

25.8 A*-Algorithmus

Disclaimer

Diese Folien beinhalten die wichtigsten Formalien zum A*-Algorithmus und dessen Korrektheit. In der Vorlesung wird der Algorithmus motiviert und mit Beispielen unterlegt.

Eine sehr schöne Motivation des Algorithmus findet sich zum Beispiel hier:
<https://www.youtube.com/watch?v=bRvs8r0QU-Q>

A*-Algorithmus

Voraussetzungen

- Positiv gewichteter Graph $G = (V, E, c)$
- G endlich oder δ -Graph: $\exists \delta > 0 : c(e) \geq \delta$ für alle $e \in E$
- $s \in V, t \in V$
- Abstandsschätzung $\hat{h}_t(v) \leq h_t(v) := \delta(v, t) \forall v \in V$.
- Gesucht: kürzester Pfad $p : s \rightsquigarrow t$

A*-Algorithmus(G, s, t, \hat{h})

Input: Positiv gewichteter Graph $G = (V, E, c)$, Startpunkt $s \in V$, Endpunkt $t \in V$, Schätzung $\hat{h}(v) \leq \delta(v, t)$

Output: Existenz und Wert eines kürzesten Pfades von s nach t

foreach $u \in V$ **do**

$d[u] \leftarrow \infty$; $\hat{f}[u] \leftarrow \infty$; $\pi[u] \leftarrow \text{null}$

$d[s] \leftarrow 0$; $\hat{f}[s] \leftarrow \hat{h}(s)$; $R \leftarrow \{s\}$; $M \leftarrow \{\}$

while $R \neq \emptyset$ **do**

$u \leftarrow \text{ExtractMin}_{\hat{f}}(R)$; $M \leftarrow M \cup \{u\}$

if $u = t$ **then return success**

foreach $v \in N^+(u)$ with $d[v] > d[u] + c(u, v)$ **do**

$d[v] \leftarrow d[u] + c(u, v)$; $\hat{f}[v] \leftarrow d[v] + \hat{h}(v)$; $\pi[v] \leftarrow u$

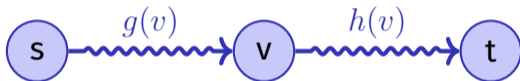
$R \leftarrow R \cup \{v\}$; $M \leftarrow M - \{v\}$

return failure

Notation

Sei $f(v)$ die Distanz eines kürzesten Weges von s nach t über v , also

$$f(v) := \underbrace{\delta(s, v)}_{g(v)} + \underbrace{\delta(v, t)}_{h(v)}$$



Sei p ein kürzester Weg von s nach t .

Dann gilt $f(s) = \delta(s, t)$ und $f(v) = f(s)$ für alle $v \in p$.

Sei $\hat{g}(v) := d[v]$ die Schätzung von $g(v)$ in obigem Algorithmus. Es gilt, dass $\hat{g}(v) \geq g(v)$.

$\hat{h}(v)$ ist eine Schätzung von $h(v)$ mit $\hat{h}(v) \leq h(v)$.

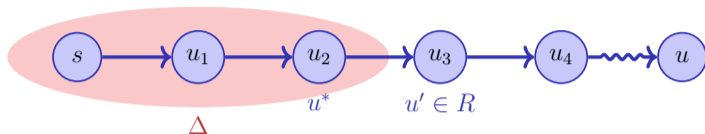
Warum der Algorithmus funktioniert

Lemma 24

Sei $u \in V$ und, zu einem Zeitpunkt des A^ -Algorithmus, $u \notin M$. Sei p ein kürzester Pfad von s nach u . Dann existiert ein $u' \in p$ mit $\hat{g}(u') = g(u')$ und $u' \in R$.*

Das Lemma besagt, dass es immer einen Knoten in der offenen Menge R gibt, dessen wahre Entfernung von s schon berechnet wurde und der zum kürzesten Pfad gehört (sofern ein solcher existiert).

Illustration und Beweis



Beweis: Wenn $s \in R$, dann $\hat{g}(s) = g(s) = 0$. Sei also $s \notin R$.

Sei $p = \langle s = u_0, u_1, \dots, u_k = u \rangle$ und $\Delta = \{u_i \in p, u_i \in M, \hat{g}(u_i) = g(u_i)\}$.
 $\Delta \neq \emptyset$, denn $s \in \Delta$.

Sei $m = \max\{i : u_i \in \Delta\}$, $u^* = u_m$. Dann $u^* \neq u$, da $u \notin M$. Sei $u' = u_{m+1}$.

1. $\hat{g}(u') \leq \hat{g}(u^*) + c(u^*, u')$ weil u' schon relaxiert wurde
2. $\hat{g}(u^*) = g(u^*)$ (da $u^* \in \Delta$)
3. $\hat{g}(u') \geq g(u')$ (Konstruktion von \hat{g})
4. $g(u') = g(u^*) + c(u^*, u')$ (da p optimal)

Also: $\hat{g}(u') = g(u')$ und somit auch $u' \in R$ da $u' \notin \Delta$.



Folgerung

Corollary 25

Wenn $\hat{h}(u) \leq h(u)$ für alle $u \in V$ und A- Algorithmus hat noch nicht terminiert. Dann existiert für jeden kürzesten Pfad p von s nach t ein Knoten $u' \in p$ mit $\hat{f}(u') \leq \delta(s, t) = f(t)$.*

Wenn es einen kürzesten Weg p von s nach t gibt, steht also stets ein Knoten in der offenen Menge bereit, der die Gesamtentfernung maximal unterschätzt und der auf dem kürzesten Weg liegt.

Beweis des Corollars

Beweis:

Nach Lemma $\exists u' \in p$ mit $\widehat{g}(u') = g(u')$.

Also:

$$\begin{aligned}\widehat{f}(u') &= \widehat{g}(u') + \widehat{h}(u') \\ &= g(u') + \widehat{h}(u') \\ &\leq g(u') + h(u') = f(u')\end{aligned}$$

Da p optimal: $f(u') = \delta(s, t)$. ■

Zulässigkeit

Theorem 26

Wenn es einen kürzesten Weg von s nach t gibt und $\hat{h}(u) \leq h(u) \forall u \in V$, dann terminiert der A^* -Algorithmus mit $\hat{g}(t) = \delta(s, t)$

Beweis: Wenn der Algorithmus terminiert, dann terminiert er in t mit $f(t) = \hat{g}(t) + 0 = g(t)$. Denn \hat{g} überschätzt g höchstens und nach obigem Korollar findet der Algorithmus stets ein Element $v \in R$ mit $f(v) \leq \delta(s, t)$. Der Algorithmus terminiert in endlichen vielen Schritten. Für endliche Graphen ist die maximale Anzahl an Relaxierschritten beschränkt.

44

⁴⁴Für einen δ -Graphen ist die maximale Anzahl an Relaxierschritten bevor R nur noch Knoten mit $\hat{f}(s) > \delta(s, t)$ enthält, auch beschränkt. Das genaue Argument findet sich im Originalartikel Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths".

Erneutes Besuchen von Knoten

- Der A*-Algorithmus kann Knoten mehrfach aus der Menge R entnehmen und sie später wieder einfügen.
- Das kann zu suboptimalem Verhalten im Sinne der Laufzeit des Algorithmus führen.
- Wenn \hat{h} zusätzlich zur Zulässigkeit ($\hat{h}(v) \leq h(v)$ für alle $v \in V$) auch noch monoton ist, d.h. wenn für alle $(u, u') \in E$:

$$\hat{h}(u') \leq \hat{h}(u) + c(u', u)$$

dann ist der A* Algorithmus äquivalent zum Dijkstra-Algorithmus mit Kantengewichten $\tilde{c}(u, v) = c(u, v) + \hat{h}(u) - \hat{h}(v)$ und kein Knoten wird aus R entnommen und wieder eingefügt.

- Es ist allerdings nicht immer möglich, eine monotone Heuristik zu finden.