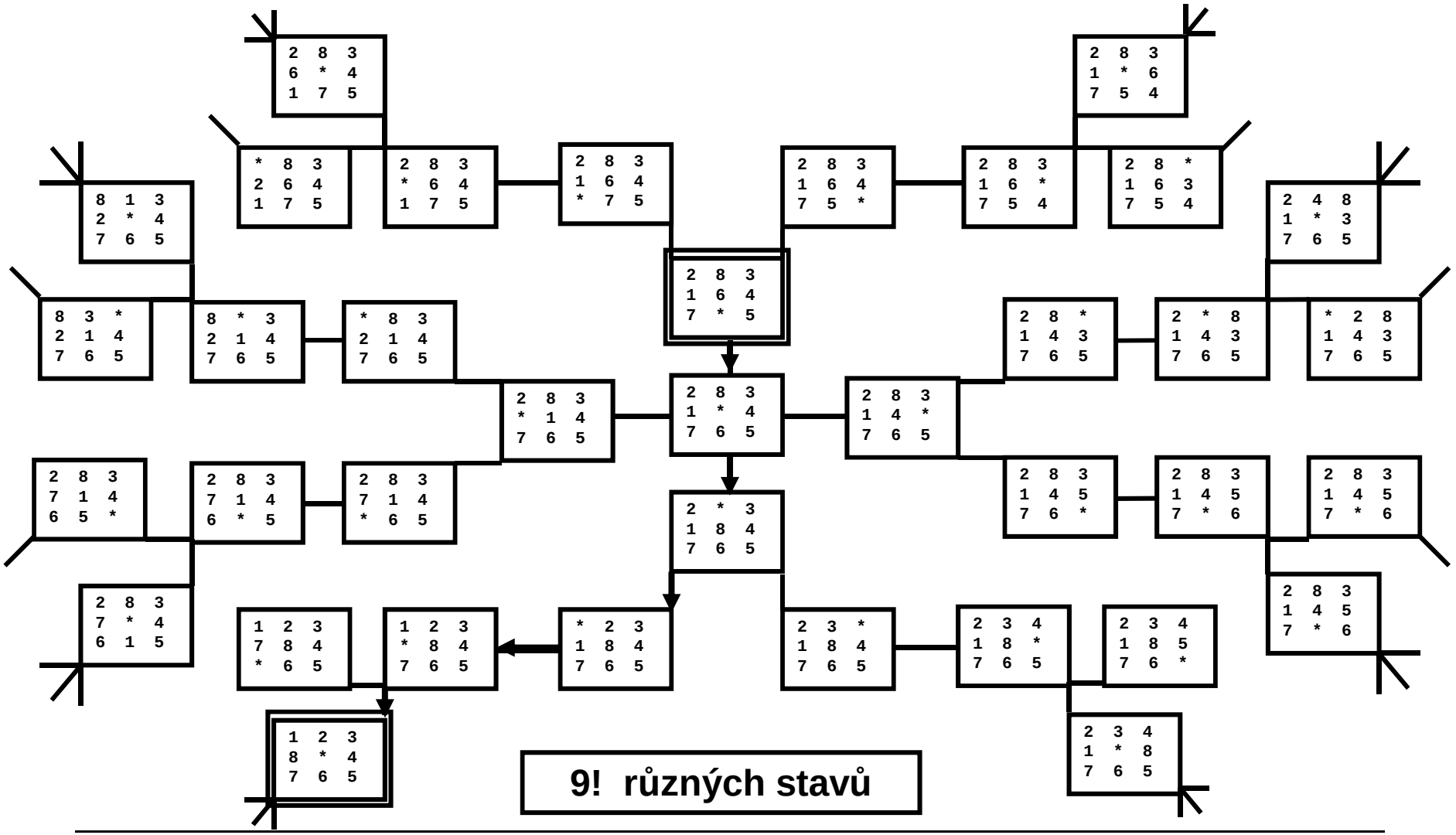
8 nebo 15 - problém

Zadání: Tabulka/krabička 3×3 nebo 4×4 s kameny číslovanými 1 až 8 (nebo 15) a jedním volným místem.

Úkol: posouváním seřadit kameny.



Stavový prostor 8-problému



Neinformované hledání

? Co nás zajímá na zvolené strategii hledání ?

- **úplnost** - zaručuje nalezení řešení (pokud existuje)
- **výpočetní** (časová) složitost
- **paměťová** složitost
- **optimálnost** - nalezne se nejlepší řešení?

Neinformované (slepé) vs. **informované** (heuristické) hledání

Běžné strategie neinformovaného hledání:

- prohledávání do šířky
- prohledávání uspořádaným výběrem (uniform cost)
- prohledávání do hloubky
- obousměrné prohledávání

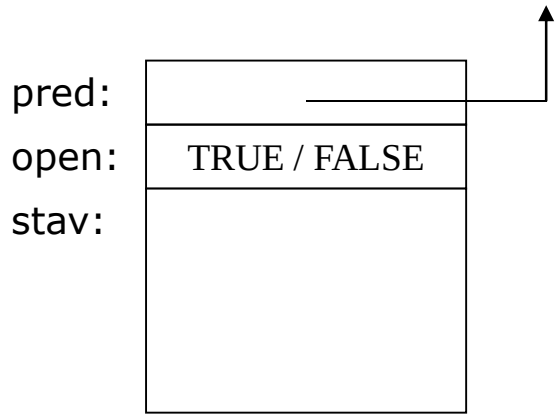
DS 14.4.2010

Rozdíl - graf problému je implicitní a (potenciálně) nekonečný

Důsledek:

- nejsou FRESH uzly, jen **OPEN** a **CLOSED**
- musíme uchovávat **stav** problému
- struktura grafu je zadána **operačně** (generováním následníků)

Data ukládaná pro každý uzel



Prohledávání do šířky

```
   boolean BFS (Succ Γ, State s, State t) {
1    ps = CreateNd(s);
2    ps.open = true;  ps.pred = null;
3    Queue.Init(); Queue.Push(ps);  Found = false;
4    while  ( !(Queue.Empty() || Found) ) {
5      pu = Queue.Pop(); pu.open = false;
6      if (pu.stav == t) Found = true;
7      else {
8        for (State v in Γ(u)) {
9          if ("v je nový stav")  {
10           pv = CreateNd(v); pv.open = true;
11           pv.pred = pu; Queue.Push(pv);
12     } } }
13   }
14   return Found;
15 }
```

! problém !

? Složitost ?

Prohledávání uspořádaným výběrem (uniform cost)

Předpokládá **ocenění cest** k uzlům **f(n)** - hodnotící funkce

```
   boolean UniformCost (Succ Γ, State s, State t) {
1    Queue.Init();
2    Queue.Push(CreateNode(s)); Found = false;
3    while (!(Queue.Empty() || Found)) {
4      pu = Queue.ExtMin(); pu.open = false;
5      if (pu.stav == t) Found = true;
6      else for (State v in Γ(u)) {
7        if ("v je nový stav")
8          "zařaď uzel v do fronty s hodnotou f(v)";
9        else "přepočti f(v) a možná zařaď do fronty";
10   } }
11   return Found;
12 }
```

Dikstrův algoritmus: f(n) = d(s,u)

? Nalezne se optimální řešení ?

<u>Podmínka:</u> hodnoty **f(n)** neklesají podél cesty řešení.

Je splněno automaticky:

- když se cena uzlu počítá jako **součet cen** jednotlivých hran (akcí) podél odpovídající cesty a
- když jsou tyto **ceny nezáporné**.

Prohledávání do hloubky a obousměrné

? Jak prohledávat do hloubky nekonečný stavový prostor ?

Řešení:

- omezení hloubky hledání (? úplnost ?)
- **iterativní prohlubování**

Další možnosti ?

Obousměrné prohledávání (do šířky)

Podmínka: explicitně zadaný cílový stav

Srovnání neinformovaných metod

	BFS	Unif. Cost	DFS	Lim. Depth	Iter. Deep.	Bi-direct
čas	b^d	b^d	b^m	b^k	b^d	$b^{d/2}$
paměť	b^d	b^d	b.m	b . k	b.d	$b^{d/2}$
optimal	ANO	ANO	NE	NE	ANO	ANO
úplnost	ANO	ANO	NE	ANO pro k>=d	ANO	ANO

b=koeficient větvení, d = délka cesty řešení, m = hloubka hledání,
k = omezení hloubky

Heuristické hledání

Předpokládejme, že jsme schopni odhadovat délku (zbývající) cesty k cíli - **heuristická funkce h(u).**

Označení:
g(u) ... délka cesty od počátku do u, d(s,u)
h(u) ... délka cesty od u do cíle, d(u,t)
f(u) ... hodnotící funkce uzlu (kombinuje g a h)
g'(u), h'(u) , f'(u) ... aproximace g, h, f

Použijeme uspořádané hledání s hodnotící funkcí f(u):
f '(u) = h'(u) ... Best-First

f '(u) = h'(u) + g'(u) ... A* (A-star) algoritmus

f '(u) = α . h'(u) + (1- α) . g'(u)

Heuristické hledání

Jak nejlépe odhadovat délku (zbývající) cesty k cíli –
výpočet **heuristické funkce h(u)?**

Pro 15-problém:

- počet kostek mimo domovskou pozici
- součet vzdáleností všech kostek od domovské pozice

Hledání v rovinné síti:

- vzdálenost "vzdušnou čarou"

Přípustnost heuristického hledání

Přípustný algoritmus hledání - najde optimální řešení (min. cestu), pokud existuje.

Lemma: Nechť h'(u) ≤ h(u) pro všechny uzly. Potom až do skončení A* existuje na minimální cestě P(s,t) otevřený uzel u' tak, že

$$\mathbf{f'(u') \leq f(s)}$$

D: $P=\langle s=u_0, u_1, u_2, \ldots v, \ldots, u_n=t\rangle$ je min. cesta, v otevřený

$f'(v) = h'(v) + g'(v)$,

přitom g'(v) = g(v) - předchůdci CLOSED, $h'(v) \leq h(v)$ tedy

$f'(v) = h'(v) + g'(v) \leq h(v) + g(v) = f(v) = f(s)$

v je tedy hledaným uzlem u'

V: Nechť h'(u) ≤ h(u) pro všechny uzly a délky všech hran jsou větší než jisté $\delta > 0$. Pak je algoritmus A* přípustný.

h'(u) je **konzistentní** ... pro každé u,v platí

h'(u) - h'(v) ≤ d(u,v)

V: Nechť h'(u) je konzistentní a u byl uzavřen během A*. Potom je g'(u) = g(u).

V: Nechť

- A_1* je A* s heuristickou funkcí h_1'(u),
- A_2* je A* s heuristickou funkcí h_2'(u),
- h_1'(u) ≥ h_2'(u) pro všechny uzly,
- h_1'(u) je konzistentní

Potom každý uzel expandovaný algoritmem A_1* bude expandován také algoritmem A_2*.

více informované hledání

Hledání s omezenou pamětí

Paměťová složitost A* ... počet uzlů roste s b^d, d = délka cesty řešení.

? Jak dosáhnout zlepšení ? Pokud

$$|h'(u) - h(u)| = O(\log h(u)),$$

pak roste počet uzlů pomaleji. Jak ale nalézt takové h'(u) ?

? Jak vyjít s danou velikostí paměti ?

Dvě varianty A*:

- **IDA* - iterative deepening A***
- **SMA* - simplified memory-bounded A***

IDA*

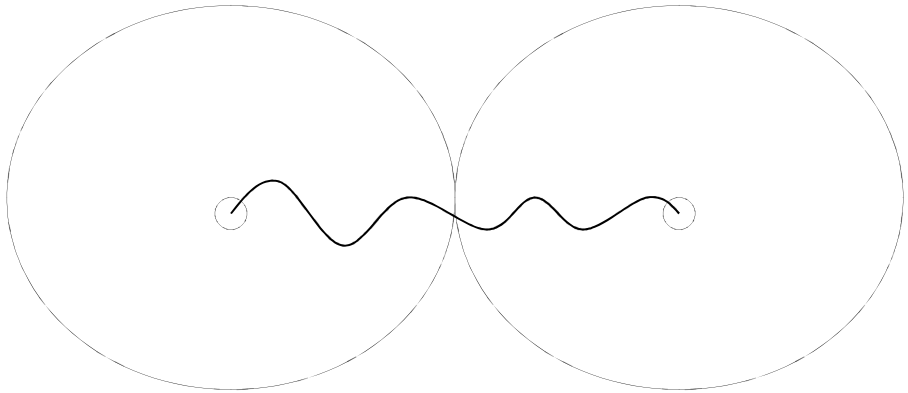
```
1 function IDA* (Node s, Succ Γ) {
2    f_limit = f(s); solution = null;
3    while ((solution == null) && (f_limit < ∞))
4      solution, f_limit = DFS(s,f_limit);
5    if (solution == null) return failure;
6    else return solution;
```

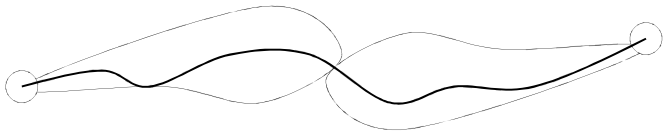
```
1   DFS (Node u, double f_lim);
2     if f(u) > f_lim then return nil, f(u);
3     if ( goal(u) ) return u, f_lim;
4     next_f = ∞;
5     for (Node x in Γ(u)) {
6       sol, fnew = DFS(x, f_lim);
7       if (sol != null) return sol, f_lim;
8       next_f = min(next_f, fnew);
9     }
10    return nil, next_f;
11  }
```

Vrátí řešení nebo nejbližší hodnotu f za limitem

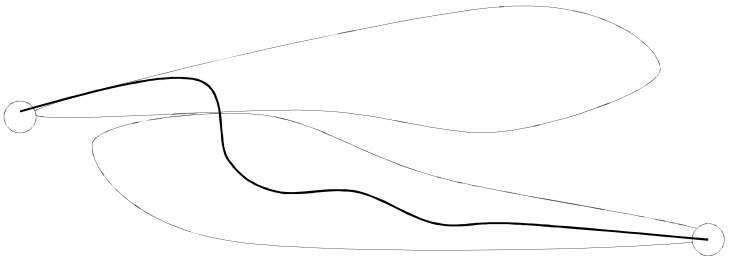
Obousměrné heuristické hledání

do šířky





heuristické - ideál



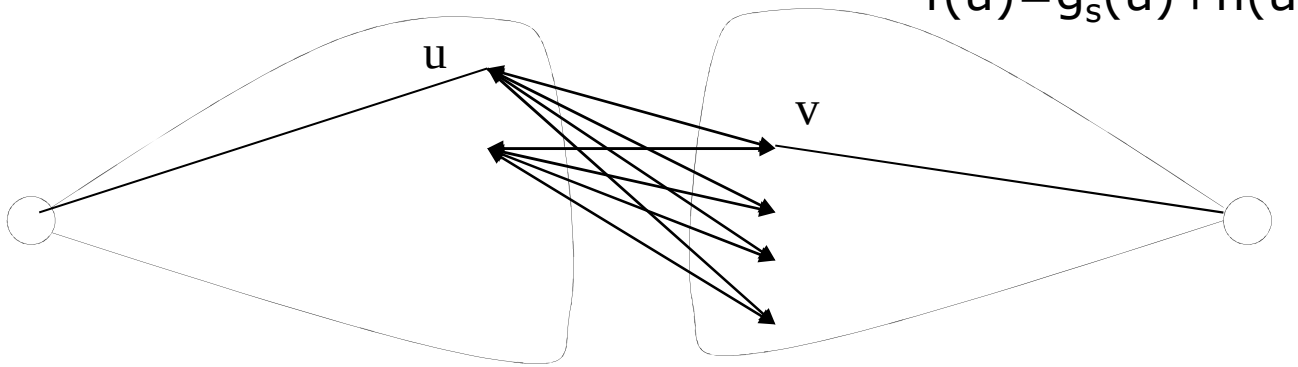
heuristické - skutečnost

Kriteria: optimálnost cesty / počet generovaných uzlů

Zlepšení obousměrného heuristického hledání

směrování vln

$f(u)=g_s(u)+h(u,v)+g_t(v)$



přibližné směrování vln

D_s, D_t - uzly nejvzdálenější od s / t, úprava každou n-tou iteraci