Dijkstra und A*

Alexander Burger, Simon Heiden, Martin Martius

November 24, 2008

Inhaltsverzeichnis

Der Informatiker Dijkstra

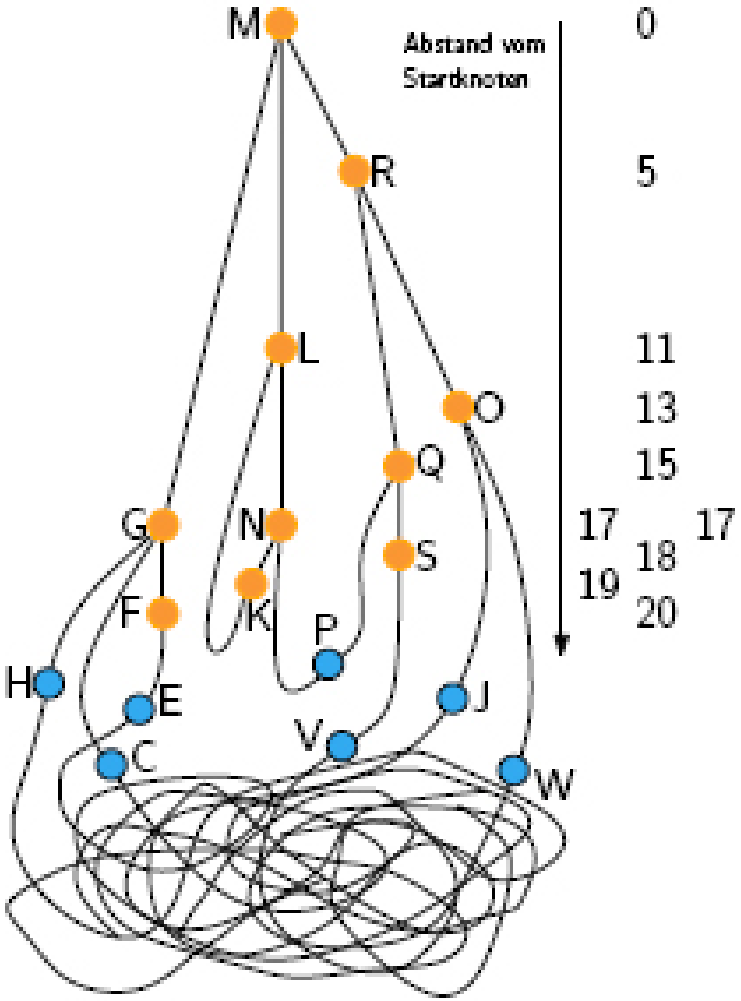
- 1930 in Rotterdam geboren
- studierte Mathematik und theoretische Physik

Finden eines kürzesten Weges

Die Idee:

- Bindfäden auf Stadtplan entlang der Straßen legen
- An Kreuzungen verknoten
- Startknoten anheben
- Netz weiter anheben, bis Endknoten sich hebt
- Ein Faden spannt den Weg zwischen Start- und Zielknoten, dieser ist der kürzester Pfad
- Nachmessen bringt Entfernung

Finden eines kürzesten Weges



Voraussetzungen

- Graph vollständig gegeben
  - alle Knoten bekannt
  - alle Kanten mit (positiven) Gewichtungen bekannt

Algorithmus

- S := s D[s] := 0;
- für alle Knoten $v \neq s$: $D[v] := C(s, v)$;
- while V $\setminus S \neq \emptyset$; *do*
- wähle den Knoten w $\in V \setminus S$ mit minimalem D[w];
- $S := S \cup w$;
- für alle u $\in$ V $\setminus$ S, u adjazent zu w do
- D[u] := min (D[u] ,D[w] + C (w, u))

Korrektheit Teil 1

für alle v∈S ist D[v] Länge des kürzesten Weges von s nach v
sei d(v) = Länge des kürzesten Weges von s nach v

- k=0; S=$\{s\}$ und D[s]=0
- k $\rightarrow$ k+1; sei w Knoten, der in der (k+1)-ten Iteration zu S kommt. Angenommen Behauptung ist falsch: D[w] > d[w]
- sei x kürzester Weg von s nach w
- sei (u,v) die Kante, die als erste aus S_k herausführt (S_k = S nach k Iterationen)
- nach Induktionsvoraussetzung
  $d[u] = D[u] d(v) = d(u) + c(u,v) = D[u] + c(u,v) \leq D[v]$

Korrektheit Teil 2

- beim Einfügen von u in S wurde D[v] auf das Minimum von [D[v];D[u] + c(u,v)] gesetzt
- dann ist d(v) $\leq D[v] \Rightarrow$ d(v) = D[v]
- aber D[w] $\geq d(w) \geq$ d(v) = D[v]
- Widerspruch zu wähle den Knoten w $\in V \setminus S$ mit minimalem D[w]

Laufzeit bei Verwendung eines Heaps

- Initialisierung in O(n) (für alle Knoten v $\neq$ s : D[v] := C (s, v))
- Minimum streichen O(log n) (wähle den Knoten w $\in$ V S mit minimalem D[w])
- Wert vermindern O(log n) (für alle u $\in$ V $\setminus$ S, u adjazent zu w do)
- ergibt O (n log n + m log n) = O (n+m log n); n - Anzahl Knoten; m - Anzahl Kanten
- O (n^2 log n)

Probleme mit negativen Gewichten

- Bereits besuchter Knoten könnte zu späterem Zeitpunkt durch kostengünstigeren Pfad erneut besucht werden.
- Der Übergang von einem Zustand in den nächsten kostet Ressourcen. Daher sollten Kosten nicht negativ sein.
- Bsp.: 3 Knoten; c(1,2)=1; c(1,3)=2; c(2,3)=-2

DijkstraVis

Simulationssoftware für die Routenfindung mit Dijkstra

Einordnung A*

//hier kämen die Folien von Alex

Algorithmus von A*

Der Algorithmus für A* benutzt 2 Listen:

- OPEN LIST: enthält die Knoten, die schom geöffnet, aber noch nicht expandiert wurden.
- CLOSED LIST: enthält die Knoten, die schon expandiert wurden.

Anfangszustand:

- OPEN LIST enthält nur den Startknoten.
- CLOSED LIST ist leer.

Funktionsweise

- Der "beste" Knoten (bestimmt durch eine Funktion f) aus der OPEN LIST wird herausgenommen und in die CLOSED LIST eingefügt.
- Dessen Nachbarknoten werden in die OPEN LIST eingefügt:
  - Wenn ein Knoten schon in der OPEN LIST steht, dann wird ggf. sein f-Wert aktualisiert, wenn ein kürzerer Pfad zu ihm gefunden wurde.
  - Wenn ein Knoten schon in der CLOSED LIST steht, dann wird er ggf. aktualisiert in die OPEN LIST eingefügt, falls ein kürzerer Pfad zu ihm gefunden wurde.
- Wenn der Zielknoten expandiert wird, dann bekommt man einen Lösungspfad durch Zurückgehen über die expandierten Knoten zum Startknoten.

Die Funktion "f"

Die Reihenfolge, in der die Knoten expandiert werden, wird bestimmt durch eine Funktion

$$f(n) = g(n) + h(n), \text{wobei}$$

- n der derzeitige Knoten ist,
- $g(n)$ die Kosten des bisher kürzesten Pfades vom Startknoten zu n, und
- $h(n)$ die geschätzten Kosten des kürzesten Pfades von n zum Endknoten sind.

heuristische Funktion $h(n)$

- heuristische Abschätzung von $h^*(n)$, den realen Kosten des kürzesten Pfades von n zum Zielknoten
- sollte/darf die realen Kosten nie überschätzen, dann
- ist der erste Pfad, den A* zum Zielknoten findet, optimal.

Beispiel bei Routenfindung: $h(n) =$ Luftlinie von n zum Ziel

Definitionen

Def.: $h(n)$ zulässig:

$h(n)$ überschätzt nie $h^*(n)$, d.h.: $h(n) \leq h^*(n) \ \forall\ n$.

Def.: $h(n)$ konsistent:

$h(n)$ ist zulässig, und es gilt

$$h(n) \leq c(n, n') + h(n') \ \forall\ n, n',$$

wobei

$$c(n, n')$$

die Kosten sind, um von n nach n' zu gelangen.

A* terminiert

Bew.: Knoten können mehrfach in die OPEN LIST eingefügt werden, von verschiedenen Pfaden aus. Aber es kann angenommen werden, dass der Algorithmus keine Zyklen erlaubt.
Es gilt: Es gibt in einem endlichen Graphen nur endlich viele verschiedene Pfade. (In einem vollständigen Graphen mit n Knoten sind es z.B. $n!$ verschiedene Pfade.)

$\Rightarrow$ Entweder wird der Zielknoten erreicht, oder der Algorithmus endet spätestens nach Prüfung aller Pfade.

□

A* liefert kostenminimalen Pfad, falls h zulässig

Bew.: Sei L_1 eine optimale Lösung mit den Kosten K^*, und sei L_2 mit den Kosten K_2 keine optimale Lösung.
$h(n)$ ist zulässig. $\Rightarrow$ Für alle Zielknoten z gilt $h(z) = 0$, also gilt auch insbesondere $h(L_2) = 0$.

L_2 ist nicht optimal. $\Rightarrow K_2 > K^*$

$$K_2 = g(L_2) = g(L_2) + h(L_2) = \underline{f(L_2) > K^*}$$

A* liefert kostenminimalen Pfad, falls h zulässig

Für alle Knoten x auf dem Pfad von L_1 gilt:

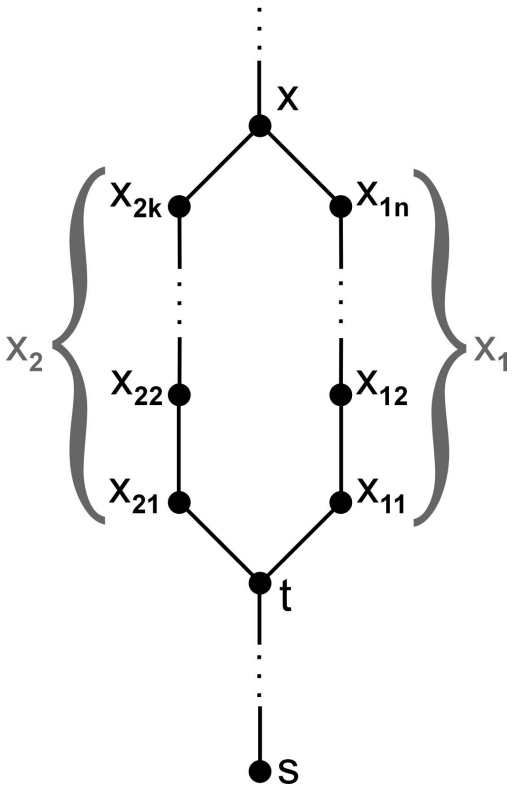
$$f(x) = g(x) + h(x) \leq K^* \quad \text{(h zulässig)}$$

$\Rightarrow f(x) \leq K^* < f(L_2)$, insbesondere: $f(L_1) < f(L_2)$.
$\Rightarrow L_1$ wird zuerst gefunden.

□

Keine Reexpansion von Knoten, falls h konsistent

Bew.: Annahme: Der Knoten x wird zum 2. Mal expandiert.
$\Rightarrow$ Es existieren 2 Pfade (x_1 und x_2) zu x (siehe Abbildung).



Keine Reexpansion von Knoten, falls h konsistent

Zeigen: Wenn x über einen der Pfade x_1 oder x_2 expandiert wird, dann wird zuerst ein optimaler Pfad zum Zielknoten gefunden, bevor x über den anderen Pfad expandiert wird.

Sei x_1 der Pfad, über den x zuerst expandiert wird.
Zeigen: $g_{x_2}(x) \geq g_{x_1}(x)$.

Annahme des Gegenteils:

$$g_{x_2}(x) < g_{x_1}(x)$$
$$g_{x_2}(x) + h(x) < g_{x_1}(x) + h(x)$$
$$f_{x_2}(x) < f_{x_1}(x).$$

Keine Reexpansion von Knoten, falls h konsistent

Sei p ein beliebiger Knoten in x_2. Es gilt:

$$f_{x_2}(p) = g_{x_2}(p) + h(p) \leq g_{x_2}(p) + c(p, x) + h(x)$$

$$= g_{x_2}(x) + h(x) = f_{x_2}(x) < f_{x_1}(x),$$

also $f_{x_2}(p) < f_{x_1}(x)$.

$\Rightarrow x$ wird zuerst über x_2 expandiert. Das ist aber ein Widerspruch, also folgt $g_{x_2}(x) \geq g_{x_1}(x)$.

Keine Reexpansion von Knoten, falls h konsistent

Noch zu zeigen:

1. Für alle Nachfolgerknoten q von x gilt: $f_{x_1}(q) \leq f_{x_2}(x)$ (unter der Voraussetzung, dass x zuerst über x_1 expandiert wird.)
2. Der erste Pfad zum Zielknoten, der gefunden wird, ist optimal.

Bemerkung: Wenn die erste Behauptung gezeigt werden kann, dann folgt daraus, dass zuerst alle Nachfolgerknoten eines Knoten expandiert werden, bevor dieser über einen zweiten Pfad expandiert werden kann.

Keine Reexpansion von Knoten, falls h konsistent

1. Annahme des Gegenteils:

$$f_{x_2}(x) < f_{x_1}(q) = g_{x_1}(q) + h(q) \leq g_{x_1}(q) + c(q,x) + h(x) = f_{x_1}(x),$$

also $f_{x_2}(x) < f_{x_1}(x)$,
was einen Widerspruch darstellt zu $g_{x_2}(x) \geq g_{x_1}(x)$.

$\Rightarrow$ Falls ein Pfad zum Zielknoten existiert, dann wird er gefunden, bevor x_2 expandiert wird, da insbesondere gilt, dass der Zielknoten ein Nachfolgeknoten von x_1 ist.

2. h konsistent $\Rightarrow$ h zulässig $\Rightarrow$ Der erste Lösungspfad, der gefunden wird, ist optimal.

$\square$