Prostředky umělé inteligence

Řešení úloh ve stavovém prostoru II.

Neinformované metody prohledávání

Metody neinformovaného prohledávání se dělí z hlediska pořadí, v jakém jsou uzly expandovány, na:

- ***slepé prohledávání do šířky***
  - nejdříve se expanduje uzel s nejmenší hloubkou,
  - nalezne nejmenší řešení (s nejmenším počtem operátorů – nejkratší cestu).
- ***slepé prohledávání do hloubky***
  - nejdříve se expanduje uzel s největší hloubkou,
  - má nižší nároky na paměť (uchovává pouze uzly na cestě od počátečního stavu ke stavu právě expandovanému),
  - často spojeno s omezením maximální prohledávané hloubky, při jejím dosažení se používá mechanismu navracení.

Pozn.:

Hloubka uzlu = počet hran od počátečního uzlu k uzlu.

Neinformované metody prohledávání

Pozn.:

Kompromis mezi prohledáváním do šířky a do hloubky:

Algoritmus $DFID$ ***(algoritmus iterativního prohlubování)***

- Úplné prohledávání do hloubky tak, že se v každé iteraci zvyšuje povolená hloubka prohledávání o 1.
- První nalezené řešení je optimální (ve smyslu nejkratší délky cesty).

Pozn.:

- Neinformované metody jsou použitelné jen v nejjednodušších úlohách.
- Nepoužívání znalostí o úloze vede na prohledávání příliš velkých částí stavového prostoru.
- Představují metodologický podklad pro složitější, informované strategie.

Informované metody prohledávání

Hodnotící funkce f :

- pro každý uzel stromu řešení určí jeho ohodnocení,
- hodnoty se používají pro výběr uzlu k expanzi,
- pokud hodnotící funkce dobře postihují vlastnosti a charakter úlohy, budou vždy expandovány „nejperspektivnější" uzly a zabrání se prohledávání cest, které nevedou k cíli,
- čím kvalitnější heuristické znalosti o dané úloze se v f využijí, tím efektivnější bude prohledávání.

Informované metody prohledávání

Gradientní algoritmus

- Vždy se expanduje ten uzel, který měl dosud nejlepší hodnotící funkci, a vyhodnocují se jeho následníci.
- „Rodič" stejně jako „sourozenci" tohoto uzlu jsou okamžitě zapomenuty. Pracuje se pouze s právě expandovaným uzlem.
- Hledání je zastaveno, když je dosaženo stavu, který má lepší ohodnocení funkcí f než jeho následníci.

Pozn.:

- Základní nevýhodou všech gradientních metod je riziko uvíznutí v lokálním extrému.
- Není vyloučeno zacyklení (pohyb po nekonečně dlouhé cestě, např. s konstantním ohodnocením).

Informované metody prohledávání

Algoritmus uspořádaného prohledávání

- Je gradientní algoritmus rozšířený o paměť.
- Expandují se stavy s minimální hodnotou funkce f .
- Používá obvyklé seznamy:
  - OPEN – seznam neexpandovaných stavů
  - CLOSED – seznam již expandovaných stavů
- Prvky seznamů jsou trojice $\langle \text{jméno uzlu}, f, \text{jméno rodičovského uzlu} \rangle$ (zápis stavu do seznamu označuje zápis uvedené trojice).

Informované metody prohledávání

1. Počáteční stav zapiš do seznamu OPEN, seznam CLOSED je prázdný.
2. Pokud je seznam OPEN prázdný, řešení neexistuje, ukonči prohledávání.
3. Ze seznamu OPEN vyber stav i s nejmenší hodnotou $f(i)$. V případě většího počtu stavů se stejnou minimální hodnotou $f(i)$ prověř, zda některý z těchto stavů není stavem cílovým, v takovém případě jej vyber; jinak vyber mezi stavy se stejnou minimální hodnotou $f(i)$ libovolně.
4. Vymaž stav i ze seznamu OPEN a zařaď jej do seznamu CLOSED.
5. Je–li stav i cílovým stavem, řešení je nalezeno, ukonči prohledávání.
6. Expanduj stav i; pro každého následníka j stavu i vypočítej hodnotu $f(j)$. Pokud stav j není ani v seznamu OPEN, ani v seznam CLOSED, zařaď jej do seznamu OPEN. Pokud je stav j v seznamu OPEN nebo CLOSED, avšak s ohodnocením větším než právě vypočtené $f(j)$, změň jeho ohodnocení na $f(j)$, změň jméno rodičovského uzlu v zápisu uzlu a zařaď ho do seznamu OPEN.
7. Pokračuj krokem č.2.

Informované metody prohledávání

Pozn.:

Algoritmus uspořádaného prohledávání má celou řadu modifikací. Velmi známý je:

Algoritmus paprskového prohledávání

- pracuje se seznamem OPEN konečné délky m, do seznamu OPEN se zapisují pouze ty nově expandované uzly, které mají lepší ohodnocení než uzly dosud obsažené v seznamu.
- To má za následek značnou redukci expandovaného stromu, ale nemusí vést k nalezení cílového stavu.

Informované metody prohledávání

Algoritmus A

jde o algoritmus uspořádaného prohledávání s hodnotící funkcí

$f(i)=g(i)+h(i)$,

kde $g(i)$ je cena přechodu z počátečního stavu s_0 do stavu i, $h(i)$ je cena optimální cesty ze stavu i do některého z cílových stavů $s \in C$.

Pozn.:

Funkci $f(i)$ lze chápat jako cenu přechodu z počátečního stavu do cílového stavu přes stav i.

Informované metody prohledávání

- Ve většině úloh $g(i)$, $h(i)$ není známo, používá se jejich odhadů $g^*(i)$, $h^*(i)$.
- Funkci $g(i)$ odhadneme minimální dosud zjištěnou cenou $g^*(i)$ přechodu z počátečního stavu do stavu i
- Funkci $h(i)$ nahrazujeme funkcí $h^*(i)$, která kvantitativně vyjadřuje náš odhad ceny cesty z uzlu i do některého z cílových stavů. Odhad vyjadřuje naši heuristickou znalost o tom, jaké jsou šance nalézt (optimální) řešení, pokračovali–li bychom expanzí daného uzlu. Funkce $h^*(i)$ je proto *nositelem heuristické informace* a bývá proto nazývána heuristickou funkcí.

Pozn.:

Heuristická funkce je pro efektivnost prohledávání podstatná.

Informované metody prohledávání

Řekneme, že algoritmus prohledávání je přípustný, jestliže vždy nalezne optimální cestu, pokud tato cesta existuje.

Algoritmus A s přípustnou heuristickou funkcí je algoritmus A^*.

Heuristická funkce je přípustná, je–li $h^*(i) \geq 0$ a $h^*(i) \leq h(i)$, pro všechny uzly i (tj. h^* je nezáporný dolní odhad h).
Čím je h^* lepším dolním odhadem h, tím menší část stavového prostoru se prohledává při hledání optimálního řešení (při $h^*=h$ expanduje algoritmus A^* pouze stavy na cestě k cílovému stavu).

Říkáme, že algoritmus A_1^* je *lépe informovaným algoritmem* než A_2^*, jestliže pro každý stav i je $h_1^*(i) \geq h_2^*(i)$. A_2^* pak expanduje všechny stavy, které expanduje A_1^*. Mohou však existovat stavy, které expanduje A_2^* a neexpanduje A_1^*.

Informované metody prohledávání

Funkce $g^*(i)$

- Umožňuje vybírat stavy k expanzi na základě posouzení jak „dobrá" byla cesta z počátečního do daného stavu. Tj. ne vždy jsou k expanzi vybírány uzly, které se zdají být nejblíže k cíli.
- Velmi důležitá, pokud nejde pouze o nalezení řešení, ale i o optimalizaci procesu nalézání řešení. Pro nalezení řešení jakýmkoliv způsobem stačí volba $g^*=a$ (konstantní funkce), obvykle $a=0$.

Informované metody prohledávání

Funkce $h^*(i)$

- ***Algoritmus se stejnoměrnou cenou*** $h^*(i)=a$ (prohledávání řízeno pouze funkcí $g^*(i)$)
- ***Algoritmus větví a mezí***
  je algoritmus se stejnoměrnou cenou, který si po nalezení cílového stavu zapamatuje jeho cenu a pokračuje v prohledávání. Ze seznamu OPEN však automaticky vyřazuje uzly z vyšší cenou, než je zapamatovaná cena cílového stavu. Pokud je nalezen cílový uzel s nižší cenou, pamatuje se nižší cena a algoritmus pokračuje až do okamžiku, kdy je seznam prázdný.
- ***Algoritmus náhodného prohledávání***
  $g^*(i)=a$, $h^*(i)=b$

Pozn.

Při $g^*(i)=\text{hloubka}(i)$, $h^*(i)=0$ degeneruje algoritmus A^* na algoritmus slepého prohledávání do šířky.

Informované metody prohledávání

- Nelze ověřit, že jde o A^* (tj. podmínku přípustnosti $0\le h^*\le h$). Přípustnost heuristiky lze prokázat pouze kompletním prozkoumáním stavového prostoru, což lze provést pouze u jednoduchých úloh. Pak ale není heuristického prohledávání zapotřebí.
- Nemožnost posouzení přípustnosti heuristické funkce v praktických úlohách je největší slabinou algoritmů A^*.

Zmírnění požadavku přípustnosti

Pokud h^* někdy (ale málokdy) nadhodnocuje h a hodnotu větší než δ, pak algoritmus A (nejde už o A^*) zřídkakdy nachází řešení, jehož cena je větší o více než δ ve srovnání s cenou skutečně optimální cesty.

Informované metody prohledávání

Algoritmus A^* s monotonní $h^*(i)$

Přípustnost algoritmu A^* nezávisí na volbě $g(i)$ (viz 6. krok algoritmu uspořádaného prohledávání, který prověřuje, jestli během prohledávání nebyla nalezena kratší cesta do již dříve expandovaného stavu).

Je–li $h^*(i)$ monotonní, lze 6. krok algoritmu A podstatně zjednodušit (je možné vypustit větu *„Pokud stav j je již v seznamu OPEN nebo CLOSED, ... a zařaď ho do seznamu OPEN.“*)

Pozn.:

- $h^*(i)$ je monotonní, platí–li $h^*(i)-h^*(j)\le\text{cena}(i,j)$, kde $\text{cena}(i,j)$ znamená cenu přechodu od stavu i do stavu j.
- Podstatné zjednodušení nemusí znamenat obecně zrychlení prohledávání.

Informované metody prohledávání

Algoritmus IDA^*

- je kombinací algoritmu A^* s algoritmem $DFID$.
- Algoritmus provádí iterativní prohledávání do hloubky, přičemž vylučuje ty uzly, jejichž cena přesahuje prahovou hodnotu.
- Hodnota prahu se dynamicky mění, na začátku je rovna odhadu f^* ceny počátečního uzlu.
- V každé iteraci se práh zvyšuje o minimální hodnotu, o kterou byl v předchozím kroku překročen.

Pozn.:

- Využívá–li přípustnou heuristickou funkci, zaručuje nalezení optimálního řešení.
- Nízká paměťová náročnost (preferováno prohledávání do hloubky).

Informované metody prohledávání

Dosud uvedené algoritmy:

- mají charakter základních (elementárních) algoritmů prohledávání stavového prostoru,
- prakticky bývají kombinovány s některými dalšími metodami, vycházejícími z představy stavového prostoru se složitostí redukovanou znalostmi o řešené úloze.

Další metody informovaného prohledávání

- Pravidla se složitějšími předpoklady,
- Metaznalosti,
- Metoda generování a testování,
- Metoda užití analogie,
- Rozklad úlohy na podúlohy.

Další metody informovaného prohledávání

Pravidla se složitějšími předpoklady

- Zahrnují znalosti do předpokladů pravidel v podobě dalších předpokladů. Tím se sníží počet stavů, na které je pravidlo použitelné (a tak i nově generovaných).
- Prohledává se menší část stavového prostoru.

Pravidla se složitějšími předpoklady / Příklad

Přelévání vody

Znalosti o úloze:

1. Nevyprazdňuj obě nádoby.
2. Nenaplňuj obě nádoby.
3. Nenaplňuj nádobu, je–li druhá prázdná a není–li naplňovaná prázdná
4. Nevyprazdňuj nádobu, je–li druhá plná.
5. Nepřelévej, pokud by potom byla jedna nádoba plná a druhá prázdná.

Pravidla se složitějšími předpoklady / Příklad

Produkční pravidla

$\{(c_A>0)\wedge(c_B>0)\wedge(c_B\neq b)\}\rightarrow\{A\rightarrow\}$	Vylej A
$\{(c_A>0)\wedge(c_B>0)\wedge(c_A\neq a)\}\rightarrow\{B\rightarrow\}$	Vylej B
$\{(c_A<a)\wedge(c_B\neq b)\wedge((c_A=0)\vee(c_B\neq 0))\}\rightarrow\{\rightarrow A\}$	Naplň A
$\{(c_A\neq a)\wedge(c_B<b)\wedge((c_A\neq 0)\vee(c_B=0))\}\rightarrow\{\rightarrow B\}$	Naplň B
$\{(c_A>0)\wedge(c_B<b)\wedge(c_A+c_B\neq b)\}\rightarrow\{A\rightarrow B\}$	Přelej A do B
$\{(c_A<a)\wedge(c_B>0)\wedge(c_A+c_B\neq a)\}\rightarrow\{A\leftarrow B\}$	Přelej B do A

Metaznalosti

jsou pravidla (metapravidla) používání pravidel definujících úlohu.

Pozn.:

Metapravidla mají nejčastěji tvar

$\{\text{situace S}\}\rightarrow\{\text{preference A}\}$

s interpretací

„Nastala – li v bázi dat situace S , preferuj pravidlo A“.

Metoda generování a testování

Znalosti o úloze jsou rozděleny mezi

- ***generátor***, který expanduje stav (nabízí jednotlivé stavy), a
- ***testér***, který stavy nabízené generátorem testuje a odmítá, nesplňují–li zadaná omezení.

Pozn.:

- Velmi efektivní je ***metoda hierarchicky uspořádaného generování a testování***, která umožňuje odmítnout stav na základě jenom jeho částečného popisu (jediným testem lze naráz zamítnout větší počet nevyhovujících stavů).

Metoda užití analogie

Řešení úlohy v prostoru I lze použít jako podklad pro řešení úlohy v prostoru II

- je–li odhalena určitá podobnost mezi dvěma stavovými prostory pro dvě, jinak i zcela odlišné úlohy, nebo
- byl–li řešen obdobný případ.

Pozn.:

Toho využívá tzv. ***usuzování na základě příkladů*** (***případové usuzování***)

Rozklad úlohy na podúlohy

Pro řadu úloh je velmi obtížné (ne–li nemožné) definovat heuristiky ve formě hodnotící funkce, pravidel, či metapravidel. Někdy je však možné získat heuristiky, kterými lze původní úlohu rozložit na (snadněji řešitelné) podúlohy. Není–li podúloha elementární, můžeme se ji znovu pokusit rozložit na podúlohy atd.

Rozklad úlohy lze reprezentovat orientovaným ***AND/OR grafem***, jehož uzly znázorňují úlohu a podúlohy. Uzly, které nejsou listy, mohou být typu:

- ***AND*** – představují konjunkci podúloh (k řešení úlohy je nezbytné, aby každá z podúloh byla řešitelná),
- ***OR*** – představují disjunkci podúloh (k řešení úlohy stačí, aby aspoň jedna z podúloh byla řešitelná).

Rozklad úlohy na podúlohy

Jestliže víme, že výsledné řešení musí procházet jistým mezilehlým stavem (tzv. mostem), lze původní úlohu rozdělit na dvě podúlohy:

- podúlohu I s počátečním stavem s_0 a cílovým stavem s_M,
- podúlohu II s počátečním stavem s_M a cílovým stavem $s_c \in C$.

Pozn.:

Uvažujeme–li právě 2 aplikovatelné operátory v každém ze stavů, je v nejhorším případě slepého prohledávání prostorů obou podúloh třeba vygenerovat

$2^{n_I}+2^{n_{II}}$ uzlů,

kde n_I, n_{II} je přípustná hloubka stromu ve stavovém prostoru podúlohy I, resp. II.

V nejhorším případě slepého prohledávání prostoru celé úlohy je třeba vygenerovat

$2^{n_I+n_{II}}$ uzlů.

Protože $2^{n_I}+2^{n_{II}} < 2^{n_I+n_{II}}$, je výhodnější rozdělení úlohy, přičemž úspora generování může být značná.

Rozklad úlohy na podúlohy

Nechť je každý stav popsán R složkami

$s_i \equiv \left(s_{i,1}, s_{i,2}, \ldots, s_{i,R}\right)$.

Potom lze úlohu U stručně zapsat jako

$U\colon\ s_0 \equiv \left(s_{0,1}, s_{0,2}, \ldots, s_{0,R}\right) \rightarrow s_c \equiv \left(s_{c,1}, s_{c,2}, \ldots, s_{c,R}\right)$.

Úloha U bude vyřešena nalezením operátoru, který odstraňuje všechny diference. V případě

- triviální úlohy, kdy $s_{0,r} = s_{c,r}$, $r = 1,2,\ldots,R$, je úloha vyřešena okamžitě,
- netriviální úlohy, kdy $s_{0,r} \neq s_{c,r}$, $r \in \{1,2,\ldots,R\}$, tj. „existuje diference aspoň jedné z odpovídajících si složek", lze diference někdy měřit, jindy lze pouze konstatovat „diference existuje/neexistuje".

Rozklad úlohy na podúlohy

Nejjednodušší by bylo rozložit úlohu na R podúloh, z nichž by každá zajišťovala odstranění diference jedné složky. To není obecně použitelné díky možným ***vedlejším (postranním) efektům*** jednotlivých operátorů, kdy kromě odstranění diference jedné složky může operátor změnit i jiné diference.

Předem není známo, která pravidla budou použita, proto díky vedlejším efektům není možno stanovit ani cílové stavy podúloh a počáteční stavy navazujících podúloh (mosty). Vedlejší efekt může navíc obnovit dříve odstraněnou diferenci.

Řídicí algoritmus musí být schopen obecně formulovat podúlohy podle předchozího průběhu řešení úlohy.

Rozklad úlohy na podúlohy / systém GPS

Metoda analýzy prostředků a cílů (systém GPS)
Používá procedury formulované na základě poznatků a psychologie lidského myšlení:

- TRANSFORM sestavuje a řeší (znovu) podúlohu, tj. převedení současného stavu do cílového,
- REDUCE vybírá pravidlo, které by odstranilo (zmenšilo) nejdůležitější diferenci mezi současným a cílovým stavem,
- APPLY aplikuje vybrané pravidlo na daný stav.

Rozklad úlohy na podúlohy / systém GPS

Cílový stav poslední vyřešené podúlohy se nachází na vrcholu zásobníku. Poslední vyřešená podúloha měla za úkol odstranit jistou diferenci (v té době nejzávažnější). Vedlejší efekty však mohly způsobit vznik ještě závažnější diference. Takto vzniklá, nyní nejzávažnější má být odstraněna korekční podúlohou sestavenou prostřednictvím TRANSFORM.

Vlastní řešení

- začíná pomocí REDUCE, která však nebere v úvahu vedlejší efekty vybraného pravidla.
- Po vybrání pravidla se aktivuje APPLY.
- Nejde–li pravidlo použít, je procedurou určeno proč a znovu použito REDUCE. Ta nyní hledá pravidlo, které by odstranilo diference bránící použití původně vybraného pravidla.

Rozklad úlohy na podúlohy / systém GPS

Proces se rekurzívně opakuje tak dlouho, až je buď

- možné celou sekvenci nalezených pravidel aplikovat (a tím odstranit diferenci pomocí TRANSFORM, tj. vyřešit danou úlohu) nebo
- se prokáže, že úloha není řešitelná.

Pozn.:

Zásobník je struktura LIFO.

Hlavní nevýhoda

Je nutno definovat diference a uspořádat je podle závažnosti a dodat návod, jaká pravidla vybírat pro odstranění diferencí.

Rozklad úlohy na podúlohy / systém STRIPS

Má schopnost samostatně vybírat diference a pravidla. Sám zjišťuje diference mezi stavy, interpretuje je jako cílové a vybírá vhodná pravidla.

Stavy jsou formulovány jako množiny formulí predikátové logiky. Každý operátor je tvořen trojicí

$\langle B_\varphi, D_\varphi, F_\varphi \rangle$,

kde B_φ je podmínka aplikovatelnosti operátoru,

D_φ je množina klauzulí, které mají být ze stávající množiny vypuštěny,

F_φ je množina klauzulí, které mají být přidány.

Rozklad úlohy na podúlohy / systém STRIPS

Zadat je třeba pouze počáteční množinu platných formulí, cílovou formuli a množinu operátorů.

V případě

- triviální úlohy je cílová formule obsažena v počáteční množině formulí a úloha je vyřešena,
- netriviální úlohy jsou hledány diference mezi množinou platných formulí a cílových formulí a pak je hledán vhodný operátor k odstranění. Je–li nalezen, je jeho podmínka aplikovatelnosti považována za nový podcíl, vzhledem k němuž se celý postup opakuje.

Rozklad úlohy na podúlohy / systém PLANNER

Explicitně využívá procedurální reprezentaci znalostí. Striktně odlišuje

- Databázi obsahující fakta, která se týkají konkrétní, právě řešené úlohy a jsou reprezentována deklarativně,
- Bázi znalostí obsahující obecné znalosti z problémové oblasti ve formě pravidel. Tato pravidla jsou vlastně procedury působící na databázi a představují procedurální reprezentaci znalostí.

Pozn.:

- PLANNER nebyl nikdy implementován, pouze zjednodušené verze PROPLER, MICRO-PLANNER. V jistém smyslu na něj navazuje programovací jazyk PROLOG.
- Systémy GPS, STRIPS, PLANNER nedosáhly praktického rozšíření, jsou teoretickým zdrojem prací v oblasti řešení úloh.
- Moderní systémy UI kladou daleko větší důraz na využívání vysoce speciálních znalostí. Do popředí vystupují mj. otázky efektivní reprezentace znalostí.