Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs rendszerek Tanszék

Mesterséges intelligencia – VIMIAC10

Informált keresés

Előadó: Hullám Gábor

Előadás anyaga: Pataki Béla
Dobrowiecki Tadeusz

A neminformált keresési stratégiák – általános (nem problémaspecifikus) módszerek

b - elágazási tényező, d - a megoldás mélysége,
m - a keresési fa maximális mélysége, l - a mélység korlát.

Jellem-ző	SzK	EgyKK	MK	MKK	IMK	KK
Idő-igény	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Tár-igény	b^d	b^d	bm	bl	bd	$b^{d/2}$
Opt.?	Igen (ha…)	Igen	Nem	Nem	Igen	Igen
Teljes?	Igen	Igen	Nem	Igen, ha $l \geq d$	Igen	Igen

~~**Az exponenciális tár-, illetve időigény komoly problémát jelent!**~~
Jobb módszerek kellenek!

Keresési stratégiák

Heurisztikus, vagy más néven **informált keresés**

- Büntessük a hátra (céltól elfele) keresést!
- De ugyanaz, és könnyebb díjazni az előrekeresést a célállapot irányába.

Ehhez kell valami elképzelés, hogy a cél:

- ***merrefelé és,***
- ***nagyjából milyen messzire fekszik.***

Ez az információ az un. **heurisztika**, **heurisztikus függvény h(*n*)**,

Keresési stratégiák – 2

Heurisztika, heurisztikus függvény h(*n*)

- a probléma minden *n* állapotára ki kell tudnunk számítani kifejezi a **célig előrehaladás becsült költség**ét

- ha pontos, akkor elvben fölöslegessé teszi a keresést

(ha nagyon pontatlan, akkor viszont semmit sem segít)

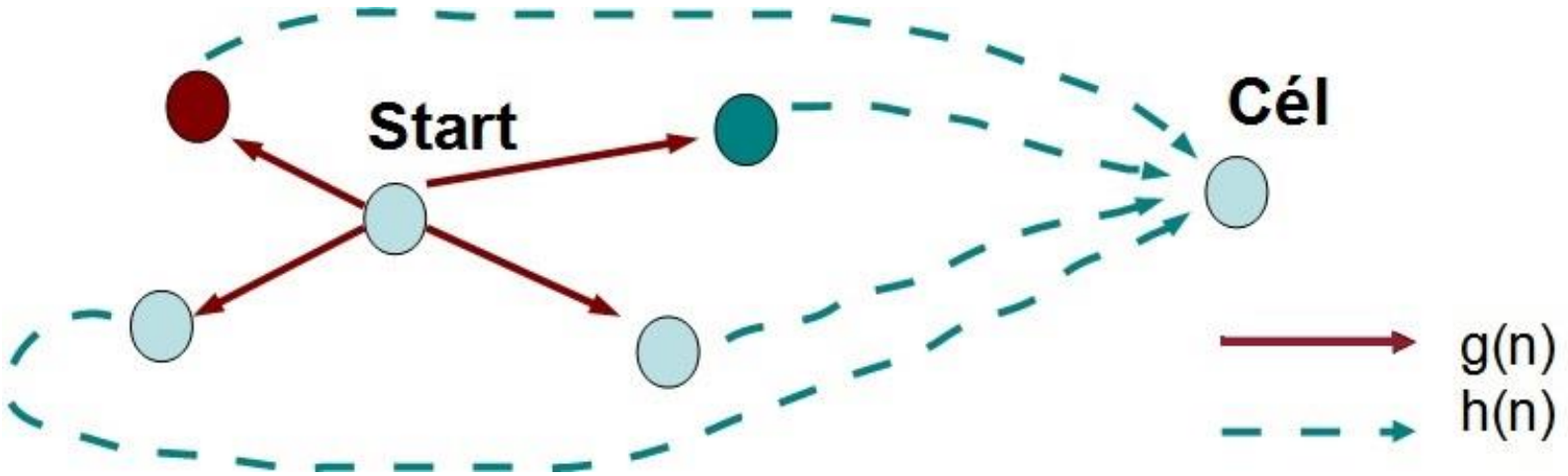
Ez minden problémára más, problémaspecifikus!

A heurisztikus függvény, h(*n*) becslést ad arra, hogy mekkora a hátralévő út legkisebb költsége.

- Ezáltal becslést ad a teljes út legjobb költségére is (f(n)).

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

Heurisztikus függvény

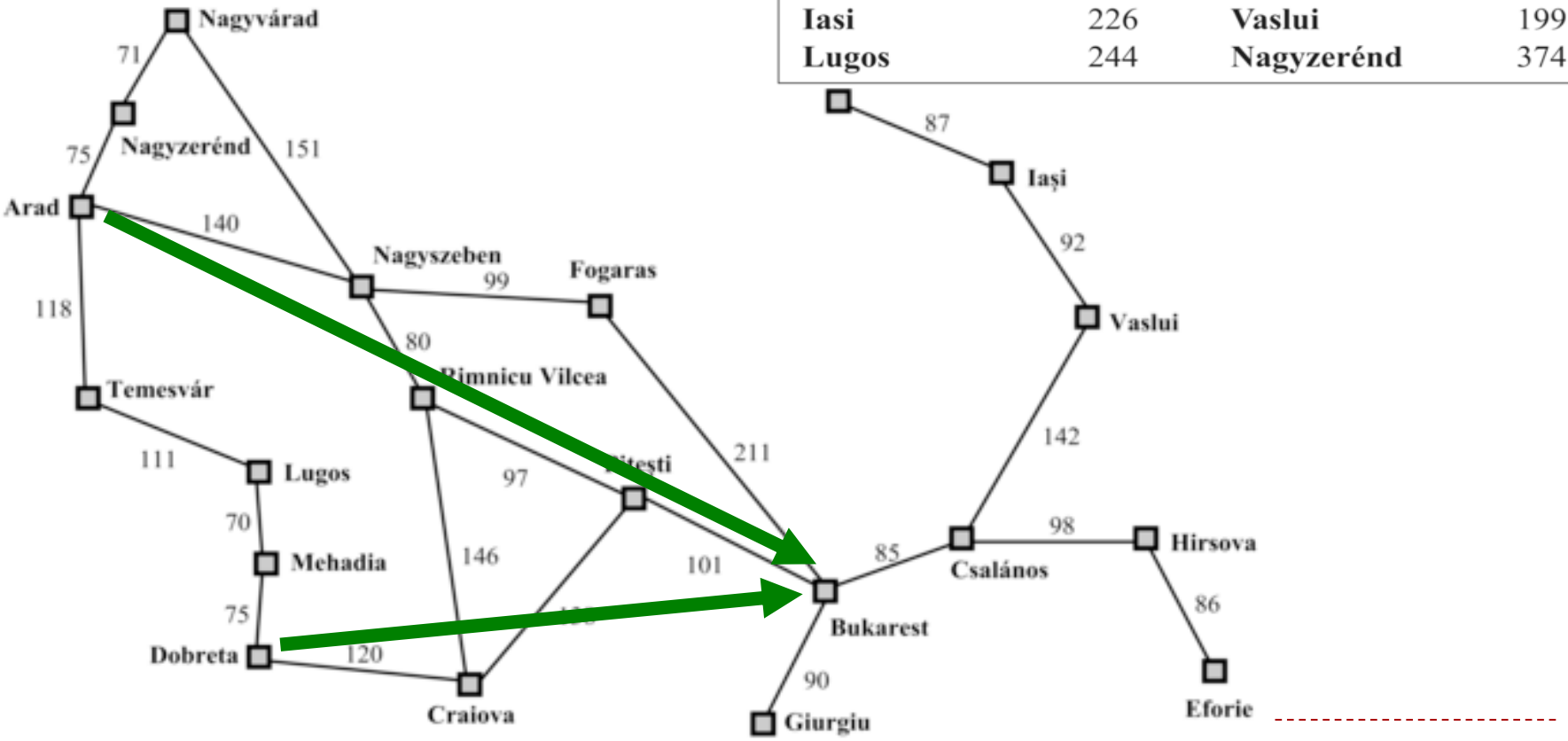


$$f(n) = g(n) + h(n)$$

- Jó heurisztikus függvénnyel a tár- és időigény jelentősen csökkenthető,
- a csökkentés az adott problémától és a *h* függvény minőségétől függ.
- Tökéletes információ (heurisztika) ⇒ előretartás elágazások nélkül (hiszen minden csomópontra pontosan tudjuk, hogy mibe kerül ezen az úton menni) ⇒ lineáris tárigény = lineáris időigény!

Legyen az útkeresési feladatunkban a heurisztikus függvény a légvonalban mért távolság (h_{LMT})
Az igényeket teljesíti?
Mi mondható a hibájáról?

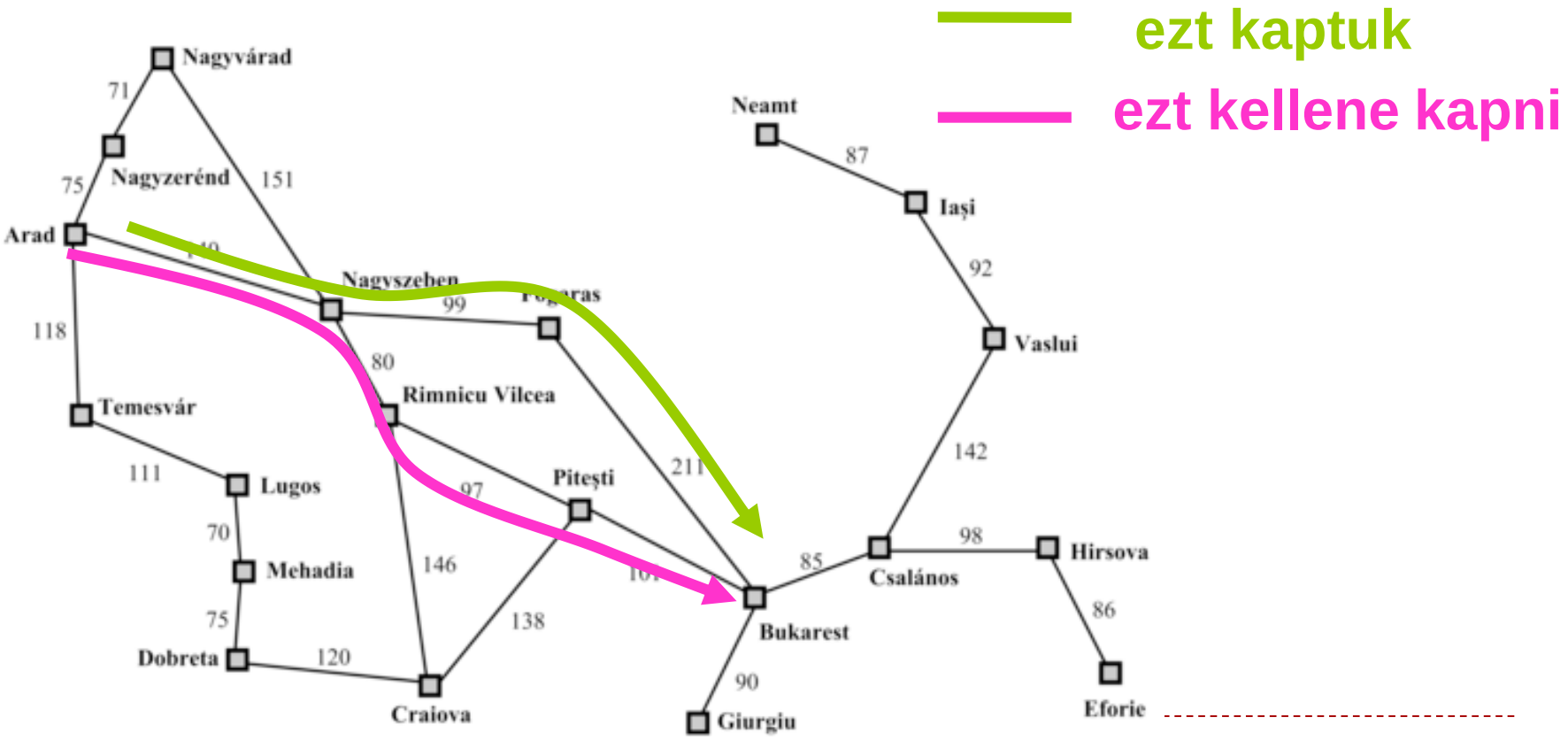
Arad	366	Mehadia	241
Bukarest	0	Neamt	234
Craiova	160	Nagyvárad	380
Dobreta	242	Piteşti	100
Eforie	161	Rimniu Vilcea	193
Fogaras	176	Nagyszeben	253
Giurgiu	77	Temesvár	329
Hirsova	151	Csalános	80
Iasi	226	Vaslui	199
Lugos	244	Nagyzerénd	374



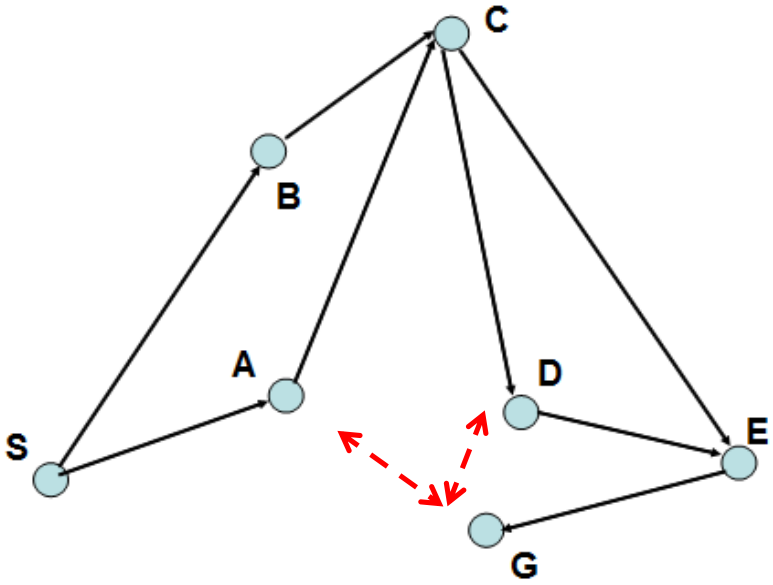
Mohó keresés:

a célhoz legközelebbinek tűnő csomópont először (a legjobbnak becsültet először)

Stratégia: a következő lépésben azt a csp-t fejti ki, amelyhez rendelt probléma-állapotot a legközelebbinek ítéli a célállapothoz (legkisebb az *n*-dik csp-hoz rendelt *h*(*n*) heurisztikus érték).

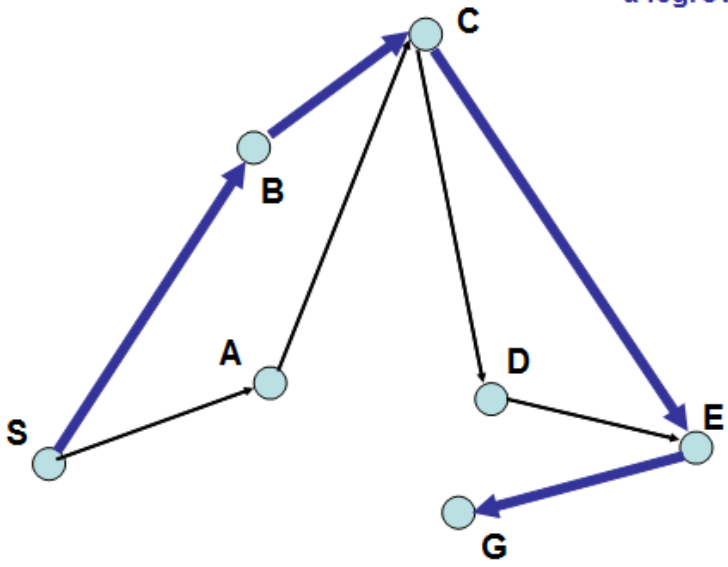


Mohó algoritmus általában **gyorsan** megtalálja a megoldást, de **nem mindig az optimális** megoldást találja meg. A mohó keresés érzékeny a hibás kezdő lépésekre is.

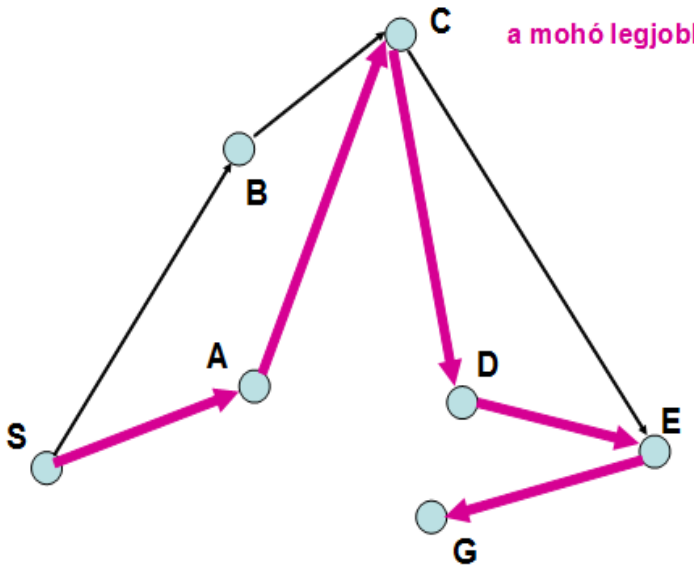


Probléma: A és D légvonalban nagyon közel van a G célhoz, de az úthálózaton nem

a legrövidebb út



a mohó legjobbat először út



Mohó keresés

- mélységi keresésre hasonlít, egyetlen út végigkövetését preferálja a célig, zsákutcából visszalép.
- Ua. a problémák, mint a mélységi keresésnél:
  - nem optimális,
  - nem teljes (elindul egy végtelen úton, és ez esetben nem tér vissza új lehetőséget kipróbálni).
- Az összes csomópontot a memóriában tartja: ezért a legrosszabb (worst-case) idő- és tárigény:
  - $O(b^m)$ – ha m a problématér mélysége

Mohó keresés

- Persze jó heurisztikus függvénnyel a tár- és időigény jelentősen csökkenthető, a csökkentés az adott problémától és a *h* függvény minőségétől függ.

- Szóval a mohó keresés ígéretes, egyszerű, sokszor akár használják is, de mégsem igazán jó.

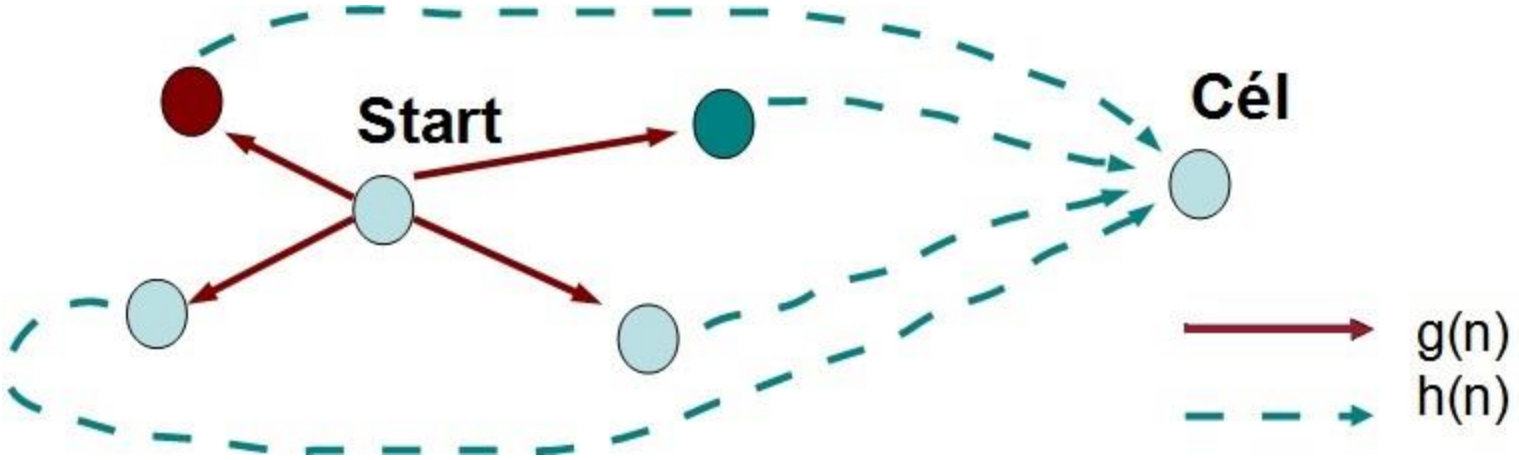
A* keresés: a teljes útköltség minimalizálása

(de facto standard az útkeresésben)

- **mohó**: **min {*h*(*n*)}** a célhoz vezető útköltség becslőjét minimalizálja
  nem teljes (nem optimális)

- **egyenletes költségű**: **min {*g*(*n*)}** a megtett út költségét minimalizálja
  optimális (egyben **teljes** is), de nagyon **rossz hatékonyságú**

a két stratégia ötvözése: **min {*f*(*n*)} = min {*h*(*n*) + *g*(*n*)}**



A* keresés: a teljes útköltség minimalizálása

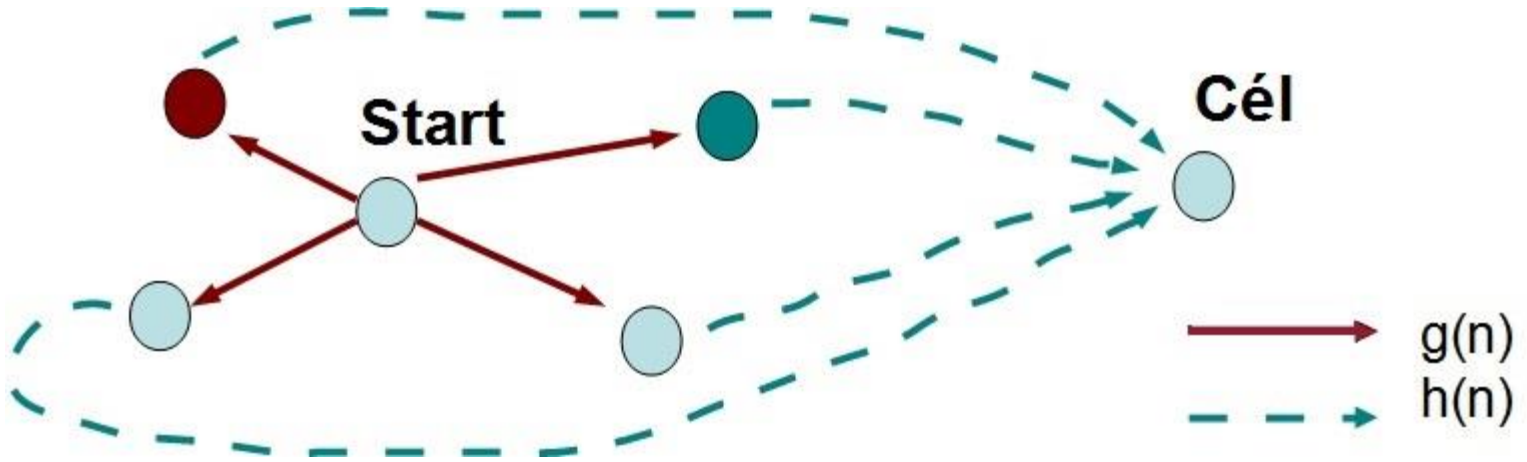
(de facto standard az útkeresésben)

min {$f(n)$} = min {$h(n)$ + $g(n)$}

***g*(*n*)**: a kiinduló cs-ponttól az *n* cs-pontig számított út **tényleges** költsége

***h*(*n*)**: az *n* cs-ponttól a célba vezető legolcsóbb költségű út **becsült** értéke

***f*(*n*)**: a kiinduló csp-tól az adott *n* csp-on át a célba vezető legolcsóbb költségű út **becsült** értéke



Elfogadható heurisztika

Ha a ***h*** függvény soha nem becsüli felül a cél eléréséhez szükséges költséget

optimista: a cél közelebbinek tűnik vagy legalább is nem távolabbinak, mint amilyen

Ha ***h*** elfogadható, akkor ***f(n)*** sohasem becsüli túl az ***n***-dik csomóponton át vezető legjobb megoldás valódi költségét

A* keresés:
*az **f*** függvényt alkalmazó legjobbat-először keresés, ahol ***h*** elfogadható

Heurisztika monotonitása

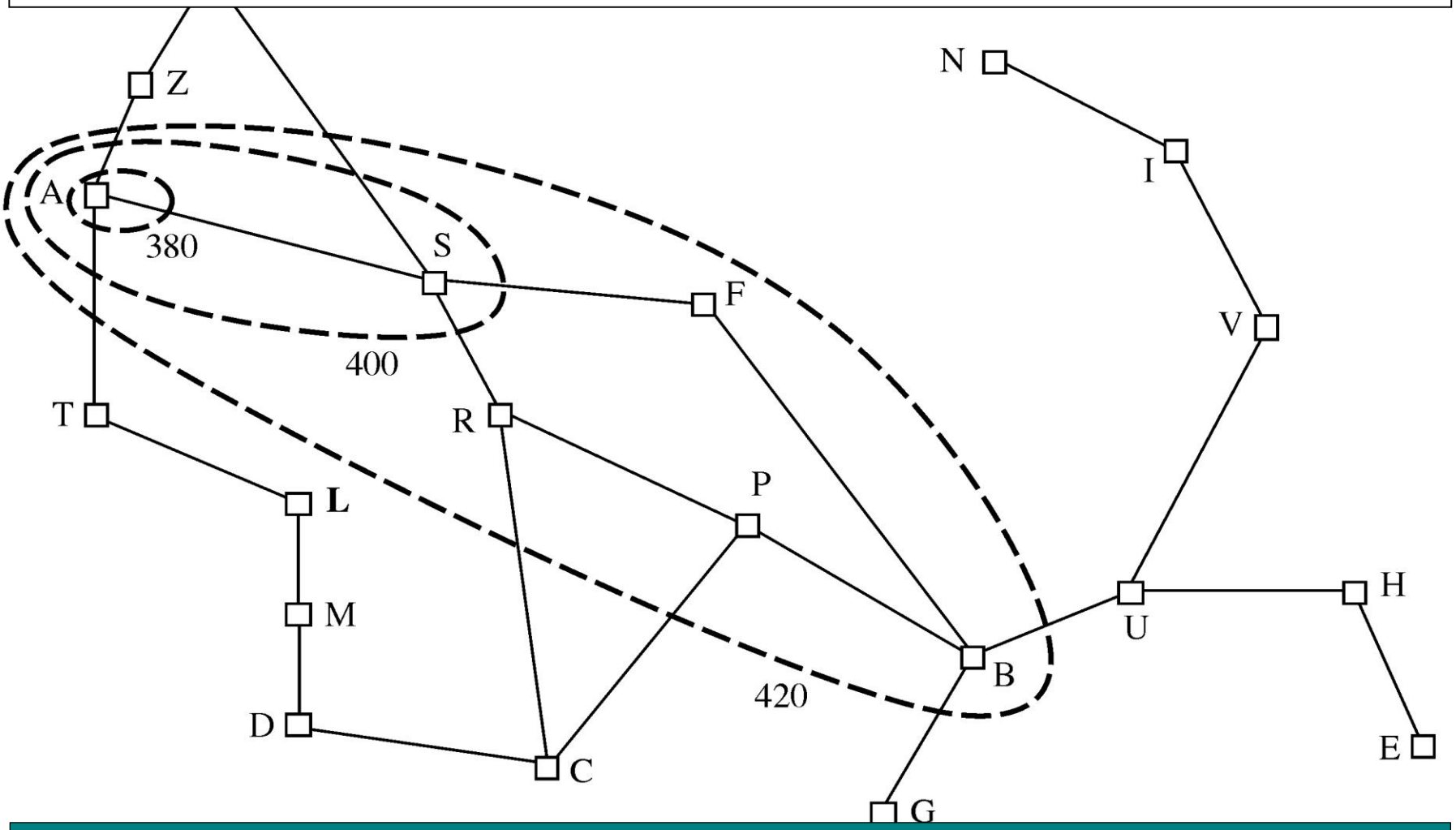
- Ha a gyökérből nézve egyetlen út ***f*** értéke sem csökken – a heurisztika **monoton nő**.

- Egy heurisztikus függvény aka. monoton, ha teljesíti a háromszög egyenlőtlenséget (**konzisztens heurisztika**).

- **Monotonná tehető**, a trükk: ha az *n'* gyermekcsp-ra f(*n'*) kisebb lenne, mint a szülőé (*n* csp) , akkor a szülő f(n) értékét használhatjuk *n'*-re is: ***n⇒n'*** ***f(n')= max(f(n), g(n') + h(n'))***

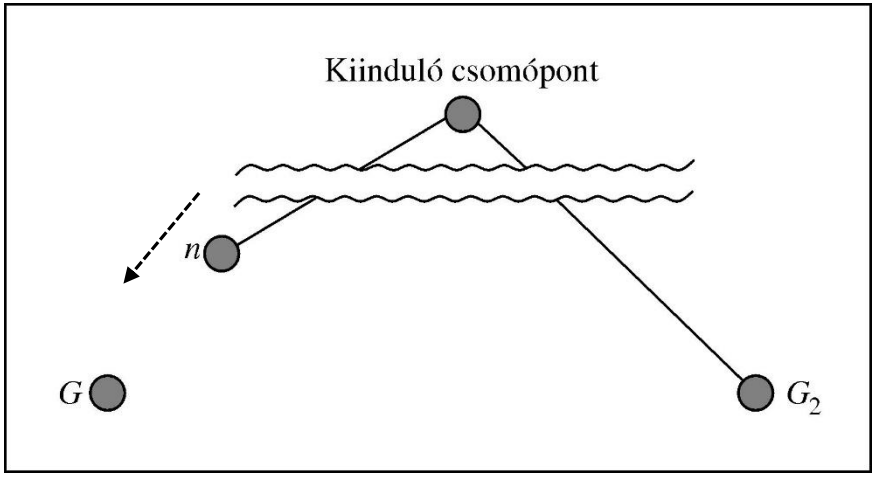
- Ha ***f*** a gyökérből kiinduló utak mentén soha sem csökken ⇒ **határvonalak** rajzolhatók

Egyenletes költségű keresés: olyan A* keresés lenne, amelynél ***h*** = 0 minden csomópontra, ezért a vizsgálandó csp-sávok a kiinduló csp köré húzott többé-kevésbé koncentrikus „körök”. Pontosabb heurisztikus függvény esetén: az A* keresésnél sávok a **cél-állapot felé nyúlnak, fókuszálódnak az optimális út körül.**



http://mialmanach.mit.bme.hu/demonstraciok/a_kereses_usa_terkepen_konturkovetessel

Az A* **teljes** és **optimális**



$f(G_2) = g(G_2) > g(G) = f^* >= f(n)$

G: optimális cél, G_2: szuboptimális cél
f*=g(G) költség,

A G fele vezető úton *n* nem lett kifejtve G2 előtt

Ha f^* az **optimális** megoldási út költsége, akkor:
- A* kifejti az összes $f(n) < f^*$ csp-t
- ezek után egy cél-csp kiválasztása előtt még kifejthet néhány csp-t a „cél-határvonalon", amelyekre $f(n) = f^*$

- Az ilyen típusú optimális algoritmusok közül az A* keresés bármely adott heurisztikus függvény mellett **optimális hatékonyságú.**

- Egyetlen optimális algoritmus sem fejt ki **garantáltan kevesebb** csomópontot az A* keresés által kifejtett csomópontnál.

Az A* **teljes** és **optimális**

A* teljessége - pontosan: az A* algoritmus **lokálisan véges gráfokon** – (véges elágazási tényező) teljes, ha létezik δ pozitív konstans, amelynél semelyik operátor költsége sem kisebb.

(**teljes**, **optimális** és az összes ilyen közül **optimálisan hatékony**)

Buktató: a legtöbb esetben a csp-tok száma a keresési tér cél-határvonalán belül a megoldás hosszának még mindig **exponenciális** függvénye.

Exponenciális - **kivéve**, ha a heurisztikus függvény hibája legfeljebb az aktuális útköltség logaritmusával nő:

$$|h(n) - h^*(n)| \leq O(\log(h^*(n)), \qquad (h^*(n) \text{ a valódi költség})$$

Az A* algoritmus komplexitása

- Majdnem minden, gyakorlati heurisztikus függvény esetén a hiba legalább arányos az út költséggel - exponenciális növekedés.

- Egy jól megválasztott heurisztikus függvény ettől függetlenül a nem informált keresési algoritmushoz képest jelentős megtakarítást eredményezhet!

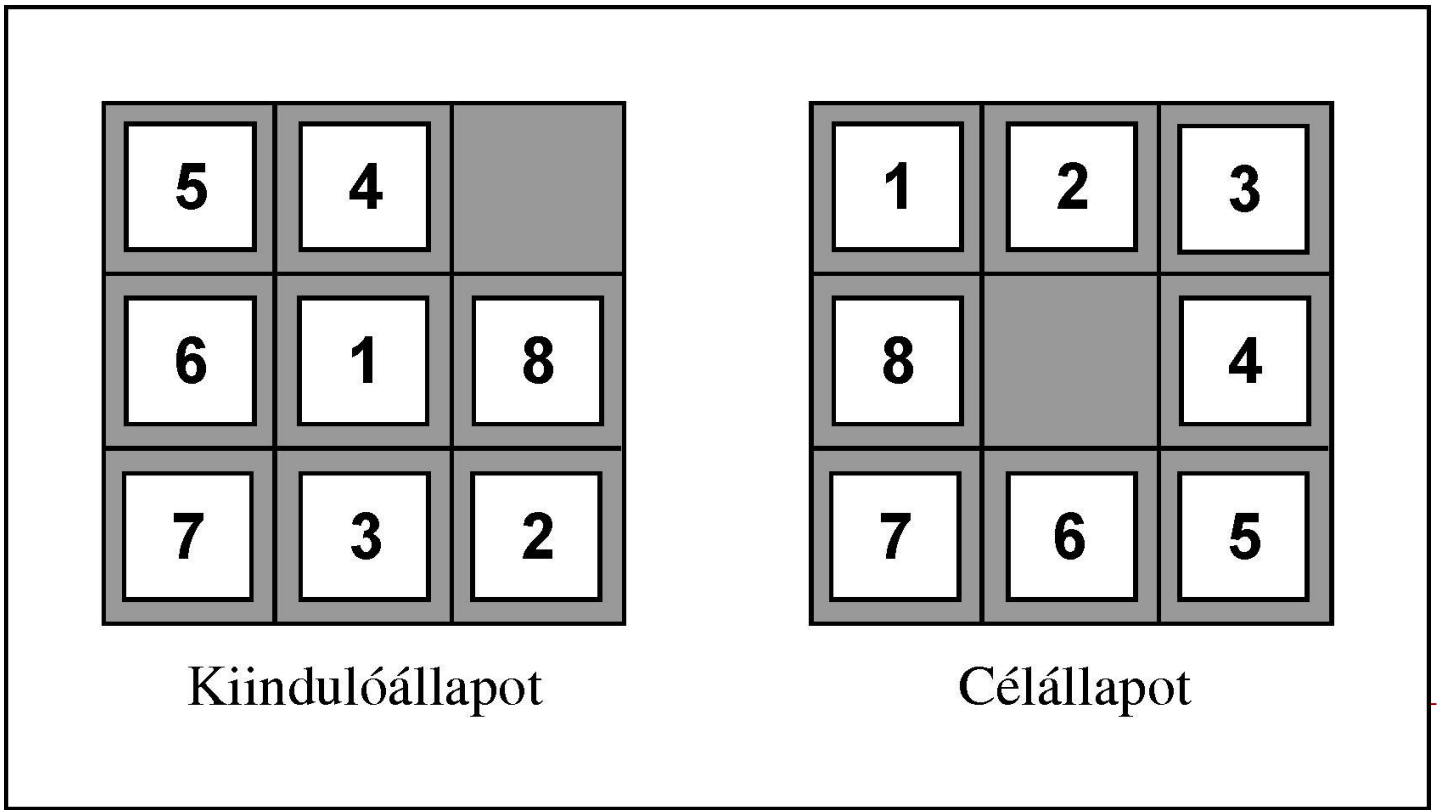
- A* algoritmus nagy problémája: az összes legenerált csomópontot eltárolja, lényegesen hamarabb felemészti a memóriát, mintsem kifutna az időből.

Heurisztikus függvények létrehozása

Nagyon jól bemutatható egy demóproblémán, 8-as tili-toli logikai játék:

- kb. 20 lépésben lehet átlagosan megoldani,
- $b \sim 3$,
- kimerítő keresés $\Rightarrow$ kb. $3^{20} = 3.5 \times 10^9$ állapot

Elfogadható heurisztikus függvény kell!

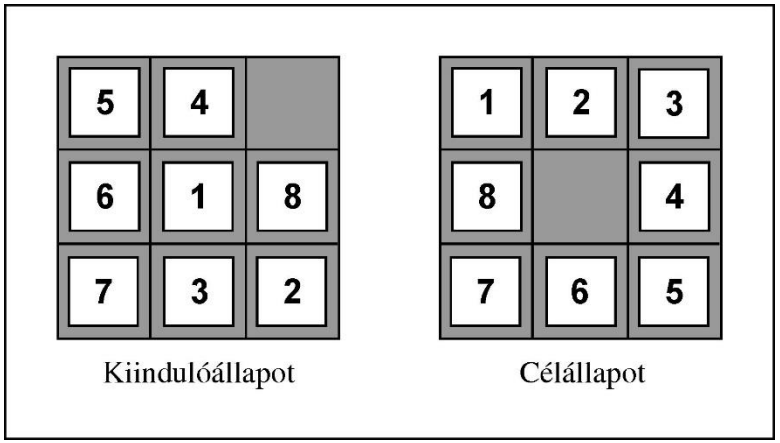


Heurisztikus függvények létrehozása (legyen sok és jó)

8-as tili-toli: kb. 20 lépés, $b \sim 3$, kimerítő keresés = kb. $3^{20} = 3.5 \times 10^9$ állapot

gyorsan és a legrövidebb megoldás?

- elfogadható heurisztikus függvény kell!



h1 = **a rossz helyen lévő lapkák száma**.

elfogadható: minden rossz helyen lévő lapkát legalább egyszer mozgatni kell

h2 = **a lapkák céltól mért** (vízsz. és függ.) **távolságainak összege: háztömb-** vagy **Manhattan-távolság**

elfogadható: minden egyes mozgatással egy lapkát csak egy (vízszintes és függőleges) lépéssel lehet közelebb vinni a célhoz.

.....

$h1 = 1 + 1 + 1 + 1 + 1 + 1 + 1 = 7$

$h2 = 2 + 3 + 3 + 2 + 4 + 2 + 0 + 2 = 18$

$h3 = ... = ... ?$

Jó lesz pl.:

h3 = (h1 + h2)/2 ?
stb.

Heurisztikus függvény pontossága és hatékonysága

A heurisztikus függvények minősítése (általában nem könnyű feladat a minősítés, a megoldások összehasonlítása – skalár mérőszám kell!): b^* **effektív elágazási tényező**

ha A* által kifejtett összes csomópont száma N, a megoldás mélysége d, akkor b^* annak a d mélységű kiegyensúlyozott fának az elágazási tényezője, amely N csomópontot tartalmazna:

$$N = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$$

(pl. 5 mélységben fekvő megoldás 52 cs-pont kifejtésével:
az effektív elágazási tényező 1.91)

$$1 + 1^1 + 1^2 + 1^3 + 1^4 + 1^5 = 6 < 52 < 63 = 1 + 2 + 2^2 + 2^3 + 2^4 + 2^5$$

Egy adott heurisztikus függvény által generált fa effektív elágazási tényezője általában nagyjából állandó egy adott problémaosztály számos egyedére.
A b^* kis számú problémahalmazon végzett kísérleti mérése jó becslés.
Egy jól megtervezett heurisztikus függvény effektív elágazási tényezője **1 körüli érték.** (Ideális lenne az 1.)

Heurisztikus függvény: pontosság és hatékonyság

a ***h1***, ***h2*** heurisztikus függvények tesztelése:

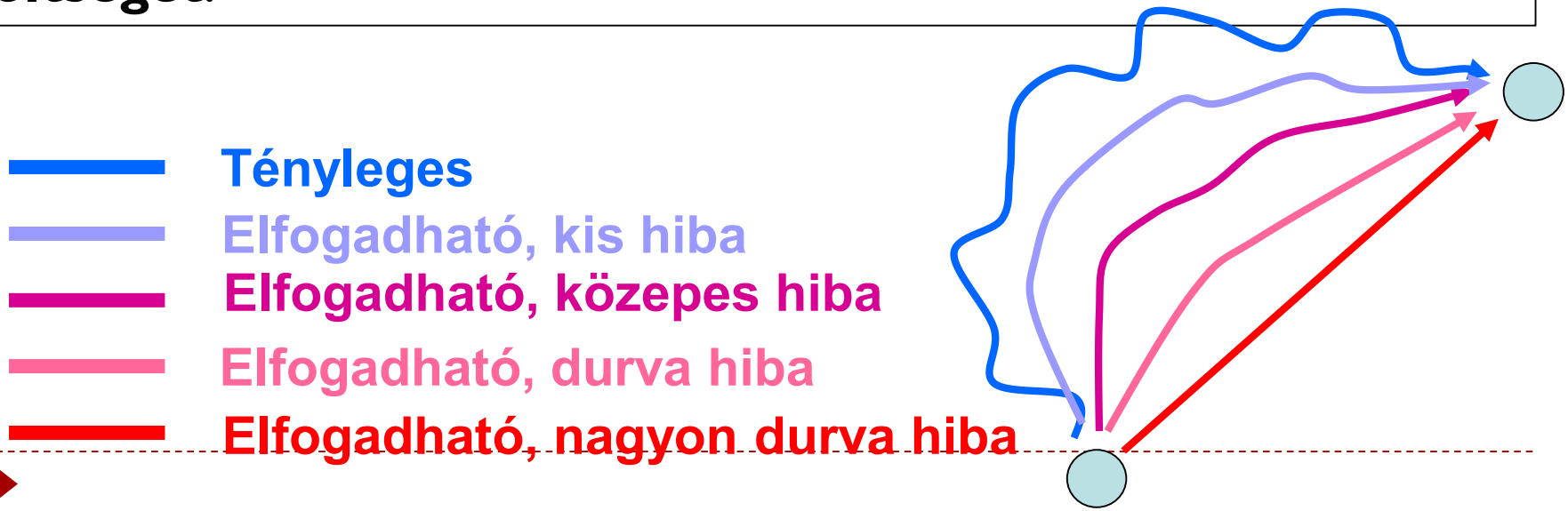
vajon a ***h2*** mindig jobb-e, mint a ***h1***? **Igen** (miért?).

minden *n* csp-ra ***h2 (n)*** $\geq$ ***h1 (n)***:

h2 **dominálja** ***h1***–et, **dominancia ⇒ hatékonyság**

a ***h2***-t használó A* *kevesebb csp-t fog kifejteni, mint a* ***h1***–et használó!

Mindig jobb, ha nagyobb értékeket adó heurisztikus függvényeket alkalmazunk, amíg nem becsüljük túl a valódi költséget.



h1 és ***h2*** tesztelése (8-as kirakójáték) : 100-100 random problémapéldány, 2, 4, …, 24 mélységű megoldással, A*, illetve a neminformált iteratívan mélyülő keresés

	Keres. ktg.			b*		
d	**IMK**	**A*(*h1*)**	**A*(*h2*)**	**IMK**	**A*(*h1*)**	**A*(*h2*)**
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	-	1301	211	-	1.45	1.25
18	-	3056	363	-	1.46	1.26
20	-	7276	676	-	1.47	1.27
22	-	18094	1219	-	1.48	1.28
24	-	39135	1641	-	1.48	1.26

(Jó) Heurisztikus függvények kitalálása

Szabály: Egy lapka az A-ról a B-re mozgatható, ha A és B szomszédok és a B mező üres.

Ha egy vagy több feltételt törlünk, akkor ún. relaxált problémákat hozhatunk létre (lazítunk - nem minden szabályt tartunk be)

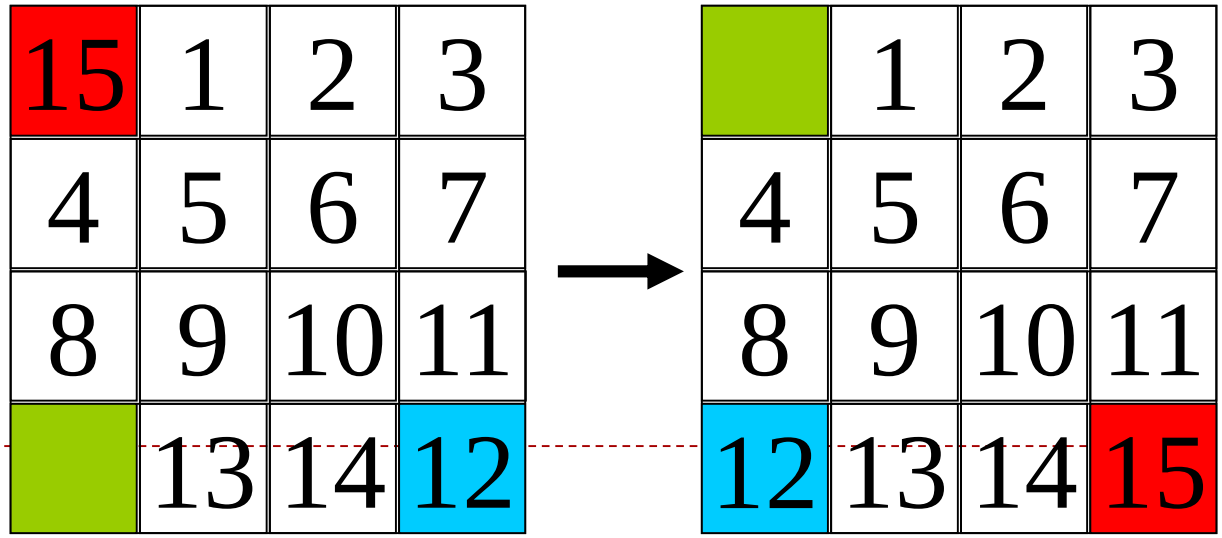
(a) Egy lapka az A-ról a B-re mozgatható, ha A és B szomszédosak (h2 betartja, h1 nem)

(b) Egy lapka az A-ról a B-re mozgatható, ha a B mező üres (sem h1 sem h2 nem törődik vele)

(c) Egy lapka az A-ról a B-re mozgatható (mindkettő betartja)

csak (a): 6 + 3 = 9 lépés

csak (c): 1 + 1 = 2 lépés



(Jó) Heurisztikus függvények kitalálása

Relaxált probléma: olyan probléma, amelyben az operátorokra kevesebb megkötést teszünk, mint az eredeti problémában.

Gyakori, hogy a relaxált probléma pontos megoldásának költsége jó heurisztikus függvény az eredeti problémára.

Gyakori, hogy – a relaxált probléma egyszerűsége miatt – az optimális megoldás a relaxált feladatra analitikusan is meghatározható.

A relaxált probléma pontos megoldása mindig egy elfogadható heurisztika a kevésbé relaxált problémára.

(Jó) Heurisztikus függvények kitalálása

Nehéz felismerni a „nyilvánvalóan legjobb” heurisztikus függvényt.

Ha egy problémához adottak a $h_1,...,h_m$ elfogadható heurisztikák

és semelyik sem dominálja a másikat, melyiket kellene választanunk?

Nem kell választanunk. A lehető legjobb:

$$h(n) = \max\{ h_1(n),...,h_m(n) \}$$

mindig azt a függvényt használja, amelyik az adott csomópontra

a legpontosabb. Mivel mind elfogadhatóak, ezért ***h*** is elfogadható lesz.

h dominálja az összes heurisztikus függvényt.

Probléma: az elemezhetőség, hiszen a konkrét h mindig változik!
(….mikor h_3 adja az eredő *h*-t, mikor h_7 stb.)

Ha a heurisztikus függvény olyan összetett, hogy értékének egy csp-ra történő meghatározása sok időt vesz igénybe, akkor baj van.

Egy jó heurisztikus függvény: pontos és hatékonyan számítható

Iteratívan Mélyülő A* keresés (IMA*)

Minden egyes iteráció egy mélységi keresés, mélységkorlát helyett egy ***f***-költség korláttal.

Minden egyes iteráció kifejti az összes - az adott ***f***-költség határvonalon belül fekvő - csomópontot, átnézve a határvonalon, hogy megtalálja, hol fekszik a következő határvonal.

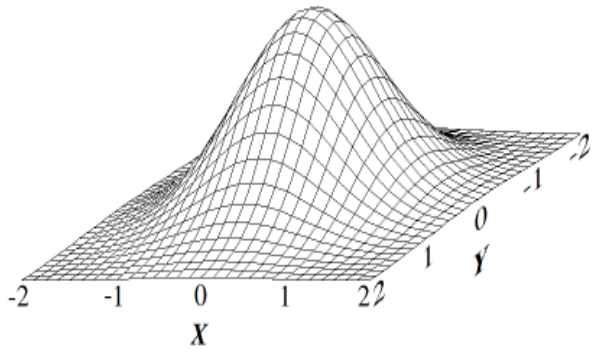
Az IMA* ugyanazokkal a kitételekkel **teljes** és **optimális**, mint az A* algoritmus, de a **mélységi keresés jellege** miatt csak a leghosszabb felderített út hosszával arányos memóriát igényel.

Lokálisan kereső algoritmusok

- Állapotok jósága: egy hiperfelület az állapotok N-dim paramétertérében.
- A felület magassága = az adott állapot optimalitása.

- A felületen a legmagasabb (legalacsonyabb) csúcsot keressük, ami egyben az optimális megoldásnak felel meg.
- A csúcs helyén a kívánt paramétertekék = a probléma megoldása.

Általában csak az aktuális állapot tárolt, és csak a közvetlen környezetében nézünk előre.

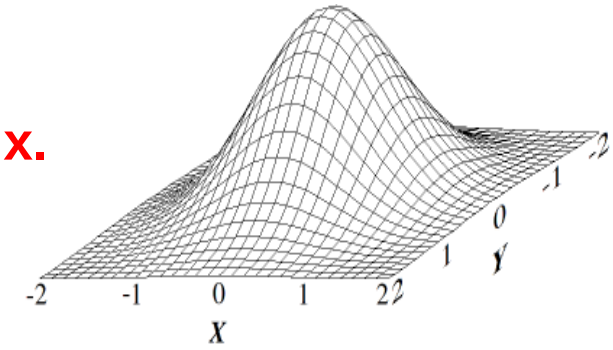


Lokálisan kereső algoritmusok

Általában csak az aktuális állapot tárolt, és csak a közvetlen környezetében nézünk előre. Tipikus probléma: optimalizálás, optimalizáló tervezés.

A keresési tér különleges ...

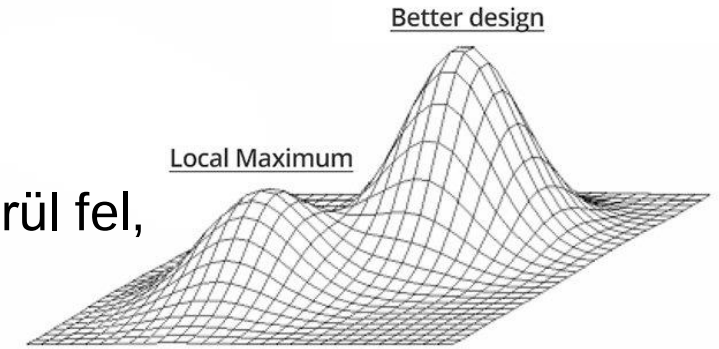
Visszalépés nincs, amiatt a memóriaigény kicsi, fix.
Az időigény lineáris.
Garancia megoldásra? (teljesség?, optimalitás?)



- **hegymászó** (diszkrét gradiens): mindig amerre javít az aktuális állapoton
- **szimulált lehűtés**: néha megenged rossz lépéseket is
- **lokális nyaláb keresés**: mindig k csomópontot tart számon
- **véletlen indítású**
- **genetikus algoritmusok**:mint a lokális nyaláb, speciális (genetikus) lépésoperátorokkal

Hegymászó keresés

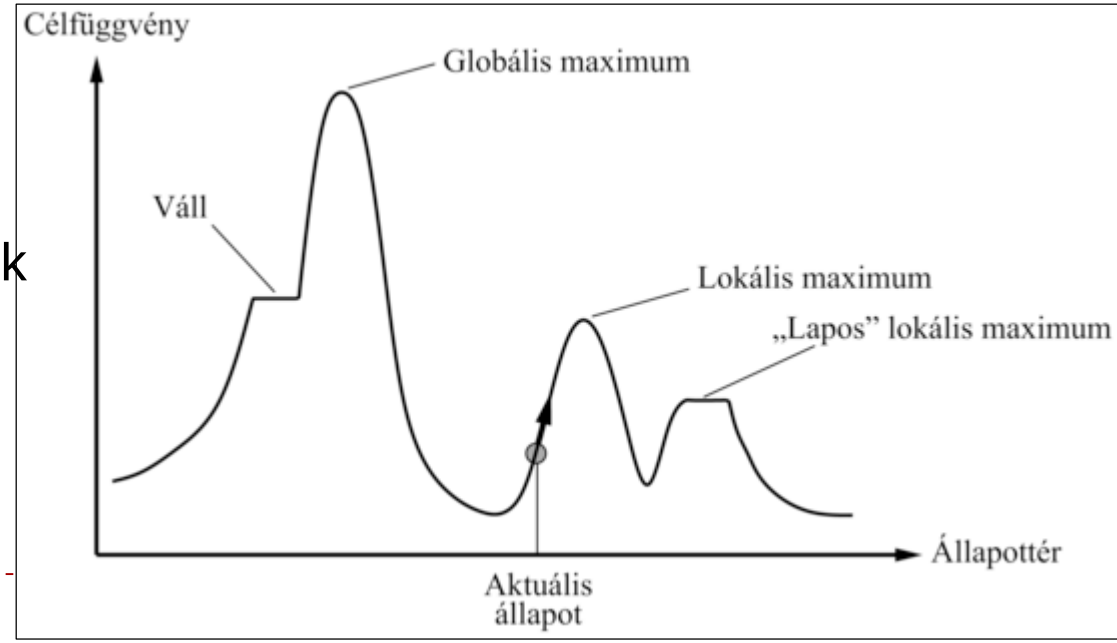
- ciklikusan lép fel mindig a javuló értékek felé
- nem tart nyilván keresési fát
- ha egynél több legjobb követő csomópont merül fel, véletlenszerűen bármelyiket kiválaszthatja.

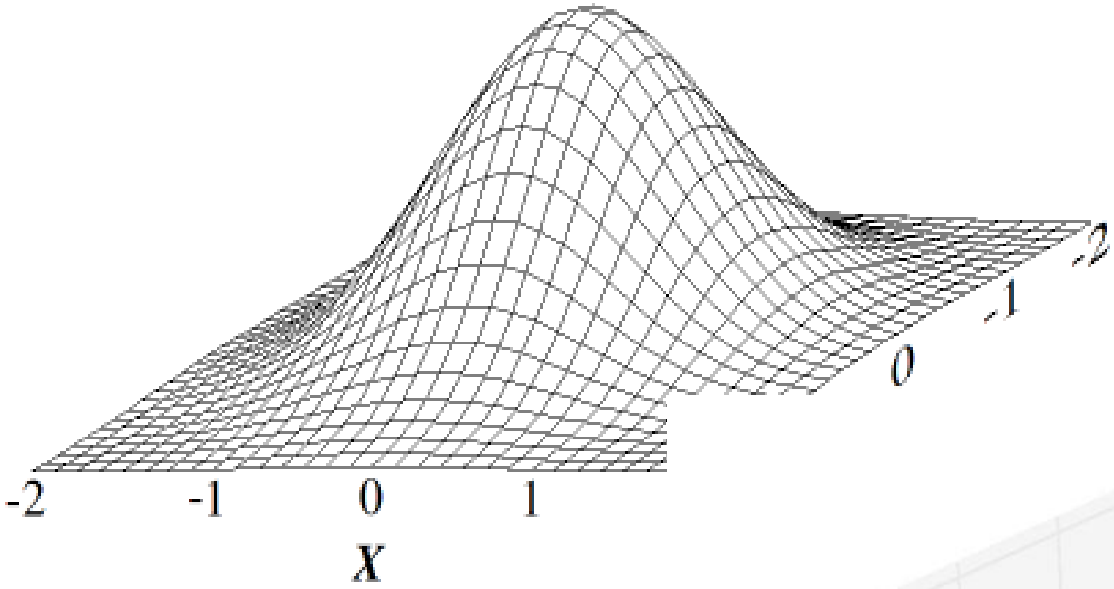


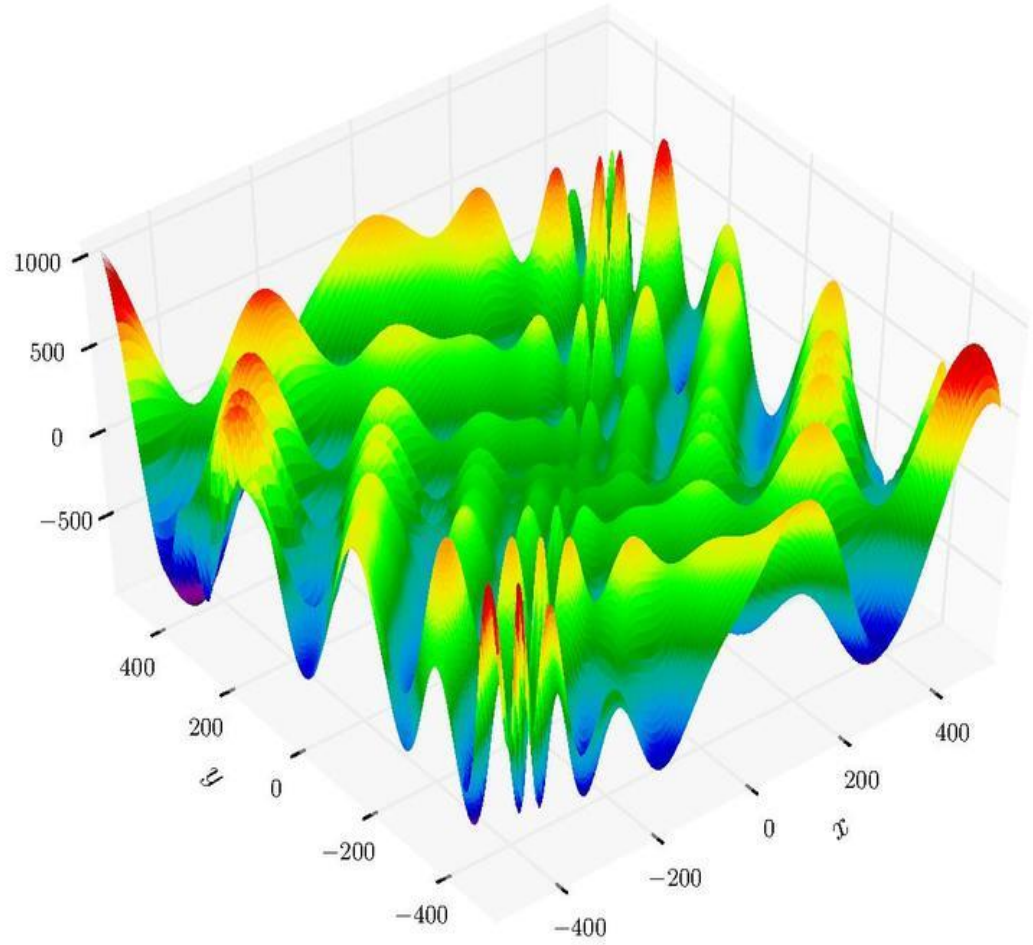
3 fő probléma:

lokális maximum: ha egy lokális maximumba ér, akkor ott megáll
fennsík: ott a kiértékelő függvény lapos, véletlen irányválasztás
hegygerinc:
egy hegygerinc oldalai meredekek, a keresés könnyen eljut a gerincre, de előfordulhat, hogy a gerinc maga csak finoman emelkedik a csúcs felé

(Sok lépés „kifárasztja” a rendszert).







Lokális szélsőérték

$f'(x) = 0$

Ha több, melyik a globális?

Véletlen újraindítású hegymászás (és sok más változat):

véletlenül generált kiinduló állapotokból hegymászó keresés,
amíg **nem jár le az idő**, vagy **már nincs észrevehető előrelépés**

a hegymászás sikere: az állapottér „**felszínének" alakjától függ**

- ha csak **néhány** lokális maximum: gyors megoldás.
- egy valódi probléma = egy „sündisznó" felszín.

Ha a probléma NP-teljes, akkor az exponenciális időigénynél
jobb nem lesz = exponenciálisan sok lokális maximum.

Általában kisszámú iteráció után már elfogadhatóan jó megoldást
lehet találni.

Szimulált lehűtés

Véletlen újraindítás helyett, ha a keresés egy lokális maximumban beragad, megengedhetjük, hogy néhány lefelé vezető lépést tegyen, hogy kimeneküljön a lokális maximumból.

A szimulált lehűtés ciklusa:

- a legjobb lépés megtétele helyett egy **véletlen lépést** tesz.
- **ha a lépés javít**, akkor az mindig végrehajtásra kerül.
- ellenkező esetben az algoritmus a lépést csak valamilyen, 1-nél kisebb **valószínűséggel** teszi meg (Boltzmann-eloszlás).

$$P \approx \exp(-\Delta E / T)$$

A valószínűség exponenciálisan csökken a lépés "rontó" képességével
– azzal a ΔE mennyiséggel, amivel a kiértékelő függvény romlott, és az idővel, a hűtés függvényében.

Szimulált lehűtés

***T*(idő) - hűtési karakterisztika**

- magas ***T***-nél a "rossz" lépések nagy valószínűséggel fordulnak elő,
- ahogy ***T*** tart nullához, a rossz lépések egyre kevésbé valószínűek, az algoritmus többé-kevésbé egy hegymászó algoritmusként viselkedik.

Hűtési karakterisztika - a hőmérséklet csökkentésének mértéke

Az egyedi lépések - termikus zaj okozta véletlen fluktuáció.

Ha a hőmérsékletet kellően lassan csökkentjük, akkor az anyag a legkisebb energiájú állapotba jut.
Ha a **hűtési karakterisztika *T***-t kellően lassan csökkentjük, az algoritmus egy globális minimumot fog találni.

Nyaláb-keresés

Lokális nyaláb keresés - k állapotot követ nyomon

Indulás: k véletlen módon generált állapot

Minden lépésben a k állapot mindegyikének összes követőit kifejti

Ha ezek valamelyike egy cél, az algoritmus leáll

Különben a teljes listából kiválasztja a legjobb k követőt,
és ezt az eljárást ismétli.

Megnövelt tár - … - megnövelt esély a „jó” extrémumra.