Note om A^*-algoritmen

Olav Geil

(written in handtex)

Vi betragter vægtede sammenhængende grafer $G(V,E)$. Lad v_s og v_e være punkter i V. Vi ønsker at finde en korteste vej fra v_s til v_e. For hvert punkt $v_i \in V$ knyttes det ikke-negative tal $h(v_i)$. Tallet $h(v_i)$ er vores gæt på afstanden fra v_i til v_e (altså vores gæt på længden af en korteste vej fra v_i til v_e). Funktionen $h : V \to \{c \in \mathbb{R} \mid c \text{ er ikke negativ}\}$ kaldes en heuristisk funktion.

Definition

En heuristisk funktion h på V kaldes tilladelig (admissible) hvis den aldrig overestimerer. Dvs. $h(v_i) \leq d(v_i, v_e)$ for alle $v_i \in V$.

Definition

En heuristisk funktion kaldes monoton hvis der for alle par af naboer v_i, v_j i grafen G gælder

$$h(v_i) - h(v_j) \leq w(v_i v_j)$$

Proposition

En monoton heuristiske funktion med $h(v_e)=0$ er tilladelig (admissible)

Bevis:

Lad $P: v_{i_1} v_{i_2} \cdots v_{i_n} = v_e$ være en korteste vej fra v_{i_1} til v_e.

$$w(P) = w(v_{i_1}, v_{i_2}) + w(v_{i_2}, v_{i_3}) + \cdots + w(v_{i_{n-1}}, v_{i_n})$$

$$\geq h(v_{i_1}) - h(v_{i_2}) + h(v_{i_2}) - h(v_{i_3}) + \cdots + h(v_{i_{n-1}}) - h(v_{i_n})$$

$$= h(v_{i_1}) - h(v_e) = h(v_{i_1})$$

og den heuristiske funktion er altså tilladelig

$\square$

Eks 1

Søger korteste vej mellem byerne Arad og Bucharest. For alle byer v_i på vores kort vælges $h(v_i)$ til at være afstanden i fugleflugtslinée fra v_i til Bucharest. Vi sætter naturligvis $v_s = \text{Arad}$ og $v_e = \text{Bucharest}$.

Ovenstående heuristiske funktion er oplagt tilladelig – men er den også monoton? Som vi straks skal se er svaret ja.

Lad v_i og v_j være nabobyer. Den sædvanlige trekantsulighed siger, at fugleflugtsafstanden fra v_i til Bucharest er kortere eller lig fugleflugtsafstanden fra v_i til v_j plus fugleflugtsafstanden fra v_j til Bucharest. Men fugleflugtsafstanden fra v_i til Bucharest er $h(v_i)$, den faktiske afstand $w(v_i, v_j)$ er ej mindre en fugleflugtsafstanden fra v_i til v_j og fugleflugtsafstanden fra v_j til Bucharest er $h(v_j)$. Så

$$h(v_i) \leq w(v_i, v_j) + h(v_j)$$

$$\Downarrow$$

$$w(v_i, v_j) \leq h(v_j) - h(v_i)$$

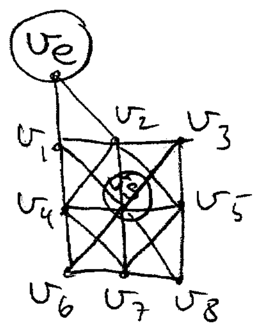
og h er monoton.

Eks 2

Dette eksempel er hentet fra [Rab] p. 107–108.

v_e		
v_1	v_2	v_3
v_4	v_s	v_5
v_6	v_7	v_8

svarende til



Hvis afstandene vertikalt hhv. horisontalt er 1 ($w(v_6, v_7) = 1$, $w(v_5, v_2) = 1$ osv) så er afstanden på skrå lig $\sqrt{2}$.

($w(v_4, v_2) = \sqrt{2}$ osv.)

Manhattan-afstanden som beskrives i [Ra] er defineret på følgende måde. Hvis man kan komme fra v_i til v_e ved at rykke a gang til venstre (eller højre) og b gange op (eller ned) så er $h(v_i) = a + b$. Bemærk at Manhattan-afstanden er meget let at beregne.

Men Manhattan-afstanden er ikke monoton. For $h(v_e) = 0$, $h(v_2) = 1+1 = 2$ men $w(v_2, v_e) = \sqrt{2} < 2$.

Manhattanafstanden kan laves monoton ved at ændre reglen til $h(v_i) = a \cdot c + b \cdot c$, hvor c er et eller andet fastholdt tal, $c \in]0; \frac{\sqrt{2}}{2}]$.

Så $h(v_i) = a \cdot 0.7 + b \cdot 0.7$ ville være et udmærket bud på en monoton Manhattan-lignende heuristisk funktion.

A^*-algoritmen fra [Rabin] lettere modificeret.

Følgende algoritme finder en korteste vej fra v_s til v_e forudsat at en sådan findes og forudsat at heurestikken er monoton og tilladelig (admissible).

Algoritme

Input: Vægtet graf G samt punkter v_s og $v_e \in V(G)$. Heuristisk funktion h.

Output: En vej fra v_s til v_e eller "vej kan ej findes".

Step 1 (Initialisering):

$g(v_s) = 0, f(v_s) = g(v_s) + h(v_s)$, $Open = \{v_s\}$, $Closed = \emptyset$

Step 2:

If $Open = \emptyset$ then

begin

Returner "vej kan ej findes" og Quit.

end

If der findes elementer i $Open$ then

begin

Lad B være et punkt i $Open$ med mindst mulig f-værdi.

If $B = v_e$ then

begin

Lad P være vejen givet som følger:

$P: \; v_e, Parent(v_e), Parent(Parent(v_e)), \ldots, Parent(\cdots(Parent(v_e))\cdots) = v_s$

Returner vejen modsat P fra v_s til v_e og Quit.

end

end

Step 3: Lad Successor være liste med alle naboer til B. For alle C i Successor:

begin

If $C \in Open$

begin

beregn $g'(C) = g(B) + w(BC)$.

If $g'(C) < g(C)$

begin

Opdater $g(C)$, $f(C)$ og $Parent(C)$ som følger:

$g(C) = g'(C)$, $f(C) = g'(C) + h(C)$ og $Parent(C) = B$.

end

end

If hverken $C \in Closed$ eller $C \in Open$

begin

Beregn $g(C) = g(B) + w(BC)$, $f(C) = g(C) + h(C)$. Sæt $Parent(C) = B$.

Tilføj C til $Open$

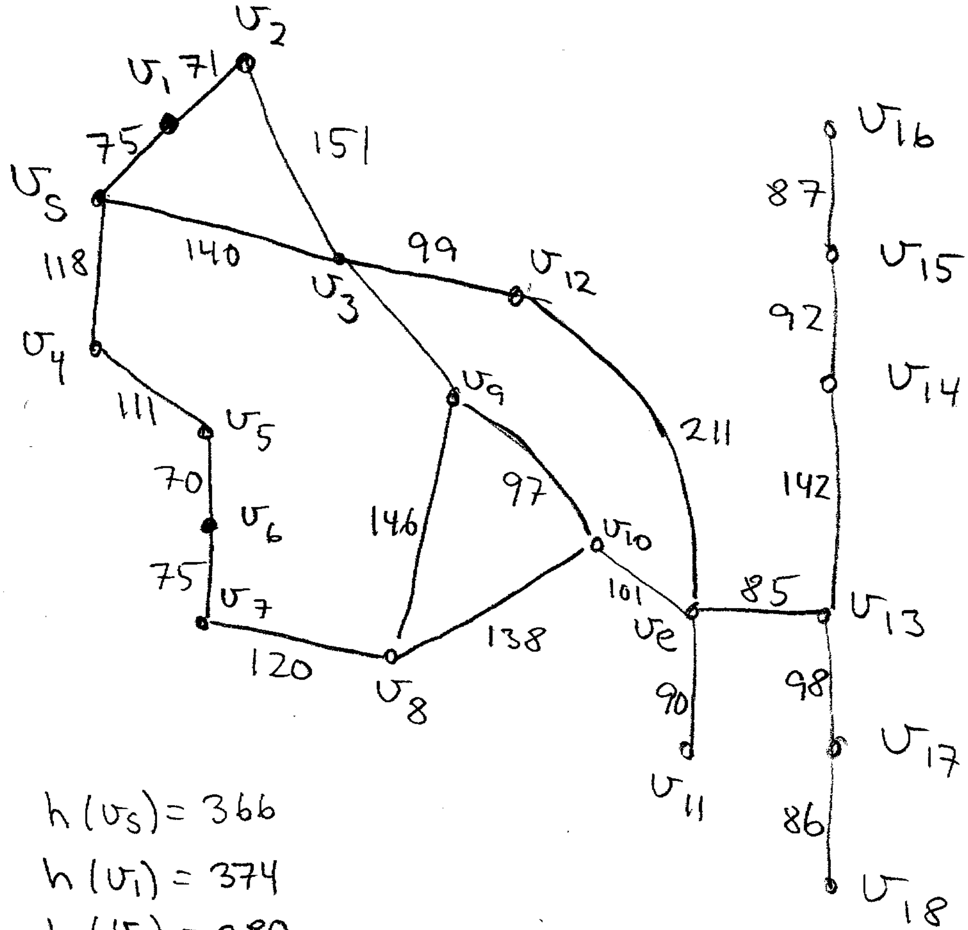
end

end

Step 4:

Flyt B fra $Open$ til $Closed$. Gå til Step 2.

Eks 3



$h(v_s) = 366$

$h(v_1) = 374$

$h(v_2) = 380$

$h(v_3) = 253$

$h(v_4) = 329$

$h(v_5) = 244$

$h(v_6) = 241$

$h(v_7) = 242$

$h(v_8) = 160$

$h(v_9) = 193$

$h(v_{10}) = 100$

$h(v_{11}) = 77$

$h(v_{12}) = 176$

$h(v_{13}) = 80$

$h(v_{14}) = 199$

$h(v_{15}) = 226$

$h(v_{16}) = 234$

$h(v_{17}) = 151$

$h(v_{18}) = 161$

$h(v_e) = 0$

v_s $g(v_s) = \underline{0}$ $h(v_s) = 366$ $f(v_s) = \underline{366}$

open = $\{v_s\}$

$B = v_s$

Successors = $\{v_1, v_3, v_4\}$

v_1 $g(v_1) = \underline{75}$ $h(v_1) = 374$ $f(v_1) = \underline{449}$

parent $(v_1) = v_s$

v_3 $g(v_3) = \underline{140}$ $h(v_3) = 253$ $f(v_3) = \underline{393}$

parent $(v_3) = v_s$

v_4 $g(v_4) = \underline{118}$ $h(v_4) = 329$ $f(v_4) = \underline{447}$

parent $(v_4) = v_s$

Open = $\{v_1, v_3, v_4\}$

closed = $\{v_s\}$

$B = v_3$

Successors = $\{v_s, v_2, v_9, v_{12}\}$

$v_s \in$ closed

v_2 $g(v_2) = 140 + 151 = \underline{291}$ $h(v_2) = 380$ $f(v_2) = \underline{671}$

parent $(v_2) = v_3$

v_9 $g(v_9) = 140 + 99 = \underline{239}$ $h(v_9) = 193$ $f(v_9) = \underline{432}$

parent $(v_9) = v_3$

v_{12} $g(v_{12}) = 140 + 99 = \underline{239}$ $h(v_{12}) = 176$ $f(v_{12}) = \underline{415}$

parent $(v_{12}) = v_3$

Open = $\{v_1, v_2\ v_4, v_9, v_{12}\}$

Closed = $\{v_s, v_3\}$

$B = v_{12}$

successors $= \{v_3, v_e\}$

$v_3 \in$ closed

v_e $g(v_e) = 239 + 211 = \underline{450}$ $h(v_e) = 0$ $f(v_e) = \underline{450}$

parent$(v_e) = v_{12}$

Open $= \{v_1, v_2, v_4, v_9, v_e\}$

Closed $= \{v_S, v_3, v_{12}\}$

$B = v_9$

Successors $= \{v_3, v_8, v_{10}\}$

$v_3 \in$ closed

v_8 $g(v_8) = 239 + 146 = \underline{385}$ $h(v_8) = 160$ $f(v_8) = \underline{545}$

parent$(v_8) = v_9$

v_{10} $g(v_{10}) = 239 + 97 = \underline{336}$ $h(v_{10}) = 100$ $f(v_{10}) = \underline{\mathbf{436}}$

parent$(v_{10}) = v_9$

Open $= \{v_1, v_2, v_4, v_8, v_{10}, v_e\}$

Closed $= \{v_S, v_3, v_9, v_{12}\}$

$B = v_{10}$

Successors $= \{v_8, v_9, v_e\}$

$\cancel{v_8}$ $g'(v_8) = 336 + 138 = 474 > g(v_8)$

$v_9 \in$ closed

v_e $g'(v_e) = 336 + 101 = 437 < g(v_e) = 450$

opdateres $g(v_e) = 437$ $h(v_e) = 0$ $f(v_e) = 437$

parent$(v_e) = v_{10}$

Open $= \{v_1, v_2, v_4, v_8, v_e\}$

Closed $= \{v_S, v_3, v_9, v_{10}, v_{12}\}$

$B = v_e$

Path found.

length $= f(v_e) = 437$

P: v_e, parent$(v_e) = v_{10}$, parent$(v_{10}) = v_9$,

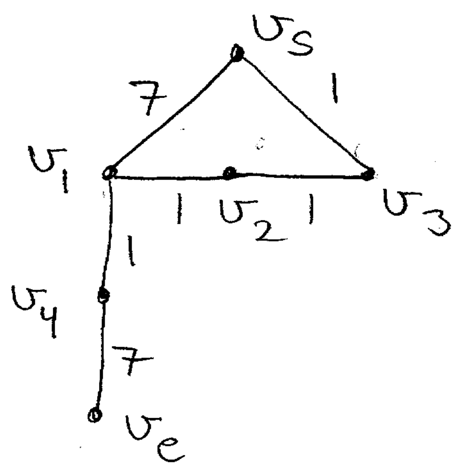
parent$(v_9) = v_3$, parent$(v_3) = v_S$

alltså

Q: $v_S, v_3, v_9, v_{10}, v_e$

Eks. 4

Dette er et eksempel, hvor A^*-algoritmen fra [Rab] svarer forkert

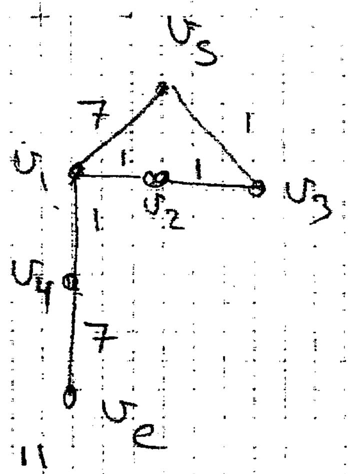


$h(v_s) = 11$
$h(v_1) = 2$, $h(v_2) = 2$, $h(v_3) = 10$,
$h(v_4) = 7$, $h(v_e) = 0$.

Så h er en tilladelig (admissible) heuristisk funktion. Men h er ikke en monoton heuristisk funktion, da
$h(v_3) - h(v_2) = 10 - 2 = 8 > w(v_2, v_3) = 1$.

Som vi skal se finder A^* fra [Rab]
ej $P: v_s v_3 v_2 v_4 v_e$ af længde 11
men $P': v_s v_1 v_4 v_e$ af længde 15
Problemet er netop, at h ej er monoton

Eksempel hvor A^* algoritmen fra [Rab] svarer forkert



$h(v_s) = 11$

$h(v_1) = 2$, $h(v_2) = 2$, $h(v_3) = 10$,

$h(v_4) = 7$ $h(v_e) = 0$

Så h er en tilladelig (admissible) heuristisk funktion

Men h er ikke en monoton heuristisk funktion, da

$h(v_3) - h(v_2) = 10 - 2 = 8 > w(v_2 v_3) = 1$

Som vi skal se finder A^* fra [Rab] ej $P: v_s v_3 v_2 v_4 v_e$ af længde 11 men $P': v_s v_1 v_4 v_e$ af længde 15.

Problemet er netop, at h ej er monoton

v_s $g(v_s)=0$ $h(v_s)=11$ $f(v_s)=11$
Open $=\{v_s\}$
$B = v_s$
successors $=\{v_1, v_3\}$
v_1 $g(v_1)=7$ $h(v_1)=2$ $f(v_1)=\underline{9}$
parent $(v_1) = v_s$
v_3 $g(v_3)=1$ $h(v_3)=10$ $f(v_3)=\underline{11}$
Open $=\{v_1, v_3\}$
Closed $=\{v_s\}$
$B = v_1$
successors $=\{v_s, v_2, v_4\}$
$v_s \in$ Closed
v_2 $g(v_2)=7+1=\underline{8}$ $h(v_2)=2$ $f(v_2)=\underline{10}$
Parent $(v_2) = v_1$
v_4 $g(v_4) = 7+1 = \underline{8}$ $h(v_4)=7$ $f(v_4)=\underline{15}$
parent $(v_4) = v_1$
Open $=\{v_2, v_3, v_4\}$
Closed $=\{v_s, v_1\}$
$B = v_2$
Successors $=\{v_1, v_3\}$
$v_1 \in$ Closed
$\cancel{v_3}$ $g'(v_3) = 9 > g(v_3)$
Open $=\{v_3, v_4\}$
Closed $=\{v_s, v_1, v_2\}$

$B = v_3$

successors $= \{v_s, v_2\}$

$v_s, v_2 \in$ closed

Open $= \{v_4\}$

closed $= \{v_s, v_1, v_2, v_3\}$

$B = v_4$

successors $= \{v_1, v_e\}$

$v_1 \in$ closed

v_e $g(v_e) = g(v_4) + 7 = 15$ $h(v_e) = 0$ $f(v_e) = 15$

parent $(v_e) = v_4$

Open $= \{v_e\}$

closed $= \{v_s, v_1, v_2, v_3, v_4\}$

$B = v_e$

$P := v_e$, parent $(v_e) = v_4$, parent $(v_4) = v_1$, parent $(v_1) = v_s$.

Så Q: v_s, v_1, v_4, v_e er korteste vej fra v_s til v_e og denne er af længde $f(v_e) = 15$.

(Men ovenstående er jo ikke den korteste vej !!!)

Sætning

Hvis den heuristiske funktion i algoritmen side 6 og 7 er monoton, da returnerer algoritmen en korteste vej, hvis en sådan findes.

Bevis:

Vi tilføjer midlertidigt følgende på side 7 mellem linie 2 og linie 3.

" If $c \in$ Closed så tjek om den nye vej er billigere (har lavere f-værdi). Hvis dette er tilfældet opdater da $g(c)$ og parent(c) i overensstemmelse hermed og flyt c fra closed til open. Gå til næste punkt "

Bemærk, at også denne modificerede algoritme stopper, da et punkt som flyttes tilbage i Open får nedsat sin g-værdi og dermed sin f-værdi

Vi konstaterer, at der i et vilkårligt step i den modificerede algoritme gælder, at hvis $B = v_i$ og $v_j \in$ successor får opdateret sin f-værdi, da haves $f(v_i) \le f(v_j)$. Der gælder nemlig, at hvis $f(v_j)$ opdateres (eller tildeles værdi for første gang) da haves

$$\begin{aligned} f(v_j) &= g(v_j) + h(v_j) \\ &= g(v_i) + w(v_i v_j) + h(v_j) \\ &\ge g(v_i) + h(v_i) = f(v_i). \end{aligned}$$

Vi indser nu, at dette medfører at ovenstående modifikation af algoritmen alene bliver effektueret.

Lad $w \in$ Closed ved algoritmens afslutning. Betragt det tidspunkt i algoritmeforløbet, hvor w blev valgt som aktivt punkt $B = w$. Lad den åbne liste hvorfra $B = w$ blev valgt være

$$\{w = u_1, u_2, \dots, u_s\}.$$

Vi har på dette tidspunkt

$$f(w) \le f(u_i) \quad i = 2, \dots, s.$$

Lad de punkter i successors til w, som får deres f-værdi opdateret

være $\{a_1, \dots, a_t\}$.
Af resultatet tidligere i beviset fås $f(w) \leq f(a_i)$, $i = 1, \dots, t$.

Enhver ny vej genereret af algoritmen efter denne opdatering må være fremkommet ved at udvide fra et af punkterne i

$$\{u_2, \dots, u_s\} \cup \{a_1, \dots, a_t\}.$$

og da alle disse punkter har f-værdi mindst $f(w)$ og da f-værdierne ej aftager langs de nye veje, vil w aldrig senere få tildelt nogen mindre $f'(w)$-værdi. Derfor opdateres w ej igen, når w først en gang er flyttet over i Closed. De foreslåede modifi-ceringer er uden betydning. Denne viden siger os, at den faktiske algoritme søger efter billige veje i hele grafen (uden begrænsninger). Når til sidst $B = v_e$ er et punkt i Open med mindste f-værdi, da ville alle fortsatte søgninger give veje med f-værdier mindst $f(v_e) = g(v_e)$ og der findes ingen billigere vej

$\square$