3. Gráfkereső stratégia

A gráfkereső rendszer olyan KR, amelynek globális munkaterülete a startcsúcsból kiinduló már feltárt utakat (részgráfot) tárolja

- kiinduló értéke: a startcsúcs;
- terminálási feltétel: megjelenik egy célcsúcs vagy megakad az algoritmus;
- keresés egy szabálya: egy csúcs rákövetkezőit állítja elő (kiterjeszti);
- vezérlés stratégiája: a legkedvezőbb csúcs kiterjesztésére törekszik;

3.1. A gráfkereső alapalgoritmus, *Jelölések:*

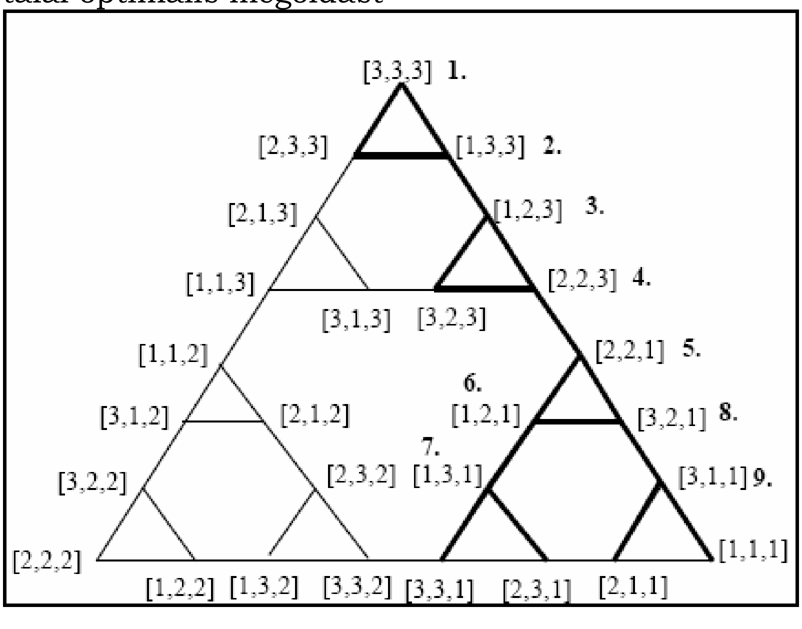
- G – kereső gráf;
- *NYÍLT* - nyílt csúcsok halmaza;
- Γ - kiterjesztés;
- kiterjesztett csúcsok - zárt csúcsok halmaza;

Az absztrakt keresési tér a továbbiakban is egy nem feltétlenül véges δ-gráf.

Procedure *GK0*

1. $G \leftarrow \{s\}$: $NYÍLT \leftarrow \{s\}$
2. while not *üres*(*NYÍLT*) loop
3. $n \leftarrow elem(NYÍLT)$
4. if *cél*(n) then return *van megoldás*
5. $G \leftarrow G \cup \Gamma(n)$
6. $NYÍLT \leftarrow NYÍLT - \{n\}$: $NYÍLT \leftarrow NYÍLT \cup \Gamma(n)$
7. endloop
8. return *nincs megoldás* end

Megjegyzés: Körökre érzékeny**,** Zárt csúcs ne lehessen újra nyílt?**,** Nehezen olvasható ki a megoldás**,** Jelölni kellene az utakat**,** Nem feltétlenül talál optimális megoldást



3.2. Általános gráfkereső algoritmus

Módosítások:
A következő kiterjesztés eldöntése – kiértékelő függvény
A csúcsokhoz (különösen a célcsúcshoz) vezető út nyilvántartása – szülő poiterek. A csúcsokhoz vezető minél kisebb költségű út nyilvántartása - út költségek

- Körök kizárása
- Optimális út megtalálásának igénye

Kiértékelő függvény

- f:*NYÍLT* $\rightarrow$ **R**
- a 3. lépésben $n \leftarrow \min_f(\text{NYÍLT})$
- f egy dinamikus függvény

Feszítőfa és költsége

Pointerek feszítőfája: a G egy s gyökerű feszítőfája

$\pi : G \rightarrow G \quad \pi(n) = n$ csúcs egyik szülője

$\pi(s) = nil$

Az n csúcshoz vezető, nyilvántartott $\alpha \in \{s \rightarrow n\}$ út költsége

$$g: G \rightarrow \mathbf{R} \quad g(n) := c^{\alpha}(s,n)$$

Az n csúcshoz vezető optimális út költsége (a teljes reprezentációs gráfra nézve)

$$g^*: R \rightarrow \mathbf{R} \quad g^*(n) := c^*(s,n) \leq g(n)$$

Konzisztens és optimális költségű feszítőfa

A G feszítő fája konzisztens, ha a g függvény minden G-beli n csúcshoz a feszítőfában nyilvántartott $s \rightarrow n$ út költségét adja meg.
A G feszítőfája optimális, ha minden G-beli n csúcshoz G-beli $s \rightarrow n$ optimális utat tárol.
Vigyázat! Nem az R-re nézve optimális. Mindkét tulajdonság fenntartásához szükséges, hogy egy n csúcs kiterjesztésekor - amennyiben van egy (n,m) él - megvizsgáljuk az m csúcs π illetve g értékét.

Az m csúcs három esete

Új csúcs

- Ha $m \notin G$ akkor
  - $\pi(m) \leftarrow n$, $g(m) \leftarrow g(n) + c(n,m)$
  - $NYÍLT \leftarrow NYÍLT \cup \{m\}$

Régi csúcs, amelyhez olcsóbb utat találtunk

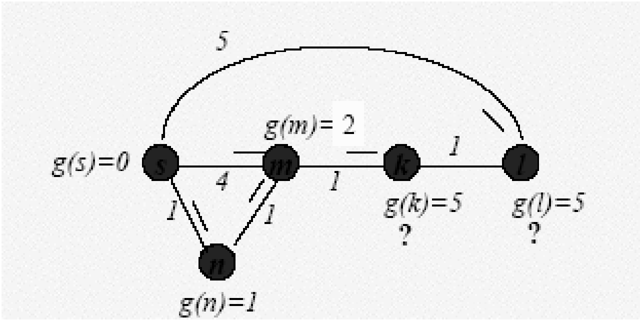
- Ha $m \in G$ és $g(n)+c(n,m)<g(m)$ akkor
  - $\pi(m) \leftarrow n$, $g(m) \leftarrow g(n) + c(n,m)$

Régi csúcs, amelyhez nem találtunk olcsóbb utat

- Ha $m \in G$ és $g(n)+c(n,m) \geq g(m)$ akkor
  - *SKIP*

Optimális költségű konzisztens feszítőfa?

Ha $m \in G$ és $g(n)+c(n,m)<g(m)$, és m csúcsnak vannak leszármazottai



Lehetséges megoldások

1) Az *m* csúcs összes keresőgráfbeli leszármazottját átvizsgáljuk valamilyen gráfbejárási technikával.

2) Ügyes kiértékelő függvénnyel biztosítjuk, hogy soha nem forduljon elő ilyen eset

3) Nem törődünk a zárt *m* csúcs leszármazottaival, de magát az *m* csúcsot visszahelyezzük a *NYÍLT* halmazba.

Procedure *GK*

1. $G \leftarrow \{s\}$: $NYÍLT \leftarrow \{s\}$: $g(s) \leftarrow 0$: $\pi(s) \leftarrow nil$
2. **while** not *üres*(*NYÍLT*) **loop**
3. $n \leftarrow minf(NYÍLT)$
4. **if** *cél*(*n*) **then return** megoldás
5. $NYÍLT \leftarrow NYÍLT - \{n\}$
6. **for** $\forall\, m \in \Gamma(n)$ **loop**
7. **if** $m \notin G$ or $g(n)+c(n,m)<g(m)$ **then**
8. $\pi(m) \leftarrow n$, $g(m) \leftarrow g(n)+c(n,m)$, $NYÍLT \leftarrow NYÍLT \cup \{m\}$
9. **endloop**
10. $G \leftarrow G \cup \Gamma(n)$
11. **endloop**
12. **return** nincs megoldás **end**

3.1. Lemma

A *GK* működése során egy csúcsot legfeljebb véges sokszor terjeszt ki.

Bizonyítás:

1. Egy *n* csúcs legfeljebb annyiszor kerülhet be a *NYÍLT*-ba, ahányszor egy minden addiginál olcsóbb utat találunk hozzá.

2. Ilyen útból legfeljebb véges sok van:

Először egy *C* költségű utat találunk az *n* csúcshoz. Ennél olcsóbb utak a *C*/d korlátnál biztos rövidebbek. (d) Megadott korlátnál rövidebb utak száma véges. (s)

3.1. Tétel

A *GK* véges reprezentációs gráfban mindig terminál.

Bizonyítás: A *GK* véges sok csúcsot (*3.1.lemma*) véges lépésben végleg kiterjeszt, azaz üres *NYÍLT* halmazzal áll meg, hacsak már korábban nem terminál másként.

3.2. Invariáns lemma

Legyen n egy tetszőleges s-ből elérhető csúcs. A *GK* az n csúcs kiterjesztése előtt bármelyik $s^*\rightarrow n$ optimális úton mindig nyilvántart egy olyan m csúcsot, amelyikre teljesül, hogy

(1) $m\in NYÍLT$
(2) $g(m)=g^*(m)$
(3) minden m csúcsot megelőző csúcs végleg zárt, azaz minden ilyen k csúcsra $g(k)=g^*(k)$.

Bizonyítás

Teljes indukció a lépések (kiterjesztések) száma szerint

1. lépés előtt: Bármelyik s-ből elérhető n csúcsra kezdetben fennáll, hogy

$$s\in NYÍLT \text{ és } s\in\ s^*\rightarrow n \text{ és } g^*(s)=0=g(s)$$

i. lépés előtt: Tegyük fel, hogy az állítás igaz minden s-ből elérhető, az *i*-dik lépésben még ki nem terjesztett n csúcsra. Rögzítsünk egy ilyen n csúcsot, és egy $\alpha=s^*\rightarrow n$ utat, amelyre tehát

$$\exists\, m\in s^*\rightarrow n \text{ úgy, hogy (1)(2)(3) fennáll.}$$

i. lépésben:

1. lehet, hogy nem az m csúcsot terjesztjük ki: Ekkor m-re továbbra is fennáll (1)(2)(3)

2. ha az m csúcsot terjesztjük ki ($m\neq n$), akkor kell keresni egy olyan csúcsot a úton, amelyik átveszi az invariáns állításban szereplő m csúcs szerepét. Az m csúcsnak az a úton fekvő utóda az m', amelyre az m csúcs kiterjesztése után két eset állhat fenn: $m' \in NYÍLT$, vagy $m' \notin NYÍLT$

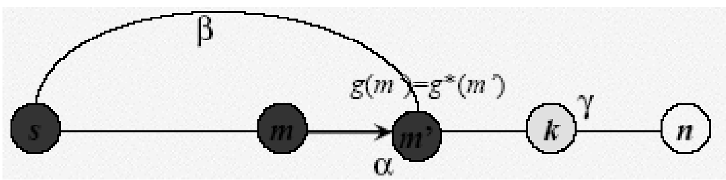
a) $m' \in NYÍLT$

ekkor (1) nyilván teljesül m'-re

$g(m') = g(m)+c(m,m') = g^*(m)+c(m,m')= g^*(m')$ miatt (2) is

(3) pedig azért, mert az m előtti csúcsok már eddig is zártak voltak, és most m vált végleg zárttá.

b) $m' \notin NYÍLT$ akkor kell lenni egy $\beta=s^*\rightarrow m'$ optimális útnak, amely mentén már korában elértük az m'-őt, azaz β minden l csúcsa zárt, és a $g(l)=g^*(l)$.



Legyen γ az μ út $m'\rightarrow n$ szakasza.

A $\beta\gamma$ egy $s^*\rightarrow n$ út, ezért az indukciós feltevés miatt tartalmaz egy(1)(2)(3) tulajdonságú k csúcsot, amely biztosan a g szakaszon van.

Ez a k csúcs az μ útnak is kívánt tulajdonságú eleme.

3.2. Tétel

Ha egy véges reprezentációs gráfban létezik megoldás akkor a *GK* egy célcsúcs megtalálásával terminál.

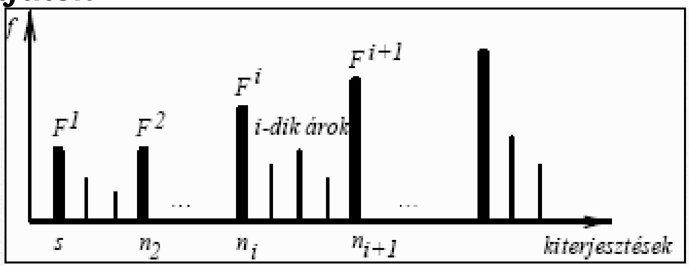
Bizonyítás: A *3.2. lemma* szerint (legyen az ottani *n* csúcs most a célcsúcs) a *NYÍLT* halmaznak mindig van eleme a célcsúcs elérése előtt, így nem terminálhat az algoritmus üres *NYÍLT* halmazzal. *3.1. tétel* miatt viszont az algoritmusnak mindenképpen terminálnia kell.

Csökkenő kiértékelő függvény

a *NYÍLT* halmazbeli csúcsok kiértékelő függvényértéke az ott tartózkodás alatt nem nő, és a *NYÍLT* halmazba visszakerülő csúcs kiértékelő függvényértéke határozottan kisebb, mint a csúcs megelőző kiterjesztésekor felvett függvényértéke.

A nevezetes gráfkereső algoritmusok ilyen csökkenő kiértékelő függvényt használnak.

Működési grafikon



Jelöljük meg a startcsúcsot, majd azt a legközelebbi csúcsot (a küszöbcsúcsot), amely kiterjesztésekor mért függvényérték (a küszöbérték) nagyobb vagy egyenlő, mint a megelőző küszöbérték.

3.3. Tétel

Csökkenő kiértékelő függvény használata mellett a *GK* a küszöbcsúcsok kiterjesztésének pillanatában optimális költségű konzisztens feszítőfát tart nyilván.

Bizonyítás: HF (A küszöbcsúcsok száma szerinti teljes indukcióval lássuk be, hogy egy küszöbcsúcs kiterjesztésekor nincs zárt csúcs a *NYÍLT* halmazban! Ehhez azt kell megmutatni, hogy amikor egy zárt csúcs újra nyílt lesz, akkor az még a következő küszöbcsúcsot megelőzően biztosan kiterjesztődik.)

3.3. Nevezetes gráfkereső algoritmusok

Most az *f* kiértékelő függvény megválasztása következik.

- Neminformált
    - mélységi,
    - szélességi,
    - egyenletes
- Heursztikus

- Előre tekintő,
- A, A*, A^c,
- B

Mélységi gráfkeresés

Mélységi gráfkeresésnek (depth-first) a *GK*-t akkor nevezzük, ha az élköltségeket egységnyinek vesszük (*GK 7. és 8. lépés*), a kiértékelő függvényt pedig az alábbi módon definiáljuk:

$$f(n) = -g(n) \quad \forall\, n \in NYÍLT \text{ csúcsra.}$$

Használhatunk mélységi korlátot. Mind a mélységi gráfkeresés, mind a visszalépéses keresés mélységi bejárást végez.

Szélességi gráfkeresés

Szélességi gráfkeresésnek (breadth-first) a *GK*-t nevezzük, ha az élköltségeket egységnyinek vesszük (*GK 7. és 8. lépés*), a kiértékelő függvényt pedig az alábbi módon definiáljuk:

$$f(n) = g(n) \quad \forall\, n \in NYÍLT \text{ csúcsra.}$$

Mindig a legrövidebb megoldást adja. Egy csúcsot legfeljebb egyszer terjeszt ki.

Egyenletes keresés

Egyenletes keresésnek (uniform-cost) a *GK*-t akkor nevezzük, ha a kiértékelő függvényt az alábbi módon definiáljuk:

$$f(n) = g(n) \quad \forall\, n \in NYÍLT \text{ csúcsra.}$$

Egy csúcsot legfeljebb egyszer terjeszt ki. Mindig az optimális megoldást adja.

Heurisztikus függvény

Azt $h{:}N \rightarrow \mathbf{R}$ függvényt, amelyik minden n csúcsra az abból a célba vezető út költségére ad becslést heurisztikus függvénynek hívjuk.

$$h(n) \approx h^*(n) = \min_{t \in T}\{c^*(n,t)\} = c^*(n,T)$$

Egy $s=n0, n1, \ldots, nk=t$ optimális megoldási út esetén

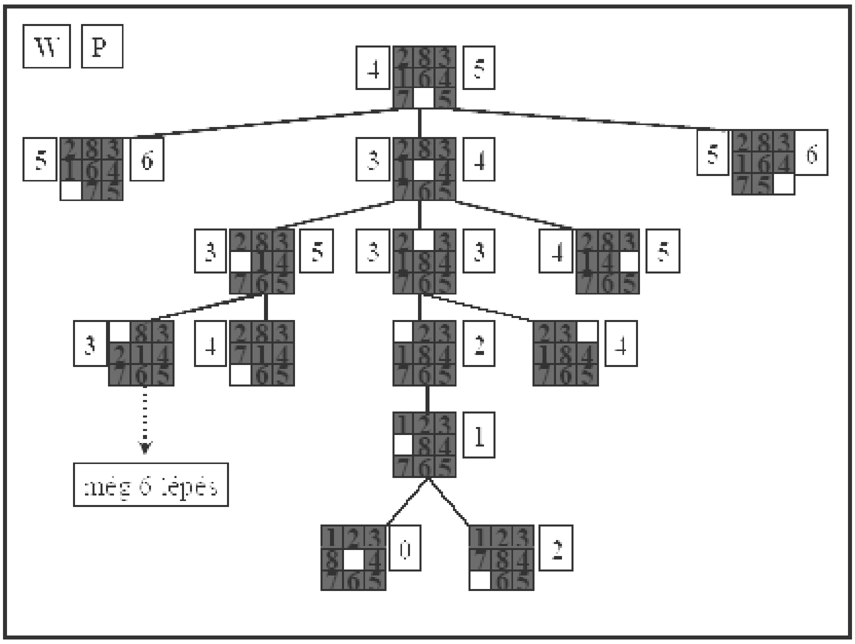
$$h^*(n_i) = \sum_{j=i}^{k-1} c(n_j, n_{j+1})$$

Előre tekintő keresés

Azt a *GK*-t nevezzük előretekint keresésnek, amelyre

$$f(n) = h(n) \quad \forall\, n \in NYÍLT$$

Szeszélyes, eredményessége és hatékonysága erősen függ a heurisztikus függvénytől

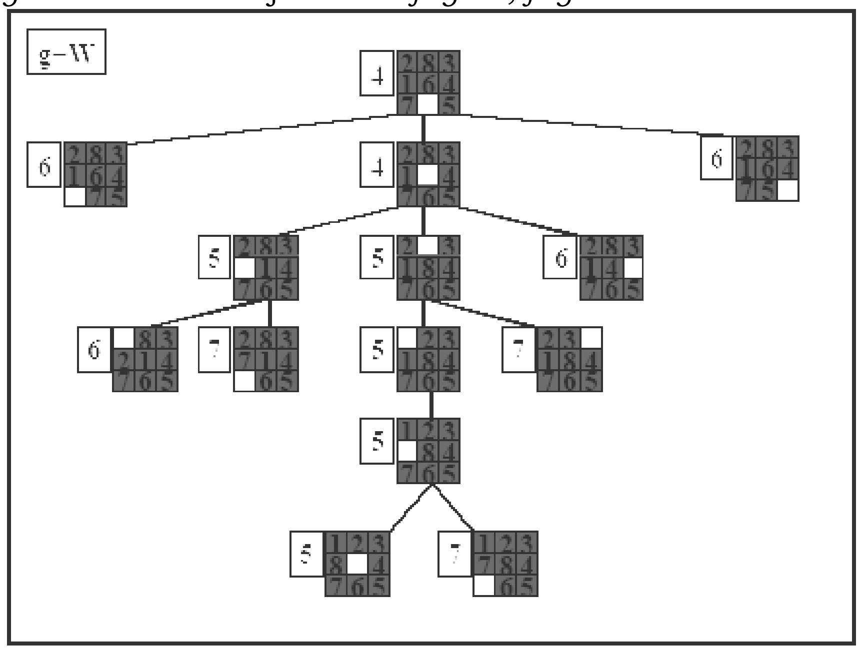


A algoritmus

Azt a *GK*-t nevezzük *A algoritmusnak*, amelyre az

$$f(n)=g(n)+h(n) \ \ \forall\, n \in NYÍLT \text{ , és}$$
$$h(n) \geq 0 \ \ \forall\, n \in N$$

Példa: $f=g+0$ illetve a 8-as játéknál: $f=g+W$, $f=g+P$



Megjegyzés

Legyen $f^*: N \to R$ optimális költségfüggvény olyan, hogy "$\forall\, n \in N$ -re $f^*(n)=g^*(n)+h^*(n)$, azaz $f^*(n)$ a legolcsóbb n csúcson átvezető megoldás költsége.

$f \approx f^*$, ahol $g \geq g^*$ és $h \approx h^*$,
$f^*(s)$ az optimális megoldás költsége.

3.3. Lemma

Az f * optimális költségfüggvény egy optimális megoldási út mentén állandó.

Bizonyítás: Legyen $s=n0, n1, \ldots, nk = t$ egy optimális megoldás.

$f^*(ni) = g^*(ni) + h^*(ni) =$

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

$$= \sum_{j=0}^{i} c(n_j, n_{j+1}) + \sum_{j=i+1}^{k-1} c(n_j, n_{j+1}) =$$

$= g^*(ni+1) + h^*(ni+1) = f^*(ni+1)$

3.4. Lemma

Ha az *A algoritmus* nem terminál, akkor minden *NYÍLT* halmazba bekerült *m* csúcs véges sok lépés után kiterjesztődik

Bizonyítás: Egy csúcs kiértékelő függvényértéke arányos a csúcs mélységével.

$$f(n) = g(n) + h(n) \geq g^*(n) \geq d(n)^*\delta \geq d^*(n)^*\delta$$

ahol $d(n)$ az $s \to n$ optimális út hossza, $d^*(n)$ a legrövidebb $s \to n$ út hossza. A $D = \lceil f(m)/\delta \rceil$ korlátnál mélyebben elhelyezkedő csúcsok nem előzik meg az *m* csúcsot a kiterjesztésben.

$$f(k) \geq d^*(k)^*\delta > D^*\delta > f(m)$$

Tehát csak a *D*-nél magasabban fekvő csúcsokra állhat fenn, hogy $f(k) \leq f(m)$

De egy δ-gráfban a σ-tul. miatt *D*-nél magasabban fekvő csúcsból csak véges (legfeljebb σ^D) sok van, és ezek véges sok lépésben végleg (lásd *3.1. lemma*) kiterjesztődnek.

3.4. Tétel

Az *A algoritmus* mindig talál egy megoldást, ha az létezik.

Bizonyítás: Jelölje α az $s=n0, n1, \ldots, nk=t$ optimális megoldást. Kezdetben az *s* nyílt csúcs.

Ha n_i egy nyílt csúcs ($i=0\ldots k$), akkor az a *3.4. lemma* miatt véges lépésben kiválasztódik, és $i<k$ esetén az n_{i+1} legkésőbb ekkor (*ni* kiterjesztésekor) bekerül a *NYÍLT* halmazba.

Tehát az algoritmus véges lépés múlva kiválasztja a *t* célcsúcsot, hacsak korábban nem terminál.

De korábban csak egy másik megoldás megtalálásával terminálhat - üres *NYÍLT* halmazzal nem -, hiszen a *t* kiterjesztése előtt a *NYÍLT* halmaz biztos tartalmazza az α egyik csúcsát (*3.2. lemma*).

Megjegyzés: Az *A algoritmus* nem mindig terminál, csak ha van megoldás, de akkor megtalál egyet.

A* algoritmus

Azt az *A algoritmust* nevezzük *A* algoritmusnak*, amelynek heurisztikája *optimális* (admissible = megengedhető), azaz *h*(*n*)£*h*(n)* "*n*Î*N*

Megjegyzés:

$$0 \leq h(n) \leq h^*(n) \;\; \forall\, n \in N$$

$$h(t)=0 \;\; \forall\, t \in T$$

Példa: $f=g+0$, *illetve a 8*-as játékesetében: $f=g+W$, $f=g+P$

3.5. Lemma

Az *A* algoritmus* által kiterjesztésre kiválasztott bármely *n* csúcsra teljesül, hogy $f(n) \leq f^*(s)$.

Bizonyítás: Tegyük fel, hogy a reprezentációs gráfban létezik megoldás, így $s^* \rightarrow t$ optimális út is. Ellenkező esetben az $f^*(s)=\infty$.

$\exists\, m \in s^* \rightarrow t$ úgy, hogy $m \in NYÍLT$ és $g(m)=g^*(m)$ (*3.2.lemma*).

De az algoritmus az *n* csúcsot választotta ki az *m* helyett $f(n) \leq f(m) =$ $=g(m)+h(m) = g^*(m)+h(m) \leq g^*(m)+h^*(m) = f^*(m) = f^*(s)$

3.5. Tétel

Az *A* algoritmus* mindig optimális megoldás megtalálásával terminál feltéve, hogy létezik megoldás.

Bizonyítás: Az *A* algoritmus*, mint speciális *A algoritmus*, biztos talál megoldást. (*3.4. tétel*). Tegyük fel indirekt módon, hogy ez a megoldás nem optimális, azaz a termináláskor kiválasztott $t \in T$ célcsúcsra $g(t)>f^*(s)$. A *3.5. lemma* szerint (*n* helyébe *t*) : $f(t) \leq f^*(s)$

De *t* célcsúcs: $f(t) = g(t)+0$

A^c algoritmus

Azt az *A algoritmust* nevezzük *A^c algoritmusnak*, amelynek heurisztikája *következetes*, azaz *monoton megszorításos*: $h(n)-h(m) \leq c(n,m) \;\; \forall\, (n,m) \in A$

célcsúcsokban pontos: $h(t)=0 \;\; \forall\, t \in T$

Példa: $f=g+0$ illetve a *8*-as játék esetében : $f=g+W$, $f=g+P$

3.6. Lemma

Ha a *h* heurisztika monoton megszorításos, akkor bármely $n,m \in N$ csúcsra fennáll a $h(n)-h(m) \leq c(n,m)$.

Bizonyítás.

Ha nincs $n \rightarrow m$ út, akkor $c(n,m) = \infty$. Ha van, akkor jelölje azt $\alpha=(n=n0, n1, \ldots, nk=m)$. Ennek éleire teljesül:

$$h(n)-h(n1) \leq c(n,n1)$$

$$h(n1)-h(n2) \leq c(n1,n2)$$

$$\ldots$$

$$h(nk-1)-h(m) \leq c(nk-1,m)$$

Adjuk össze ezeket az egyenlőtlenségeket:

$$h(n)-h(m) \le \sum c(n_{i-1},n_i) = c^{\alpha}(n,m)$$

3.6. Tétel

A következetes heurisztika egyben optimális is.

Bizonyítás: Kell: $h(n) \le h^*(n)\ \forall\, n \in N$

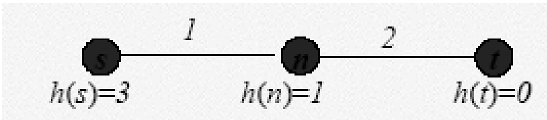
Ha nincs $n \to T$ út, akkor $h^*(n) = \infty$, tehát $h(n) \le h^*(n)$.

Egyébként $h^*(n) = \min_{t\in T} c^*(n,t) = c^{\alpha}(n,t)$ *ahol* α a egy $n^* \to T$ út.

A következetes heurisztika monoton megszorításos, így a *3.6. lemma* miatt $h(n)-h(t) \le c^{\alpha}(n,t)$, továbbá célcsúcsban pontos, azaz $h(t)=0$. Összeolvasva $h(n) \le h^*(n)$.

Következmény

Minden A^c *algoritmus* egyben *A* algoritmus* is, azaz optimális megoldást talál, ha van megoldás. Viszont fordítva nem igaz, azaz nem minden *A* algoritmus* A^c *algoritmus*.



Általában könnyebb igazolni a következetességet, mint az optimalitást.

3.7. Tétel

Amikor az A^c *algoritmus* egy csúcsot kiterjesztésre kiválaszt, akkor már ismeri hozzá az optimális utat:

$$g(n)=g^*(n).$$

Bizonyítás: TF indirekt: n kiterjesztésekor $g(n)>g^*(n)$.

$\exists\, m \in s^* \to n$ úgy, hogy $m \in NYÍLT$ és $g(m)=g^*(m)$ (*3.2.lemma*) (Az indirekt feltevés miatt az $n \ne m$.) A *3.6. lemma* miatt $h(m)-h(n) \le c^*(m,n)$ De az algoritmus az n csúcsot választotta ki m ellenében:

$$f(n) \le f(m) = g(m)+h(m) = g^*(m)+h(m) \le$$
$$\le g^*(m)+c^*(m,n)+h(n) = g^*(n)+h(n) < g(n)+h(n) = f(n)$$

Következmény

Az A^c *algoritmus* egy csúcsot egynél többször nem terjeszt ki.

3.4. A* algoritmus hatékonysága

Hatékonyság: Memória igény Futási idő Zárt csúcsok száma a termináláskor

- különböző heurisztikák
- különböző algoritmusok

Kiterjesztések száma a zárt csúcsok számához viszonyítva

A továbbiakban olyan problémákat vizsgálunk, amelyeknek van megoldása, így az *A* algoritmus* terminál.

Különböző heurisztikájú A* algoritmusok összehasonlítása

Az *A1* (*h1*) és *A2* (*h2*) *A* algoritmusok* közül az *A2* jobban informált, mint az *A1*, ha minden $n \in N \setminus T$ csúcsra teljesül, hogy $h1(n)<h2(n)$.
Megmutatjuk, hogy egy jobban informált *A* algoritmus* nem terjeszt ki több csúcsot, mint a kevésbé informált.

3.8. Tétel

Legyen *A2* jobban informált *A* algoritmus*, mint az *A1*. Ekkor *A2* nem terjeszt ki olyan csúcsot, amelyet *A1* sem terjeszt ki.
Bizonyítás: Teljes indukció az *A2* terminálásakor nyilvántartott feszítőfa mélysége (szintjei) szerint. A (*0*)-dik szinten csak a startcsúcs van, amit vagy mindkét algoritmus kiterjeszt, ha az nem célcsúcs, vagy egyik sem.
(*d*)-edik szintig minden csúcsról feltesszük, hogy ha azt *A2* kiterjesztette, akkor *A1* is. Indirekt tegyük fel, hogy a (*d+1*)-ik szinten van olyan $m \in N$ csúcs, amit csak az *A2* terjeszt ki. (*m* nyilván nem célcsúcs.)
Egyrészt $f2(m) \leq f^*(s)$ fennáll a *3.5.lemma* miatt. (Az $f2(m)$ az *m* csúcs *A2* általi kiterjesztéskor mért érték.) Másrészt $f1(m) \geq f^*(s)$. (Az $f1(m)$ az *A1* terminálásakor mért érték.) (ld. később). Harmadrészt $g2(m) \geq g1(m)$ (ld. később)

$$\text{Összegezve } f2(m) \leq f^*(s) \leq f1(m) = g1(m) + h1(m) \leq$$
$$\leq g2(m) + h1(m) < g2(m) + h2(m) = f2(m)$$

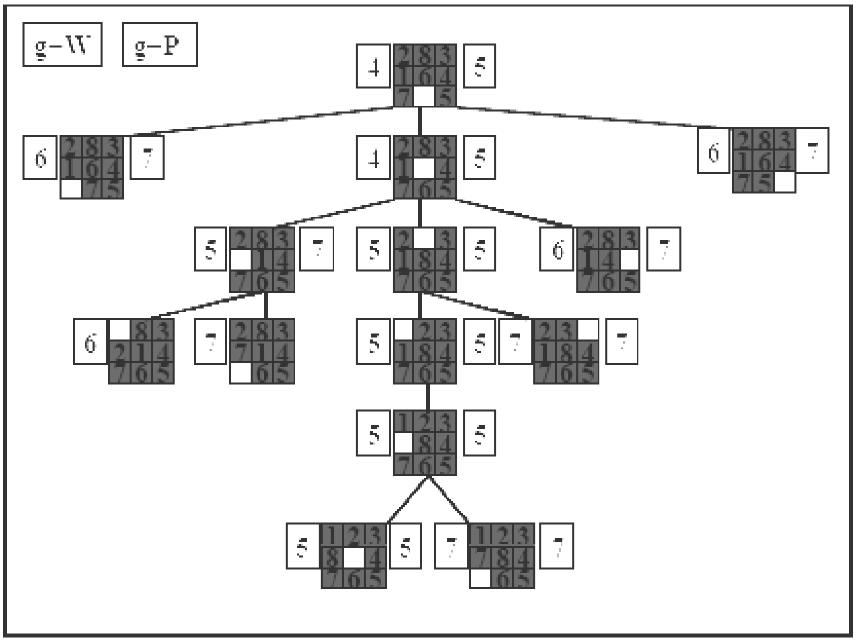
Az $f1(m) \geq f^*(s)$, mert az *m d* szinten levő *A2* által kiterjesztett szülőjét (az indukciós feltevés miatt) az *A1* is kiterjeszti, és így az *A1* algoritmus működése alatt az *m* legkésőbb ekkor nyílt csúcs lesz. Ugyanakkor $f1(m) < f^*(s)$ sohasem lehet, mert ekkor az *A1* kiterjesztené az *m* csúcsot, ami ellentmond az indirekt feltevésnek.
A $g2(m) \geq g1(m)$, mert az *A1* biztosan feltárta az *m* csúcshoz vezető *A2* által talált legolcsóbb (tehát *d* szint alatti) utat, de lehet, hogy talált egy még olcsóbbat.

Megjegyzés

A gyakorlatban sokszor enyhébb feltételek mellett látványosabb különbségekkel is találkozhatunk: Már $h1 \leq h2$ esetén is több csúcsot terjeszt ki az *A1*, mint *A2*

$$W \leq P$$

Minél jobban becsli alulról a heurisztika a h^*-ot, várhatóan annál kisebb lesz a memória igénye.



Különböző gráfkereső algoritmusok összehasonlítása

Optimális heurisztikájú feladatokon hasonlítjuk össze az A algoritmust (amely ilyenkor természetesen egy A* algoritmus) más algoritmusokkal.
Egy nem-determinisztikus algoritmus determinisztikus változatai („tie-breaking rule") *algoritmusosztályt* alkotnak, ezért valójában algoritmusosztályokat hasonlítunk össze.

Jobb algoritmusosztály

Az *X* és *Y* algoritmusosztályok. Az *X jobb Y* -nál egy adott feladatosztályra nézve**,** ha a feladatosztály minden feladatára van az *X*-nek egy olyan algoritmusa, amely csak olyan csúcsokat terjeszt ki (értékel ki), amelyeket az *Y* minden algoritmusa kiterjeszt (kiértékel).

Optimális algoritmusok

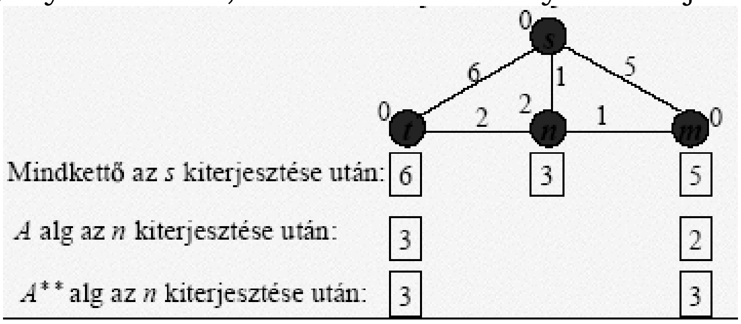
Optimálisnak nevezzük azt az algoritmust/algoritmusosztályt (nem feltétlenül gráfkeresést), amely optimális heurisztikájú feladatokra optimális megoldást talál feltéve, ha van megoldás.
Példák:

- *Egyenletes keresés*
- *A(*) algoritmus*
- *A** algoritmus:* $f(n) = \max_{m \in s \rightarrow n} \{g(m) + h(m)\}$
- *IDA* algoritmus*

Jó lenne, ha az *A algoritmus* lenne a legjobb algoritmus az optimálisak között.

*A algoritmus nem jobb az A**-nál*

Van olyan feladat és optimális heurisztika, ahol az *A(*)* minden verziója kiterjeszt egy olyan csúcsot, amit az *A*** valamelyik verziója nem.



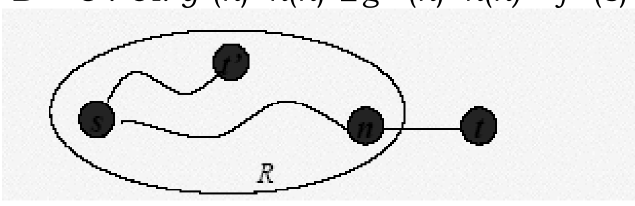
Megjegyzés

Melyik optimális algoritmus a legjobb az optimális heurisztikájú feladatosztályon?
A fentiek közül egyik sem. (az A algoritmusnál nincs jobb) Egy szűkebb feladatosztályon, a következetes heurisztikájú feladatokon viszont az *A algoritmus* a legjobb optimális algoritmus.

3.9. Tétel

Következetes heurisztikájú feladatok esetén az *A* (A^c) *algoritmus* jobb az optimális algoritmusoknál.
Bizonyítás: Tegyük fel indirekt, hogy van olyan *Y* optimális algoritmus, és olyan (R,s,T) feladat a *h* következetes heurisztikával, hogy az *R* egy $n \in N$ csúcsát az *A algoritmus* minden verziója kiterjeszti, de az *Y* algoritmus nem.
Van olyan optimális megoldás, amely nem vezet át *n*-en, hiszen csak ezt tudja az *Y* megtalálni. Mivel *A* minden verziója eljut az *n* csúcshoz, ezért van olyan $s^{\alpha} \to n$ út, amelynek minden *m* csúcsára (többek között az *n*-re is) fennáll, hogy $g^{\alpha}(m)+h(m)<f^*(s)$. (Ekkor ugyanis ennek az útnak minden csúcsa biztos bekerül a *NYÍLT* halmazba, és $f(m) \le g^{\alpha}(m)+h(m)<f^*(s)$ miatt még a célcsúcs előtt kiterjesztésre kerül.)
Jelöljük: $C= f^*(s)$, $D= g^*(n)+h(n)$

$$D < C \text{. Ui: } g^*(n)+h(n) \le g^{\alpha}(n)+h(n) < f^*(s)$$



Bővítsük az indirekt feltevésben szereplő feladatot egy új éllel:

$$\left(R \cup \{(n,t)\}, s, T \cup \{t\}\right)$$

$$c(n,t)=h(n)+(C-D)/2$$

$$h(t)=0$$

A h heurisztika az új feladaton is következetes (elég csak az új élt vizsgálni)

$$h(n)-h(t) = h(n) \leq h(n) +(C - D)/2 = c(n,t)$$

Az új feladat egyetlen optimális megoldása az $s \to t$ út. Az $s \to t$ út költsége kisebb az előző feladat optimális megoldásának költségénél (C)

$$g^*(t) = g^*(n) + c(n,t) = g^*(n) + h(n) + (C - D)/2 =$$
$$D + (C - D)/2 = (C + D)/2 < C$$

Az Y algoritmus az n csúcsot nem terjeszti ki, ezért az $s \to t$ megoldást nem találhatja meg: Ez ellentmondás, hiszen egy optimális algoritmusnak ezen az új feladaton is optimális megoldást kell találnia.

Az A algoritmusnál nincs jobb Ha lenne az *A algoritmusnál* jobb optimális algoritmus, akkor ez a következetes heurisztikájú feladatokra is jobb lenne (hiszen a következetes heurisztika is optimális). Ennek ellent mond a *3.9. tétel.*) Az *A algoritmus* nem a legjobb a optimális algoritmusok között, de nincsen nála sem jobb.

Futási idő

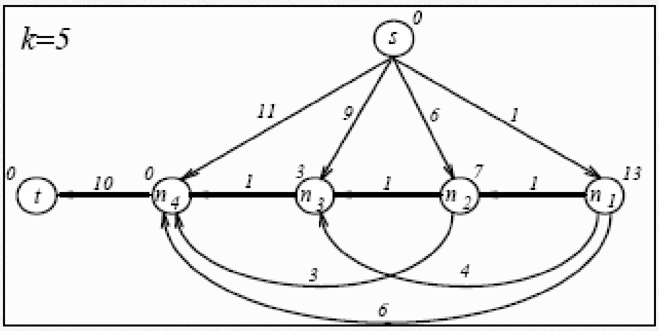
Zárt csúcsok száma: k

Alsókorlát: k

A következetes heurisztika mellett egy csúcs legfeljebb csak egyszer terjesztődik ki, habár ettől még a kiterjesztett csúcsok száma igen sok is lehet (egyenletes keresés)

Felső korlát: 2^{k-1}

Martelli példája

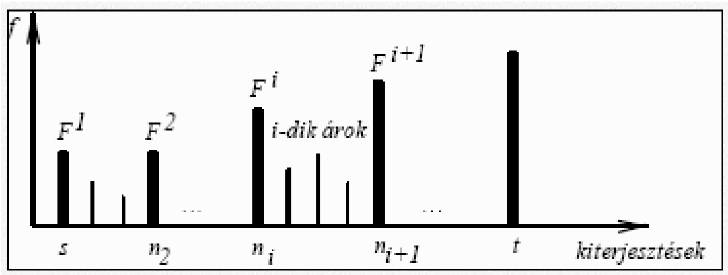


Az $n_1,\dots,n_{k-1}$ csúcsokhoz rendre $2^0,2^1,\dots,2^{k-2}$ különböző út vezet.

Megjegyzés

Másik heurisztikával természetesen javítható a kiterjesztések viszonylagos száma, de nem biztos, hogy ez tényleges javulás lesz. A kiterjesztések száma egy viszonylagos szám $h1$ heurisztika mellett $k1$ darab zárt csúcs, és 2^{k1-1} kiterjesztés $h2$ heurisztika mellett $k2$ darab zárt csúcs, és $k2$ kiterjesztés mégis $2^{k1-1} < k2$, mivel $k1 << k2$.

A probléma oka



Általában egy árkon belül egy csúcs sokszor kiterjesztésre kerülhet.
Az árkokban használjunk más kiértékelő függvényt!

AB algoritmus

Az *AB algoritmust* az *A algoritmusból* kapjuk úgy, hogy bevezetjük az *F* aktuális küszöbértéket és a *q*:*NYÍLT*→*R* belső kiértékelő függvényt , majd az *1. lépést* kiegészítjük az *F* ← *f(s)* értékadással,
a *3. lépést* pedig helyettesítjük az
if *minf(NYÍLT)<F*
then *n* ← *minq(mÎNYÍLT ½ f(m)<F)*
– else *n* ¬ *minf (NYÍLT); F* ¬ *f(n)*
– endif elágazással.

Megjegyzés

Nevezetes *AB algoritmusok* :
A algoritmus (egyenletes keresés) : *q=f*
B algoritmus (Martelli) : *q=g*
Az *AB algoritmusok* csökkenő kiértékelő függvényt használnak.

3.10. Tétel

Az eltérő belső kiértékelő függvényt használó *AB algoritmusok* működésük során ugyanazokat a küszöbcsúcsokat, ugyanabban a sorrendben és ugyanazon küszöbértékkel választják ki, és ekkor ugyanazt a keresőgráfot, ugyanazon feszítőfával és költségértékekkel tartják nyilván.
Bizonyítás: Teljes indukció a küszöbcsúcsok számára.
Az *i+1*-dik küszöbcsúcsnál bekövetkező állapot attól függ, hogy előtte mely csúcsokat terjeszti ki a keresés az az *i*-edik árokban. (Ez a kiterjesztések sorrendjétől és számától nem függ.)
Egy az *i*-edik árokban kiterjesztett *m* csúcsnak a *NYÍLT* halmazban kell lennie az *ni* kiterjesztése után, de még az *ni+1* kiterjesztése előtt úgy, hogy *f(m)* < *Fi* fennálljon. Az *m* csúcs akkor kerülhet be a *NYÍLT* halmazba, ha *i*-edik árokhoz tartozó csúcsok kiterjesztései során találunk hozzá egy *s*-ből induló *ni* -n keresztül vezető utat, amely vagy az első hozzá talált út, vagy egy minden eddigénél olcsóbb út.
Ha *m* csúcs már benn volt a *NYÍLT* halmazban az *ni* kiterjesztésekor, akkor az *f(m)* ³ *Fi* állt fenn. Ahhoz,
hogy ez megváltozzon (*f*=*g*+*h*) kell, hogy találjunk egy

minden eddigénél olcsóbb utat *m*-hez *i*-edik árokhoz
tartozó csúcsok kiterjesztései során.
Az i-edik árokban kiterjesztett csúcsok
74
_ *Di* = { *m* Î *N* ½
– $aÎ{*s* ® *ni* ® *m*}:"*n*Î *ni* ® *m* \{*m*}: *n*Î*Di*
– *gi*(*ni*)+*c*a(*ni*,*m*)<*gi*(*m*) ha *m*Î*Gi*
• *gi* az algoritmus által nyilvántartott *g* értékeket
mutatja *ni* kiterjesztésének pillanatában.
– *f(m)= g(m)+h(m)* £ *gi*(*ni*)+*c*a(*ni*,*m*)+*h*(*m*) < *Fi* }
Az i-edik árok csúcsai függetlenek q-tól
75
Megjegyzés
_ Az *AB algoritmusok* megoldással terminálnak, ha van
megoldás. (Ui: az *A* is egy *AB*)
– HF: az árkok (az utolsó árok is) véges hosszú.
_ Az optimális heurisztikát használó *AB algoritmusok*
optimális megoldással terminálnak, ha van megoldás. (Ui:
az *A** is egy *AB*)
- HF: *A** m
_
ködésekor a célcsúcs az utolsó küszöbcsúcs.
_ A következetes heurisztikát használó *AB algoritmus* egy
csúcsot legfeljebb egyszer terjesztenek ki. (Ui: az *Ac* is
egy *AB*)
– HF: *Ac* árkai üresek.
76
Megjegyzés
_ Az *AB algoritmusok* memória igénye az *A*
algoritmussal azonos. (Optimális heurisztikájú
feladatokon nincs náluk jobb.)
_ Az *AB algoritmusok* futási ideje eltér
_
, mert az
árkon belüli kiterjesztések száma és sorrendje más.
_ A futási id
_
szempontjából az *A algoritmus* nem jó
(*A* algoritmus* futási ideje legrosszabb esetben
exponenciális).
_ A legjobb futási idej
_
AB algoritmus a *B algoritmus*
(futási ideje legrosszabb esetben polinomiális)
77

3.11. Tétel
_ A *B algoritmus* egy árkon belül egy csúcsot csak egyszer terjeszt ki.
_ Bizonyítás:
_ Tegyük fel indirekt, hogy egy *m* csúcs kétszer terjeszt
_
dik ki a *Di* árkon belül: el
_
ször az *ni*
küszöbcsúcsból egy drágább aút mentén érjük el az *m* csúcsot, majd egy olcsóbb bút mentén, azaz
*c*b(*ni m*) < *c*a(*ni m*)
78
ni m
k
s
a
b
_ Amikor az aút mentén elérjük, majd kiterjesztjük az *m* csúcsot (*m*Î*NYÍLT* és *g*(*m*) =*c*(*s,ni*)+*c*a(*ni m*)), addigra elértünk a búton egy *k* csúcshoz (*k*Î*NYÍLT* és *g*(*k*) £ *c*(*s,ni*)+*c*b(*ni k*)).
_ A *B algoritmus* az *m* csúcsot választotta:*g*(*m*)£*g*(*k*)
_ *g*(*k*) £ *c*(*s,ni*) +*c*b(*ni k*) £ *c*(*s,ni*)+*c*b(*ni m*) <
< *c*(*s,ni*)+*c*a(*ni m*) = *g*(*m*)
14
79
B algoritmus futási ideje
_ *k* zárt csúcs esetén legfeljebb *k+1* küszöbcsúcs van (az utolsó a célcsúcs, ha a heurisztika optimális), ilyenkor *k* darab árok van.
_ A *B* algoritmus legrosszabb esetben egy árokban az összes *k* csúcsot pontosan egyszer kiterjeszti.
_ Ezért a kiterjesztések összes száma legfeljebb *k2*.
80
*A**
Ac
A
Neminformált Heurisztikus
Algoritmus osztályok
El_
re
tekint_
Mé

lységi
Szélességi
AB B
A**
81
3.5. Heurisztika szerepe
_ Milyen a jó heurisztika?
– optimális : $h(n)$ £ $h^*(n)$
• Nincs mindig szükség az optimális megoldásra.
– jólinformált: $h(n) \sim h^*(n)$
– monoton megszorításos: $h(n)-h(m)$ £ $c(n,m)$
• Ekkor *Ac algoritmus*, különben *B algoritmus*
_ Változó heurisztikák:
– $f=g+f^*h$ ahol f~d
– *B'* algoritmus
82
Gyakorlat
83 84
15
85
Állítások
_ Csökken
_
kiértékel
_
függvény mellett a *GK*
küszöbcsúcsai mind különböznek, és egy csúcs csak az
els
_
kiterjesztésekor lehet küszöbcsúcs.
_ Ha a *GK* terminálásakor egyetlen nyílt csúcs (a célcsúcs)
van, akkor a *GK* optimális megoldást talált.
_ Az *A** minden olyan *n* nyílt csúcsot kiterjeszt, amelyre
$f(n)<f^*(s)$.
_ Ha a *h* monoton megszorításos, akkor egy tetsz
_
leges *n*
csúcsba vezet
_
optimális út mentén a g^*+h értéke
monoton növekv
_
.
86
Tétel

_ Csökken
_
kiértékel
_
függvény használata mellett a
GK csak konzisztens csúcsot választ kiterjesztésre.
_ Bizonyítás: HF (Képzeljük el, hogy miután az *n* csúcs egy
a út mentén kiterjeszt
_
dött, egy olcsóbb b út mentén újra
nyílt lesz. Eközben bekerült a *NYÍLT* halmazba a g út menti *m*
is. Az *m* csúcs az nyílttá válásakor inkonzisztensé válik. Meg
kell mutatni, hogy az algoritmus nem terjeszti ki az *m* csúcsot
az *n* csúcs el
_
tt! Ehhez kövessük nyomon a b és a g utak
felfedezését, az *n* csúcs korábbi kiterjesztése és újra nyílttá
válása között.
n
b
s m a g
*c*b(*s, n*) < *c*a(*s, n*)
87
*A** algoritmus*
_ Azt a *GK*-t nevezzük *A** algoritmusnak*, amelyre
az
– *f*(*n*)=max*m*Îs®*n*(*g*(*m*)+*h*(*m*)) "*n*Î*NYÍLT* ,
– *h* optimális
_ Optimális megoldást talál, ha van megoldás
– Talál megoldást. Ehhez 3.5. lemma:
_ *f*(*n*) ³ *g*(*n*)+*h*(*n*) ³ *g**(*n*) ³ *d**(*n*) *d
– Optimális a megoldás. Ehhez 3.6. lemma:
_ *f*(*n*) £ *f*(*m*) = *g*(*mi*)+*h*(*mi*) £ *f**(*mi*) = *f* *(*s*).
IDA algoritmus*
c ¬ *f*(*start*)
loop
(*megoldás, c*) ¬ *VL2.1*(<*s*>, *c*)
if *megoldás* ¹*hiba* then return *megoldás*
if *c*= ¥then return *hiba // ha VL2.1 –ben nem volt vágás*
endloop
– *VL2.1*:
• *f*=*g*+*h*
• *n* csúcsot kiértékelés nélkül levágja, ha *f*(*n*)>*c*
• *c*-ben visszaadja a levágott csúcsok *f* értékeinek
minimumát; ha nincs ilyen, akkor a ¥-t.

89
_ Az *IDA* algoritmus* optimális heurisztika mellett optimális megoldást talál, ha van megoldás!
– Talál megoldást: Egyrészt a *c* értéke nem válhat ¥naggyá megoldás megtalálása el
_
tt. Másrészt a keresés nem akadhat meg egy megoldási útnak egy csúcsánál.
– Optimális a megoldás : Ehhez elég belátni, hogy *c*£*f**(*start*) mindig fennáll, hiszen ekkor nem találhat a *VL2.1* olyan célcsúcsot, amelyre *g*(*t*)> *f**(*start*).
• Legyen *n*Î*s**®*t* az a csúcs, amit levágunk (*g*(*n*) = *g**(*n*))
Ekkor *c* £*f*(*n*) = *g*(*n*)+*h*(*n*) = *g**(*n*)+*h*(*n*) £ *f**(*n*) = *f* *(*s*).
B' algoritmus
if *h*(*n*)<*minm*ÎG(*n*) (*c*(*n,m*)+*h*(*m*))
then *h*(*n*)¬ *minm*ÎG(*n*) (*c*(*n,m*)+*h*(*m*))
else for "*m*ÎG(*n*)-re **loop**
if *h*(*n*)- *h*(*m*)>*c*(*n,m*) **then** *h*(*m*) ¬*h*(*n*)- *c*(*n,m*)
endloop
A *h* optimális marad
A *h* nem csökken
A mononton megszorításos élek száma n
_
16
91
Mohó A algoritmus
_ Nincs mindig szükség az optimális megoldásra.
– Ilyenkor a mohó *A* algoritmus is használható.
– Ha *h* optimális és "*t*Î*T*: "(*n,t*)Î*A*: *h*(*n*)+a³*c*(*n,t*)
akkor *g*(*t*)£*f**(*s*)+a
_ A mohó *A algoritmus* optimális heurisztika mellett akkor garantálja az optimális megoldást, ha
– "*t*Î*T*: "(*n,t*)Î*A*: *h*(*n*)=*c*(*n,t*) vagy
– *h* monoton és $a³*0*: "*t*Î*T*: "(*n,t*)Î*A*: *h*(*n*)+a=*c*(*n,t*)
92
Lemma
_ Az *Ac algoritmus* (!) által kiterjesztésre választott bármely *n* csúcsra teljesül, hogy *f(n)*£ *f *(s).*
_ Bizonyítás: Tegyük fel, hogy a reprezentációs gráfban létezik megoldás, így *s**®*t* optimális út is. Ellenkez
_
esetben az *f* *(*s*)=¥.
_ $*m*Î*s**®*t* úgy, hogy *m*Î*NYÍLT* és *g*(*m*)=*g**(*m*) (*3.2. lemma*).
_ De az algoritmus az *n* csúcsot választotta ki az *m* helyett

– *f*(*n*) £ *f*(*m*) = *g*(*m*)+*h*(*m*) =
– = *g**(*m*)+*h*(*m*) £ *g**(*t*)+*h*(*t*) = *g**(*t*) = *f* *(*s*).
93
Tétel
_ Az *Ac algoritmus* (!) optimális megoldás
megtalálásával terminál feltéve, hogy létezik
megoldás.
_ Bizonyítás:
_ Megegyezik a *3.5. tétel* bizonyításával, csak most a *3.5.*
lemma helyett az el
_
z
_
lemmára kell hivatkoznunk.
94
Tétel
_ Minden *Ac algoritmus A* algoritmus* is.
_ Bizonyítás: Be kell látni: *h(n)* £ *h*(n)* "*n*Î*N*
_ Ha az *n* csúcsból nem vezet út a célcsúcsba, akkor a *h**
értékét végtelen nagynak véve magától értet
_
dik az
állítás.
_ Ha viszont létezik ilyen út, akkor van *n=n0 ,n1 ,...,nk =t*
optimális út is.
95
_ Ennek éleire írjuk fel a monoton megszorítást feltételét:
– *h*(*n*)-*h*(*n1*) £ *c*(*n,n1*)
– *h*(*n1*)-*h*(*n2*) £ *c*(*n1,n2*)
– ...
– *h*(*nk-1*)-*h*(*t*) £ *c*(*nk-1,t*)
_ Adjuk össze ezeket az egyenl
_
tlenségeket:
– *h*(*n*)-*h*(*t*) £ S *c*(*ni-1,ni*) = *h**(*n*).
_ Mivel *t* egy célcsúcs, ezért *h*(*t*)=*0*, tehát *h*(*n*) £ *h**(*n*)