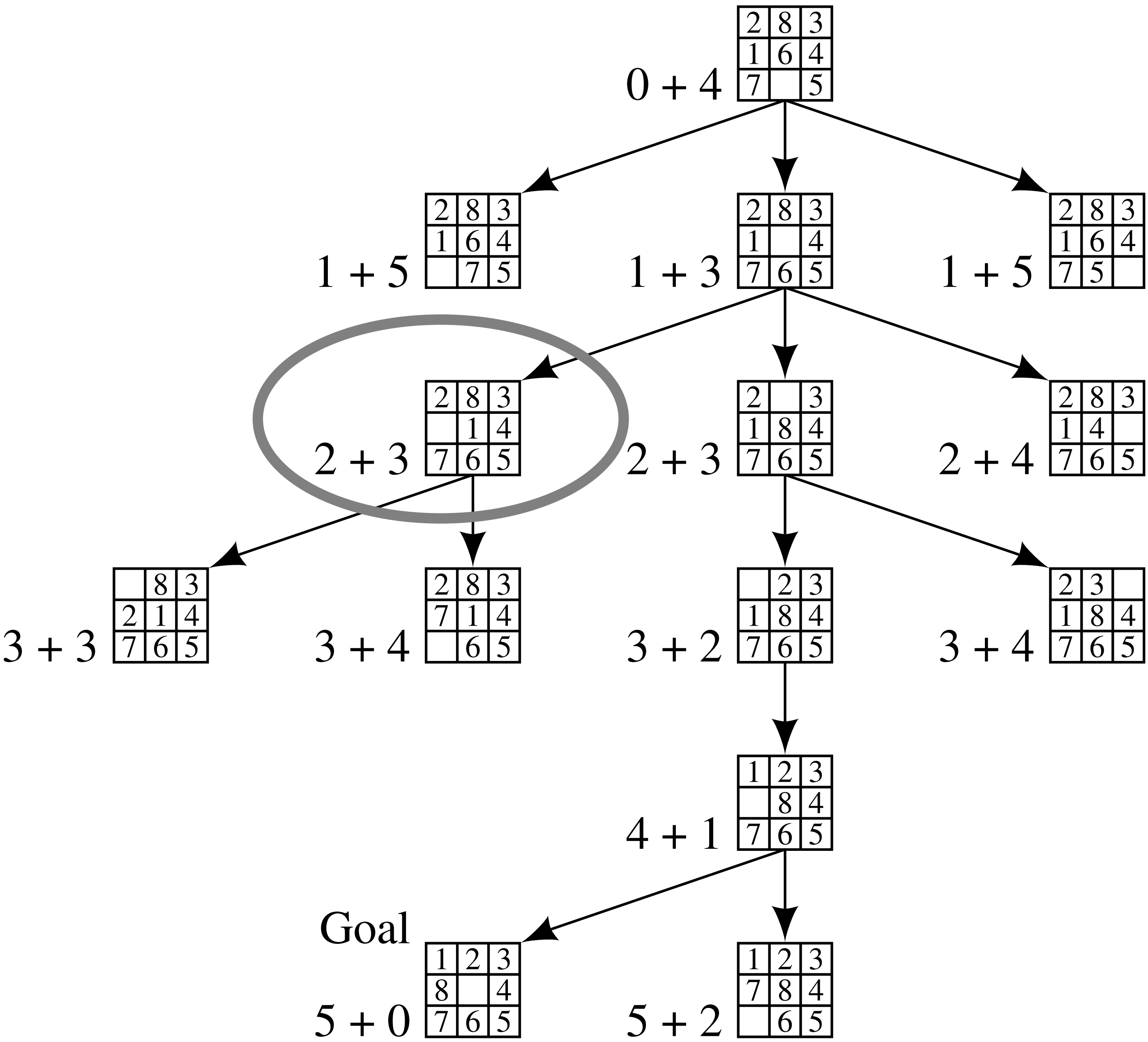
Tai jos 8-liukupelissä

$$\hat{h}(n) = \text{laudalla } n \text{ väärissä paikoissa olevien tiilten määrä}$$

niin etsintä onnistuu nopeasti (lukuunottamatta rengastettua haaraa):



© 1998 Morgan Kaufman Publishers

Kuva 31: Heuristinen etsintä 8-liukupelissä [Nil98, kuva 9.2].

- g on *leveys*suuntainen osa joka pyrkii *huolellisuuteen*.
- h on *syvyys*suuntainen osa joka pyrkii *tehokkuuteen*.

A*-algoritmi on. . .

optimaalinen: Kuvan 30 juuresta alkavilla poluilla funktion f antamat arvot *eivät vähene*.

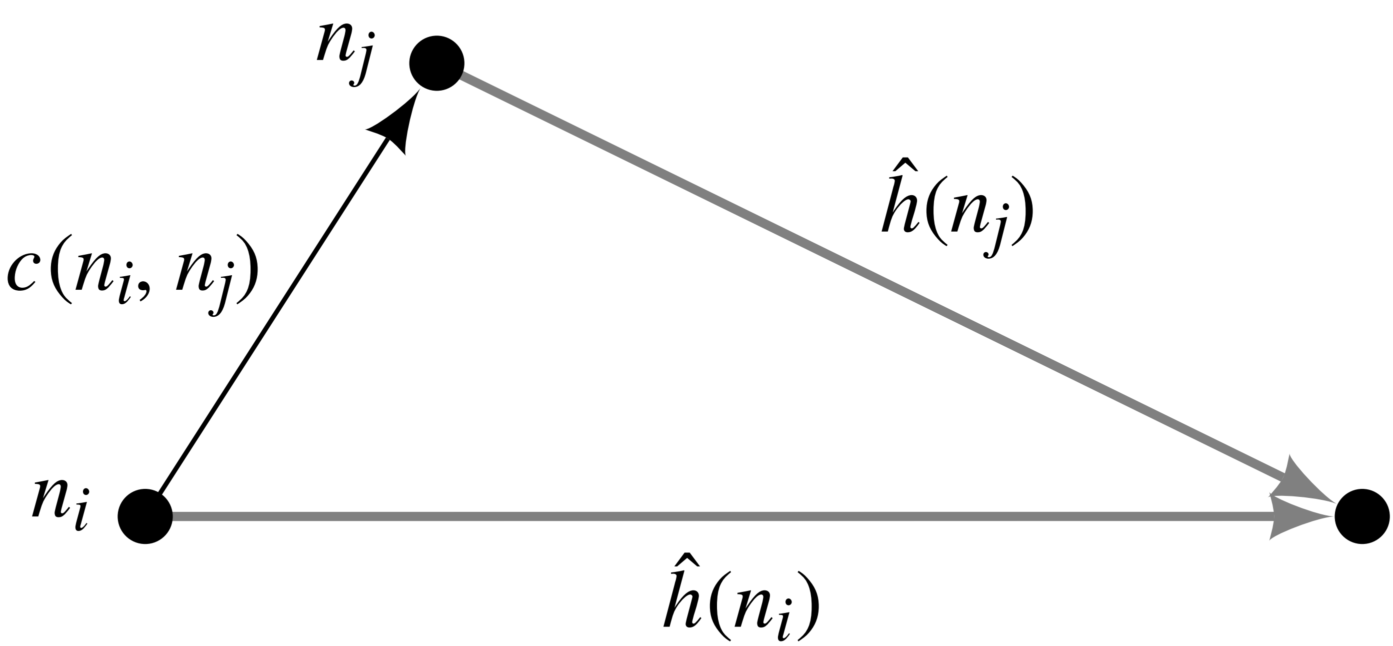
Tällaista f kutsutaan *monotoniseksi*, tarkemmin sanoen *kasvavaksi* [Myr74a, määritelmä 2.11.1].

Koska polkuhinta g oletettiin kasvavaksi ja '$+$' säilyttää kasvavuuden, riittää tarkastella heuristiikkaa h.

Funktio f on monotoninen jos ja vain jos h toteuttaa *kolmio-epäyhtälön*, joka saa nyt muodon

$$h(n) \leq k(n, n') + h(n')$$

missä $k(n, n')$ on pienin polkuhinta solmusta n solmuun n' (todistus sivuutetaan).



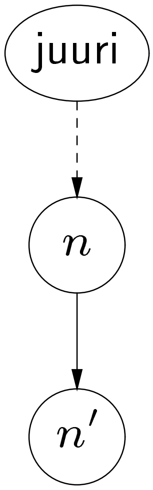
$$\hat{h}(n_i) \leq c(n_i, n_j) + \hat{h}(n_j)$$

© 1998 Morgan Kaufman Publishers

Kuva 32: Kolmioepäyhtälö [Nil98, kuva 9.7].

Esimerkiksi kuvan 15 "linnuntie-etäisyys" täyttää kolmioepäyhtälön, onhan se geometrinen etäisyysmitta.

Jos f ei olekaan monotoninen,niin etsintäpuusta löytyy haara



jossa $f(n) > f(n')$. Mutta $f(n)$ oli optimistinen arvio *kaikille* juuresta solmun n kautta maaliin kulkeville poluille — myös niille jotka kulkevat kaaren isäsolmusta n lapsisolmuun n' kautta, mutta näitähän $f(n')$ pitää vielä parempina!

Monotonisuus voidaan siis palauttaa yhtälöön (9) *pathmax-yhtälöllä*

$$f(n') = \max(f(n), g(n') + h(n')) \tag{10}$$

jonka jälkeen f on varmasti monotoninen, kunhan h on luvallinen.

Oletetaan siis f monotoniseksi. Silloin optimaalisuus voidaan näyttää seuraavasti:

Algoritmin voidaan nyt ajatella piirtävän sisäkkäisiä *korkeuskäyriä* etsintäavaruuteen (kuva 33).

Luvun 3.1.2 etsintäalgoritmissa (jossa $h = 0$) nämä käyrät ovat renkaita alkutilan ympärillä, kun taas paremmat heuristiikat "venyttävät" niitä soikioiksi kohden maalitilaa.

Olkoon f^* parhaan ratkaisun polkuhinta. Silloin algoritmi laajentaa

- ensin kaikki solmut n käyrän f^* sisällä (eli $f(n) < f^*$) funktion f arvojen mukaan kasvavassa järjestyksessä, ja
- sitten osan solmuista itse käyrällä f^* (eli $f(n) = f^*$),

kunnes itse käyrältä f^* löytyy (jokin) maalisolmu.

Formaalisti: olkoon G optimaalinen maalisolmu, ja olkoon G_2 jokin toinen ei-optimaalinen maalisolmu, jolla siis

$$f(G_2) > f^*. \tag{11}$$

Voisiko algoritmi vastata G_2? Silloinhan se *ei* olisikaan optimaalinen! Olkoon n juuresta solmuun G vievän optimaalisen polun P tämänhetkinen lehti; jos sellaista ei ole, niin P on jo käsitelty ja G vastattu. (Kuva 34 näyttää tilanteen.)

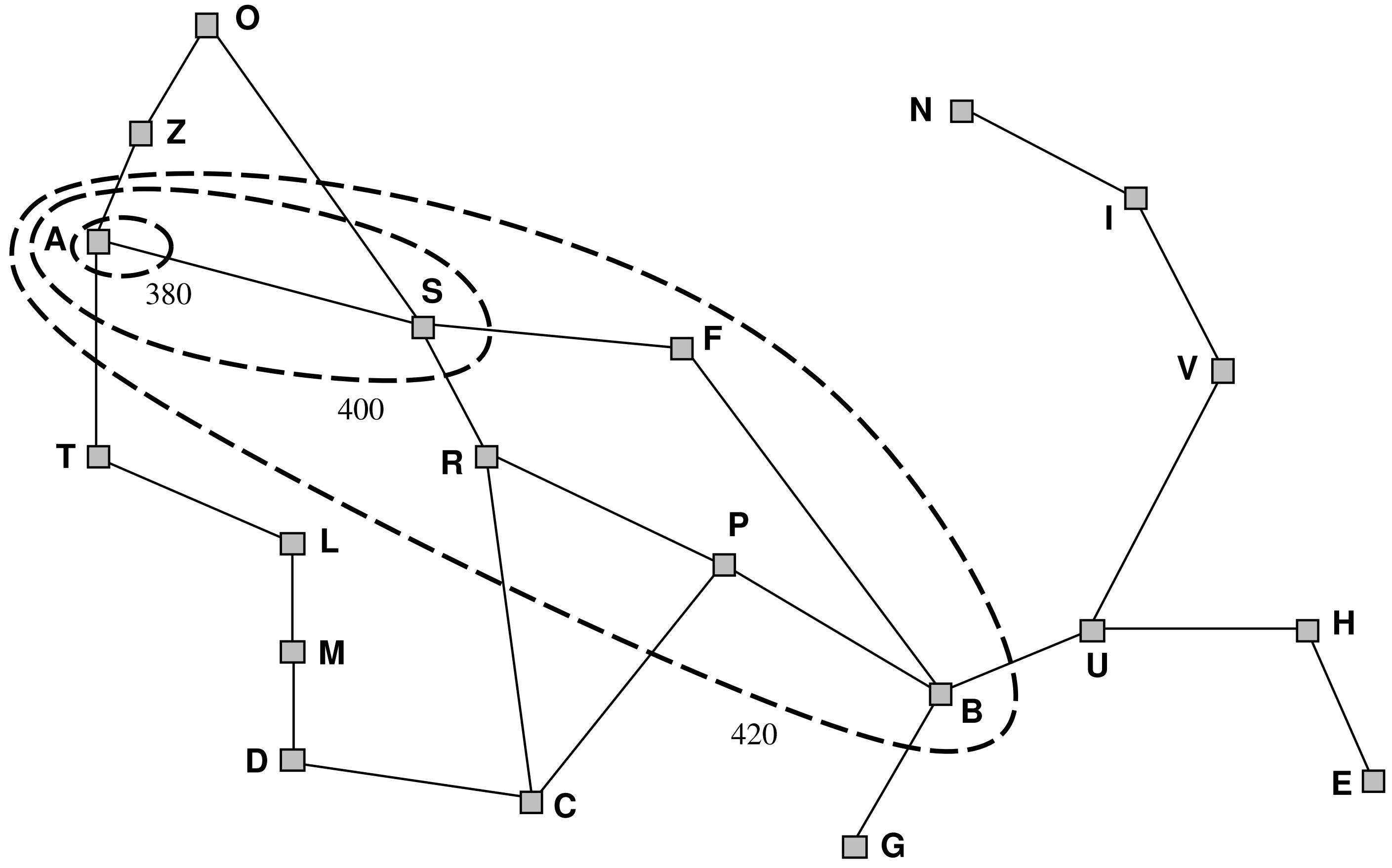
$$\begin{aligned} f^* &= f(G) \text{ koska } G \text{ on optimaalinen.} \\ &\geq f(n) \text{ koska } f \text{ on monotoninen.} \\ &\geq f(G_2) \text{ koska } G_2 \text{ valittiin ennen kuin } n. \\ &= g(G_2) + h(G_2) \text{ yhtälöstä (9).} \\ &= g(G_2) \text{ koska } h \text{ on luvallisena } 0 \text{ kaikille maaleille.} \end{aligned}$$

Tämä on kuitenkin ristiriidassa yhtälön (11) kanssa, joten algoritmi on optimaalinen.

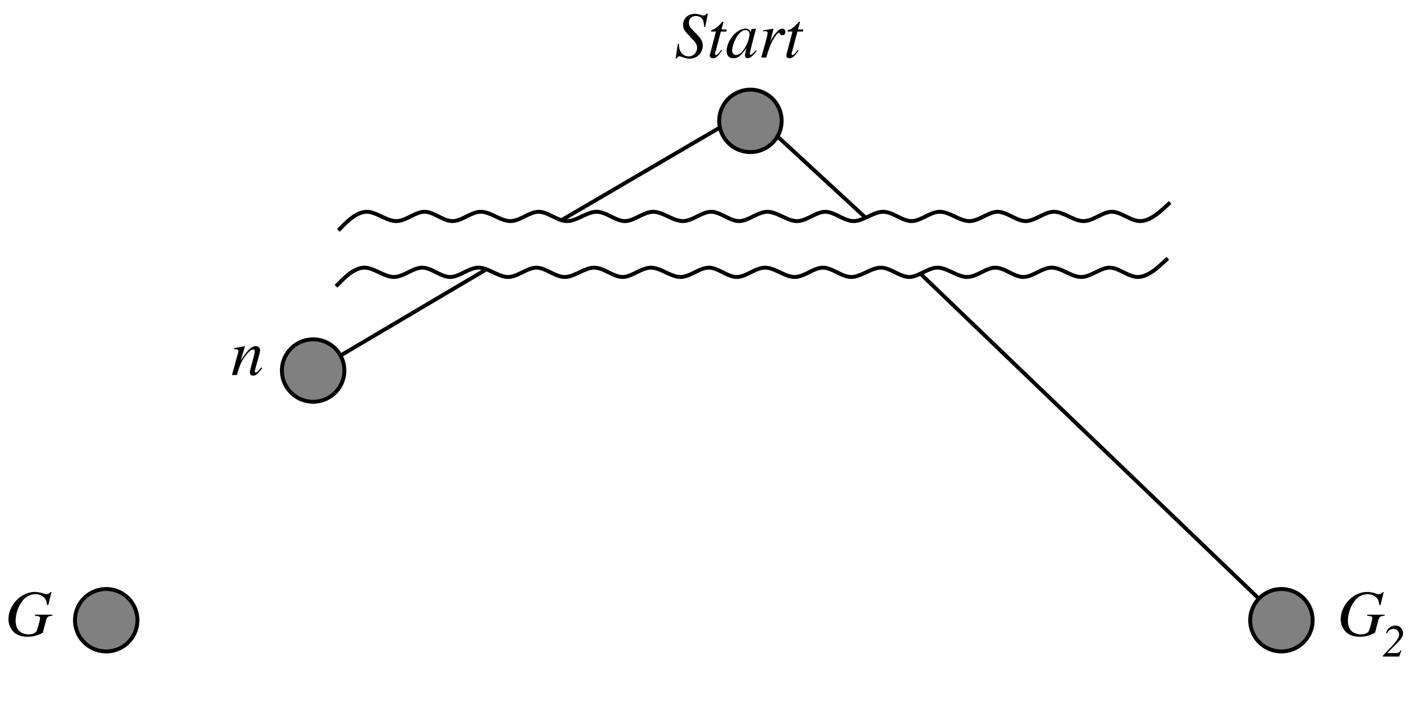
täydellinen: laajentamisjärjestyksensä vuoksi algoritmi löytää maalisolmun *ellei* käyrälle f^* tai sen sisälle jää äärettömän monta solmua, jotka algoritmi siis yrittäisi käsitellä ensin. Tämä on puolestaan mahdollista vain jos etsintäavaruudessa on. . .

- solmu, jolla on ääretön haarautumisaste.
  Tällöin etsintää tekevällä toimijalla olisi äärettömän monta eri vaihtoehtoa vastaavassa tilanteessa.
- ääretön polku $P = n_0 \xrightarrow{c_1} n_1 \xrightarrow{c_2} n_2 \xrightarrow{c_3} n_3 \cdots$ jonka hinta $g(P) = c_1 + c_2 + c_3 + \cdots \leq f^*$ pysyy äärellisenä.
  Tällöin etsintää tekevällä toimijalla olisi tarjolla mielivaltaisen "halpoja" tekoja.
  Tästä ehdosta mainittiin jo luvussa 3.1.4.

Algoritmi on siis täydellinen sellaisissa avaruuksissa joissa *haarautumisaste on äärellinen* ja *kaikkien tekojen kustannuksilla on jokin alaraja* $\delta \in \mathbb{R}$.



Kuva 33: Korkeuskäyrät Aradista Bukarestiin [RN95, kuva 4.5].



Kuva 34: Algoritmin **A*** optimaalisuus [RN95, kuva 4.6].

tehokas: algoritmi on *optimaalisen tehokas (optimally efficient)* seuraavassa mielessä:

Olkoon $\mathbf{B}^*$ minkä tahansa kilpaileva etsintäalgoritmi, joka. . .

- käyttää samaa heuristista ohjausinformaatiota h kuin $\mathbf{A}^*$,
- on optimaalinen kuten $\mathbf{A}^*$, ja
- etenee samoin kuten $\mathbf{A}^*$ laajentamalla polkuja juuresta alkaen.

Silloin on väistämättä olemassa sellaisia etsintäongelmia, joissa $\mathbf{B}^*$ *joutuu laajentamaan ainakin yhtä monta solmua kuin* $\mathbf{A}^*$:

Muuten $\mathbf{B}^*$ jättäisi laajentamatta solmuja käyrältä f^* tai sen sisältä, ja saattaisi näin jättää optimipolun laajentamatta. (Todistus sivuutetaan.)

Toisaalta käyrälle f^* ja sen sisälle jää yleensä eksponentiaalisen monta solmua verrattuna optimipolun solmujen määrään, joten $\mathbf{A}^*$ on huolellisuutensa vuoksi yleensä *eksponentiaalinen*.

Tämä eksponentiaalisuus voidaan varmasti välttää vain jos

$$\left|h(n) - h^*(n)\right| \leq O(\log h^*(n))$$

eli jos heuristiikka h on niin tarkka, että sen tuottama *virhe on aina korkeintaan logaritminen*. (Todistus sivuutetaan.)

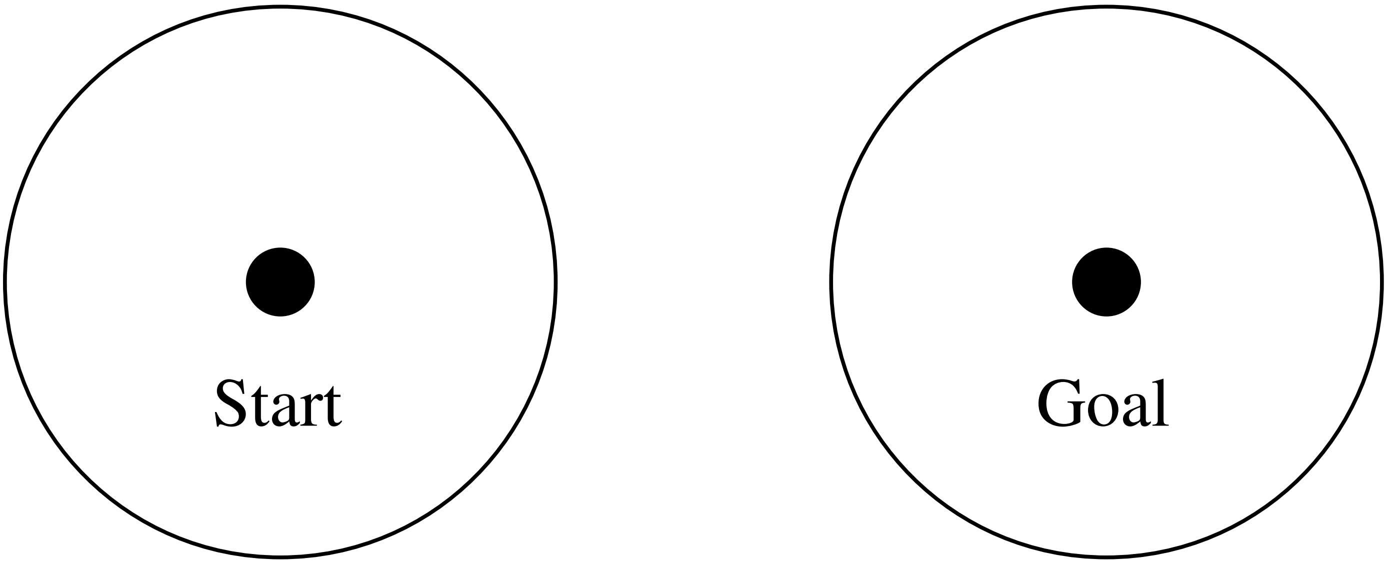
Käytännössä virhe tapaa olla ainakin luokkaa $g(n)$.

Algoritmin $\mathbf{A}^*$ suurempi ongelma on kuitenkin (jälleen) *muistintarve*, koska se pitää kaikki (lehti)solmut muistissa. Tähän ongelmaan palataan luvussa 3.2.4.

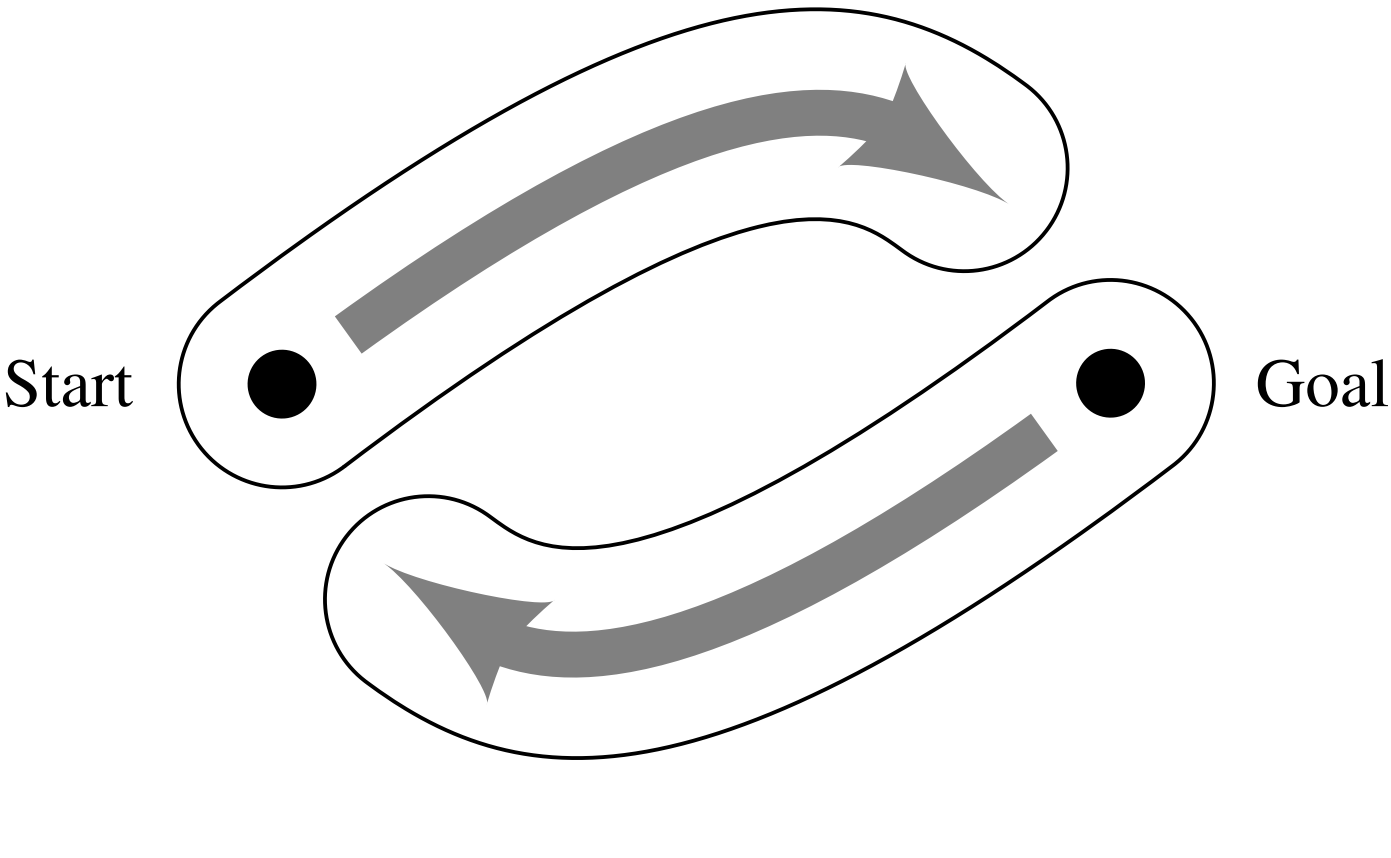
Jos heuristista etsintää (esimerkiksi $\mathbf{A}^*$) käyttää luvun 3.1.6 kaksisuuntaisessa etsinnässä, voivat eri suuntiin etenevät, yksinään hyvät heuristiikat kumota toisensa [Nil98, kuva 9.10] (sivulla 86):

Kumpikin etenee omaan hyvään suuntaansa, ja ohittaa toisen. Etsintätyö ei putoakaan murto-osaan, vaan kaksinkertaistuu, koska kumpikin suunta päätyy tekemään työn kokonaan yksin, eikä puoliksi vastasuunnan kanssa.

Tästä syystä suositellaankin, että toinen kaksisuuntaisista etsintäalgoritmeista olisikin leveyshaku eikä heuristiikka.



(a) Breadth-first search



(b) Heuristic search

© 1998 Morgan Kaufman Publishers

Entä jos heuristiikkamme h *ei* olekaan luvallinen?

- Silloin $h(n)$ voikin arvioida solmusta n lähimpään ratkaisuun vievää hintaa $h^*(n)$ *pessimistisesti* ylöspäin (eli $h(n) > h^*(n)$).
- Silloin saattaakin olla olemassa jokin toinen solmu n' jolla ylöspäin pyöristystä ei tapahdukaan (eli $h(n') \leq h^*(n)$).
- Silloin saattaakin olla

$$\underbrace{g(n') + h(n')}_{f(n')} < \underbrace{g(n) + h(n)}_{f(n)}$$

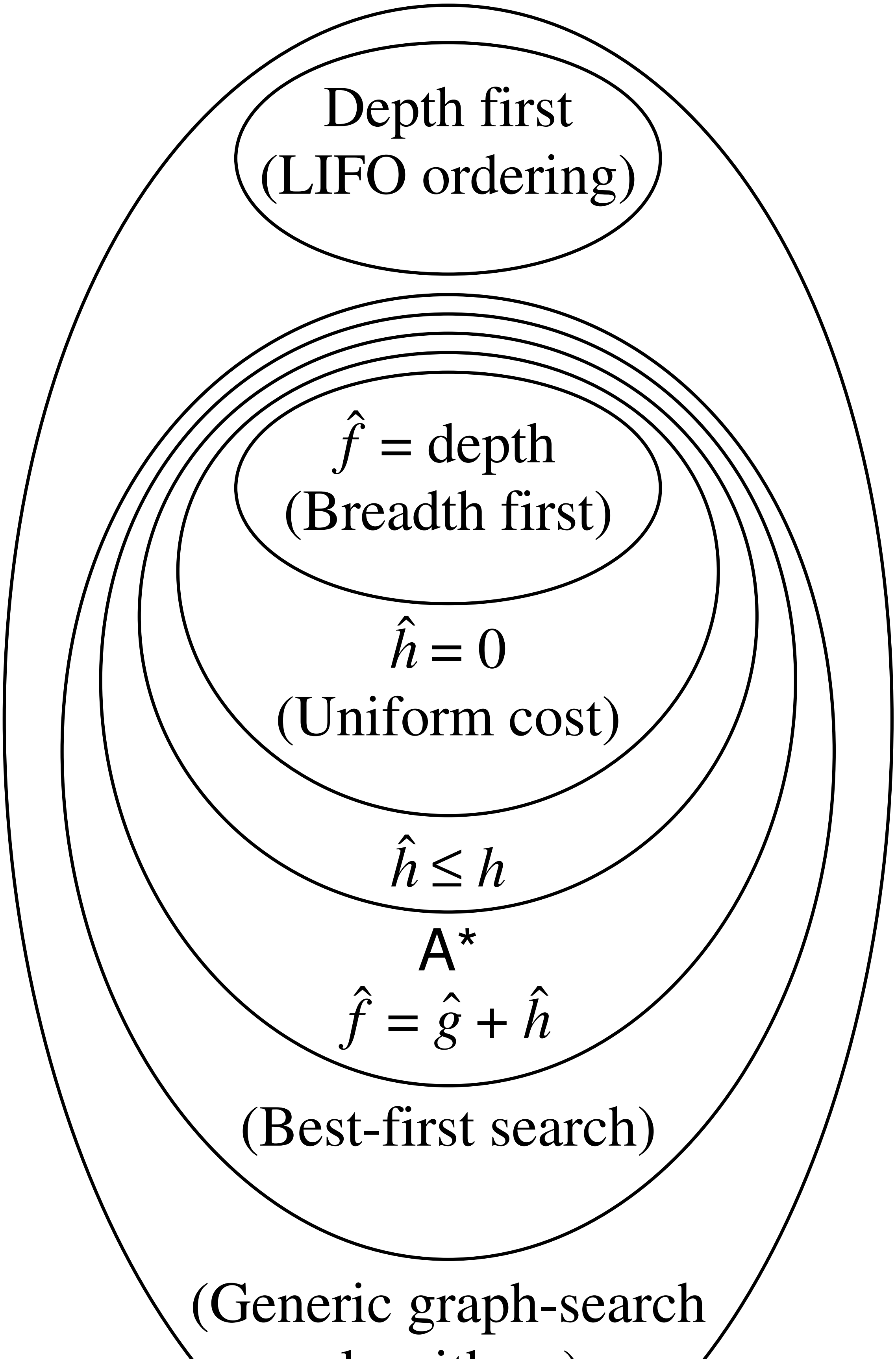
vaikka todellisille hinnoille

$$g(n') + h^*(n') > g(n) + h^*(n).$$

- Silloin etsintä valitseekin todellisuudessa huonomman solmun n' eikä parempaa solmua n.
- Siten etsintä voi päätyä tulokseen, joka ei olekaan halvin mahdollinen — enää etsintä *ei ole optimaalinen*.

Sivun 88 kuva [Nil98, kuva 9.6] kertaa eri etsintämenetelmien välisiä suhteita seuraavin merkinnöin:

[Nil98]	[RN95]
$\hat{h}$	h
h	h^*
$\hat{f}$	f
$\hat{g}$	g



© 1998 Morgan Kaufman Publishers

3.2.3. Miten tutkia heuristisia funktioita?

Tarkastellaan kuvan 20 8-liukupeliä. Luvallisia heuristiikkoja ovat esimerkiksi

$$h_1(n) = \hat{h}(n) \text{ luvusta 3.2.2}$$

$$\begin{aligned} h_2(n) &= \sum_{1 \leq i \leq 8} \text{tiilen } i \textit{ Manhattan-etäisyys} \text{ oikeasta paikastaan} \\ &= 2+3+3+2+4+2+0+2 \text{ kuvan 20 lähtötilassa.} \end{aligned}$$

Eräs heuristiikan h hyvyyden mitta on sen aikaansaama *todellinen haarautumisaste (effective branching factor)* b^*: Lasketaan ensin

- montako solmua N algoritmi **A*** laajentaa heuristiikalla h ja
- näin löydetyn ratkaisun syvyys d.

Ratkaistaan sitten b^* yhtälöstä (7)

$$N = \frac{(b^*)^{d+1} - 1}{b^* - 1}.$$

Tulos kertoo

- millainen haarautumisaste (reaalilukuna) olisi saman kokoisella *täydellisellä* puulla, eli
- kuinka leveää luvun 3.1.1 *leveyssuuntaista* etsintää **A*** vastaa heuristiikalla h.

Tulos mittaa heuristiikan h laatua koska

- leveyshaku oli hidas mutta huolellinen menetelmä joka ei huomioinut mitään lisäinformaatiota, joten

- mittausarvo kertoo kuinka kapeaksi heuristiikka pystyi kaventamaan leveyshaun, eli
- kuinka monta sellaista vaihtoehtoa yhdessä etsintäaskeleessa on, joista pitäisi valita "umpimähkään".

  Pienempi mittausarvo on siis parempi heuristiikka!

Jos esimerkiksi

$$N = 52 \text{ ja}$$
$$d = 5 \text{ niin}$$
$$b^* \approx 1.91$$

eli

- hakupuussa on $d = 5$ sukupolvea isäsolmuja, joista kullakin on (keskimäärin) $b^* = 1.91$ lapsisolmua, joten
- 9 etsintäaskeleen jälkeen on noin 9 tutkimatonta "yhtä hyvää" vaihtoehtoa, binääripuussa taas 10.

Taulukossa 2 verrataan mitalla b^* heuristiikkoja h_1 ja h_2 toisiinsa ja luvun 3.1.5 iteratiiviseen syventämiseen.

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
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

Taulukko 2: Heuristiikkojen vertailua [RN95, kuva 4.8].