Karlsruher Institut für Technologie

Algorithmen für Routenplanung

7. Vorlesung, Sommersemester 2012

Daniel Delling | 16. Mai 2012

MICROSOFT RESEARCH SILICON VALLEY

Hierarchische Techniken

Ideensammlung:

- identifiziere wichtige Knoten mit Zentralitätsmaß
- überspringe unwichtige Teile des Graphen

Overlay Graph

Gegeben

- Eingabegraph $G = (V, E, \text{len})$
- Knotenmenge $V \supseteq V_L$

Berechne

- Berechne $G_L = (V_L, E_L, \text{len}_L)$, so dass Distanzen in G_L wie in G

Multilevel-Overlay Graphs

Gegeben

- Eingabegraph $G = (V, E, \text{len})$
- Folge $V := V_0 \supseteq V_1 \supseteq \ldots \supseteq V_L$ von Teilmengen von V.

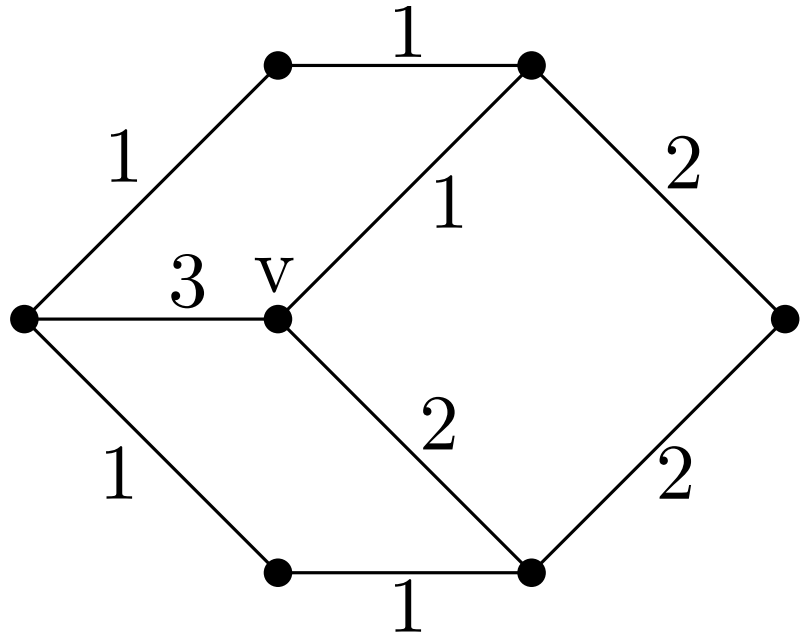
Berechne

- Folge $G_0 = (V_0, E_0, \text{len}_0), \ldots, G_L = (V_L, E_L, \text{len}_L)$ von Graphen, so dass Distanzen in G_i wie in G_0

Wahl der Knotenmenge

- Randknoten
- können auch beliebige Knotenmengen sein

Contraction Hierarchies



Zutaten:

- Ordne Knoten nach Wichtigkeit $r : V \to [0, \ldots, n-1]$
- Shortcut Operation

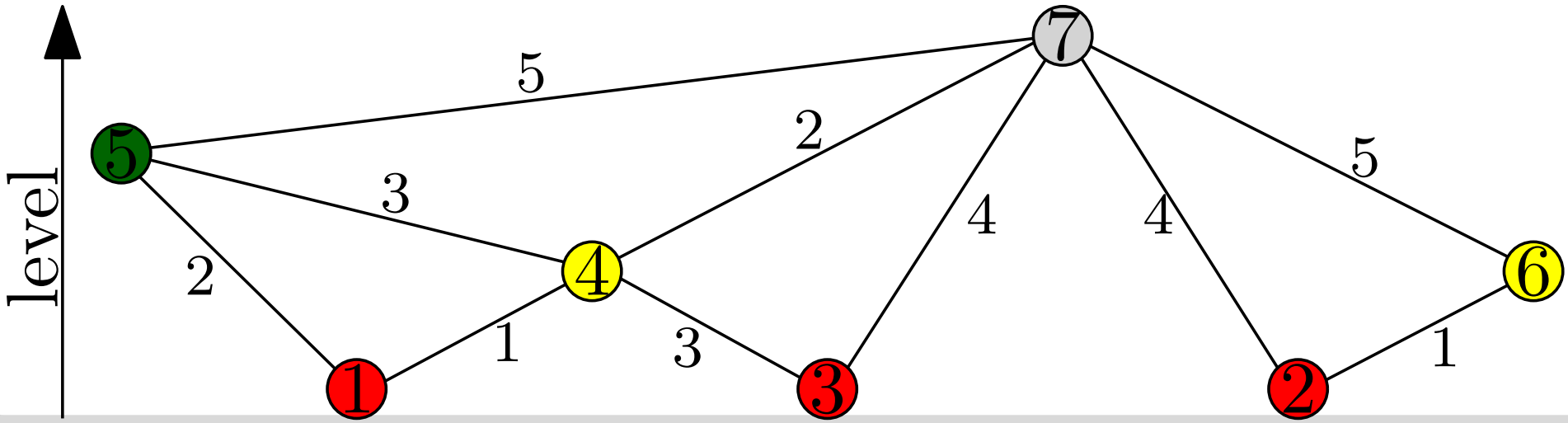
Shortcut Operation (von Knoten v):

- entferne v
- für jedes Paar u, w füge Kanten (u, w) zum Graphen wenn v auf dem einzigen kürzesten Weg von u nach w liegt
- kann durch lokale Dijkstra Suchen von den Nachbarn berechnet werden

Contraction Hierarchies

Vorberechnung:

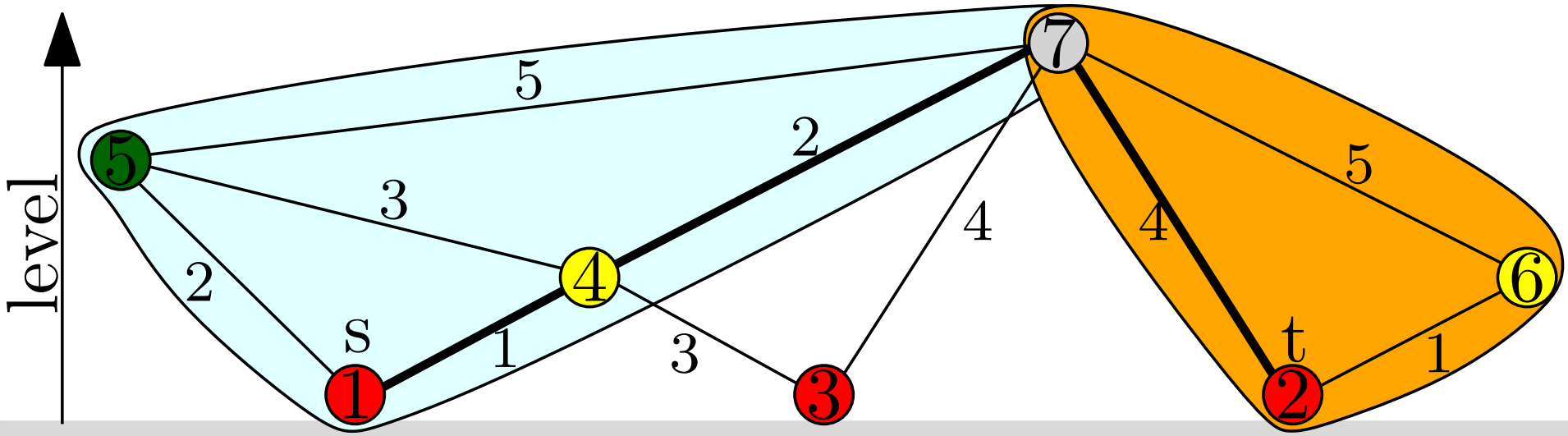
- ordne Knoten
- bearbeite in der Reihenfolge
- shortcutten des Knoten, hinzufügen von Shortcuts (A^+)
- bestimme Level des Knoten
- erzeugt einen Graph $G^+ = (V, A \cup A^+)$



Contraction Hierarchies

Anfrage

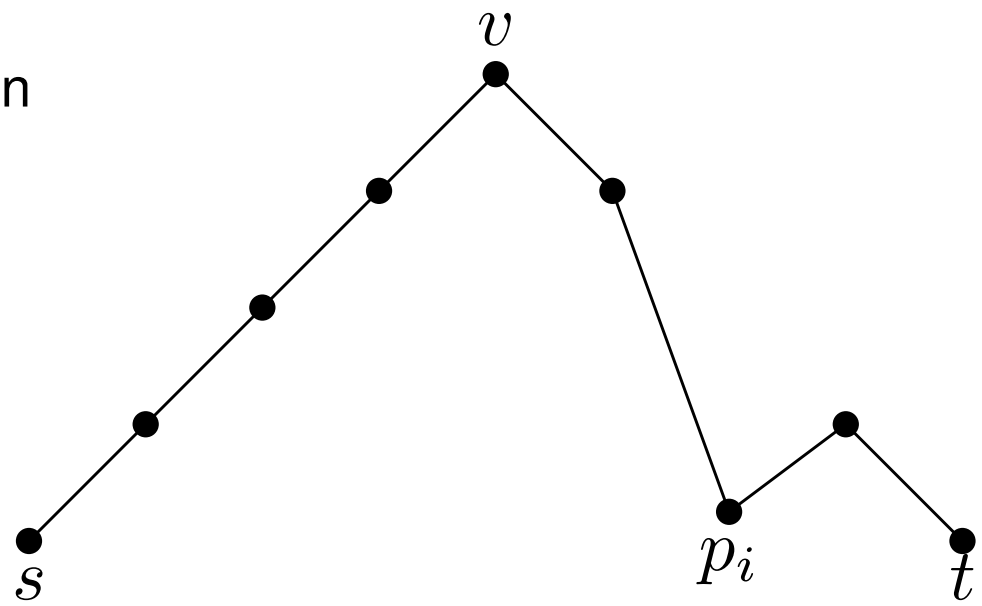
- modifizierter bidirektionaler Dijkstra
- folge nur Kanten zu wichtigeren Knoten
- konservatives Stoppkriterium
- jede Suche stoppt wenn Queue leer oder $\mathit{minKey}(Q) > \mu$



Korrektheit

Beobachtung:

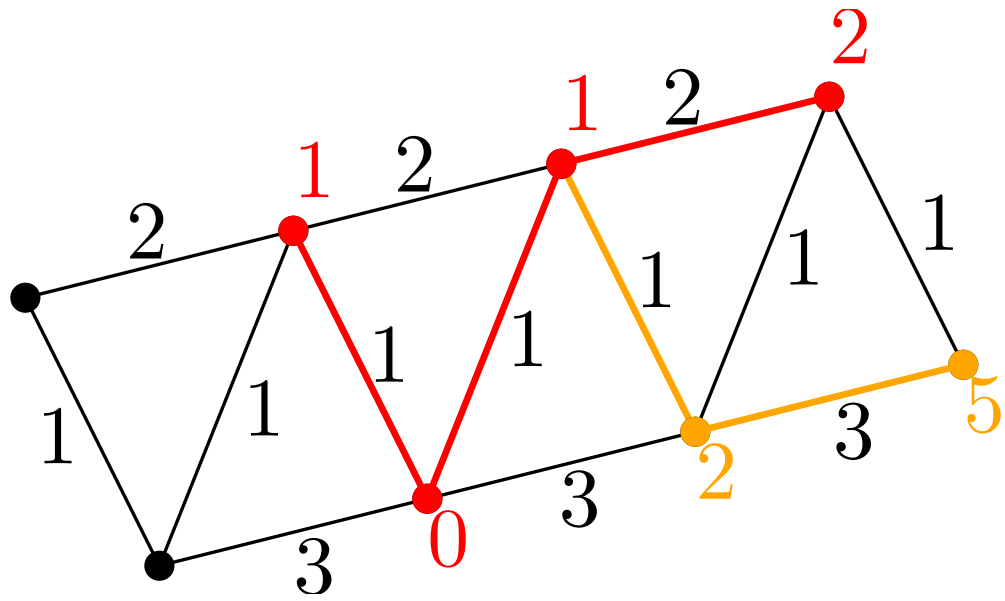
- der kürzeste s–t Weg P existiert auch in G^+
  (wir fügen nur Kanten hinzu)
- aber wir suchen nur aufwärts
- es gibt einen wichtigsten Knoten v auf P
- wir müssen zeigen, dass es einen s–aufwärts–v–abwärts–t Pfad gibt
- verändertes Stoppkriterium:
  es kann $v = s$ oder $v = t$ sein



Stall-On-Demand

Beobachtung:

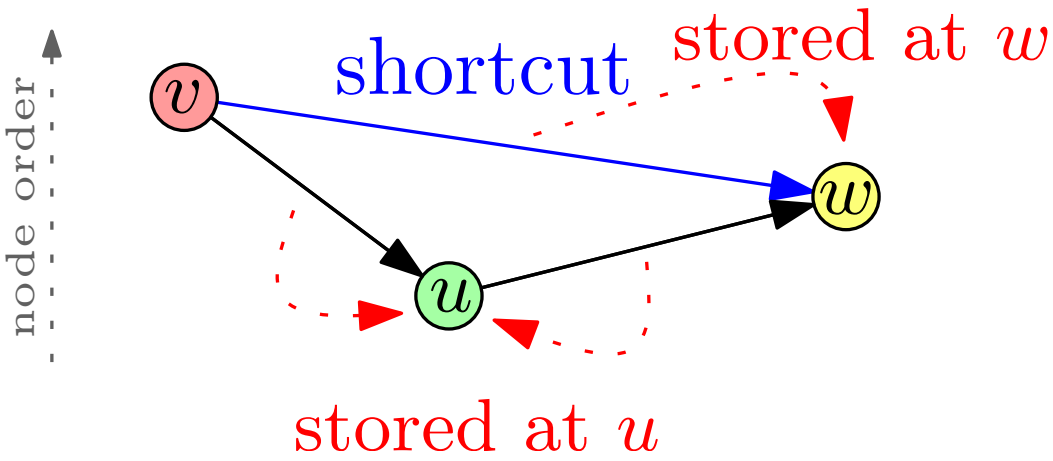
- Aufwärtssuchen können Knoten mit falscher Distanz scannen
- Heuristik versucht das durch scannen von Abwärtskanten zu verhindern
- diese Knoten werden deaktiviert, können aber wieder aktiviert werden
- reduziert Suchraum um einen Faktor 4



Datenstruktur

Suchgraph:

- normalerweise: speichere Kanten (v, w) in den Adjacenz-Arrays von v und w
- für die Suche reicht es aus, die Kante nur an den Knoten $\min\{r(v), r(w)\}$ zu speichern
- durch ungerichtete Kanten negativer Speicherverbrauch möglich (!)



Knotenordnung

Wie Knoten ordnen?

- top-down
- bottom-up

Top-Down

Vorgehen:

- bestimme alle kürzesten Wege U, die noch nicht von gewählten Knoten überdeckt sind
- U_v: alle kürzesten Wege, die v enthalten
- S_v: Menge der Startknoten von U_v, T_v der Zielknoten
- wähle Knoten v mit maximalem $|U_v|/(|S_v| + |T_v|)$
- aktualisiere $U = U \setminus U_v$
- wiederhole bis U leer

Diskussion

- n mal APSP
- kann auf APSP Zeit gedrückt werden
- sehr gute Qualität der Ordnung

Bottom-Up

Vorgehen:

- identifiziere aktuell unwichtigsten Knoten
- kontrahiere den Knoten
- wiederhole

Kriterien zum Bestimmen dieses Knotens:

- Differenz Hinzugefügter und Gelöschter Kanten
- Differenz der Originalkanten, die diese Kanten repräsentieren
- Level des Knoten

Diskussion

- schnell, da alles lokale Information
- kann on-the-fly berechnet werden
- aber nur, solange der Knotengrad gering bleibt
- Qualität der Ordnung ist schlechter (grössere Suchräume)

Range Optimierung

Idee

- Knotenordnung (z.B. durch bottom-up) gegeben
- zwei Parameter X, Y
- kontrahiere alle Knoten mit $r(v) \leq X$
- bestimme alle kürzesten Wege U die nicht von Knoten mit $r(v) > Y$ überdeckt sind
- ordne Knoten mit $X < r(v) \leq Y$ mit top-down Algorithmus

Diskussion

- verbessert Qualität der Ordnung
- schnell genug bei geschickter Wahl von X und Y
- kann ineinader verschachtelt ausgeführt werden

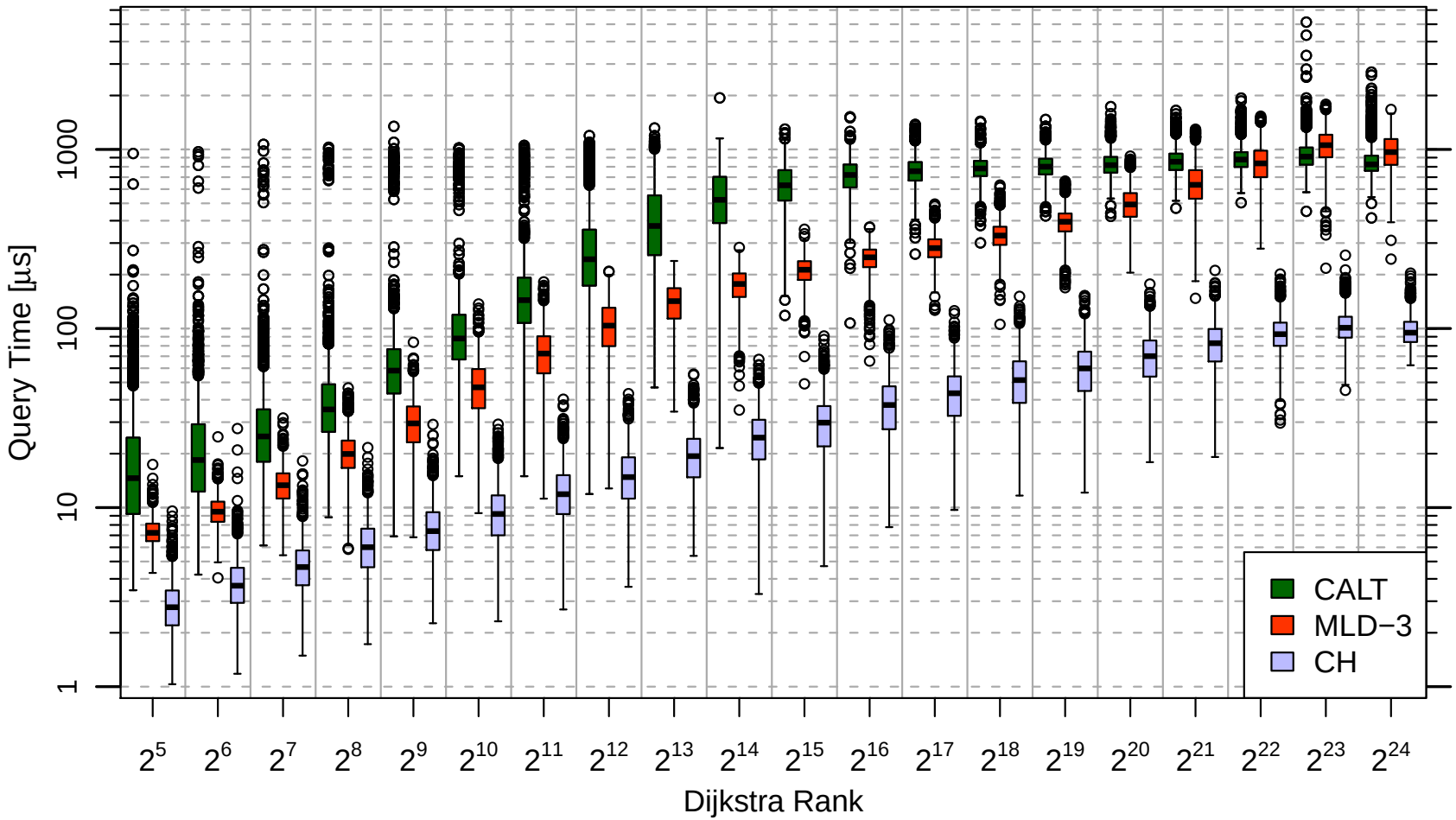
Ergebnisse

	preprocessing		query	
method	time [h:m]	space [GB]	scans	time [μs]
MLD-3	$<$ 0:01	0.4	6074	912
MLD-4	$<$ 0:01	0.4	3897	707
CH	0:02	0.4	284	96.3
CH-15	0:04	0.4	231	85.0
CH-17	0:24	0.4	217	79.7
CH-∞	5:42	0.4	200	73.9

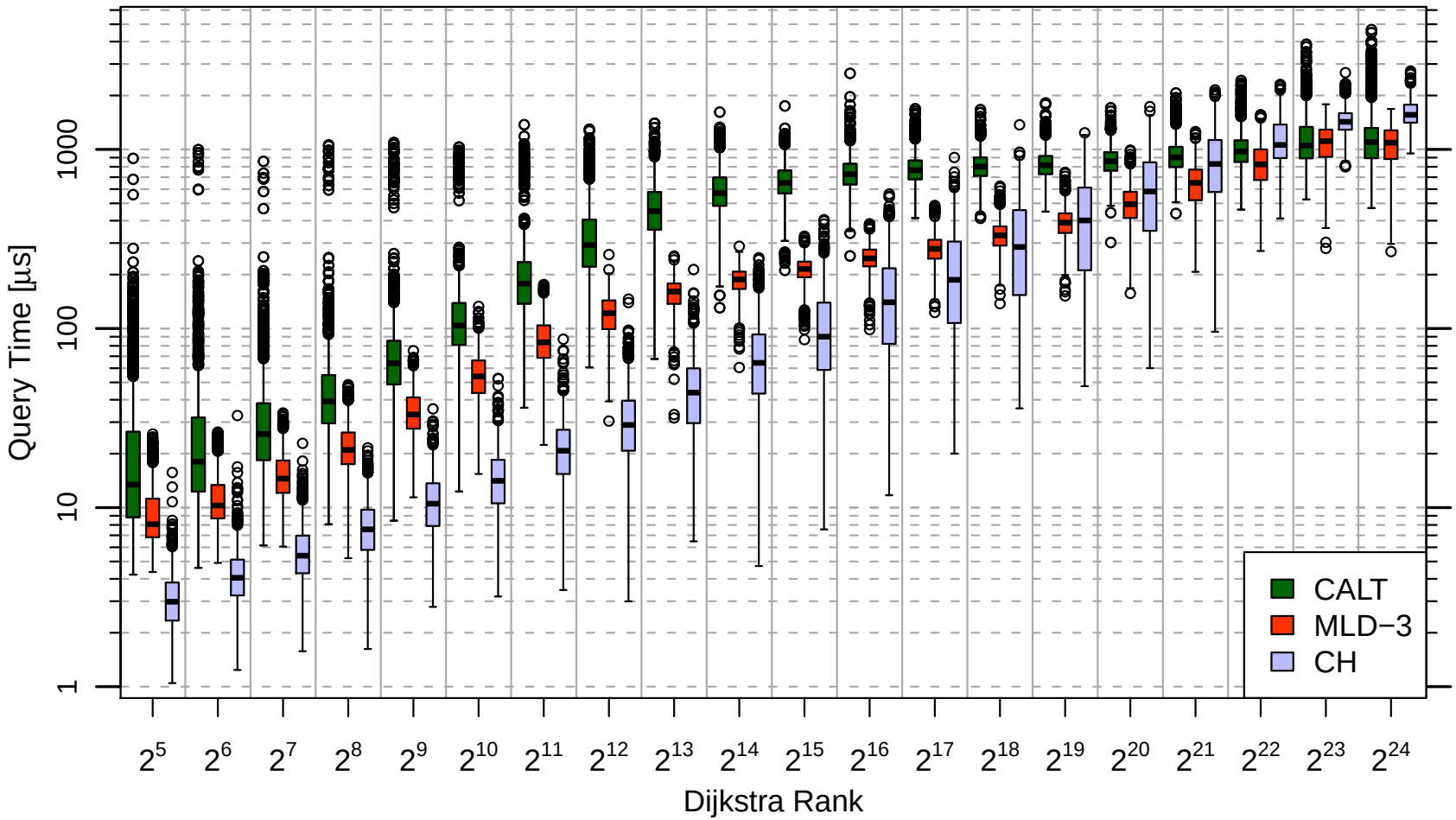
CH-x mit 2^x top-down ordnung, ∞ mit Hybrid Ordnung

- CH etwas langsamere Vorberechnung
- Faktor 10 schneller als MLD
- bottom-up Knotenordnung gut genug

Local Queries: Reisezeiten

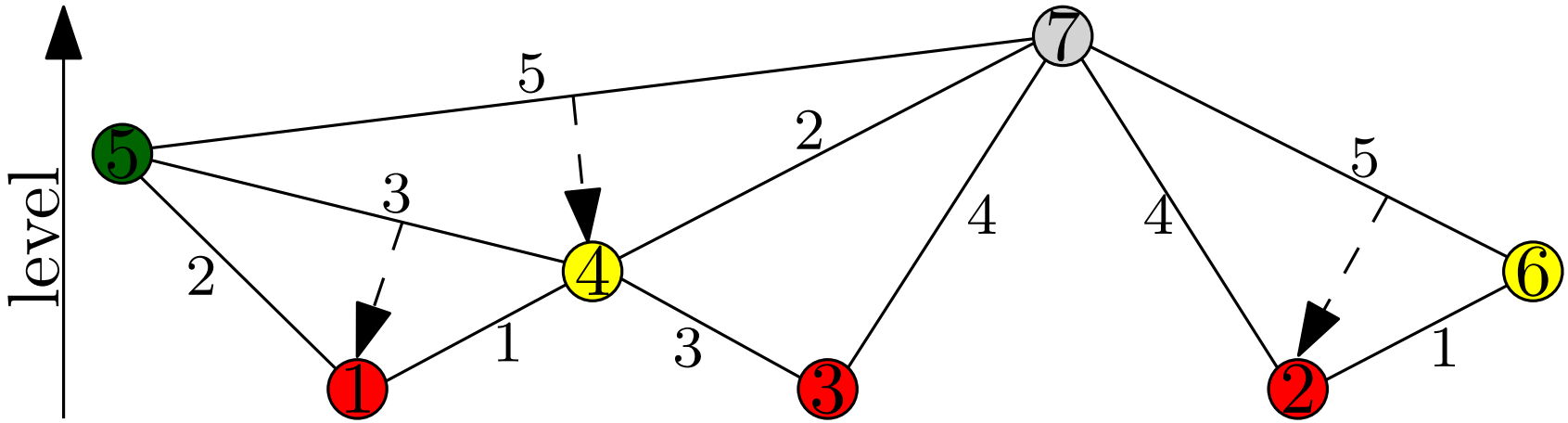


Local Queries: Reisedistanzen



Pfadentpackung

- für jeden Shortcut (u, w) eines Pfades (u, v, w), speichere Mittelknoten v an der Kante
- expandiere Pfade mittels Rekursion



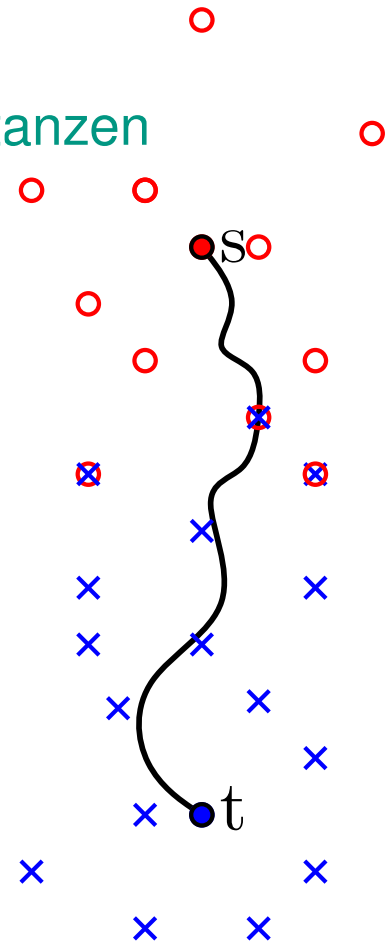
Zusammenfassung

- Knotenordnung + Shortcutting
- erzeugt n Overlay graphen
- bidirektionaler Anfragealgorithmus
- einfache und schnelle Vorberechnung
- negativer (?) Speicheroverhead
- Performance stark metrikabhängig
- hohe Beschleunigung für Reisezeiten
- in Produktion von vielen (?) Firmen

Labeling

Vorberechung:

- für jeden Knoten u, berechne zwei Label $L_f(u), L_b(u)$
- ein Label ist eine Menge von Knoten (Hubs) und Distanzen
  - $\text{dist}(u, v)$ für jeden Hub $v \in L_f(u)$
  - $\text{dist}(v, u)$ für jeden Hub $v \in L_b(u)$
- die Label müssen die cover property einhalten:
  $\forall s, t$, $L_f(s) \cap L_b(t)$ überdeckt den kürzesten s–t Pfad



s–t Anfrage:

- finde Knoten $v \in L_f(s) \cap L_b(t)$ ...
- ... der $\text{dist}(s, v) + \text{dist}(v, t)$ minimiert

Beobachtungen:

- Laufzeit hängt von Labelgrösse ab
- wie effizient berechnen?

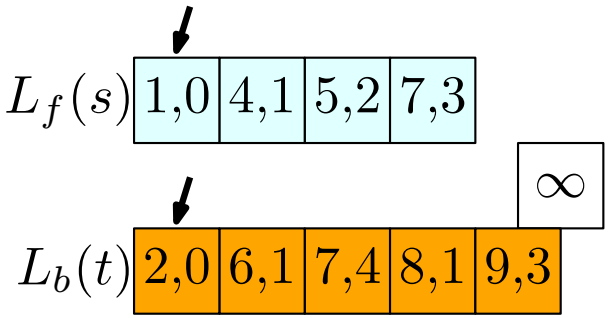
Hub Labels

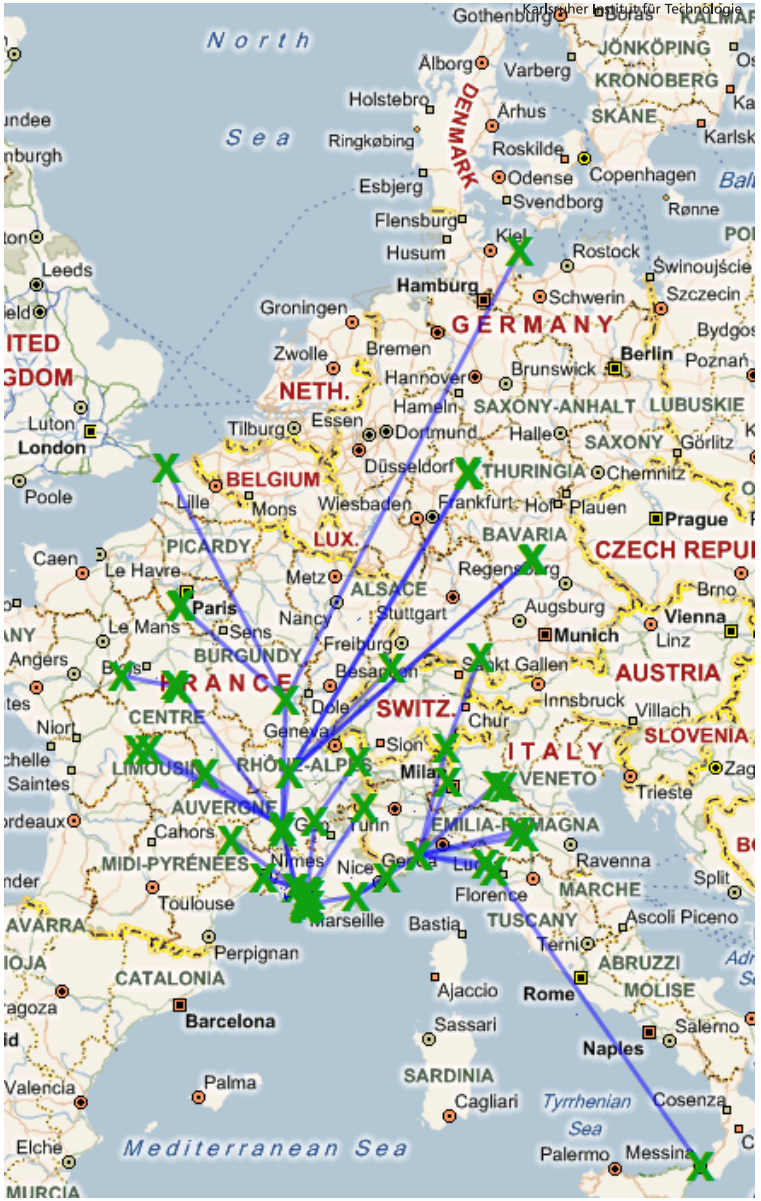
Speichern der Labels:

- als Menge von Hub,Distanz Paaren

Anfrage:

- scannen von zwei Arrays
- nur einige Speicherzugriffe nötig
- sehr hohe Lokalität

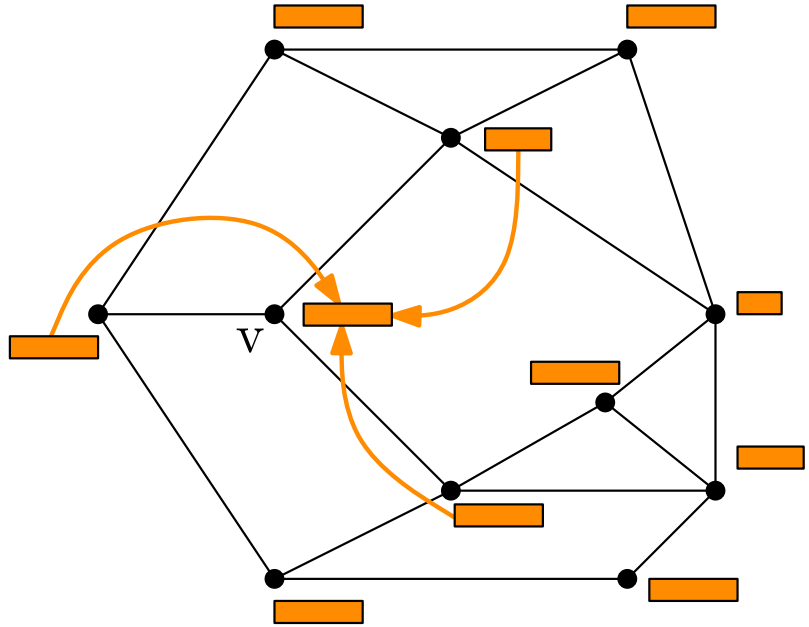




Vorberechnung

Idee:

- benutze Knotenordnung
- kontrahiere Knoten v
- berechne Labels rekursiv
- merge Labels der Aufwärts-Nachbarn von v
- dünne Label aus



Korrektheit:

- analog zu Korrektheit von CH
- Argumentation über den wichtigsten Knoten auf dem Pfad
- dieser ist im Vorwärtslabel von s und im Rückwärtslabel von t

Mergen von Labels

Generell:

- $L_f(v) = \bigcup_{(v,u) \in G^+} (L_f(u) + (v,u))$
- wenn ein Hub mehrfach im resultierendem Label, behalte nur den mit minimaler Distanz

Ausdünnen:

- manche Knoten im Label haben falschen Distanzwert
- können entfernt werden (analog zu stall-on-demand von CH)
- prunen durch HL-Queries nach Mergen

Weitere Beschleunigung

globale Anfragen:

- benachbarte Knoten haben ähnliche Hubs
- der verantwortliche Hub ist wichtig

idea:

- permutiere hub IDs so, dass wichtige hubs kleine IDs haben

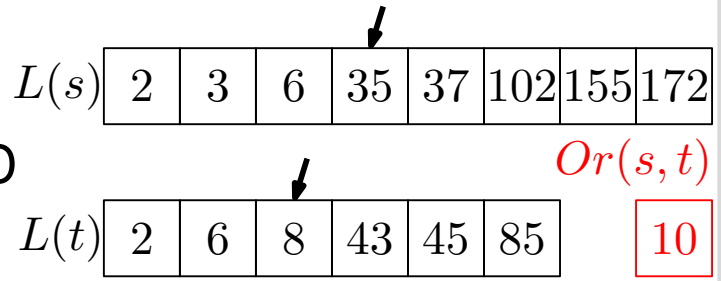
$\Rightarrow$ sind somit am Anfang des Labels

- partitioniere die Eingabe (wieder mal...)
- für jedes Paar Regionen
  - bestimme den am wenigsten wichtigen Hub
  - speichere dessen ID in einer Tabelle

query:

- bestimme ID des hubs in der Tabelle
- stoppe Iteration der Labels nach dieser ID

$\Rightarrow$ beschleunigt globale Anfragen



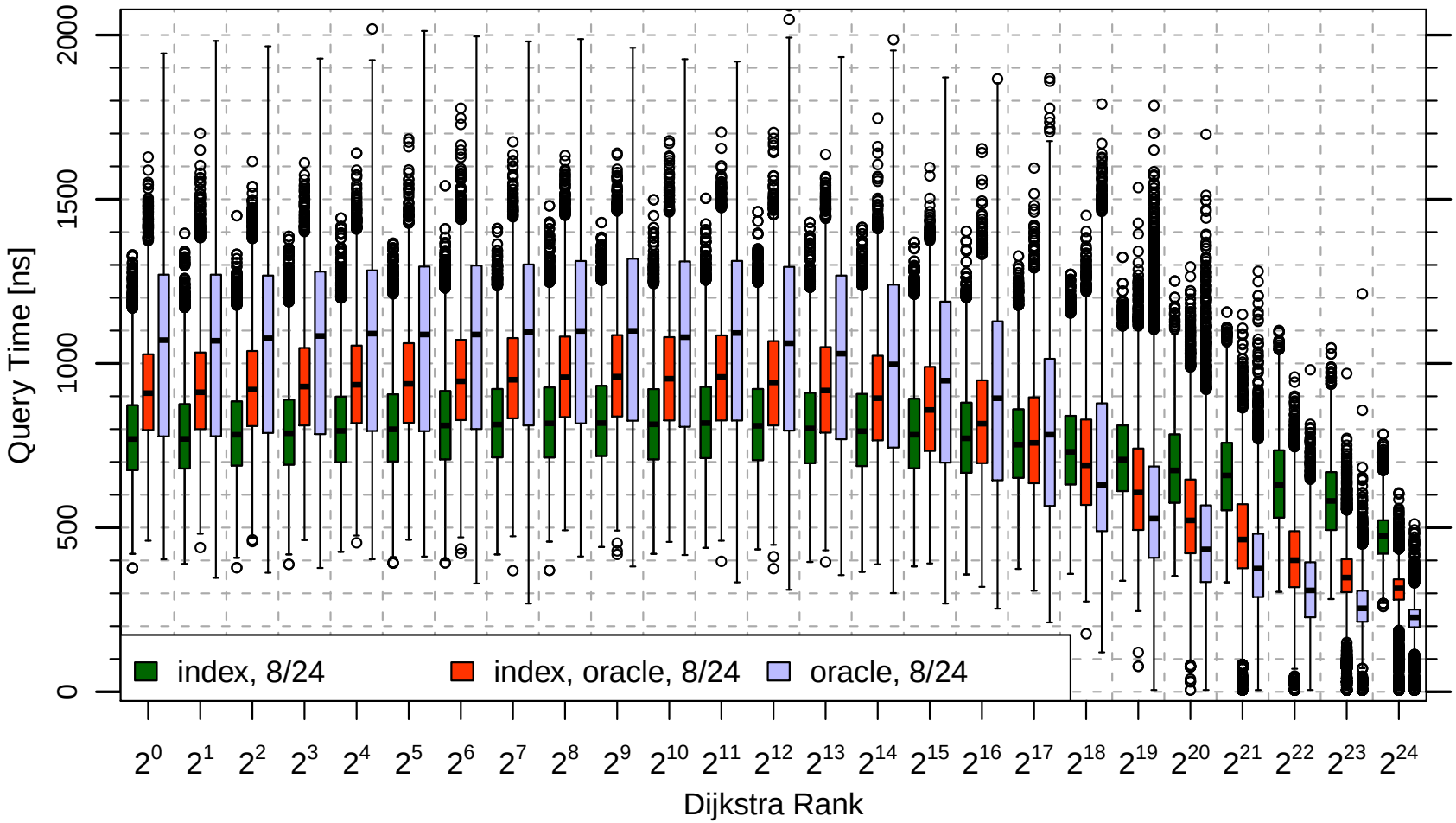
Ergebnisse

	preprocessing		query
method	time [h:m]	space [GB]	time [μs]
MLD-3	$<$ 0:01	0.4	912
CH	0:02	0.4	96.3
HL-0	0:03	22.5	0.700
HL-15	0:05	18.8	0.556
HL-17	0:25	18.4	0.545
HL-∞	5:43	16.8	0.508
HL-∞ + Oracle	6:12	17.7	0.254
Table Lookup	???	1 208 358.7	0.056

HL-x mit 2^x top-down ordnung, ∞ mit Hybrid Ordnung

- HL ist Faktor 100 schneller als CH (Speedup 10 Mio)
- hoher Speicherverbrauch (durch Kompression reduzierbar)
- nur 5 mal langsamer als ein Speicherzugriff

Lokale Queries



Zusammenfassung

- Knotenordnung definiert Labeling
- Beschleunigung gegenüber CH von Faktor mehr als 100
- durch besser Lokalität
- nur 5 mal langsamer als ein Speicherzugriff
- schnellster Algorithmus momentan
- beschleunigt lokale und globale Anfragen
- aber Speicherverbrauch sehr hoch
- wird zu einem späterem Zeitpunkt noch einmal wichtig

Literatur

Contraction Hierarchies:

- Robert Geisberger, Peter Sanders, Dominik Schultes, Christian Vetter
  Exact Routing in Large Road Networks Using Contraction Hierarchies
  In: *Transportation Science*, 2012

HubLabels:

- Ittai Abraham, Daniel Delling, Andrew V. Goldberg, Renato Werneck
  A Hub-Based Labeling Algorithm for Shortest Paths on Road Networks
  In: *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA '11)*, 2011
- Ittai Abraham, Daniel Delling, Andrew V. Goldberg, Renato Werneck
  Hierarchical Hub Labelings for Shortest Paths
  MSR Technical Report, 2012

Nächste Termine

Montag, 21.5.2012 (Thomas)

Mittwoch, 30.5.2012 (Thomas)

Montag, 4.6.2012 (Daniel)

Mittwoch, 6.6.2012 (Daniel)