Előadó: Dr. Várterész Magdolna

Módosítható megoldáskeresők osztályozozása:
1. visszalépéses (backtrack)
2. keresőgráffal

2.2.2.2. Keresőgráffal megoldást keresők

Adatbázis: keresőgráf, egy fa. A reprezentációs gráfnak a bejárt részét feszítő fa. Egy csúcspont lehet zárt (az út közbensõ csúcsa, rajtuk már továbbhaladtunk), vagy lehet nyílt (az utak végén álló levelek, itt folytathatjuk majd a keresést).
Minden csúcsnak pontosan egy szülője lesz nyilvántartva a keresőgráfban (mivel fa). Az éleket visszamutatók segítségével realizáljuk. Így tudunk majd rekonstruálni.
(A visszalépéses megoldáskereső csak egy utat tartott számon. Itt több út is számon van tartva, rajtunk múlik, hogy hol folytatjuk).

Mûvelet: kiterjesztés. Levélelemből kivitelezi a továbblépést, Így tudunk majd élnyire továbbmenni. Az összes lehetséges irányba továbbmegyünk majd. (Egy új, nem kiterjesztett csúcsra alkalmazzuk az összes alkalmazható operátort).

Vezérlõ: melyik nyílt csúcs legyen a következõ lépésben kiterjesztve. A kiválasztott nyílt csúcsot tesztelni tudja (célcsúcs, nem célcsúcs), s van egy út visszafelé is. A nyílt csúcsok közül kell választania a megoldáskeresõnek.
Nincs megoldás: egyetlenegy nyílt csúcs sincs, amelyen a keresést folytathatnánk.

Szisztematikus keresőgráffal keresők

Szisztematikus keresőgráffal keresők:

- szélességi kereső
- mélységi kereső
- optimális kereső

2.2.2.2.1. Szélességi és mélységi kereső

A reprezentációs gráf csúcsait oly módon választjuk, hogy az 1. szinten, 2. szinten, stb. lévő csúcsokat vesszük.

mélységi szám:	g(s)=0	a `start` csúcsé 0. 0 szintre van a `start`-tól
	g(m)=g(n)+1	a szülő mélységi száma + 1, ahol (n,m) $\in$ E (vagyis él)

Egy csúcs előállításakor mindig kiszámítjuk a mélységi számát.

Kiterjesztésre melyik csúcsot választjuk ki?

- szélességi: mindig a legkisebb mélységi számú nyílt csúcs kerül kiterjesztésre
- mélységi: mindig a legnagyobb mélységi számú nyílt csúcs kerül kiterjesztésre

szélességi+mélységi: egy adott csúcshoz vezetõ melyik út van éppen nyilvántartva? Mindegy, ezért a legelsõ számontartott utat fogjuk nyilvántartani.

1. adatbázis inicializálása: S
   rögzítsük a mélységi számot: 0
   visszamutató: 0
   } $S_0(0)$ nyílt (nyílttá tesszük)

   (azt is tudni kell, hogy az induló keresõgráf egyetlen csúcsa nyílt-e vagy zárt)

2. Van-e nyílt csúcs?
   nincs: be kell fejezni, nem tudjuk folytatni
   van: ki kell választani a nyílt csúcsok közül egyet (az alapján, hogy mélységi v. szélességi a keresés)
   A kiterjesztésre kiválasztott csúcsot teszteljük.
   Célcsúcs?
   nem: kiterjesztjük, õt zárttá, gyermekeit nyílttá tesszük. Gyermekek visszamutatnak a szülõre, mélységi szám beírása. Vissza a 2. pontra.
   igen: sikeres vég, mutatók alapján van egy út vissza

Szélességi, mélységi keresésnél nem cél az optimális megoldás kiválasztása: ha több út van az adott csúcsba, akkor tetszõlegesen tartjuk nyilván az egyiket (legegyszerûbb, ha maradunk az elsõnél). Mikor tesztelünk, a tesztelést elõrehozhatjuk, mert nem cél az optimális megoldás.

Szélességi: tetszõleges reprezentációs gráfban ha van megoldás, akkor a legkevesebb operátor alkalmazásával elõállítható valamelyik terminális. Mindig a legkisebb mélységi számú csúcsot teszteljük, terjesztjük ki, ezért a legkisebb hosszúságú megoldást állítjuk elõ.
Ha nincs a problémának megoldása, akkor azt a nyílt csúcsok elfogyásával felismeri.

Mélységi: tetszõleges, véges reprezentációs gráfban ha van megoldás, akkor a mélységi keresõ megoldás a terminál. Ha nincs megoldás: felismeri.

(Mint látható, nem kell megkövetelni a körmentességet - ellentétben a backtrack-kel).

Implementálás:

Szélességi esetben a nyílt csúcsokat egy sorban tartjuk nyilván, amelybõl pont mélységi szám szerint növekvõleg fognak kijönni.
Mélységi esetben: veremben való kezelés.


Ezt onnan lehet könnyen megjegyezni, hogy "a sor széles, a verem pedig mély" :-)

Ha a reprezentációs gráf fagráf: kiterjesztéskor nem kell figyelni, hogy a gyermekek szerepeltek-e

már vagy sem, mert nem szerepelhettek! (ui. fagráf esetén egy elemnek csak 1 szülõje lehet, így egy elemet csak 1 irnyból, a szülõ felõl érhetjük el, s nem kell attól félnünk, hogy már korábban egy másik irányból kiterjesztették).

2.2.2.2.2. Optimális kereső

(a szélességi keresés is egyfajta opimális megoldást adott. Költség: út hossza).

$r(n,m) \geq \delta > 0$,ahol $(n,m) \in E$

Minden csúcsnál számon tartjuk az odavezetõ út költségét.

$g(s)=0$

$g(m)=g(n)+r(n,m)$,ahol $(n,m) \in E$

$g(n)$: szülõ költsége, $g(n) \geq 0$

$r(n,m)$: a szülõ és gyermek közti él költsége

$g^*(n)$ jelöljük így a `start`-ból `n`-be jutás optimális költségét

$g(n) \geq g^*(n)$, $n \in N$

A nyílt csúcsok közül azt fogjuk kiterjesztésre kiválasztani, amelyik eddig a legolcsóbb volt (ott reméljük majd a legolcsóbb megoldást). A legkisebb költségû nyílt csúcsot választjuk ki kiterjesztésre.

Ha az új út kisebb költségûnek bizonyul mint a régi, akkor...

$n \in \Gamma m$

m szerepelt már akeresõgráfban:

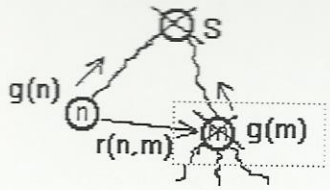
Ha az új út költsége < régi, akkor
átállítjuk a mutatót ill. a költséget (zölddel jelölve)

m lehetett nyílt: ekkor átállítjuk a mutatót ill. a költséget (lásd fentebb)

m lehetett zárt: már vannak részgráfjai! Az *m*-bõl induló részgráf összes csúcsába is kisebb költséggel juthatnánk el. Gond!

DE! Bebizonyítható, hogy ha *m* zárt, akkor nem található hozzá vezetõ kisebb költségû út!

Mivel *m* zárt (most ezt feltételezzük), akkor
az *m*-et hamarabb kellett kiterjeszteni, mint *n*-et, azaz
$g(m) \leq g(n)$



$g(m) \leq g(n)$

$g(m) < g(n) + r(n,m)$

Így tehát az elõbbi probléma nem fog minket érinteni.

Mit garantál az optimális keresõ? Tetszõleges reprezentációs gráfban ha van megoldás, akkor az optimális megoldással terminál. Ha nincs megoldás: jelzi.

Tesztelés: mikor teszteljük a csúcsokat, hogy terminális-e? Ez idõnként elõrehozható, például kiterjesztéskor, amikor elõállítunk egy csúcsot akkor rögtön tesztelhetjük. Ez viszont csak csak szélességi és mélységi keresésnél mûködik ilyeténképpen, optimálisnál nem (ott csak kiválasztás után).

Heurisztikus keresõgráf

Heurisztika: a szisztematikus keresõgráffal keresõk a gráf jóval nagyobb részét használják (egy egész fát, míg a korábbiak jóval kisebbet).
A heurisztika trükk, egyszerûsítés, mely a nagyméretû reprezentációs gráfban a keresést tudja korlátozni. Csak a gráf azon részét kelljen elõállítani, amelyben a megoldást reméljük a tudásunk alapján (míg a szélességi bizonyos mélységig az egészet feltárja).

Példa:
szélességi keresés esetén:

1 csúcsnak van d gyermeke, s l hosszú a legrövidebb megoldás. Hány csúcsú keresõgráfot állítana ez elõ?

$$1 + d + d^2 + \ldots + d^l = \frac{d^{l+1} - 1}{d - 1}$$

Vagyis a megoldás mérete a megoldás hosszának exponenciális mérete.

Heurisztikával: a nyílt csúcsok közül ha mindig azt terjesztenénk ki, ahol a megoldást reméljük:

$$1 + d + d + \ldots + d = 1 + l \cdot d$$

Ez az ún. perfekt heurisztika. Persze perfekt heurisztikánk NINCS (hiszen ha ismernénk a megoldást, akkor nem kellene keresni)! De ez lenne az ideális. A lényeg: csökkentsük a keresõgráf méretét!

2.2.2.2.3. Best-first keresõ

(A legjobb irányban haladó keresés).

$h(n) \sim n$ becslõ függvényérték, ahol n-bõl meg tudjuk becsülni a célbajutás optimális költségét

$h(n) \geq 0$

$h(t) = 0$, ha $t \in T$ (vagyis ha terminális csúcsban vagyunk)

$h(n) = \infty$, ha egy csúcsból (n-bõl) nem érhetõ el egyetlen terminális sem, akkor valamilyen nagyon nagy számmal jelöljük

A keresõgráfban minden n csúcshoz elõállítjuk h(n)-t.

Kiterjesztésre kiválasztás: legkisebb heurisztikájú nyílt csúcs
(CSAK ezt tárolja: abból a csúcsból kb. milyen hosszú lesz a hátralévõ út. A múltról NEM tárol semmit.
Míg az optimális keresõ: ami ADDIG a legolcsóbb volt, a múltat figyelte.
Ez: a jövõt nézi, milyen hosszú lesz még.)

Általában az elsõként nyilvántartott utat szoktuk nyilvántartani. Vagyis: ha kiterjesztés során elõállítunk egy csúcsot, s szerepelt a keresõgráfban: nem foglalkozunk vele; nem szerepelt: nyílt csúcs lesz, visszamutatóval.

1. Adatbázis inicializálása: `start` csúcs, heurisztika megállapítása, nyílttá tesszük
2. Van-e nyílt csúcs?
   - nincs: nincs megoldás
   - van: kiválasztjuk a legkisebb heurisztikájút, s teszteljük. Célcsúcs?
     - igen: rendben, visszamutatók alapján megvan a megoldás
     - nem: kiterjesztjük. A keresõgráfban még nem szereplõ gyermekeit felvesszük a nyílt csúcsok közé, szülõre visszamutatnak, becslés esetükben. A szülõt zárttá tesszük.

Nincs minõsített megoldáskeresés. A terminálási feltételek elõrehozhatók; például kiterjesztéskor gyermekek tesztelése.

Mit garantál? tetszõleges véges gráf esetén ha a reprezentációs gráf tartalmaz megoldást, akkor véges sok lépésen belül megoldást állít elõ. Ha nincs megoldás: jelzi.

Probléma: ha a heurisztika nem elég jó, akkor a keresés biztonságát, a garanciát elveszíthetjük.

TOC

Vissza a lap tetejére

Mesterséges intelligencia 1

Elõadó: Dr. Várterész Magdolna

2.2.2.2.4. A-algoritmusok

a) "alap" A-algoritmus

optimális keresés esetén: $g(s) = 0$

$g(m) = g(n) + k(n,m)$

,ahol $(n,m) \in E$, ill. $k(n,m) \geq \delta > 0$

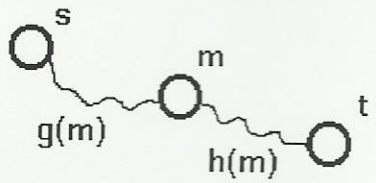
best-first: $h(m) \geq 0$

$h(t) = 0$, ha $t \in T$

$h(m) = \Gamma$, ha *m*-bõl nem érhetõ el egyetlen terminális sem

Ötlet: ötvözzük a kettõt!

$f(m) = g(m) + h(m)$ ez lesz a kiértékelõ függvény, azaz a `start`-ból *m*-en keresztül valamilyen terminálisba való jutás becsült költsége



Nyilvántartja az *m*-ig vezetõ út költségét, ill. az *m*-bõl a *t*-be vezetõ út becsült költségét.

$g^*(m)$ a `start`-ból *m*-be jutás optimális költsége

$g(m) \geq g^*(m)$

$h^*(m)$ *m*-bõl milyen költséggel juthatunk legolcsóbban célba?

célbajutás optimális költsége

$h(m) \sim h^*(m)$,vagyis a becslés közelíti ezt az opt. értéket

$f^*(m)$ `start`-ból *m*-en keresztül a célbajutás optimális költsége

$f^*(m) = g^*(m) + h^*(m)$,elméleti érték, kiszámítani nem tudjuk

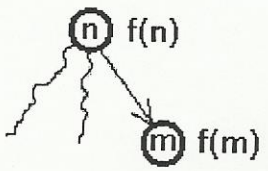
$f(m) \sim f^*(m)$, $f^*(m)$ -et szeretnénk közelíteni

$f^*(s)$ optimális megoldás költsége

`start`-on keresztül a legolcsóbb célbajutás költsége

$f^*(s) = h^*(s)$ (hiszen $g(s) = g^*(s) = 0$)

Mûködése: ezt az értéket fogjuk származtatni

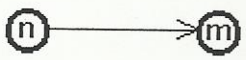


$f(n) = g(n) + h(n)$
$g(n) = f(n) - h(n)$

$f(m) = g(m) + h(m) = g(n) + k(n,m) + h(m) =$
$= f(n) - h(n) + k(n,m) + h(m)$

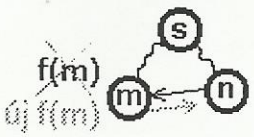
Kiterjesztésre mindig a legkisebb *f(m)* -ût kell kiválasztani!
Hátra és elõre is tekintünk! Összességében a legolcsóbbat terjesztjük ki.

Kiterjesztésre kiválasztani: a legkisebb kiértékelõfüggvény-értékû nyílt csúcsot



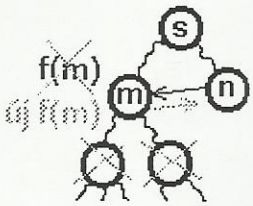
1. *m* még nem szerepelt a keresõgráfban
   m-et felfûzzük a keresõgráfban nyílt csúcsként (*n*-re visszamutató pointer)
   $f(m) = f(n) - h(n) + k(n,m) + h(m)$
2. *m* szerepelt már a keresõgráfban

a)



m-et feltártuk már, de még nem léptünk tovább (azaz nyílt csúcsként szerepel)
Ha találunk hozzá egy kisebb költségû utat, akkor tároljuk azt le.
Visszamutató nyíl *n*-re.

b)



Zárt csúcsok problémája: *m*-bõl egyszer már továbbmentünk. Amennyivel csökkent *f(m)* értéke, annyival kell csökkenteni az egész részgráf csúcsainak a költségét.
Fel kell újítani!

Zárt csúcsok problémája:

1. járjuk be a rész-keresõgráfot, s frissítsünk.
   Gond: visszamutatókat használtunk, nem elõremutatókat. (Reprezentációs változtatás).
2. bízzuk a dolgot az A-algoritmusra. Az *m* zárt csúcsból csináljunk nyílt csúcsot, s ekkor ez olyan, mintha a részgráfot még NEM jártuk volna be. Az újrabejáráskor lesz korrekció!
3. megelõzés. Az optimális keresõ olyan volt, hogy mikorra

egy csúcs zárt lett, akkor már nem található hozzá vezetõ kisebb költségû út. Vagyis az opt. keresõnél nincs ilyen gond.

Az optimális keresés egy speciális A-algoritmus, ahol **h(n) = 0**, vagyis nem számítunk heurisztikát.
Kérdés: heurisztikával rendelkezõ A-algoritmus esetében mûködik-e?
A válasz: igen, a heurisztikának kell olyannak lennie.

A vezérlõ mûködése:

1. az adatbázis inicializálása: `start` csúcs, nyílttá tesszük
   kiértékelõ-függvény megállapítása
2. van-e nyílt csúcs?
   - nincs: sikertelen a keresés
   - van: kiválasztjuk a legkisebb kiértékelõfüggvényû nyílt csúcsot.
     Teszteljük: célcsúcs?
     - igen: jó, vége
     - nem: kiterjesztjük. Ha egy gyermeke még nem szerepelt a keresõgráfban: felvesszük, pointer vissza, kiértékelõ-függvény kiszámítása a szülõ segítségével.
       Ha szerepelt már: megnézzük, hogy a hozzá vezetõ új út, vagy a már nyilvántartott (régi) út költsége-e kisebb? Ha az új út költsége (amit az új szülõ alapján számítunk) kisebb: akkor van teendõ, különben nincs.
       Ha az új út költsége kisebb: ha ez nyílt, akkor a visszamutatót átállítjuk az új szülõre, s regisztráljuk az új kiértékelõ-függvényt. Ha zárt, akkor visszaminõsítjük nyílt csúccsá, átírjuk a visszamutatót és a kiértékelõ-függvényt.
       Az épp kiterjesztett csúcsból zárt csúcs lesz.

Elvárások: tetszõleges reprezentációs gráf esetén ha van megoldás: megoldása a terminál véges sok lépésben.
Felismeri véges esetben, ha nincs megoldás.

Az optimális keresõ is ilyesmi tulajdonságokkal bír, az egy speciális A-algoritmus, a szélességi pedig egy speciális optimális keresés.

1. lemma: az A-algoritmus mûködése során bármely nyílt csúcs véges sok lépés megtétele után kiterjesztésre kerül, hacsak közben az algoritmus terminális csúcs megtalálásával nem terminál.

2. lemma: az A-alg. terminálása elõtt mindig van az optimális útnak eleme a nyílt csúcsok között.

Állítás: az A-algoritmus megoldással rendelkezõ reprezentációs gráf esetén véges sok lépésben terminálissal terminál.

Bizonyítás (1. lemma):

legyen r a keresõgráf egy csúcsa

$f(r)$ kiértékelõ-függvény (az r csúcsig vezetõ út költsége + heurisztika)

$f(r) = g(r) + h(r) \geq g(r) \geq g^*(r)$

A nyilvántartott, r-ig vezetõ út vagy az optimális, vagy sem, azaz $\geq$ mint az odavezetõ opt. út.

$d^*(r)$	r-ig vezetõ optimélis út éleinek a száma	
$g^*(r)$	optimális út költsége	
	$g^*(r) \geq d^*(r) \cdot \delta$	
összesen $d^*(r)$ él van	$f(r) \geq d^*(r) \cdot \delta$	$\forall r$ esetén

Vegyünk egy tetszõleges nyílt csúcsot: m

$f(m)$ $\quad f(m) \geq d^*(m) \cdot \delta$

Határozzunk meg ebben a keresõgráfban egy olyan mélységet, amely az ismert adatokból meghatározható!

$\left[\frac{f(m)}{\delta}\right] + 1 = d$ $\quad$ m-be vezetõ legrövidebb mélység alatti mélység

$\left[\frac{f(m)}{\delta}\right] + 1 \geq \left[\frac{f(m)}{\delta}\right] \geq d^*(m)$ $\quad$ m csúcs felette van a d szintnek

legyen r a d-szintnél mélyebben lévõ csúcs:

$d^*(r) > d$

$f(r) \geq d^*(r) \cdot \delta$ $\quad$)

$\quad$ } $\quad d^*(r) \cdot \delta > d \cdot \delta$

$d^*(r) > d$ $\quad$)

$d \cdot \delta > \frac{f(m)}{\delta} \cdot \delta = f(m)$ $\quad$ tehát $f(r) > f(m)$

d szint felett véges sok csúcsunk van. Véges sok $f(m)$ -nél $\leq$ nyílt csúcsunk van.

Nem csak kikerülhetnek nyílt csúcsok, de be is kerülhetnek, DE csak véges sokszor. (Az algoritmus mûködése során egy csúcs többször is bekerülhet a nyílt csúcsok közé; többször is ki lehet azt választani kiterjesztésre, de csak véges sokszor ⊡ biztos, hogy minden csúcs véges sok lépés után kikerül a nyílt csúcsok közül.

Bizonyítás (2. lemma):

A bizonyítás indukcióval történik.
Elsõ kiterjesztésre kiválasztás elõtt van az optimális útnak eleme (a `start` csúcs). Vegyünk az optimális utak közül egyet:

$s = n_1$	`start` csúcs
n_2	
...	
$t = n_k$	utolsó csúcs, terminális

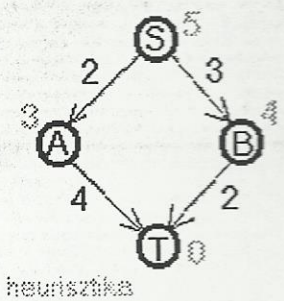
Indukciós feltétel: a *k.* kiterjesztés elõtt tegyük fel, hogy ennek az optimális megoldásnak az *i.* csúcsa eleme a nyílt csúcsok halmazának ($n_i \in$ Nyílt).
A *(k+1).* kiterjesztés elõtt mi a helyzet?

A *k.* lépésben kiterjesztéskor pont az n_i -t terjesztettük ki. Elõállítottuk az összes gyermekét, köztük n_{i+1} -et.
Ha nem szerepelt: felvesszük nyílt csúcsként.
Ha szerepelt (zárt csúcsként): akkor más úton már eljutottunk hozzá. Utódjai közül pár benne van a keresõgráfban pár nyílt csúcsként szerepel.

Ha nem az n_i -t választjuk kiterjesztésre: õ ott marad nyíltként.

A kettõ következménye:

Az A-algoritmus nem garantálja, hogy az optimális megoldással terminál. Ha van megoldás, akkkor megoldással terminál.
Bizonyítása: konstruktív módon:



Heurisztika:		Nyílt:		Zárt:
	S: 5			
	A: 3			
	B: 4	$\underline{S_0}(0+5)$		
	T: 0	$\underline{A_S}(2+3)$	$B_S(3+4)$	$S_0(0+5)$
		$\underline{T_A}(6+0)$	$B_S(3+4)$	$A_S(2+3)$