teszt: célcsúcs, kész
Holott: a másik út olcsóbb!!!

Bebizonyítottuk, hogy az A-algoritmus nem garantálja az optimális megoldást!

b) A*-algoritmus

Kérdés: A-algoritmussal elõállítható-e az optimális megoldás? (az elõbb a heurisztika rontotta el)

Ha a heurisztikára adunk egy megkötést, akkor "megjavul". ALULRÓL kell becsülni.
Minden csúcsban $\mathbf{h(n) \leq h^*(n)}$

Ekkor garancia lehet az optimális megoldás megtalálására.
Ha a heurisztikánk alsó becslésû: A*-algoritmus. A csúcsokbeli heurisztikánk alulról becsüli a hátralévõ optimális út költségét.

Állítás: az A*-algoritmus optimális megoldás megtalálását garantálja

Bizonyítás: optimális megoldás: 0 heurisztika ∎ triviális
Így a szélességire is beláttuk az állításunkat.

3. lemma: az A*-algoritmus által kiterjesztésre kiválasztott bármely csúcs kiértékelõfüggvénye nem nagyobb (kisebb egyenlõ), mint az optimális út tényleges költsége: $\mathbf{f(n) \leq f^*(s)}$

Bizonyítás (3. lemma):

1. ha nincs megoldás (opt. költség = ∎), akkor $f(n) = f^*(n) =$ ∎ ,triviális eset
2. ha van megoldás, akkor van optimális megoldás is

$s = n_0$
n_1
n_2 } problémmánk egy aktuális megoldása (opt. megoldása)
...
$t = n_k$

Ennek az útnak lesz mindig eleme. Vegyük a legelsõ olyan elemet, ami a nyílt csúcsok között van: ni. Összes elõtte levõ: zárt.

$g(n_i) = g^*(n_i)$

n - legyen ez az a csúcs, amelyet kiválasztottunk kiterjesztésre

$f(n) \leq$ f(összes_többi_nyílt_csúcs) ,azaz $f(n) \leq f(n_i)$ (például)

$$f(n) \leq f(n_i) = g(n_i) + h(n_i) \leq g^*(n_i) + h^*(n_i) = f^*(n_i) = f^*(s)$$

$$\downarrow \quad \downarrow$$

$$g^*(n_i) \quad [h(n_i) \leq h^*(n_i)] \text{ (az alsó becslés miatt)}$$

Vagyis: $\mathbf{f(n) \leq f^*(s)}$. Ezt akartuk belátni.

Az A*-algoritmus az optimális megoldással terminál, ha van megoldás.

c) Monoton A-algoritmus

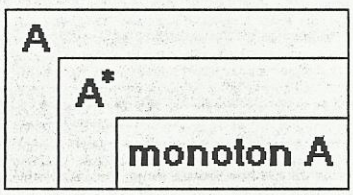
Az A-algoritmusnál a gond: zárt csúcsok. Ugyanazt a csúcsot esetleg többször is ki kell terjeszteni (zárt csúcsok problémája).
Optimális kereső: speciális A-algoritmus, ahol a heurisztika 0. De van-e nem azonos nulla heurisztikával rendelkező algoritmus, mely ilyen tulajdonsággal bír? (vagyis ahol nem merül fel a zárt csúcsok problémája)

Monoton heurisztika:

$h(n) \leq h(m) + k(n,m)$ $\forall(n,m)$ □ E esetben

Azaz: szülő heurisztikája ≤ gyermek heurisztikája + szülőből gyermekbe jutás költsége.

Az ilyen tulajdonságú A-algoritmus: monoton A-algoritmus.



Monoton heurisztikával rendelkezőA-alg. A*-alg. is egyben.

Tétel: Ha *h* monoton heurisztika, akkor $h(n) \leq h^*(n)$

Bizonyítás:

a) ha *n*-ből nem érhető el egyetlen terminális csúcs sem
$h(n) = h^*(n) =$ □
b) ha *n*-ből elérhető valamelyik terminális csúcs
$n_0, n_1, n_2, \ldots, n_l = t$

$h(n_0) \leq h(n_1) + k(n_0,n_1)$
$h(n_1) \leq h(n_2) + k(n_1,n_2)$

...

$h(n_{l-1}) \leq h(n_l) + k(n_{l-1},n_l)$

$$\sum_{i=0}^{l-1} h(n_i) \leq \sum_{j=1}^{l} h(n_j) + \sum_{i=0}^{l-1} k(n_i,n_{i+1})$$

$\downarrow$

$h^*(n)$ (a hátralévõ optimális út költsége)

$h(n_0) - h(n_1) \leq h^*(n)$ (a többi páronként 0)

$h(n) \leq h^*(n)$ (mivel n_l terminális csúcs, így heurisztikája 0)

A tételt ezzel bebizonyítottuk!

Monoton heurisztika esetén bebizonyítható, hogy tetszõleges kiterjesztésre kiválasztott nyílt csúcs olyan, hogy az adott csúcsban a kiértékelõ függvényben olyan értékkel számolunk, ami már optimális.

Tétel: a monoton heurisztikájú A-algoritmus által kiterjesztésre kiválasztott minden nyílt csúcsra igaz, hogy **$g(n) = g^*(n)$**

Vagyis optimális megoldást állít elõ, ill. a zárt csúcsok problémája itt nincs jelen! (hasonlóan mint az optimális keresõknél)

Bizonyítás:

$s = n_0,n_1,...,n_l = n$ optimális út

n □ Nyílt és kiterjesztésre n-et választjuk

Indukciós feltevés: t.f.h. még nem találtuk meg a hozzávezetõ opt. utat: $g(n) > g^*(n)$

n_i az optimális utunk elsõ nyílt csúcsok közötti eleme

$f(n_i) = g(n_i) + h(n_i) = g^*(n_i) + h(n_i) \leq$

$\leq g^*(n_i) + h(n_{i+1}) + k(n_i,n_{i+1}) = g^*(n_{i+1}) + h(n_{i+1}) \leq$

$\leq g^*(n_{i+1}) + h(n_{i+2}) + k(n_{i+1},n_{i+2}) = g^*(n_{i+2}) + h(n_{i+2}) \leq$

$\leq ... \leq g^*(n_l) + h(n_l)$

Hol is tartunk?

$f(n_i) \leq g^*(n) + h(n) < g(n) + h(n) = f(n)$ ELLENTMONDÁS!

Tehát a zárt csúcsok problémája a monoton A-algoritmusnál nem lép fel!

(Ha egy csúcs zárt csúcs lett, akkor azzal már nem kell foglalkozni).

$f(n) = \alpha \cdot g(n) + (1-\alpha) \cdot h(n)$ α - súly

$\alpha = 0$ best-first
$\alpha = 1$ optimális keresõ

A keresés során változtathatjuk is ezeket a súlyokat, s áttérhetünk az egyikről a másikra. Ilyen pl. a B-algoritmus!

Legyen P és Q két A^*-algoritmus. Hasonlítsuk össze a kettõt a heurisztika alapján.

P jobban informált mint Q, ha

$$h_Q(n) < h_P(n) \leq h^*(n)$$

Egy jobban informált nem használ nagyobb gráfot mint a kevésbé informált.

Tétel:

Legyen P és Q két A^*-algoritmus.
Ha P jobban informált, mint Q $\Rightarrow$ Q minden olyan csúcsot kiterjeszt, amit P is.

Bizonyítás:

T.f.h. P kiterjeszt egy n $\in$ Nyílt csúcsot. Ekkor (lemma 3 segítségével): $f_P(n) \leq f^*(s)$

$$n' \in s \Rightarrow n : f_P(n') \leq f^*(s)$$

mivel P jobban informált, mint Q $\Rightarrow$ $h_Q(n') < h_P(n')$

$$f_Q(n') = g(n') + h_Q(n') < g(n') + h_P(n') = f_P(n') \leq f^*(s)$$

2.2.2.2.5. B-algoritmus

Indulunk az A-algoritmussal, de regisztráljuk az addigi legnagyobb kiértékelõ függvényû csúcsot.
Addigi $L_n := f(s)$

Ha a kiértékelõfüggvény az addigi L_n-nál kisebb, akkor opt. keresõvel folytatjuk, egyébként meg A^*-gal.

Ha $h(n) \leq h^*(n)$,akkor ugyanazt állítja elõ, mint az A^*-algoritmus, csak a kiterjesztések száma lecsökken.

TOC

Vissza a lap tetejére

Elõadó: Dr. Várterész Magdolna

2.3. Probléma-redukciós reprezentáció

2.3.1. Probléma-redukció

Nagyon gyakori a feladat egyszerûsítése, részproblémákra bontása.

p: a megoldandó probléma. Ezt valahogy le kell írni. Ez akár az állapottérrel is leírható, de most nem az a fontos. Ehhez a problémához hasonló problémákat keresünk.

P: probléma-halmaz (ebbe gyûjtjük össze õket)

$p \in P$ ennek a mi problémánk is eleme lesz

$e \in E \subseteq P$ egy ismert probléma is legyen benne! Ez az egyszerû probléma.

e: 1. meg tudjuk oldani, ismerjük a megoldását
2. meg tudjuk mondani, hogy nem megoldható

Az induló problémát addig kellene gyúrni, alakítani, míg csupa egyszerû probléma nem állna elõ.

$r \in R$ redukciós operátor

$dom(r) \subseteq P$ (egy problémaosztálybeli problémát tud egyszerûsíteni) van elõfeltétele is!

$r: P \to 2^P$ P részhalmazainak a halmazába képez tehát

$r(p) = \{p_1,...,p_n\}$ részproblémákra bont, egyszerû részekre bont ($n \geq 1$)

<P,p,E,R> ezzel a négyessel tudjuk megadni a probléma probléma-redukciós reprezentációját

Def.:

$P_1 = \{p_1,...,p_{i-1},p_i,p_{i+1},...,p_n\}$
$P_2 = \{p_1,...,p_{i-1},q_1,q_2,...,q_m,p_{i+1},...,p_n\}$

P_1-bõl egy lépésben (közvetlenül) redukálható P_2, ha van:

$r \in R: p_i \in dom(r)$ és $r(p_i) = \{q_1,...,q_m\}$ $\quad P_1 \overset{r}{\Rightarrow} P_2$

Def:

A P' problémahalmazból a P'' redukálható, ha van:

- $P' = P_0,P_1,...,P_k = P''$ problémahalmaz-sorozat és
- $P_i \Rightarrow_{r_i} P_{i+1}$ $(i = 0,1,...,k-1)$

Mikor megoldható? Ha a kezdõproblémából (mint 1 elemû problémahalmazból) csupa megoldható egyszerû problémát tartalmazó halmazt redukálhatunk.

Megoldás: az azt megvalósító r_i redukciós operátorsorozat.

Megint egy keresési feladattal állunk szemben: egy operátorsorozatot keresünk. Itt is lehet költség, redukciós operátor alkalmazási költsége.

Megoldás költsége: a megoldásban szereplõ operátorsorozat alkalmazási költségeinek összege.

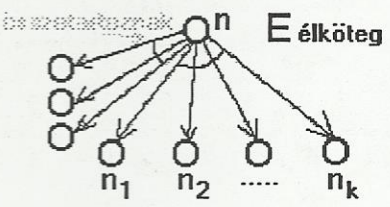
2.3.2. Szemléltetése

Problémánk probléma-redukciós reprezentációját a <P,p,E,R> négyessel tudjuk megadni.

$q \in P$	$n \in N$	egy problémát egy csúccsal jelölünk
$p \in P$	$s \in N$	start-csúcs
$e \in E \subseteq P$	$t \in T \subseteq N$	terminális csúcsok

$r \in R$

$r(q) = \{q_1,...,q_k\}$ $k \geq 1$



$r'(q) = \{z_1,z_2,...,z_l\}$

1 csúcsból annyi élköteg indul, ahány redukciós operátort alkalmazunk, s egy élköteg annyi élbõl áll, ahány részprobléma állt elõ.

ÉS, AND élköteg	minden részproblémát egy élkötegben kell megoldani
VAGY, OR élköteg	vagy az egyik redukció segítségével bontom részproblémákra, vagy a másikkal
<G,s,T>	és/vagy problémaredukciós (reprezentációs) gráf, ahol: G: gráf

s: `start` csúcs

T: terminális csúcs(ok)

hiperút: olyan összefüggõ része az és/vagy gráfnak, melyben minden csúcsból legfeljebb 1 ÉS-élköteg indul ki.
(Ha 1 ÉS-élköteget 1-nek veszünk, akkor az út fogalmához jutunk).

A hiperút a `start` csúcsból indul, s a levelek: csupa megoldható terminális levelek.
A megoldást egy olyan hiperút szemlélteti, mely a `start` csúcsból indul, s levelei csupa megoldható terminális csúcsok. A megoldáskeresés során egy általánosabb gráfban egy részgráfot keresünk.

k(n, {n1,...,nk}) redukciós operátor alkalmazási költsége

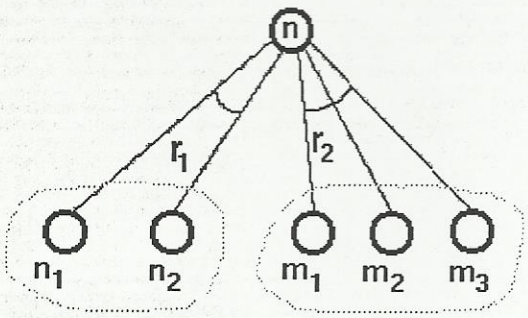
$\{n\} \xrightarrow[\text{hiperút vezet}]{} K$,ahol K - csúcshalmaz

$$g(n,K) = \begin{cases} 0, \text{ ha } n \in K \\ k(n,\{n_1,\ldots n_k\}) + \sum_{i=1}^{k} g(n_i,K) \end{cases}$$

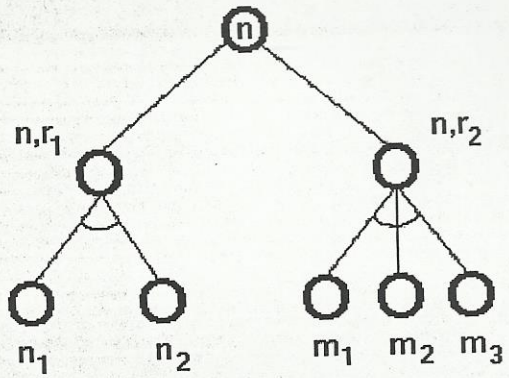
$$k(n,\{\}) = \begin{cases} c(n), \text{ ha } n \in E \text{ megoldható} \\ \infty, \text{ha } n \text{ nem megoldható} \end{cases}$$

Tiszta ÉS-VAGY gráf: olyan gráf, amelyben minden csúcsból csak ÉS vagy csak VAGY élkötegek indulnak ki.

Például adott a következõ ÉS-VAGY gráf:



Elérhetõ, hogy csak ÉS vagy csak VAGY élkötegeket tartalmazzon:



Fiktív csúcsokat kell beiktatni. Annyi kell, ahény VAGY élköteg indul ki. Mellette feltüntetve: melyik redukciós operátorral kaptuk.

Megoldáskeresõ módszerek:

I. nem-módosítható megoldáskeresõk (nem foglalkozunk velük)
II. módosítható keresõk
    a. visszalépéses (backtrack)
    b. keresõgráffal

TOC

Vissza a lap tetejére

Mesterséges intelligencia 1

Elõadó: Dr. Várterész Magdolna

2.4. Probléma-redukcióval leírt feladatok megoldását kereső módszerek

Osztályozásuk:

I. nem-módosítható megoldáskeresõk (nem foglalkozunk velük)
II. módosítható keresõk
   a. visszalépéses (backtrack)
   b. keresõgráffal

2.4.1. Backtrack (visszalépéses)

állapottér-reprezentáció esetén:	adatbázis:	`start` csúcsból induló valamilyen út a reprezentációs gráfban. Aktuális út. Aktuális állapot: az út végi csúcs. Ennek a vizsgálata döntötte el, hogy hogyan tovább.
ÉS-VAGY gráf esetén:	adatbázis:	egy hiperút, a `start` csúcsból indul. Ebbõl szeretnénk megoldást készíteni. Aktuális hiperút.
Levelei:		az eredeti problémának valamilyen szintû részproblémái. Ha a levelek mind egyszerû megoldható problémák: megoldás:
Csúcsok:		probléma, részprobléma, amit éppen reprezentál

Miket tartunk nyilván a csúcsokban?

1. mutató gyermekére, az 1. gyermekére
   köv. mutató a testvérre
   szülõre mutató mutató (a visszalépéshez)
2. a még nem alkalmazott (de alkalmazható) redukciós operátorokra valamilyen információ
3. címke (az adott probléma megoldását tartalmazza-e a hiperút vagy sem?)

Amikor egy csúcsot elõállítunk megnézzük, hogy az egyszerû problémák között szerepelt-e már? Háromféle címkét fogunk alkalmazni:

1. M (megoldható *egyszerû* probléma)
2. N (nem megoldható *egyszerû* probléma)
3. F (a többi problémának ez a címkéje, azt jelzi, hogy még folyik a munka)

Mûveletek: 1. redukciós operátorok