

Politechnika Poznańska
Instytut Informatyki

Adam Meissner

Adam.Meissner@put.poznan.pl

<http://www.man.poznan.pl/~ameis>

SZTUCZNA INTELIGENCJA

Modelowanie problemów za pomocą grafu stanów

Literatura

- [1] Russell S.J., Norvig P., *Artificial Intelligence: A Modern Approach, Third Edition*, Prentice Hall, 2010.

Wprowadzenie

- ogólny charakter prezentowanego modelu (odniesienia do systemu GPS)
- warunki stosowalności:
 - reprezentacja problemu w postaci zbioru obiektów
 - każdy obiekt ma atrybuty (cechy), którym nadaje się wartości
 - na obiektach wykonuje się działania powodujące zmianę wartości cech
 - problem ma określone „warunki początkowe” oraz „warunki docelowe”, które można reprezentować przez wartości cech obiektów modelujących
 - rozwiązanie problemu ma postać ciągu działań przeprowadzających obiekty od warunków początkowych do warunków końcowych
- przykłady zastosowań:
 - niektóre gry, jak gry planszowe (np. szachy, reversi, go)
 - łamigłówki (np. 15-tka, kostka Rubika, wilk-koza-kapusta)
 - planowanie tras, np. nawigacja robotów
 - konstruowanie układów VLSI
 - synteza białek

Model problemu (1)

Założenia ogólne

- model problemu ma postać grafu $G = (V, E)$, którego wierzchołki reprezentują stany problemu, a krawędzie odpowiadają przejściom między stanami
- dowolny stan problemu $s \in S$ jest krotką (c_1, \dots, c_n) , której każdy element jest wartością pewnej cechy c_i kolejnego obiektu modelującego dla $i = 1, n$
- wyróżnia się stan początkowy s_0 odpowiadający „warunkom początkowym”
- wyróżnia się zbiór R stanów końcowych, z których każdy spełnia „warunki docelowe” problemu; zbiór R może być dany *explicite* lub przez podanie relacji przynależności p , takiej że dla każdego stanu $s \in S$ formuła $p(s)$ jest prawdziwa wtedy i tylko wtedy, gdy s jest stanem końcowym

Model problemu (2)

Założenia ogólne (cd.)

- dowolne dwa stany s oraz s' sąsiadują ze sobą (tj, są połączone krawędzią) wtedy i tylko wtedy, gdy istnieje działanie na jakimś obiekcie modelującym, w wyniku którego stan problemu s zmienia się na s'
- wszystkie działania na obiektach reprezentuje funkcja przejścia $f: S \rightarrow 2^S$
- na zbiorze krawędzi E definiuje się funkcję kosztu $v: E \rightarrow \mathbb{R}$, przez domysł wartość funkcji v wynosi 1
- funkcję kosztu określa się także na ścieżkach czyli ciągach krawędzi, wartość tej funkcji jest wypadkową (zwykle sumą) kosztów krawędzi składowych
- **rozwiązaniem problemu** jest dowolna ścieżka prowadząca od stanu początkowego s_0 do jednego ze stanów końcowych $s \in R$; niekiedy rozpatruje się także poszukiwanie rozwiązania o minimalnym koszcie

Model problemu (3)

Przykład (łamiągówka wilk-koza-kapusta, IX w.)

Znaleźć sposób przeprowienia wilka, kozy i kapusty z lewego brzegu rzeki na prawy przy następujących założeniach.

1. Przewoźnik ma łódź, w której jest miejsce dla co najwyżej jednego z wymienionych obiektów.
2. Przewoźnik nie może zostawić na tym samym brzegu, bez osobistego dozoru, ani wilka z kozą ani kozy z kapustą.

Problem ma następujący model:

- wyróżnia się 4 obiekty, tj. przewoźnika (F), wilka (W) kozę (K) oraz kapustę (C)
- każdy obiekt ma 1 cechę - położenie na lewym brzegu rzeki (wartość l) albo na brzegu prawym (wartość r)
- każdy stan problemu jest czwórką (F, W, K, C) , której elementami są wartości cech odpowiednich obiektów, przykładowo w stanie (l, r, l, r) przewoźnik i koza znajdują się na lewym brzegu rzeki, a wilk z kapustą na brzegu prawym

Model problemu (4)

Przykład (cd.)

- stan początkowy $s_0 = (l, l, l, l)$, zbiór stanów końcowych $R = \{(r, r, r, r)\}$
- w definicji funkcji f używa się funkcji pomocniczych, tj. op oraz us

$$op(l) \Rightarrow r$$

$$op(r) \Rightarrow l$$

$$us((F, W, K, C)) \Rightarrow true \mid (W = K \wedge F \neq W) \vee (K = C \wedge F \neq K)$$

$$us((F, W, K, C)) \Rightarrow false \mid (W \neq K \vee F = W) \wedge (K \neq C \vee F = K)$$

$$f((X, W, K, C)) \Rightarrow (Y, W, K, C) \mid op(X) = Y \wedge \sim us((Y, W, K, C))$$

$$f((X, X, K, C)) \Rightarrow (Y, Y, K, C) \mid op(X) = Y \wedge \sim us((Y, Y, K, C))$$

$$f((X, W, X, C)) \Rightarrow (Y, W, Y, C) \mid op(X) = Y \wedge \sim us((Y, W, Y, C))$$

$$f((X, W, K, X)) \Rightarrow (Y, W, K, Y) \mid op(X) = Y \wedge \sim us((Y, W, K, Y))$$

Należy zauważyć, że dla niektórych stanów wartością funkcji przejścia może być zbiór stanów o mocy > 1 , przykładowo $f((l, l, r, l)) = \{(r, l, r, l), (r, r, r, l), (r, l, r, r)\}$.

Rozwiązywanie problemu

- problem o rozpatrywanym modelu rozwiązuje się przez przeszukiwanie grafu stanów; do tego celu można stosować wiele metod i ujęć, zależnych m.in. od dodatkowych warunków formułowanych w ramach problemu, np. dotyczących kosztów poszukiwanego rozwiązania
- w celu uporządkowania dalszych rozważań przyjmuje się, że graf stanów G jest wstępnie przekształcany w drzewo $T = \text{Expand}(s_0, G)$, które podlega dalszemu przeszukiwaniu.

funkcja $\text{Expand}(v, G)$

wejście: wierzchołek $v \in V$, graf $G = (V, E)$

wyjście: drzewo $T = (V, E_T)$ o korzeniu v

1. Utworzyć (multi)zbiory $V' = \{v' \mid (v, v') \in E\}$ oraz $E' = \{(v, v') \mid v' \in V\}$
 2. Jeżeli $V' = E' = \emptyset$ to $T = (\{v\}, \emptyset)$, **zwróć**(T)
 3. Utworzyć zbiór drzew $\mathcal{T} = \{T' \mid T' = \text{Expand}(v', G) \wedge v' \in V'\}$
 4. Utworzyć drzewo $T = (V, E_T)$, gdzie $E_T = E' \cup \cup \{E'' \mid T'' = (V'', E'') \wedge T'' \in \mathcal{T}\}$
- jeżeli G jest multigrafem to liczba wierzchołków w drzewie T może przyrosnąć wykładniczo
 - jeżeli graf G zawiera cykle to drzewo T jest nieskończone
 - przyjmuje się, że drzewo T jest przeszukiwane za pomocą prostych strategii siłowych takich jak:
 - przeszukiwanie wszerek (ang. *Breadth-First Search*)
 - przeszukiwanie w głąb (ang. *Depth-First Search*)
 - przeszukiwanie w głąb z iteracyjnym pogłębianiem (ang. *Depth-First Iterative Deepening Search*)
 - przeszukiwanie dwukierunkowe (ang. *Bidirectional Search*)

Strategie przeszukiwania drzew (1)

Strategia BFS

funkcja $\text{BFS}(T, p)$

wejście: drzewo $T = (V, E)$, relacja p przynależności do zbioru wierzchołków poszukiwanych

wyjście: poszukiwany wierzchołek drzewa T

1. $L \leftarrow \{r_T\}$
2. Powtarzać kroki 3–5 dopóty, dopóki $L \neq \emptyset$
3. $L' \leftarrow \emptyset$
4. Dla każdego elementu $v \in L$: jeżeli $p(v)$ to **zwróć**(v), jeżeli nie to $L' \leftarrow L' \cup \{v' \mid (v, v') \in E\}$
5. $L \leftarrow L'$

Strategia DFS

funkcja $\text{DFS}(v, T, p)$

wejście: wierzchołek v drzewa T , drzewo T , relacja p przynależności do zbioru wierzchołków poszukiwanych

wyjście: poszukiwany wierzchołek drzewa T

1. Jeżeli $p(v)$ to **zwróć**(v)
2. Dla każdego, kolejnego, nieodwiedzonego potomka v' wierzchołka v w drzewie T : jeżeli $\text{DFS}(v', T, p) \neq \text{nil}$ to **zwróć**($\text{DFS}(v', T, p)$)
3. **Zwróć**(nil)

W celu przeszukania całego drzewa T należy wywołać funkcję $\text{DFS}(r, T, p)$, gdzie r jest korzeniem drzewa.

Strategie przeszukiwania drzew (2)

Strategia DFIDS

funkcja $DFIDS(r, T, p, h_0, \delta)$

wejscie: korzeń r drzewa T , drzewo T , relacja p , głębokość początkowa h_0 , funkcja przyrostu głębokości δ

wyjście: poszukiwany wierzchołek drzewa T

1. $h \leftarrow h_0$
2. Powtarzać kroki 3–4
3. Jeżeli $RDFS(r, T, p, h) \neq nil$ to **zwróć**($RDFS(r, T, p, h)$)
4. $h \leftarrow \delta(h)$

funkcja $RDFS(r, T, p, h)$

wejscie: wierzchołek v drzewa T , drzewo T , relacja p , głębokość h ,

wyjście: poszukiwany wierzchołek drzewa T

1. Jeżeli $h = 0$ to **zwróć**(nil)
2. Jeżeli $p(v)$ to **zwróć**(v)
3. Dla każdego, kolejnego, nieodwiedzzonego potomka v' wierzchołka v w drzewie T : jeżeli $RDFS(v', T, p, h-1) \neq nil$ to **zwróć**($RDFS(v', T, p, h-1)$)
4. **zwróć**(nil)

Strategie przeszukiwania drzew (3)

Złożoność obliczeniowa – przypomnienie podstaw

- złożoność obliczeniowa czasowa i pamięciowa
- rząd złożoności obliczeniowej $O(g(n))$
 - wszystkie rozpatrywane niżej funkcje działają ze zbioru \mathbb{N} na zbiór \mathbb{R}^+
 - funkcja f jest (co najwyżej) rzędu g , co zapisuje się jako $f(n) = O(g(n))$, wtedy i tylko wtedy gdy $(\exists c \in \mathbb{R})(\exists n_0 \in \mathbb{N})(\forall n \in \mathbb{N})(n \geq n_0 \rightarrow f(n) \leq c \cdot g(n))$
 - rząd iloczynu stałej i dowolnej funkcji jest równy rządowi tej funkcji
 - rząd sumy jest równy największemu z rządów wszystkich składników
- złożoność obliczeniowa strategii przeszukiwania drzew zależy w ogólności od następujących czynników:
 - wysokość drzewa (m)
 - współczynnik rozgałęzienia drzewa (b)
 - odległość od korzenia poszukiwanego wierzchołka, najbliższego korzeniowi (d)

Strategie przeszukiwania drzew (4)

Zestawienie

Kryterium	BFS	DFS	DFIDS	BS
Pełność	tak	nie	tak	tak*
Optymalność	tak	nie	tak*	tak*
Złoż. obl. czasowa	$O(b^{d+1})$	$O(b^m)$	$O(b^d)$	$O(b^{d/2})$
Złoż. obl. pamięciowa	$O(b^{d+1})$	$O(bm)$	$O(bd)$	$O(b^{d/2})$

* Pod pewnymi warunkami dodatkowymi.

Implementacja przykładu

Założenia

- przykład implementuje się w języku Prolog
- stan (F, W, K, C) jest reprezentowany przez listę $[F, W, K, C]$, przykładowo lista $[l, r, l, r]$ odpowiada stanowi (l, r, l, r)
- rozwiązanie reprezentuje lista postaci $[[l, l, l, l], \dots, [r, r, r, r]]$
- zapytanie o rozwiązanie: `?- solve()`.

Program

```
solve(B,E,P):-
    path(B,E,[B],P1),
    reverse(P1,P).

path(E,E,W,W).
path(B,E,Wp,W):-
    f(B,B1),
    not(member(B1,Wp)),
    path(B1,E,[B1|Wp],W).

f(S,S1):-
    possible(S,S1),
    not(unsafe(S1)).
```

```
possible([X,W,G,C],[Y,W,G,C]):- opposite(X,Y).
possible([X,X,G,C],[Y,Y,G,C]):- opposite(X,Y).
possible([X,W,X,C],[Y,W,Y,C]):- opposite(X,Y).
possible([X,W,G,X],[Y,W,G,Y]):- opposite(X,Y).

unsafe([F,X,X,_]):- opposite(F,X).
unsafe([F,_X,X]):- opposite(F,X).

opposite(l,r).
opposite(r,l)
```