



Algorithmen für Routenplanung

11. Vorlesung, Sommersemester 2021

Jonas Sauer | 17. Mai 2021



Alternative
Route



flickr.com/photos/dun...

Alternativ-Routen

Wiederholung Punkt-zu-Punkt

Anfrage:

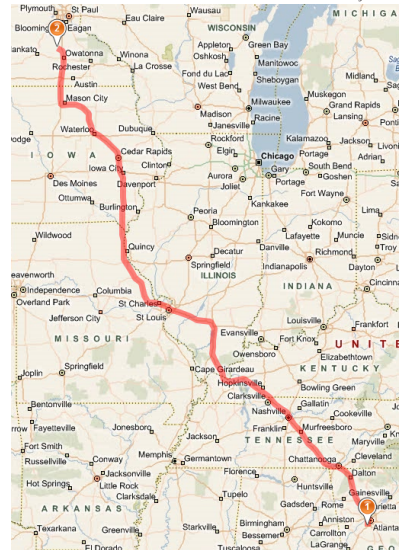
- Finde die **beste** Route in einem Transportnetz

Idee:

- Netzwerk als Graph $G = (V, E)$
- Kantengewichte sind **Reisezeiten**
- **Kürzester** Pfad in G entspricht **schnellster** Verbindung

Ergebnisse:

- Schnelle Algorithmen existieren



Alternativ-Routen

Anfrage:

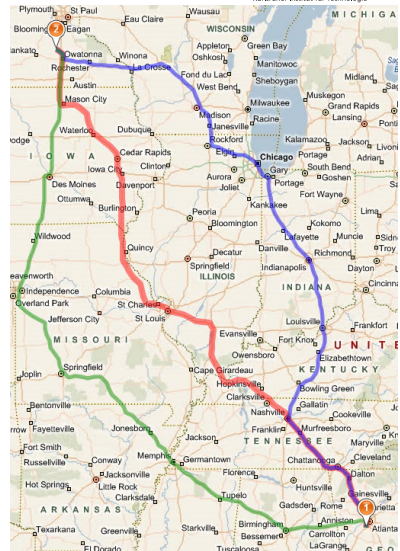
- Finde **gute Alternativen** in einem Transportnetz

Problem:

- Der kürzeste Weg ist wohldefiniert
- Was aber ist eine gute Alternative?
- Problem erscheint rein heuristischer Natur

Ziele:

- Lieber keine als schlechte Routen zeigen
- Sollte nicht deutlich langsamer als Punkt-zu-Punkt sein



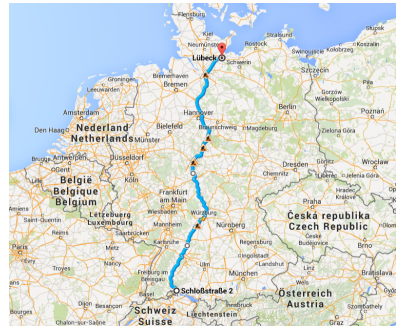
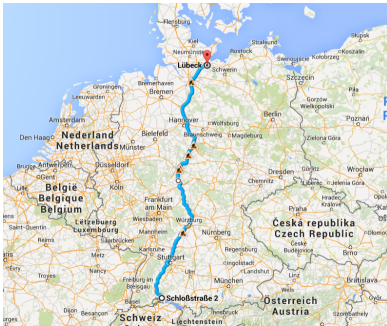
Berechne top- k kürzeste Wege

- + Wege sind möglichst kurz
- Oft erst für hohes k relevant

Erste Ansätze

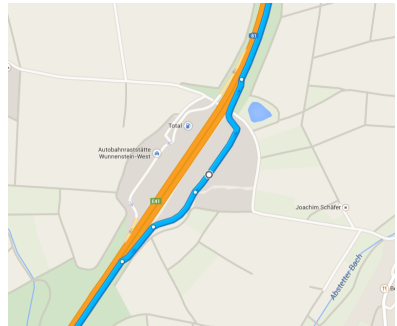
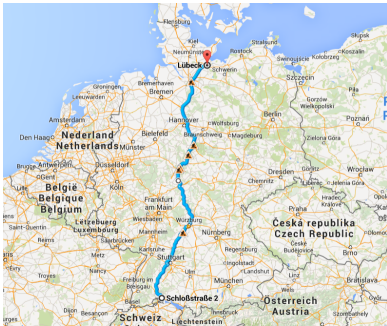
Berechne top- k kürzeste Wege

- + Wege sind möglichst kurz
- Oft erst für hohes k relevant



Berechne top- k kürzeste Wege

- + Wege sind möglichst kurz
- Oft erst für hohes k relevant

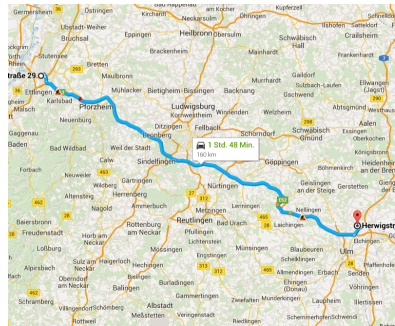
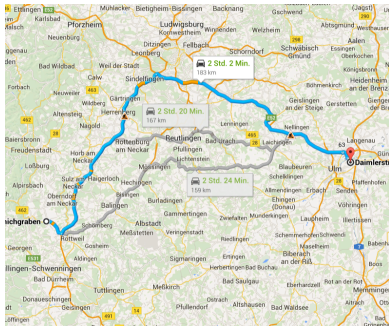


Optimiere verschiedene Metriken

- z.B. schnellster und kürzester Weg
- + bedient verschiedene Vorlieben
- verpasst interessante Alternativen

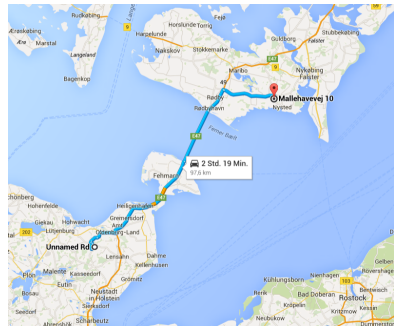
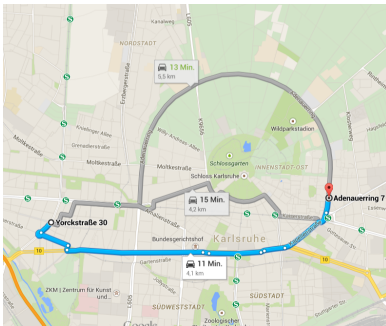
Optimiere verschiedene Metriken

- z.B. schnellster und kürzester Weg
- + bedient verschiedene Vorlieben
- verpasst interessante Alternativen



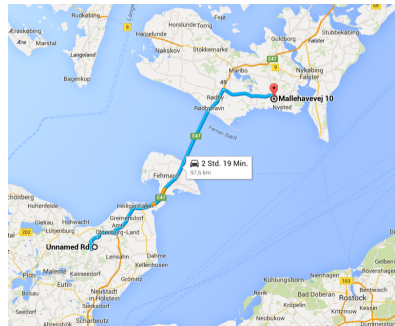
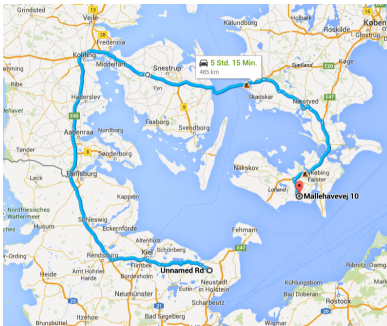
Disjunkte Pfade

- + stark unterschiedliche Pfade
- verpasst interessante Alternativen



Disjunkte Pfade

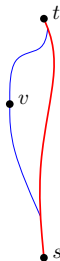
- + stark unterschiedliche Pfade
- verpasst interessante Alternativen



Fortgeschrittene Ansätze

Via-Knoten

- Nutze dritten Knoten
- Berechne zusammengesetzten Pfad
- Problem: **Woher** kommt dieser Knoten?



Simulierter Stau

- **Verlangsame** einzelne Segmente künstlich
- Gefahr vieler kleiner Umleitungen



Und wie?

- Was macht eine gute Alternative aus?
- Wie berechnen wir sie schnell?

Intuition

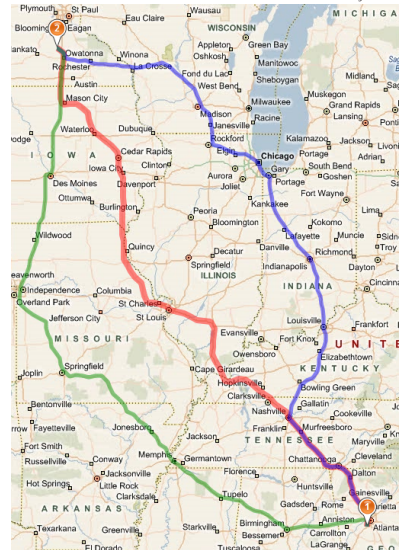
Alternativen sollten erfüllen:

- Nicht viel länger als der kürzeste Weg
- Signifikant verschieden

Erste Idee:

- Finde einen Pfad, der Länge und **Gemeinsamkeit minimiert**
 - Maximal $x\%$ länger
 - Teilt maximal $y\%$

Ist das genug?



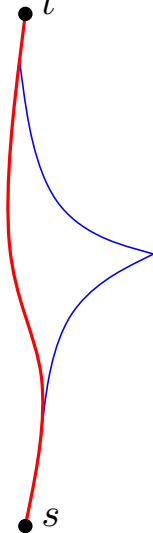
Das dritte Kriterium

Problem:

- Routen können seltsam aussehen
- Sogar wenn sie verschieden und kurz sind
- Lokale Umwege
- „Diese Strecke macht doch keinen Sinn. Das kann ich besser!“

Idee:

- Kurze Subpfade müssen kürzeste Pfade sein
 - Beliebige Paare von Knoten auf dem Pfad sollen kleinen Stretch haben
- ⇒ Keine unnötigen lokalen Umwege



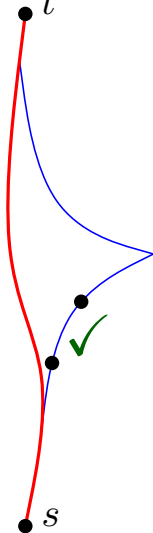
Das dritte Kriterium

Problem:

- Routen können seltsam aussehen
- Sogar wenn sie verschieden und kurz sind
- Lokale Umwege
- „Diese Strecke macht doch keinen Sinn. Das kann ich besser!“

Idee:

- Kurze Subpfade müssen kürzeste Pfade sein
 - Beliebige Paare von Knoten auf dem Pfad sollen kleinen Stretch haben
- ⇒ Keine unnötigen lokalen Umwege



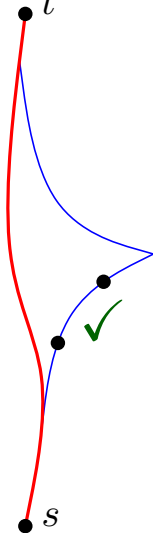
Das dritte Kriterium

Problem:

- Routen können seltsam aussehen
- Sogar wenn sie verschieden und kurz sind
- Lokale Umwege
- „Diese Strecke macht doch keinen Sinn. Das kann ich besser!“

Idee:

- Kurze Subpfade müssen kürzeste Pfade sein
 - Beliebige Paare von Knoten auf dem Pfad sollen kleinen Stretch haben
- ⇒ Keine unnötigen lokalen Umwege



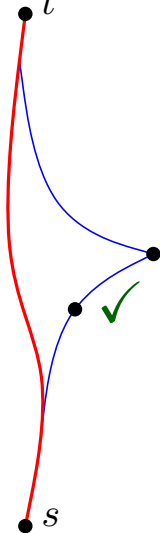
Das dritte Kriterium

Problem:

- Routen können seltsam aussehen
- Sogar wenn sie verschieden und kurz sind
- Lokale Umwege
- „Diese Strecke macht doch keinen Sinn. Das kann ich besser!“

Idee:

- Kurze Subpfade müssen kürzeste Pfade sein
 - Beliebige Paare von Knoten auf dem Pfad sollen kleinen Stretch haben
- ⇒ Keine unnötigen lokalen Umwege



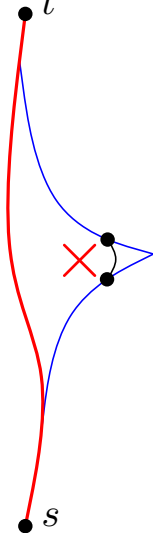
Das dritte Kriterium

Problem:

- Routen können seltsam aussehen
- Sogar wenn sie verschieden und kurz sind
- Lokale Umwege
- „Diese Strecke macht doch keinen Sinn. Das kann ich besser!“

Idee:

- Kurze Subpfade müssen kürzeste Pfade sein
 - Beliebige Paare von Knoten auf dem Pfad sollen kleinen Stretch haben
- ⇒ Keine unnötigen lokalen Umwege



Idee

Alternativ-Pfade sollen hinreichend verschieden sein.

- P ist die Kantenmenge des kürzesten s - t -Pfads
- P' ist die Kantenmenge des alternativen s - t -Pfads
- $\ell(Q)$ ist die Summe über alle Kanten-Längen in Q
- Eine Alternative ist γ -verschieden, wenn

$$\ell(P \cap P') \leq \gamma \ell(P)$$

Idee

Alternativ-Pfade sollen keine großen Umwege sein.

- Sei $\text{dist}(s, t)$ die Länge des kürzesten s - t -Pfads
- Sei P' der alternative s - t -Pfad und $\ell(P')$ seine Länge
- P' hat einen **Stretch** von höchstens ε , wenn

$$\ell(P') \leq (1 + \varepsilon) \text{dist}(s, t)$$

Kriterium für die Vorlesung vereinfacht, siehe [\[ADGW13\]](#) für Details.

Idee

Alternativ-Pfade sollen lokal sinnvoll sein.

- Sei P' der alternative Pfad und P'_{xy} der Subpfad von x bis y
- P' besteht den **T -Test**, wenn für alle P'_{xy} mit $\ell(P'_{xy}) \leq T$ gilt, dass P'_{xy} ein kürzester x - y -Pfad ist
- P' ist **α -lokal-optimal**, wenn er den Test für $T = \alpha \cdot \text{dist}(s, t)$ besteht

Kriterium für die Vorlesung vereinfacht, siehe [\[ADGW13\]](#) für Details.

- **Ziel:** Finde Pfad, der γ -verschieden ist, einen Stretch ε hat und α -lokal-optimal ist.

- **Ziel:** Finde Pfad, der γ -verschieden ist, einen Stretch ε hat und α -lokal-optimal ist.
- γ , ε , und α sind Tuningparameter des Problems
- Typische Werte sind: $\alpha = 25\%$, $\varepsilon = 25\%$, $\gamma = 80\%$

- **Ziel:** Finde Pfad, der γ -verschieden ist, einen Stretch ε hat und α -lokal-optimal ist.
- γ , ε , und α sind Tuningparameter des Problems
- Typische Werte sind: $\alpha = 25\%$, $\varepsilon = 25\%$, $\gamma = 80\%$
- Bei mehreren Alternativen:
Neue Alternative muss γ -verschieden von der Vereinigung aller bisher gefundenen Pfade sein.

Weiteres Vorgehen

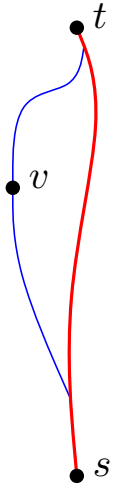
- Bisher: Wir haben gültige Alternativen formalisiert
- Nun: Wie finden wir gute Alternativen?

- Bisher: Wir haben gültige Alternativen formalisiert
- Nun: Wie finden wir gute Alternativen?
- **Vorgehen:**
 - Finde Kandidaten-Pfad nach einem heuristischen Verfahren
 - Teste, ob der Pfad eine gute Alternative ist
- Findet nicht immer eine Alternative

- Bisher: Wir haben gültige Alternativen formalisiert
- Nun: Wie finden wir gute Alternativen?
- **Vorgehen:**
 - Finde Kandidaten-Pfad nach einem heuristischen Verfahren
 - Teste, ob der Pfad eine gute Alternative ist
- Findet nicht immer eine Alternative
- **Problem:** Wie effizient testen?
- Nicht trivial
- Der Test braucht nicht vernachlässigbare Rechenzeit
- Details nicht in der Vorlesung

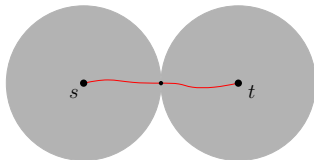
Single via paths:

- **Konkatenation** von zwei kürzesten Pfaden: $s-v$ und $v-t$
- Eigenschaften:
 - **linear** viele Pfade
 - einzelner Pfad ist definiert durch Via-Knoten v (Notation: P_v)
 - lokal optimal von s nach v und v nach t
 - Verletzungen nur um v herum
 - Alternativen können effizient berechnet werden, sobald v bekannt ist



Relaxiertes Stoppkriterium:

- Suche bidirektional
- Stoppe, sobald Radien $> (1 + \epsilon)\ell(\text{OPT})$



Finde Alternative:

- Für alle v , die von beiden Suchen gescannt wurden:
- Teste, ob P_v den Anforderungen genügt
- Finde **besten** P_v anhand einer Optimierungsfunktion

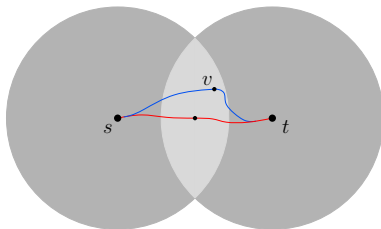
Problem:

- Anzahl der Kandidaten ist **sehr hoch**
- Interaktion mit Beschleunigungstechniken?

Prinzip Bidirektionale Suche

Relaxiertes Stoppkriterium:

- Suche bidirektional
- Stoppe, sobald Radien $> (1 + \epsilon)\ell(\text{OPT})$



Finde Alternative:

- Für alle v , die von beiden Suchen gescannt wurden:
- Teste, ob P_v den Anforderungen genügt
- Finde **besten** P_v anhand einer Optimierungsfunktion

Problem:

- Anzahl der Kandidaten ist **sehr hoch**
- Interaktion mit Beschleunigungstechniken?

Alternativen mit CH

Idee:

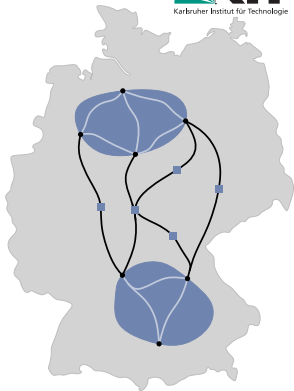
- Lasse Vorwärts- und Rückwärtssuche laufen, bis Stretch erreicht ist
- Entpacke Pfad für jeden Knoten im gemeinsamen Suchraum
- Teste Pfad

Probleme:

- gegenläufige Ziele
 - CH will Suchraum klein haben
 - Alternativrouten brauchen genug Kandidaten
- Tests und Pfadentpackungen kosten Zeit

„Lösung“:

- Relaxiere vereinzelt auch Kanten, die den Suchraum verlassen
- Je weiter man den Suchraum verlässt, desto besser werden die gefundenen Pfade, aber desto schlechter werden die Laufzeiten
- Siehe [\[ADGW13\]](#) für Details



Idee:

- Analog zu Dijkstra
- Lasse Vorwärts- und Rückwärtssuche laufen, bis Stretch erreicht ist
- Teste Pfad

Probleme:

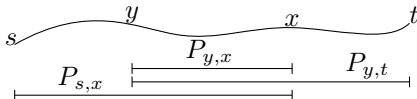
- Reach dünnt Suchraum schlechter aus als CH
- Mehr Kandidaten
- Bessere Alternativen
- Aber langsamer

Problem mit Via-Knoten:

- Lokale Optimalität am Via-Knoten verletzt

Problem mit Via-Knoten:

- Lokale Optimalität am Via-Knoten verletzt



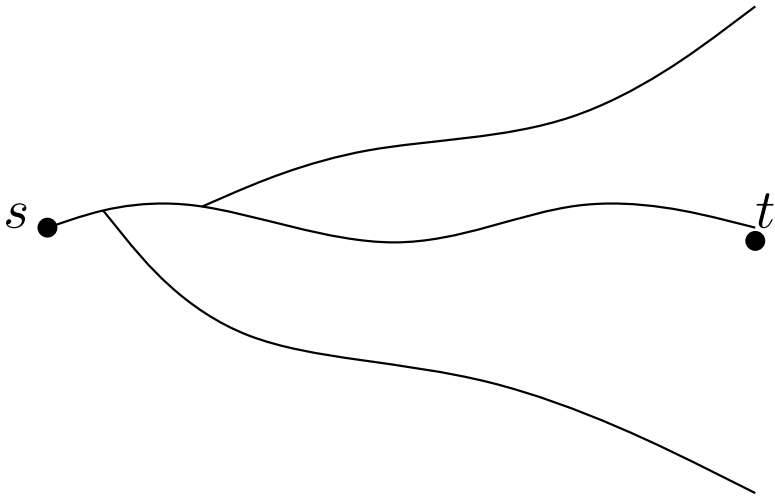
Idee:

- Suche zwei kürzeste Pfade $P_{s,y}$ und $P_{x,t}$
- mit $P_{x,y} \subseteq P_{s,y}$ und $P_{x,y} \subseteq P_{x,t}$
- $P_{x,y}$ heißt **Plateau**
- Alternative besteht T -Test für $T \leq \ell(P_{x,y})$
- Langes Plateau \rightarrow gute Alternative
- Randfall Via-Knoten: $x = y$

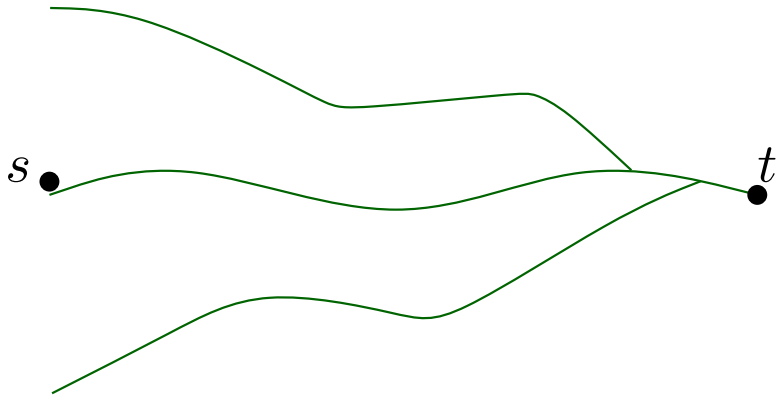
s ●

t
●

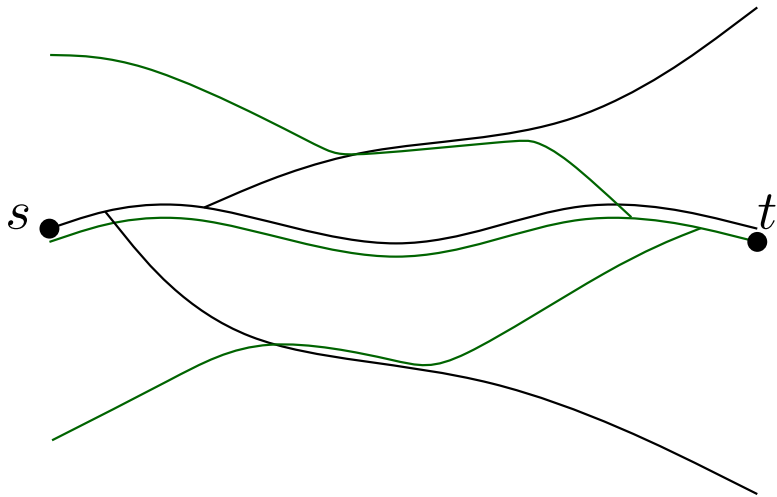
Plateau

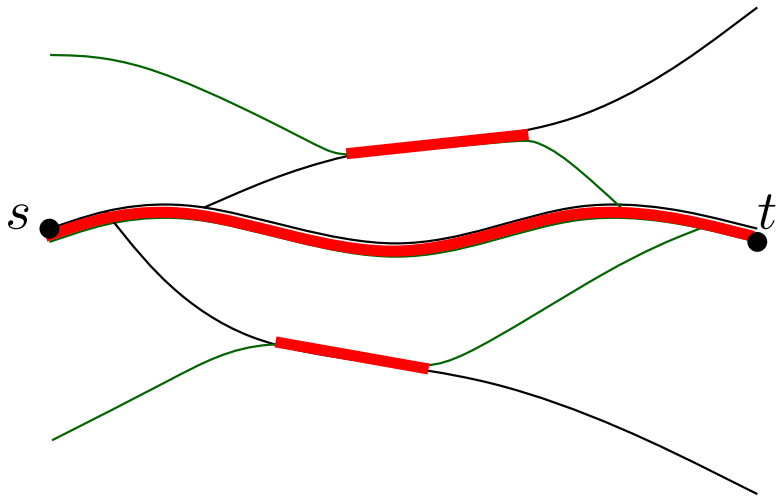


Plateau

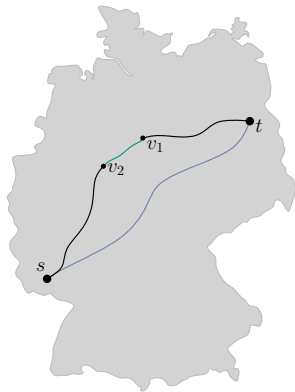


Plateau





Plateaus berechnen



- Benötigt optimale Suchbäume
- Einfach mit Dijkstra
- CH: komplex, aber möglich

(Grundlage: vollständiges Entpacken des Suchraums: [Kob13])

- Via-Knoten einfacher und schneller zu berechnen
 - Aber: Mehr Via-Knoten als Plateaus
 - d.h. mehr Zulässigkeitstests mit Via-Knoten
-
- Jede Plateau-Alternative ist auch eine Via-Alternative
 - **Beweis:**
 - Sei $P_{s,t} = s \rightarrow y \rightarrow x \rightarrow t$ die Plateau-Alternative
 - Für jeden Knoten $v \in P_{y,x}$ ist $P_{s,t}$ eine Via-Alternative mit Via-Knoten v , da $P_{s,v}$ und $P_{v,t}$ kürzeste Pfade sind

Idee: [BDGS11]

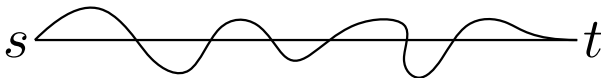
- Suche kürzesten Pfad P_1
- Simuliere Stau auf P_1
- Suche neuen kürzesten Pfad P_2 , der Stau in Betracht zieht
- Simuliere Stau auf P_1 und P_2
- Suche neuen kürzesten Pfad P_3 , der Stau in Betracht zieht
- ...
- Wiederhole, bis Alternativen zu lang werden

Idee: [BDGS11]

- Suche kürzesten Pfad P_1
- Simuliere Stau auf P_1
- Suche neuen kürzesten Pfad P_2 , der Stau in Betracht zieht
- Simuliere Stau auf P_1 und P_2
- Suche neuen kürzesten Pfad P_3 , der Stau in Betracht zieht
- ...
- Wiederhole, bis Alternativen zu lang werden

Englischer Fachbegriff: Penalty Method

Ist das eine gute Alternative?



- Nicht lokal optimal
- Bestrafe nicht nur die Kanten auf dem Pfad, sondern auch die inzidenten Kanten

Strafterme:

- Einfluss von additiven Straftermen hängt vom Graphen ab
mehr Grad-2-Knoten \rightarrow größerer Einfluss
- **Lösung:** Multiplikative Strafterme
- Anders formuliert: Setzt die Geschwindigkeit gleichmäßig runter anstatt jede Kante um x Sekunden zu verlängern

Wie implementieren?

- Benutze 3-Phasen-Vorberechnungstechnik
- Nur Teile der Metrik ändern sich → kann ausgenutzt werden
- [KRS13] baut auf MLD/CRP auf

algorithm	first		second		third	
	time [ms]	success rate [%]	time [ms]	success rate [%]	time [ms]	success rate [%]
X-BDV	11 451.5	94.5	12 225.9	80.6	13 330.9	59.6
CRP- π	130.0	96.3	130.0	84.0	130.0	62.9
X-REV	20.4	91.3	33.6	70.3	42.6	43.0
HiDAR	18.2	91.5	18.2	75.7	18.2	55.9
X-CHV-v1	16.9	90.7	20.3	70.1	22.1	42.3
X-CHV-v2	3.1	58.2	3.6	28.6	3.9	10.9

- X-BDV: Bidirektionaler Dijkstra
- X-REV: Reach
- X-CHV-v1 und X-CHV-v2: Zwei CH-Varianten, v1 sucht mehr Kanten außerhalb des Suchraums ab
- HiDAR: Plateau-CH
- CRP- π : Simulierter Stau

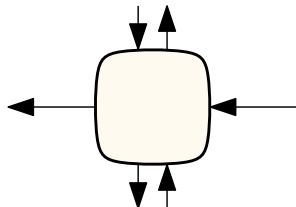
- Eine gute Alternative ist:
 - Nicht viel länger
 - Lokal-optimal
 - Hinreichend verschieden
- Alternativen findet man durch
 - Kandidaten heuristisch aufzählen
 - Schlechte Kandidaten herausfiltern
- Kandidaten findet man mit
 - Via-Knoten
 - Plateaus
 - Simuliertem Stau

Abbiegekosten

Abbiegeverbote/-kosten

Bisher:

- Kreuzungen → Knoten
- Straßen → Kanten



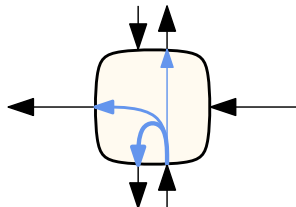
Abbiegeverbote/-kosten

Bisher:

- Kreuzungen → Knoten
- Straßen → Kanten

Aber:

- Abbiegen manchmal verboten
- Linksabbiegen teurer als rechts
- Kosten für U-Turns hoch
- Wurde als einfaches Modellierungsdetail abgetan



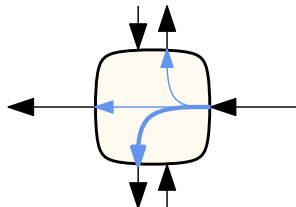
Abbiegeverbote/-kosten

Bisher:

- Kreuzungen → Knoten
- Straßen → Kanten

Aber:

- Abbiegen manchmal verboten
- Linksabbiegen teurer als rechts
- Kosten für U-Turns hoch
- Wurde als einfaches Modellierungsdetail abgetan



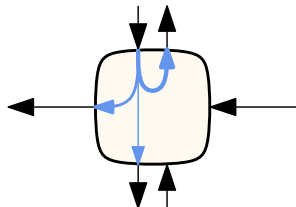
Abbiegeverbote/-kosten

Bisher:

- Kreuzungen → Knoten
- Straßen → Kanten

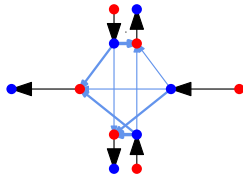
Aber:

- Abbiegen manchmal verboten
- Linksabbiegen teurer als rechts
- Kosten für U-Turns hoch
- Wurde als einfaches Modellierungsdetail abgetan



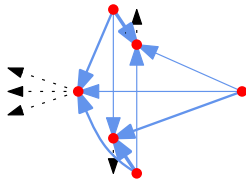
Möglichkeit I: Expandierter Graph

- Vergrößern des Graphen durch Ausmodellierung
- Kantenbasierter Graph:
 - Zwei Knoten pro Straße (Tail und Head)
 - Straße → Kante von Tail zu Head
 - Turn → Kante von Head zu Tail
- Redundante Information



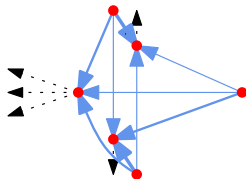
Möglichkeit I: Expandierter Graph

- Vergrößern des Graphen durch Ausmodellierung
 - Kantenbasierter Graph:
 - Zwei Knoten pro Straße (Tail und Head)
 - Straße → Kante von Tail zu Head
 - Turn → Kante von Head zu Tail
 - Redundante Information
- Kontrahiere Head-Knoten



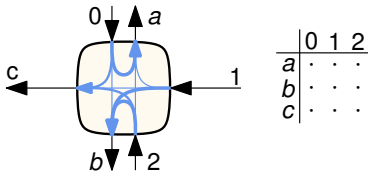
Möglichkeit I: Expandierter Graph

- Vergrößern des Graphen durch Ausmodellierung
 - Kantenbasierter Graph:
 - Zwei Knoten pro Straße (Tail und Head)
 - Straße → Kante von Tail zu Head
 - Turn → Kante von Head zu Tail
 - Redundante Information
- Kontrahiere Head-Knoten



Möglichkeit II: Kompaktes Modell

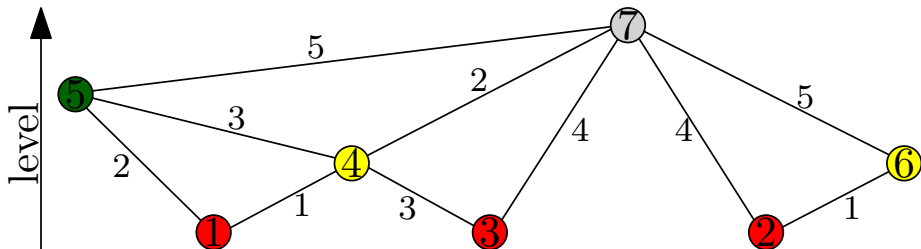
- Behalte Kreuzungen als Knoten
- Speichere Abbiegetabelle
Abb.: Einfahrt × Ausfahrt → Kosten
- Beobachtung: Viele Knoten mit identischer Abbiegetabelle
- Speichere jede Tabelle einmal, Knoten speichern Tabellen-ID



Wdh: Contraction Hierarchies

Vorbereitung:

- Ordne Knoten nach Wichtigkeit
- Kontrahiere Knoten in dieser Reihenfolge
- Füge Shortcuts hinzu
- Weise den Knoten Levels zu



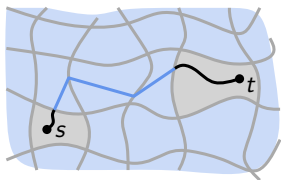
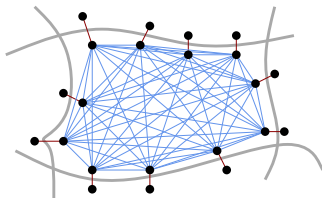
Wdh: Multilevel-Dijkstra (MLD)

Idee:

- Partitioniere Graphen
- Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay-Graph:

- Randknoten
- Cliques in jeder Zelle
- Schnittkanten



Suchgraph:

- Start- und Zielzelle...
- ...plus Overlay-Graph
- (bidirektionaler) Dijkstra

Optimierung: mehrere Levels

Dijkstra:

- Funktioniert ohne Anpassung
- Mehr Knoten zu scannen
- Faktor 3–4 langsamer

Dijkstra:

- Funktioniert ohne Anpassung
- Mehr Knoten zu scannen
- Faktor 3–4 langsamer

CH:

- Funktioniert ohne Anpassung
- Aber: größere Anzahl Knoten/Kanten erhöht Vorberechnungszeit

Dijkstra:

- Funktioniert ohne Anpassung
- Mehr Knoten zu scannen
- Faktor 3–4 langsamer

CH:

- Funktioniert ohne Anpassung
- Aber: größere Anzahl Knoten/Kanten erhöht Vorberechnungszeit

MLD:

- Anzahl Schnittkanten erhöht sich
- *jetzige* Schnittkanten = *ehemalige* Schnittknoten
(Eventuell Wechsel zu Knotenseparatoren sinnvoll?)

Dijkstra:

- Turns müssen in den Suchalgorithmus integriert werden
- Kreuzungen können mehrfach gescannt werden
label-correcting bzgl. Kreuzung, label-setting bzgl. Eingangs-/Ausgangspunkten
- Jede **Kante** wird höchstens einmal gescannt
- Suchraum gleich zu kantenbasiertem Modell
simuliert Dijkstra auf kantenbasiertem Graphen
- Vorteil: weniger Speicher für den Graphen

Kompaktes Modell

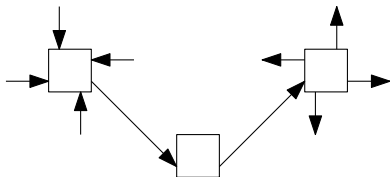
CH:

- Zeugensuche wird komplizierter

Kompaktes Modell

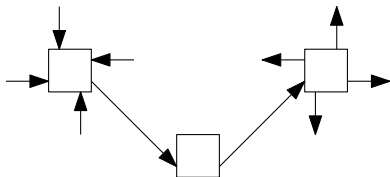
CH:

- Zeugensuche wird komplizierter
 - Eine Zeugensuche pro Paar eingehender und ausgehender Kanten



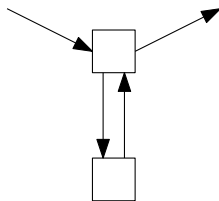
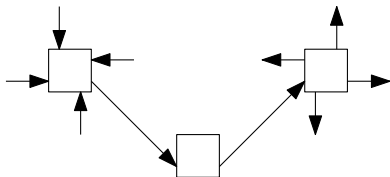
CH:

- Zeugensuche wird komplizierter
 - Eine Zeugensuche pro Paar eingehender und ausgehender Kanten
 - Es können Self-Loops entstehen



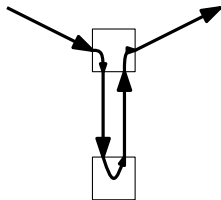
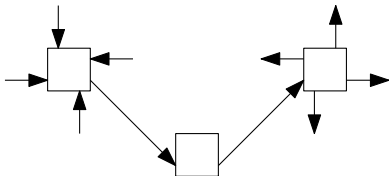
CH:

- Zeugensuche wird komplizierter
 - Eine Zeugensuche pro Paar eingehender und ausgehender Kanten
 - Es können Self-Loops entstehen



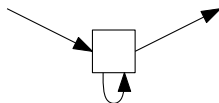
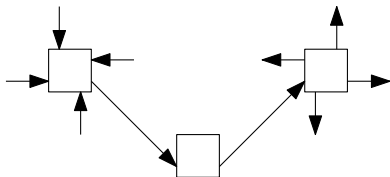
CH:

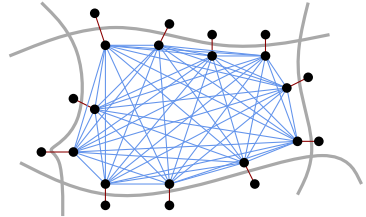
- Zeugensuche wird komplizierter
 - Eine Zeugensuche pro Paar eingehender und ausgehender Kanten
 - Es können Self-Loops entstehen



CH:

- Zeugensuche wird komplizierter
 - Eine Zeugensuche pro Paar eingehender und ausgehender Kanten
 - Es können Self-Loops entstehen








MLD:




- Schnittkanten bleiben erhalten
 - Schnittkante \rightarrow 2 Knoten auf Overlay
 - Turns müssen nur auf unterstem Level beachtet werden
 - Auf Overlay-Graphen: normaler Dijkstra
- \Rightarrow Einfache Anpassung, aber zusätzliche Fallunterscheidung in der Query

U-Turn		Customization		Queries	
cost	Algorithm	time [s]	[MB]	#scans	time [ms]
1 s	MLD [$2^8:2^{12}:2^{16}:2^{20}$]	5.8	61.7	3556	1.18
	CH expanded	3407.4	880.6	550	0.18
	CH compact	849.0	132.5	905	0.19
100 s	MLD [$2^8:2^{12}:2^{16}:2^{20}$]	7.5	61.7	3813	1.28
	CH expanded	5799.2	931.1	597	0.21
	CH compact	23774.8	304.0	5585	2.11

Beobachtung:

- (Metrikabhängige) CH-Ordnung problematisch bei Turns
- Besser: Ordnung an Turns anpassen (CH expanded vs. compact)
- MLD robust

-  Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck.
Alternative routes in road networks.
ACM Journal of Experimental Algorithmics, 18(1):1–17, 2013.
-  Roland Bader, Jonathan Dees, Robert Geisberger, and Peter Sanders.
Alternative route graphs in road networks.
In *Proceedings of the 1st International ICST Conference on Theory and Practice of Algorithms in (Computer) Systems (TAPAS'11)*, volume 6595 of *Lecture Notes in Computer Science*, pages 21–32. Springer, 2011.
-  Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck.
Customizable route planning in road networks.
Transportation Science, 51(2):566–591, 2017.

-  Robert Geisberger and Christian Vetter.
Efficient routing in road networks with turn costs.
In Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11), volume 6630 of *Lecture Notes in Computer Science*, pages 100–111. Springer, 2011.
-  Moritz Kobitzsch.
HiDAR: An alternative approach to alternative routes.
In Proceedings of the 21st Annual European Symposium on Algorithms (ESA'13), volume 8125 of *Lecture Notes in Computer Science*, pages 613–624. Springer, 2013.
-  Moritz Kobitzsch, Marcel Radermacher, and Dennis Schieferdecker.
Evolution and evaluation of the penalty method for alternative graphs.
In Proceedings of the 13th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'13), OpenAccess Series in Informatics (OASISs), pages 94–107, 2013.