

Das Kürzeste-Wege-Problem in öffentlichen Verkehrsnetzen

Seminar
Stefan Görlich

25.05.2005

- Einleitung
- Suchverfahren
- Gierige Suche (Greedy Best-first)
- A^*
- Speicherbegrenzte Suche
- Modifikationen für den Praxiseinsatz im ÖPNV

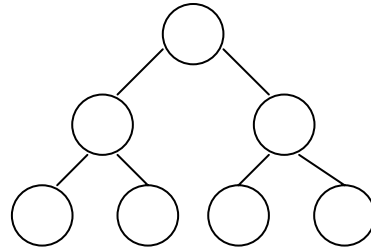
Warum suchen?

- Kürzesten Weg finden
- Problemlösen in der KI

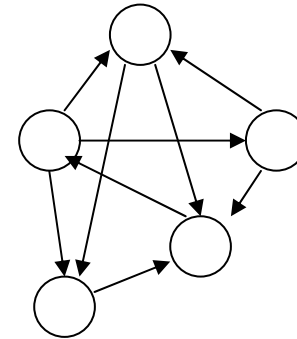
Probleme

- Häufig sehr große Suchräume
 - Z.B. HVV: 36617 Knoten
- Graphen mit Zyklen
- Begrenzte Ressourcen

- Suchgraph



- Zyklen



- Knoten entsprechen Haltestellen

- Kanten entsprechen einer Bus-/Bahnlinie

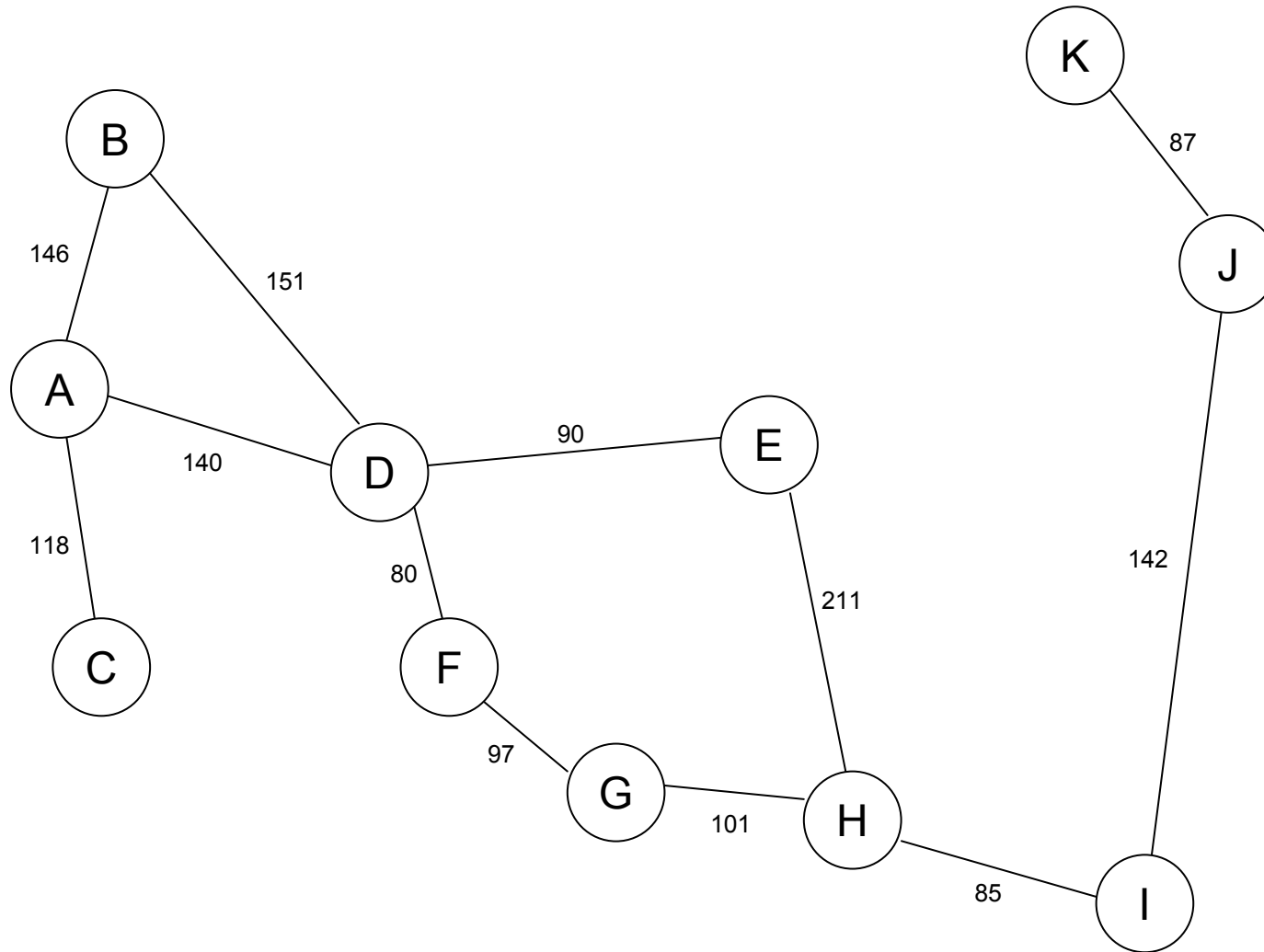
- Uninformierte Suche
 - „Blinde Suche“
 - Breitensuche, Tiefensuche, ...

- Heuristische Suche
 - Problemspezifische Suche
 - Bergsteigen, ...

Jetzt: Bestensuche

Eigenschaften eines „besten“ Suchalgorithmus:

- Vollständigkeit
- Optimalität
- optimale Effizienz



L: Menge von Knoten in Suchraum

C: Menge von Knoten, die bereits exploriert wurden

n: Ein Knoten im Suchraum

n': Kindknoten von n

f(n): Entscheidungsfunktion zur Exploration der Knoten

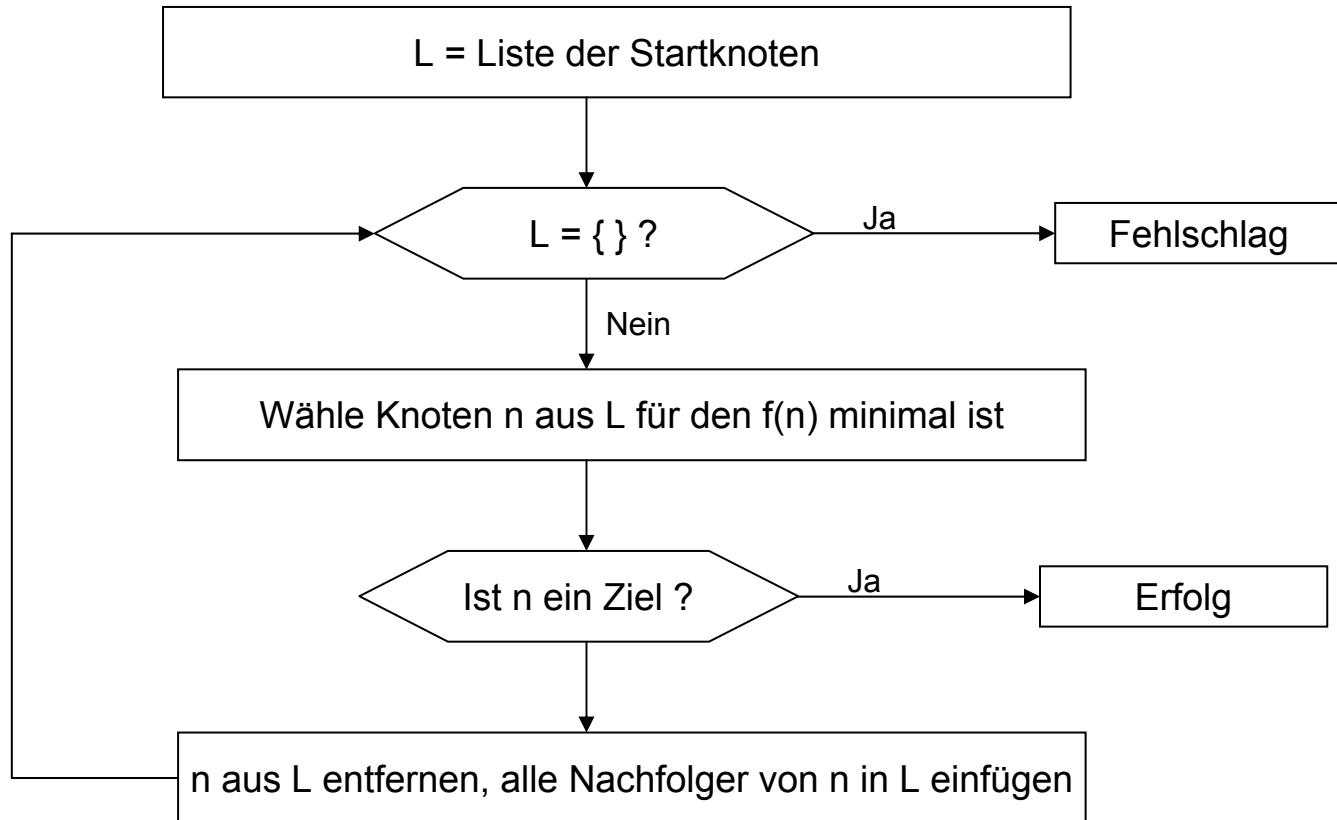
h(n): Geschätzte Kosten des billigsten Pfades von n zum Ziel

g(n): Kosten vom Startknoten zum Knoten n

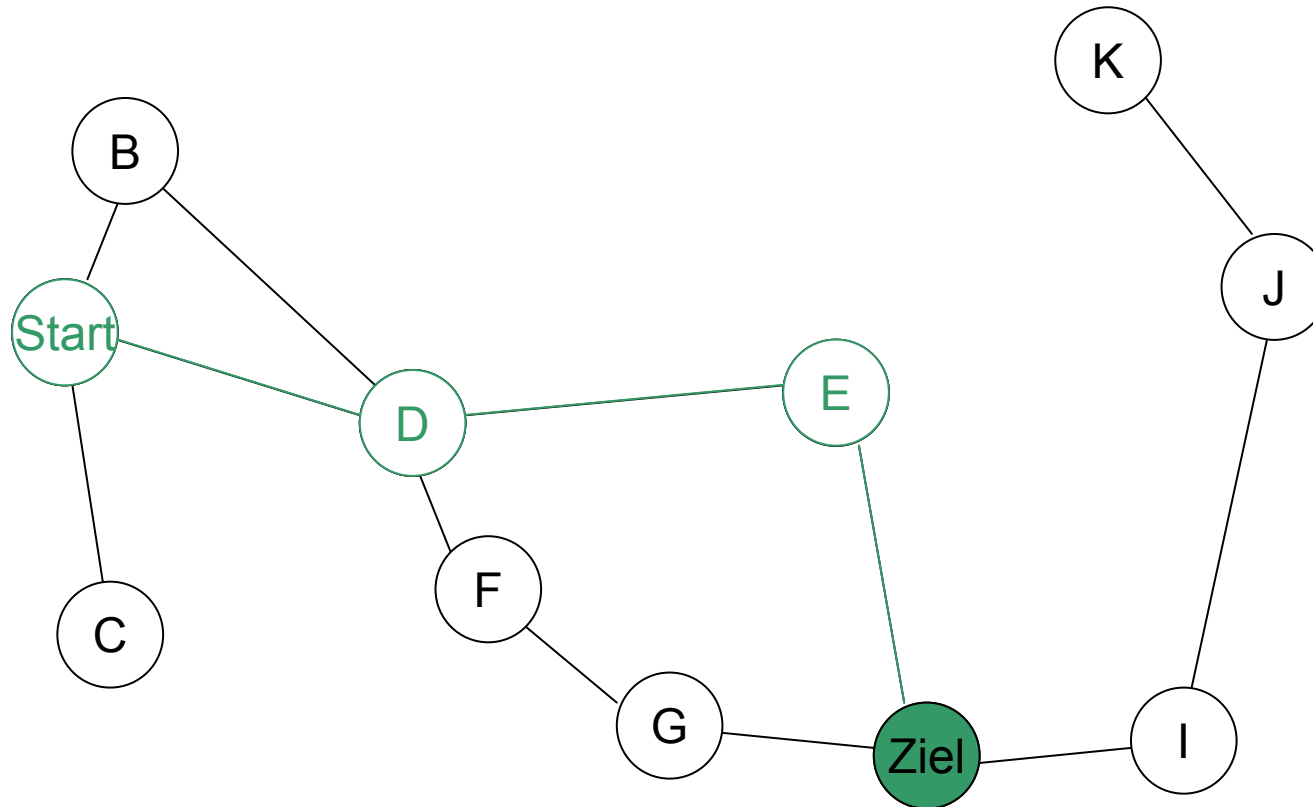
h*(n): Tatsächliche Kosten vom Knoten n zum Ziel

c(n→n'): Kosten für den Übergang von n zu n'

- Expandiert den Knoten, der dem Ziel am nächsten liegt
- $f(n) = h(n)$



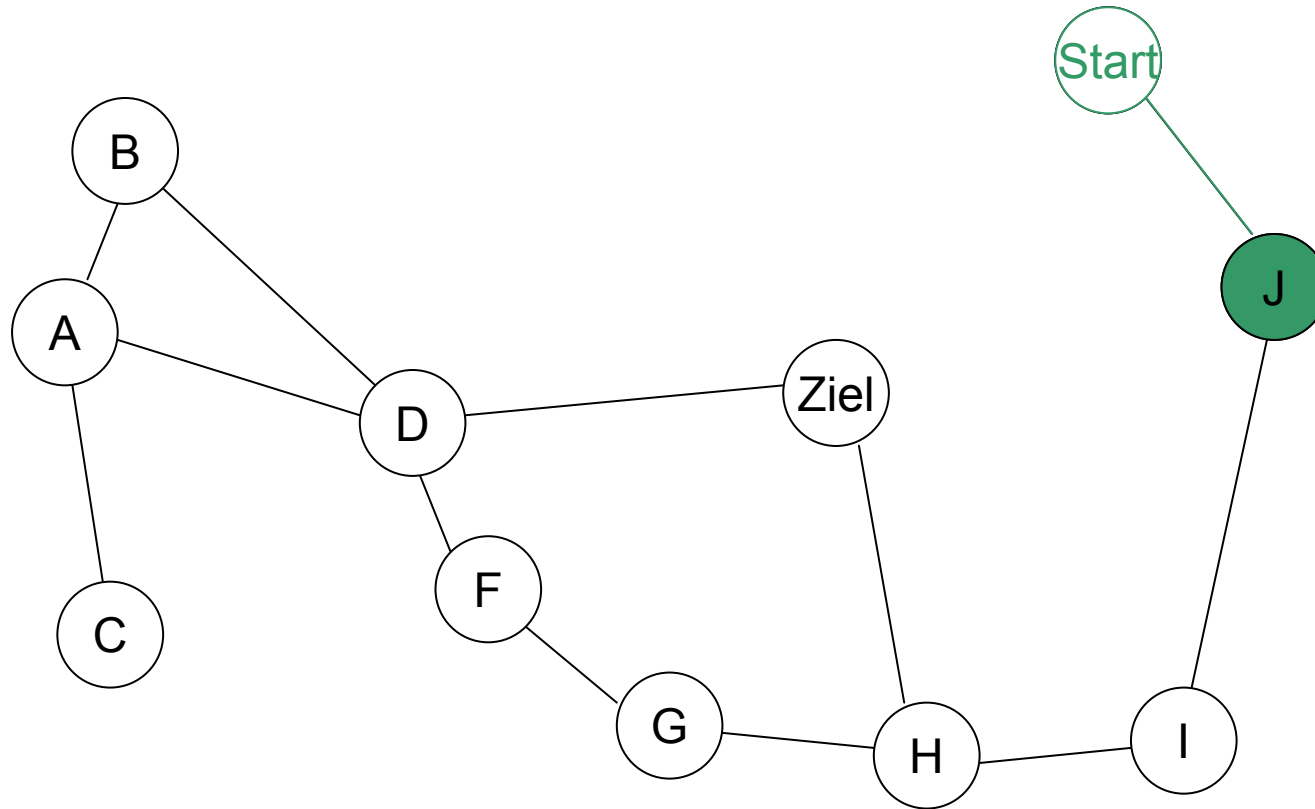
Gierige Suche (Greedy Best-First): Beispiel 1



Entfernungen zum Ziel:

Start	366
B:	380
C:	329
D:	253
E:	178
F:	193
G:	98
I:	80
J:	226
K:	234

Gierige Suche (Greedy Best-First): Beispiel 2



Entfernungen zum Ziel:

Start	120
A:	380
B:	385
C:	380
D:	99
F:	193
G:	198
H:	211
I:	230
J:	140

$L = \{I, \text{Start}\}$

Fazit:

- Nicht Vollständig
- Nicht optimal
- Nicht optimal effizient
- Zeit- und Speicherkomplexität:

Worst-case: $O(b^m)$

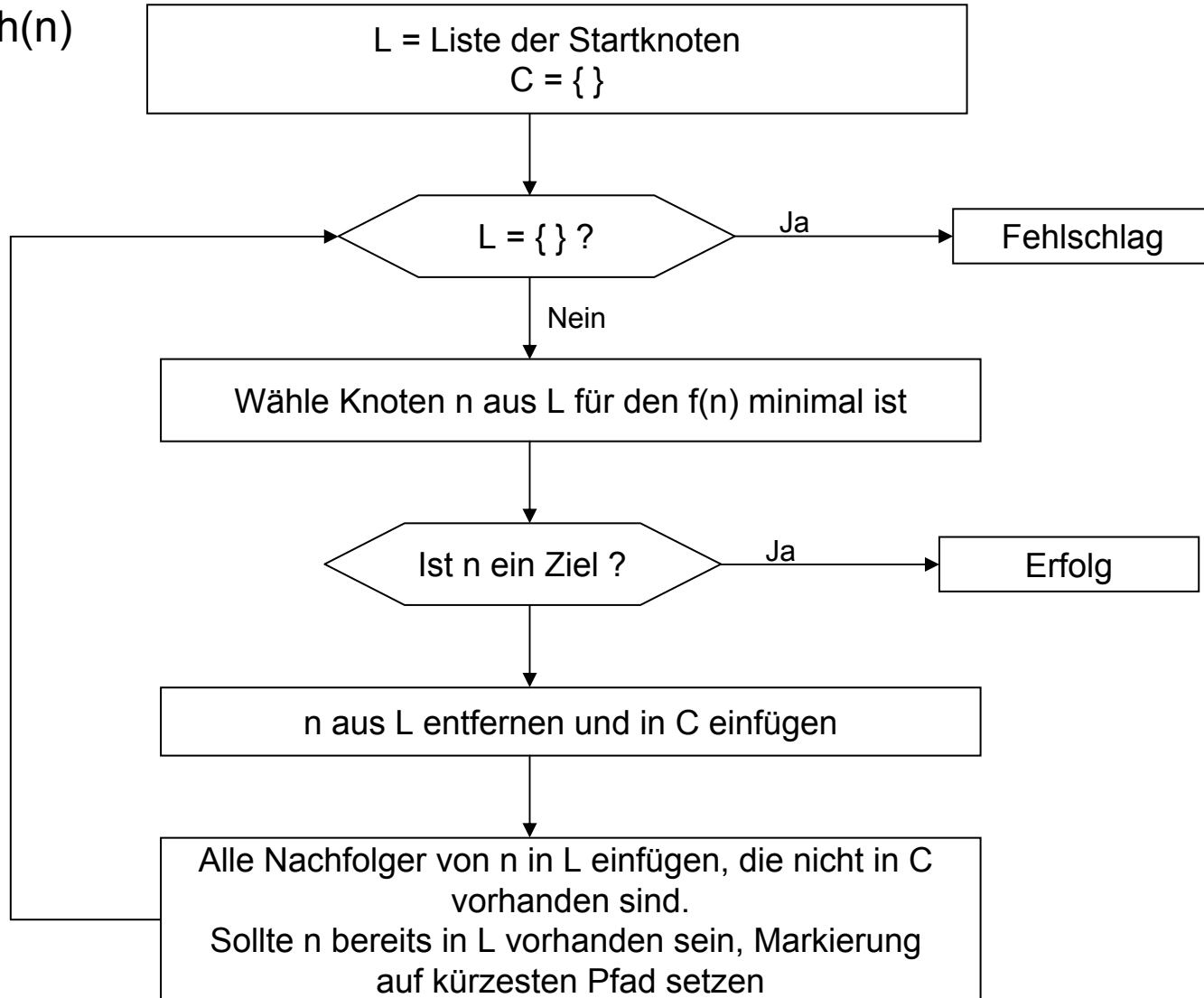
Stark von $h(n)$ abhängig

m: Maximale Tiefe des Suchraums

b: Verzweigungsfaktor

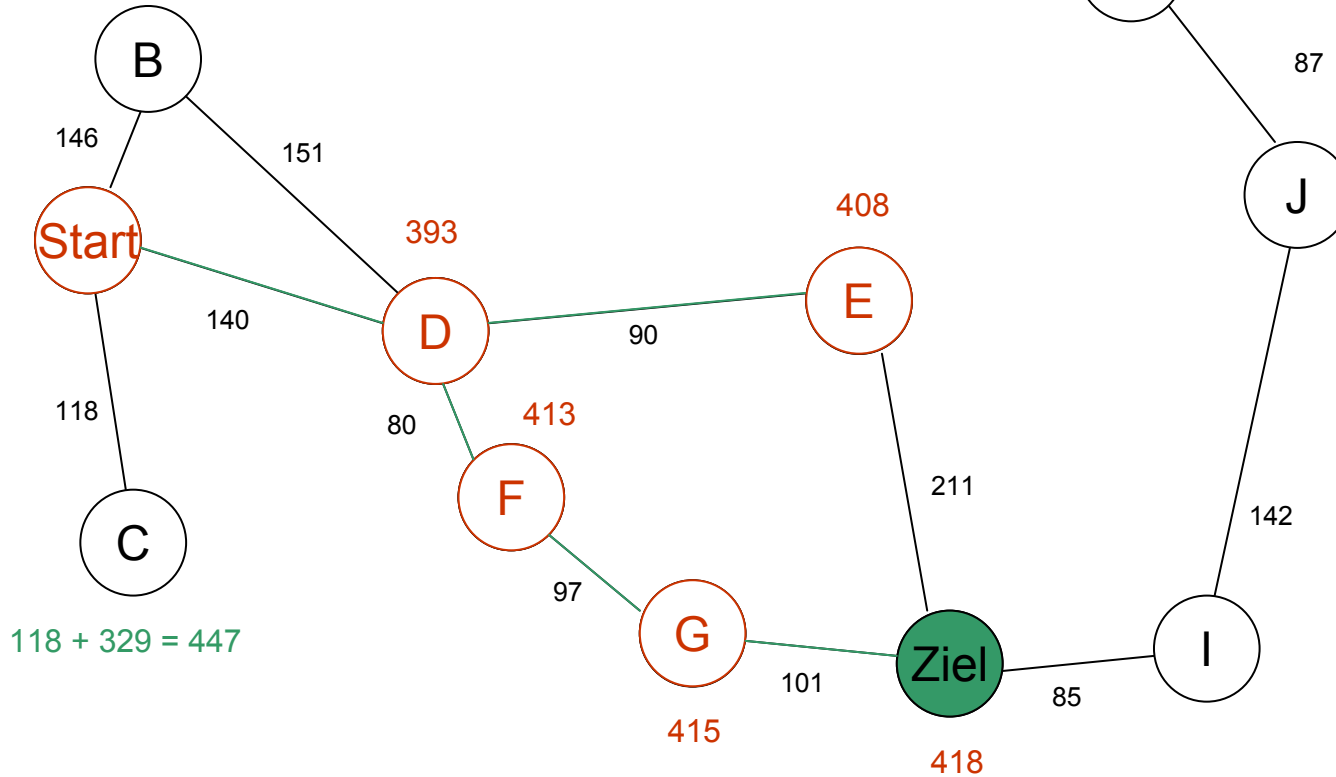
- + Minimierung der geschätzten Gesamtkosten für die Lösung
- + $f(n) = g(n) + h(n)$
- + Vollständig
- + Optimal (sofern $h(n)$ Konsistent)
- + Optimal-Effizient
- Verwaltung bereits erschlossener Knoten

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



A*: Beispiel

$146 + 380 = 526$



$118 + 329 = 447$

Entfernungen zum Ziel:

Start	366
B:	380
C:	329
D:	253
E:	178
F:	193
G:	98
I:	80
J:	226
K:	234

$L = \{B, C\}$

$C = \{\text{Start}, D, E, F, G, \text{Ziel}\}$

Da der A*-Algorithmus...

- alle Nachfolger eines Knotens in L aufnimmt,
- die bereits explorierten Knoten in C speichert,
- iteriert bis $L = \{ \}$ oder das optimale Ziel gefunden ist

...ist er vollständig**

** : vgl. Nilsson: Principles of Artificial Intelligence, 1982

Bedingung: $h(n)$ ist zulässig und konsistent (monoton)

- Zulässig: $h(n) \leq h^*(n)$
- Konsistent: $h(n) - h(n') \leq c(n \rightarrow n')$

Annahme: A* ist nicht optimal

$$\rightarrow f(t) = g(t) > g(o)$$

Da t nicht optimal, muss es vor der Terminierung ein Knoten n' in L geben, für den gilt: $f(n') \leq g(o)$

$$\text{Es folgt: } f(n') \leq g(o) < f(t) \quad \Rightarrow \quad f(n') < f(t)$$

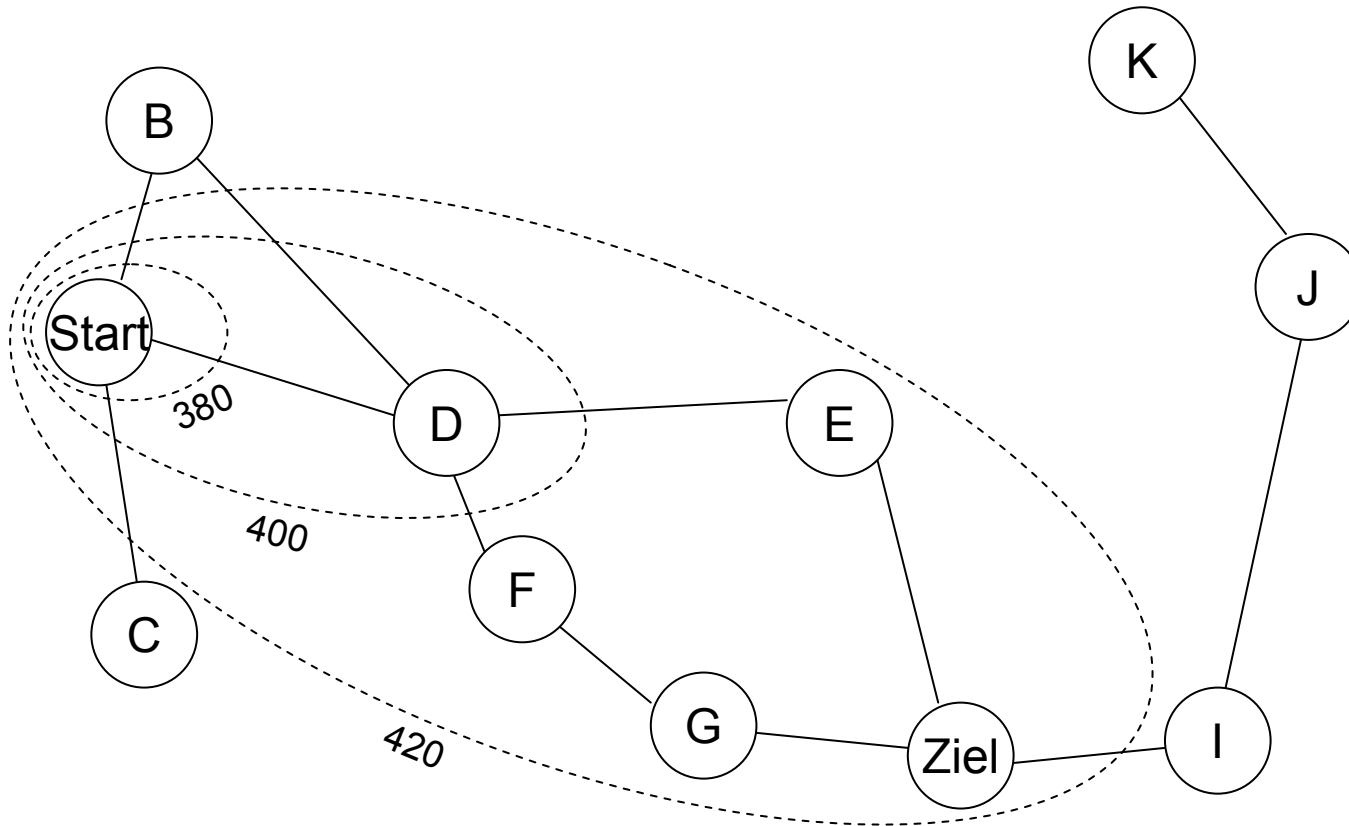
Demnach hätte n' vor t Expandiert werden müssen!

t: Nichtoptimaler Zielknoten

s: Startknoten

o: Optimaler Zielknoten

A*: Optimale Effizienz



- Jeder Algorithmus, der nicht mindestens alle Knoten exploriert, für die gilt $h(n) < h(g)$ ist nicht optimal!

=> A* ist optimal effizient**

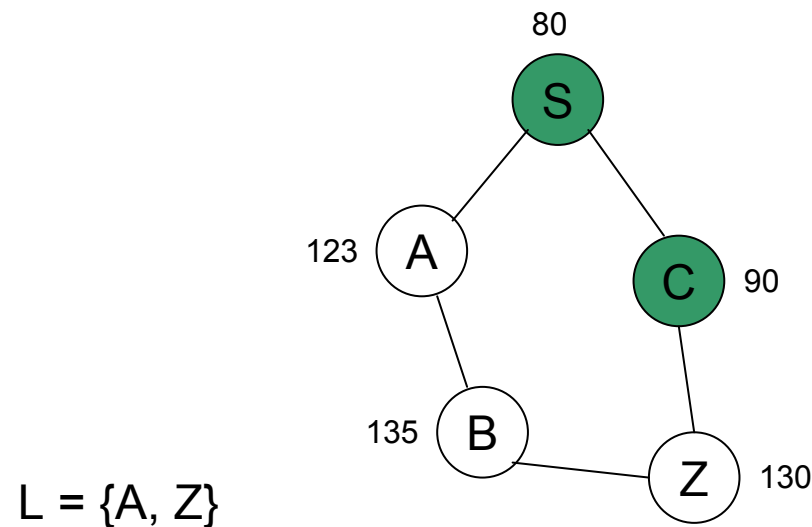
Suchkosten:

d	IDS	A* (h1)	A* (h2)
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	3644035	227	73
14	-	539	113
16	-	1301	211
18	-	3056	363
20	-	7276	676
22	-	18094	1219
24	-	39135	1641

** : vgl. Dechter, Pearl: Generalized best-first search strategies and the optimality, 1985

SMA* (simplified memory-bounded A*)

- Unter der Monotoniebedingung $h(n) - h(n') \leq c(n \rightarrow n')$ kann auf die Liste der bereits erschlossenen Knoten C verzichtet werden, wenn nur die Nachfolgeknoten in L aufgenommen werden, für die gilt: $f(n) \leq f(n')$



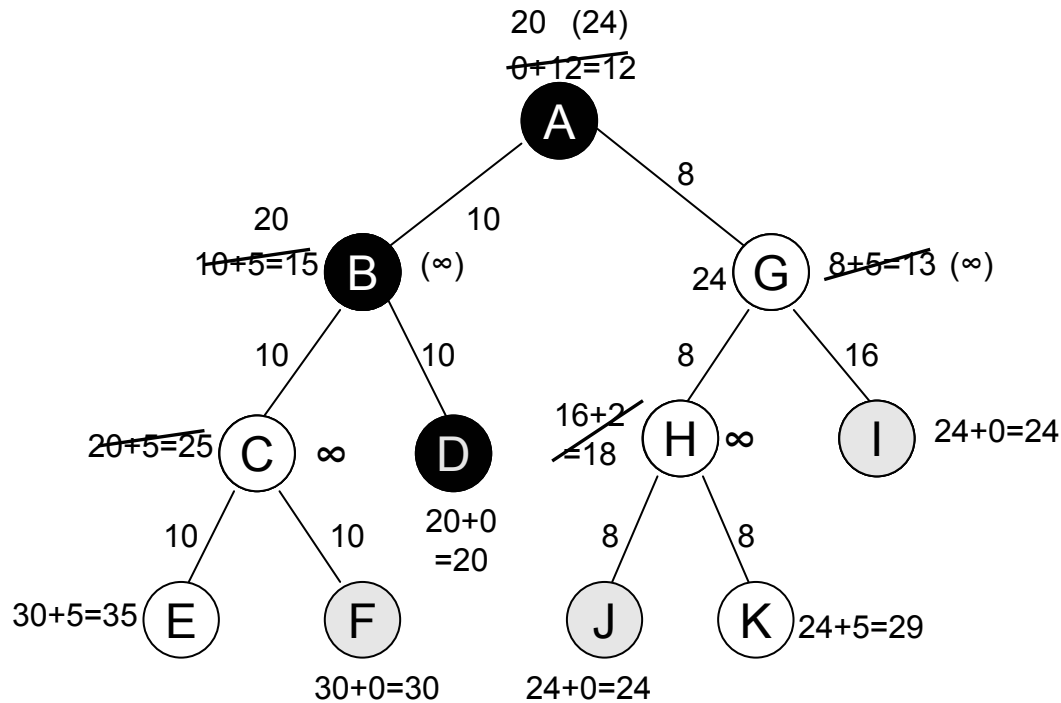
SMA* (simplified memory-bounded A*)

- Definition des maximalen Speichers, der zur Verfügung steht
- Normaler A*-Algorithmus bis Speichergrenze erreicht
- Dem Vaterknoten wird der minimale Wert der Nachfolgeknoten zugewiesen

Sobald Speichergrenze erreicht:

- Knoten mit größtem $f(n)$ -Wert wird verworfen, die Kosten des verworfenen Pfades werden zusätzlich im Vaterknoten gespeichert

SMA* (simplified memory-bounded A*)



$L = \{A, B, D\}$

Speicherkapazität: 3 Knoten

SMA* (simplified memory-bounded A*)

- + Vollständig, falls ein Zielknoten mit dem zur Verfügung stehenden Speicher erreicht werden kann
- + Optimal, falls das optimale Ziel mit dem zur Verfügung stehenden Speicher erreicht werden kann
- Erhöhter Rechenaufwand durch Neuerstellung bereits untersuchter Knoten

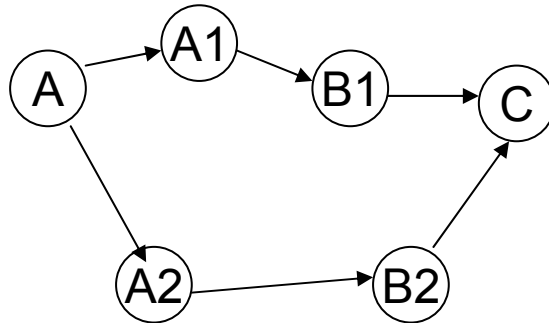
- Gesucht: Schnellste Verbindung, nicht kürzeste
- Schnellste Verbindung zur bestimmten Zeit

- Problem: Unterschiedlich getaktete Linien
 - Wartezeit beim Umsteigen minimieren

Beispiel: www.geofox.de

- Lösung: Vorwärts- mit anschließender Rückwärtssuche

- Sucheinschränkungen



- Dadurch schnellste Verbindung nicht unbedingt beste

Optionen: über Haltestelle:

Max. Anzahl Ergebnisse:

Umsteigen:

Zuschlag:

Weg zur Haltestelle:

Fußweg:

Mobilität:

- Unterschiedliche Fahrzeiten je nach Tageszeit
 - Kosten müssen angepasst werden

Beispiel: www.geofox.de

- Haltestellen können zu bestimmten Zeiten nicht erreichbar sein
 - Problem: Alle Knoten würden expandiert werden

Beispiel: DELFI

Vielen Dank für Ihre Aufmerksamkeit