

Algorithmen für Routenplanung

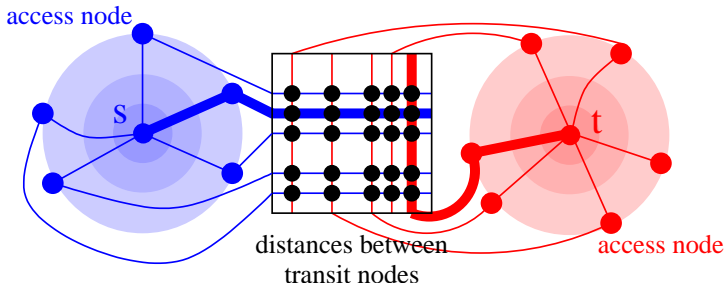
8. Sitzung, Sommersemester 2010

Thomas Pajor | 31. Mai 2010

INSTITUT FÜR THEORETISCHE INFORMATIK · ALGORITHMIK I · PROF. DR. DOROTHEA WAGNER



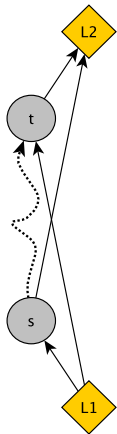
Transit-Node Routing



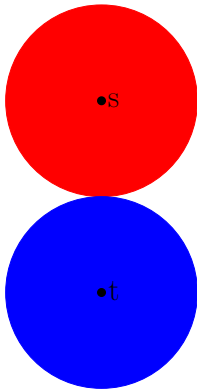
Kombinationen

Fünf Bausteine

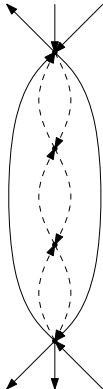
Landmarken



Bidirektionale Suche



Kontraktion



Arc-Flags

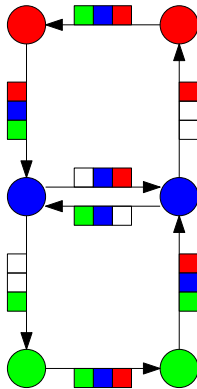
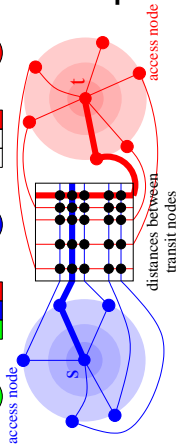


Table-Lookups

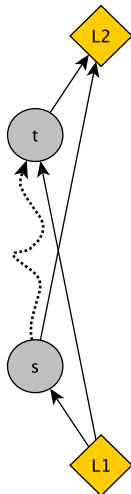


Vorbereitung:

- wähle eine Hand voll (≈ 16) Knoten als **Landmarken**
- berechne Abstände von und zu allen Landmarken

Anfrage:

- benutze Landmarken und Dreiecksungleichung um eine **untere Schranke** für den Abstand zum Ziel zu bestimmen
- verändert **Reihenfolge** der besuchten Knoten
- bevorzugt Knoten zwischen Start- und Zielknoten
- bekannt als **ALT**: **A***, **L**andmarken, **T**riangle Inequality

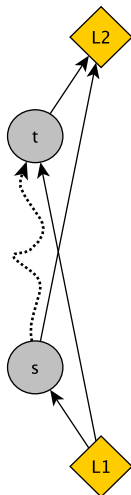


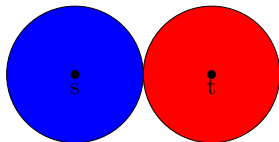
Vorteile:

- + Anpassung einfach
- + einfache und schnelle Vorberechnung
- + sehr **robust** gegenüber der Eingabe
- + zusätzliche Staus können berücksichtigt werden

Nachteile:

- **geringe** Beschleunigung ("nur" Faktor von ca. 100)
- dies auch nur mit **bidirektionaler** Suche erreichbar
- Vorberechnung benötigt viel Speicher (2 Distanzen pro Knoten und Landmarke)





- starte zweite Suche von t
- relaxiere rückwärts nur eingehende Kanten
- stoppe die Suche, wenn beide Suchräume sich treffen

Beobachtung:

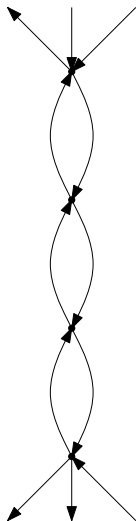
- Knoten mit niedrigem Grad sind **unwichtig**

Knoten-Reduktion:

- entferne diese Knoten **iterativ**
- füge neue Kanten (**Abkürzungen**) hinzu, um die Abstände zwischen verbleibenden Knoten zu erhalten

Kanten-Reduktion:

- behalte nur relevante Shortcuts
- lokale Suche während oder nach Knoten-reduktion



Beobachtung:

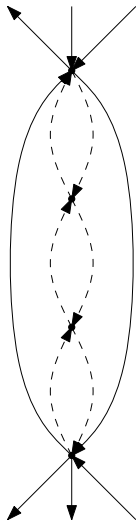
- Knoten mit niedrigem Grad sind **unwichtig**

Knoten-Reduktion:

- entferne diese Knoten **iterativ**
- füge neue Kanten (**Abkürzungen**) hinzu, um die Abstände zwischen verbleibenden Knoten zu erhalten

Kanten-Reduktion:

- behalte nur relevante Shortcuts
- lokale Suche während oder nach Knoten-reduktion

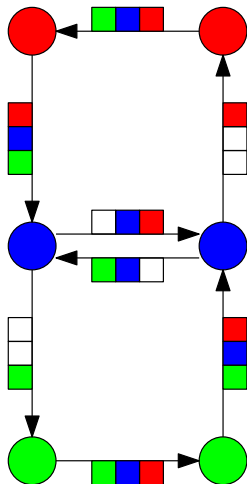


Idee:

- **partitioniere** den Graph in k Zellen
- hänge ein **Label** mit k Bits an jede Kante
- zeigt ob e wichtig für die Zielzelle ist
- **modifizierter** Dijkstra überspringt unwichtige Kanten

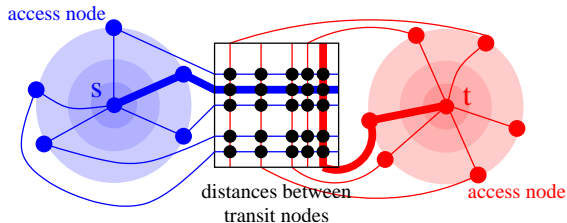
Diskussion:

- + einfacher Suchalgorithmus
- + schnell
- **lange** Vorberechnungszeit
- keine Vorteile in **Zielzelle**

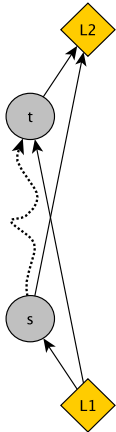


Idee:

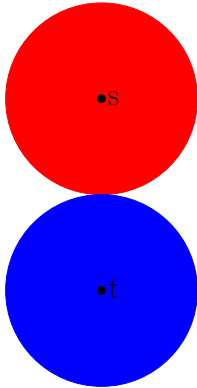
- speichere Distanztabellen
- nur für "wichtige" Teile des Graphen
- Suchen laufen nur bis zur Tabelle
- harmoniert gut mit hierarchischen Techniken



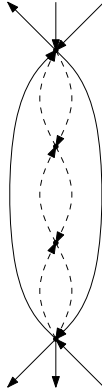
Landmarken



Bidirektionale
Suche



Kontraktion



Arc-Flags

Table-
Lookups

Motivation:

- ALT ist robust gegenüber der Eingabe
- aber hoher Speicherverbrauch

Hauptidee:

- beschränke ALT nur im Kern

Core-ALT

(Landmarken, bidirektionale Suche, Kontraktion)

Idee

- begrenze Beschleunigungstechnik auf kleinen Subgraphen (Kern)

s ●

● t

Vorbereitung

- kontrahiere Graphen zu einem Kern
- Landmarken nur im Kern

Anfrage

- Initialphase: normaler Dijkstra
- benutze Landmarken nur im Kern

Core-ALT

(Landmarken, bidirektionale Suche, Kontraktion)

Idee

- begrenze Beschleunigungstechnik auf kleinen Subgraphen (Kern)



Vorbereitung

- kontrahiere Graphen zu einem Kern
- Landmarken nur im Kern

Anfrage

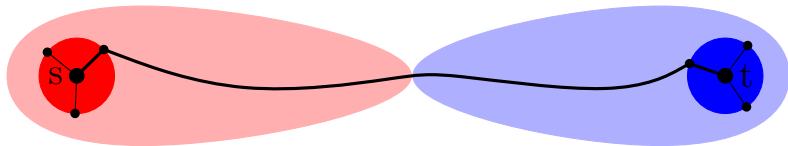
- Initialphase: normaler Dijkstra
- benutze Landmarken nur im Kern

Core-ALT

(Landmarken, bidirektionale Suche, Kontraktion)

Idee

- begrenze Beschleunigungstechnik auf kleinen Subgraphen (Kern)



Vorbereitung

- kontrahiere Graphen zu einem Kern
- Landmarken nur im Kern

Anfrage

- Initialphase: normaler Dijkstra
- benutze Landmarken nur im Kern

Problem:

- ALT braucht Potential von jedem Knoten zu t und s
- s und/oder t könnten außerhalb des Kerns liegen
- somit keine Abstandswerte von den Landmarken zu s und t

Lösung:

- bestimme Proxy-Knoten t' für t (s analog), t' im Kern
- neue Ungleichungen:

$$d(u, t) \geq d(u, l_2) - d(t', l_2) - d(t, t')$$

$$d(u, t) \geq d(l_1, t') - d(l_1, u) - d(t, t')$$

Problem:

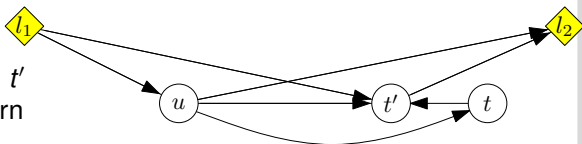
- ALT braucht Potential von jedem Knoten zu t und s
- s und/oder t könnten außerhalb des Kerns liegen
- somit keine Abstandswerte von den Landmarken zu s und t

Lösung:

- bestimme Proxy-Knoten t' für t (s analog), t' im Kern
- neue Ungleichungen:

$$d(u, t) \geq d(u, l_2) - d(t', l_2) - d(t, t')$$

$$d(u, t) \geq d(l_1, t') - d(l_1, u) - d(t, t')$$

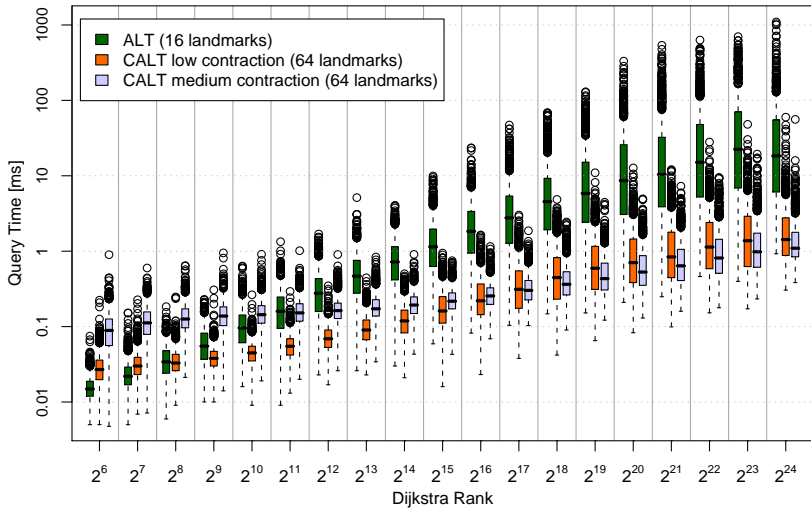


| input | c | h | PREPRO. | | | | QUERY | | |
|--------|-----|-----|------------|-------------|------------|-------------|----------------|--------------|-----------|
| | | | core nodes | $ E $ incr. | time [min] | space [B/n] | #settled nodes | #entry nodes | time [ms] |
| Europe | 0.0 | 0 | 100.00% | 0.00% | 68 | 512.0 | 25 324 | 1.0 | 19.61 |
| | 0.5 | 10 | 35.48% | 10.23% | 21 | 187.7 | 10 925 | 3.2 | 8.02 |
| | 1.0 | 20 | 6.32% | 14.24% | 7 | 38.4 | 2 233 | 8.2 | 2.16 |
| | 2.0 | 30 | 3.04% | 11.41% | 9 | 21.8 | 1 382 | 13.3 | 1.55 |
| | 2.5 | 50 | 1.88% | 9.16% | 11 | 15.4 | 1 394 | 18.6 | 1.34 |
| | 3.0 | 75 | 1.29% | 7.80% | 12 | 12.2 | 1 963 | 24.2 | 1.43 |
| | 5.0 | 100 | 0.86% | 6.94% | 18 | 9.8 | 3 126 | 34.0 | 1.67 |
| USA | 0.0 | 0 | 100.00% | 0.00% | 93 | 512.0 | 68 861 | 1.0 | 48.87 |
| | 0.5 | 10 | 28.90% | 11.40% | 20 | 154.2 | 21 544 | 3.4 | 16.61 |
| | 1.0 | 20 | 8.29% | 12.68% | 11 | 48.9 | 7 662 | 7.1 | 6.96 |
| | 2.0 | 30 | 3.21% | 10.53% | 12 | 22.5 | 3 338 | 12.6 | 4.11 |
| | 2.5 | 50 | 2.06% | 8.00% | 13 | 16.1 | 2 697 | 17.1 | 3.01 |
| | 3.0 | 75 | 1.45% | 6.39% | 14 | 12.6 | 2 863 | 22.0 | 2.85 |
| | 5.0 | 100 | 0.86% | 4.50% | 21 | 9.3 | 3 416 | 30.2 | 2.35 |

Einfluss Anzahl Landmarken

| L | no cont. ($c=0.0, h=0$) | | | | low cont. ($c=1.0, h=20$) | | | |
|----|---------------------------|----------------|-------------------|--------------|-----------------------------|----------------|-------------------|--------------|
| | PREPRO. | | QUERY | | PREPRO. | | QUERY | |
| | time [min] | space [B/n] | #settled nodes | time [ms] | time [min] | space [B/n] | #settled nodes | time [ms] |
| 8 | 26.1 | 64 | 163 776 | 127.8 | 7.1 | 10.9 | 12 529 | 10.25 |
| 16 | 85.2 | 128 | 74 669 | 53.6 | 9.4 | 14.9 | 5 672 | 5.77 |
| 32 | 27.1 | 256 | 40 945 | 29.4 | 6.8 | 23.0 | 3 268 | 2.97 |
| 64 | 68.2 | 512 | 25 324 | 19.6 | 8.5 | 36.2 | 2 233 | 2.16 |

| L | med: $c=2.5, h=50$ | | | | high: $c=5.0, h=100$ | | | |
|----|--------------------|----------------|-------------------|--------------|----------------------|----------------|-------------------|--------------|
| | PREPRO. | | QUERY | | PREPRO. | | QUERY | |
| | time [min] | space [B/n] | #settled nodes | time [ms] | time [min] | space [B/n] | #settled nodes | time [ms] |
| 8 | 10.1 | 7.0 | 4 431 | 3.98 | 17.8 | 5.9 | 4 106 | 2.51 |
| 16 | 11.0 | 8.2 | 2 456 | 2.33 | 18.3 | 6.5 | 3 500 | 2.23 |
| 32 | 10.0 | 10.6 | 1 704 | 1.66 | 17.7 | 7.6 | 3 264 | 2.01 |
| 64 | 10.5 | 15.4 | 1 394 | 1.34 | 18.0 | 9.8 | 3 126 | 1.67 |



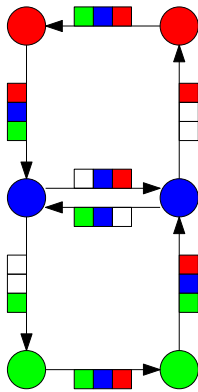
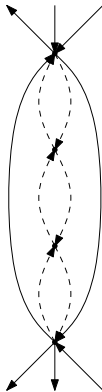
Landmarken

Bidirektionale
Suche

Kontraktion

Arc-Flags

Table-
Lookups

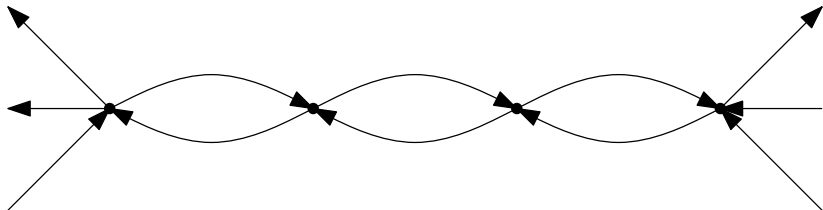


Motivation:

- unidirektional
- Vorberechnungszeiten von Arc-Flags reduzieren

Ideen:

- Multi-Level Arc-Flags
- Integration von Kontraktion
- Verfeinern von Flaggen

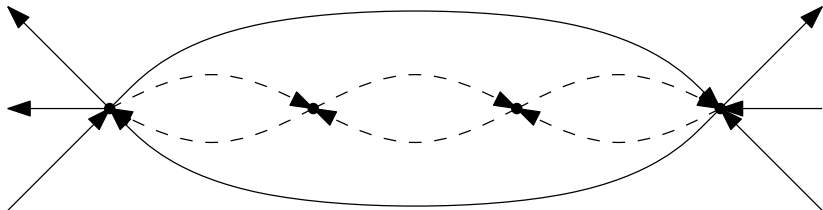


Idee:

- sub-optimale flaggen für entfernte Kanten
- in die Komponente, nur own-cell Flagge
- sonst volle Flaggen

⇒

- Vorberechnung nur auf kontrahierten Graph
- sub-optimale Flaggen nur am Anfang und Ende der Query
- Überspringen von Kanten automatisch durch Arc-Flags Query

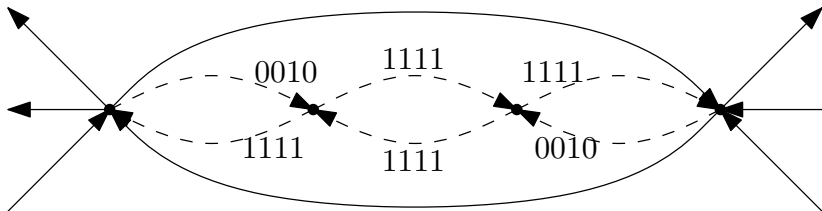


Idee:

- sub-optimale flaggen für entfernte Kanten
- in die Komponente, nur own-cell Flagge
- sonst volle Flaggen

⇒

- Vorberechnung nur auf kontrahierten Graph
- sub-optimale Flaggen nur am Anfang und Ende der Query
- Überspringen von Kanten automatisch durch Arc-Flags Query



Idee:

- sub-optimale flaggen für entfernte Kanten
- in die Komponente, nur own-cell Flagge
- sonst volle Flaggen

⇒

- Vorberechnung nur auf kontrahierten Graph
- sub-optimale Flaggen nur am Anfang und Ende der Query
- Überspringen von Kanten automatisch durch Arc-Flags Query

Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)

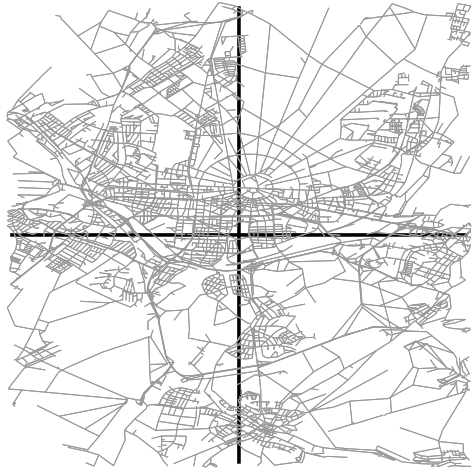


Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)

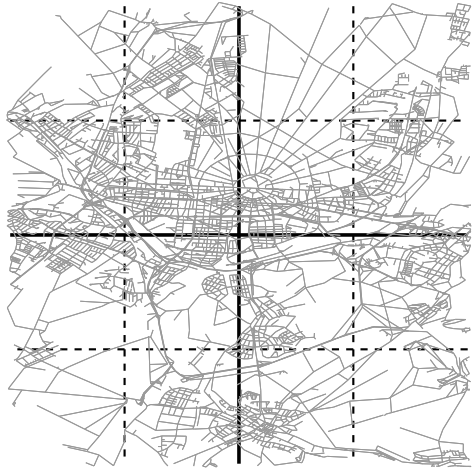


Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)

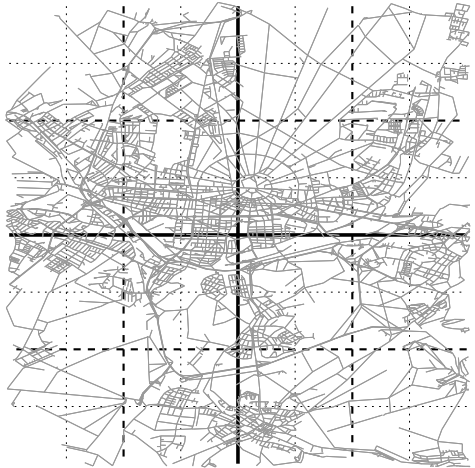


Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)

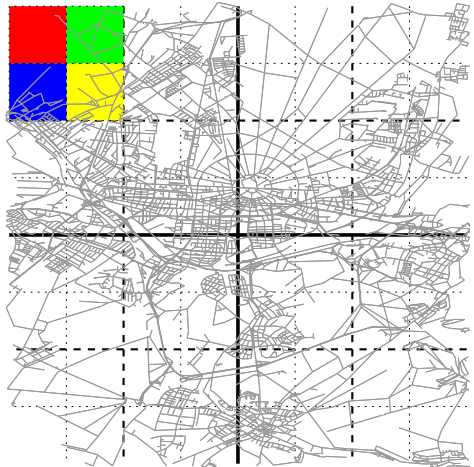


Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)

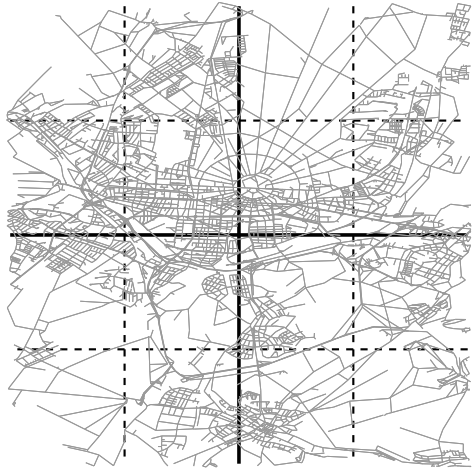


Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)



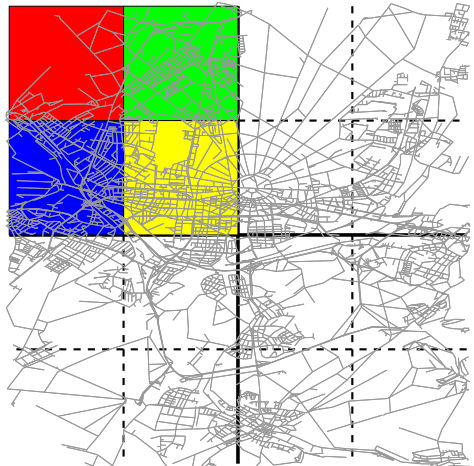
SHARC (Kontraktion, Arc-Flags)

Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)

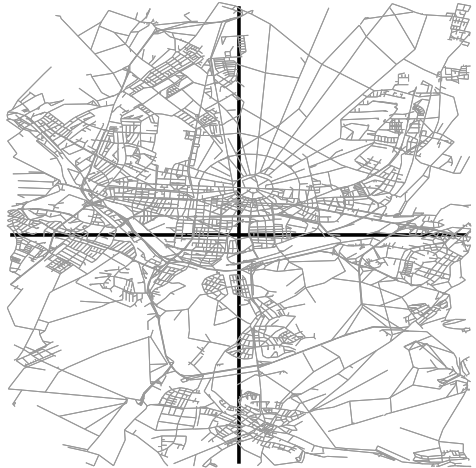


Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)

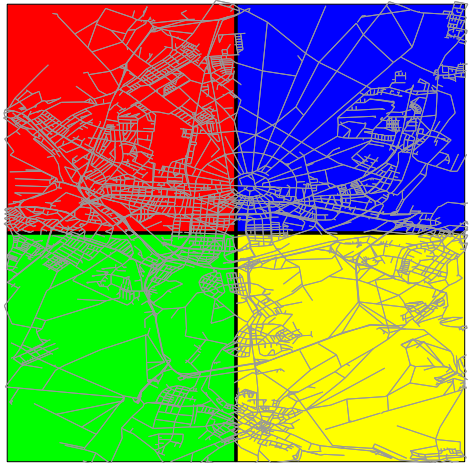


Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)

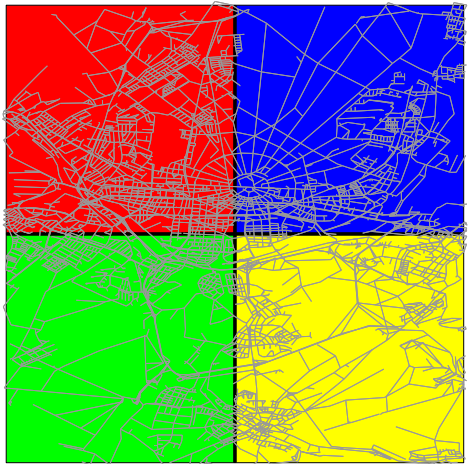


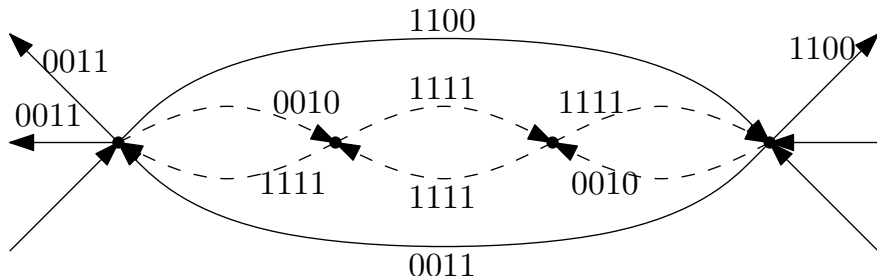
Vorbereitung:

- Multi-Level-Partition
- iterativer Prozess:
 - kontrahiere Subgraphen
 - berechne Flaggen

Anfragen:

- unidirektional
- hierarchisch (Kontraktion)
- zielgerichtet (Arc-Flags)



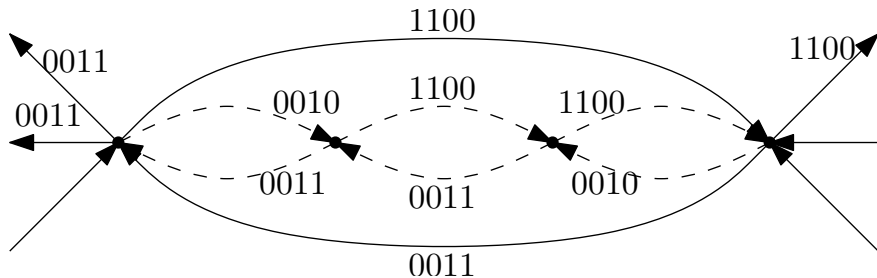


Beobachtung:

- Flaggen schlechter als nötig
- geht es besser?

Idee:

- verfeinere Flaggen
 - propagiere Flaggen von wichtigen zu unwichtigen Kanten
 - mittels lokaler Suche
- ⇒ sehr gute Flaggen



Beobachtung:

- Flaggen schlechter als nötig
- geht es besser?

Idee:

- verfeinere Flaggen
 - propagiere Flaggen von wichtigen zu unwichtigen Kanten
 - mittels lokaler Suche
- ⇒ sehr gute Flaggen

Partielle Flaggenberechnung:

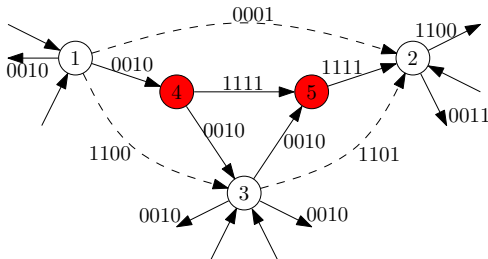
- Flaggen auf unteren Leveln unwichtig
- nur wenige werden am Ende relaxiert
- setze einfach alle Flaggen auf 1
- offensichtlich korrekt

Partielle Verfeinerung:

- verfeinere nur Knoten auf hohen Leveln
- dadurch zu Beginn einer Anfrage schlechte Flaggen
- Verfeinerung für niedrigere Level wichtiger als Flaggenberechnung

Beobachtung:

- Shortcut entspricht einem Pfad
- manche Shortcuts erscheinen unwichtig

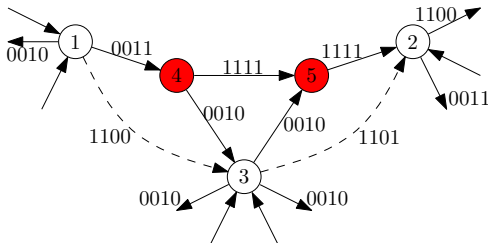


Idee:

- entferne (manche) Shortcuts nach Vorberechnung
- vererbe Flaggen an erste Kante des Pfades
- Extremfall: entferne alle Shortcuts

Beobachtung:

- Shortcut entspricht einem Pfad
- manche Shortcuts erscheinen unwichtig



Idee:

- entferne (manche) Shortcuts nach Vorberechnung
- vererbe Flaggen an erste Kante des Pfades
- Extremfall: entferne alle Shortcuts

| PARTITION | | | | | | | | | PREPRO | | QUERY | | | |
|------------------|----------|----------|----------|----------|------------|-------|-------|----------------|-------------|-------------|-------------|------------|--------------|------------|
| #cells per level | | | | | | | | | ⊙#nodes | time space | #settled | #rel. | time | |
| l_0 | l_1 | l_2 | l_3 | l_4 | l_5 | l_6 | l_7 | #total cells | per cell | [h:m] | [B/n] | nodes | edges | [μ s] |
| 128 | - | - | - | - | - | - | - | 128 | 140 704.5 | 1:52 | 6.0 | 78 429 | 178 103 | 23 306 |
| 8 | 120 | - | - | - | - | - | - | 960 | 18 760.6 | 1:14 | 9.8 | 11 362 | 26 323 | 3 049 |
| 4 | 4 | 120 | - | - | - | - | - | 1 920 | 9 380.3 | 1:24 | 10.6 | 5 982 | 14 128 | 1 637 |
| 4 | 8 | 116 | - | - | - | - | - | 3 712 | 4 851.9 | 1:25 | 10.8 | 3 459 | 8 372 | 983 |
| 8 | 8 | 112 | - | - | - | - | - | 7 168 | 2 512.6 | 1:36 | 11.5 | 2 182 | 5 389 | 667 |
| 16 | 16 | 96 | - | - | - | - | - | 24 576 | 732.8 | 2:12 | 13.1 | 1 217 | 3 169 | 428 |
| 4 | 4 | 4 | 116 | - | - | - | - | 7 424 | 2 425.9 | 1:20 | 11.2 | 2 025 | 5 219 | 625 |
| 4 | 4 | 8 | 112 | - | - | - | - | 14 336 | 1 256.3 | 1:14 | 11.6 | 1 320 | 3 544 | 441 |
| 4 | 8 | 16 | 100 | - | - | - | - | 51 200 | 351.8 | 1:17 | 13.1 | 819 | 2 357 | 319 |
| 4 | 4 | 4 | 4 | 112 | - | - | - | 28 672 | 628.1 | 1:12 | 12.0 | 957 | 2 827 | 360 |
| 4 | 4 | 8 | 8 | 104 | - | - | - | 106 496 | 169.1 | 1:18 | 13.7 | 700 | 2 153 | 294 |
| 4 | 4 | 4 | 16 | 100 | - | - | - | 102 400 | 175.9 | 1:16 | 13.7 | 703 | 2 162 | 295 |
| 4 | 8 | 8 | 16 | 92 | - | - | - | 376 832 | 47.8 | 1:30 | 16.0 | 663 | 2 046 | 288 |
| 4 | 4 | 4 | 4 | 4 | 108 | - | - | 110 592 | 162.9 | 1:15 | 13.6 | 695 | 2 263 | 299 |
| 4 | 4 | 4 | 4 | 8 | 104 | - | - | 212 992 | 84.6 | 1:21 | 14.5 | 654 | 2 116 | 290 |
| 4 | 4 | 4 | 8 | 8 | 100 | - | - | 409 600 | 44.0 | 1:28 | 16.1 | 645 | 2 087 | 290 |
| 4 | 4 | 4 | 8 | 16 | 92 | - | - | 753 664 | 23.9 | 1:31 | 17.7 | 646 | 2 028 | 289 |
| 4 | 4 | 8 | 8 | 16 | 88 | - | - | 1 441 792 | 12.5 | 1:50 | 19.8 | 663 | 2 085 | 296 |
| 4 | 4 | 4 | 4 | 4 | 4 | 104 | - | 425 984 | 42.3 | 1:27 | 15.6 | 649 | 2 209 | 299 |
| 4 | 4 | 4 | 4 | 8 | 8 | 96 | - | 1 572 864 | 11.5 | 1:46 | 19.3 | 637 | 2 092 | 294 |
| 4 | 4 | 4 | 8 | 8 | 16 | 84 | - | 5 505 024 | 3.3 | 2:00 | 21.5 | 655 | 2 035 | 294 |
| 4 | 4 | 4 | 4 | 4 | 4 | 4 | 100 | 1 638 400 | 11.0 | 1:43 | 19.2 | 650 | 2 247 | 308 |
| 4 | 4 | 4 | 4 | 4 | 4 | 8 | 96 | 3 145 728 | 5.7 | 1:56 | 20.0 | 627 | 2 113 | 293 |
| 4 | 4 | 4 | 4 | 4 | 8 | 8 | 92 | 6 029 312 | 3.0 | 1:51 | 21.1 | 649 | 2 121 | 300 |
| 4 | 4 | 4 | 4 | 4 | 8 | 16 | 84 | 11 010 048 | 1.6 | 2:03 | 21.5 | 648 | 2 035 | 296 |

| c | PREPRO | | QUERY | | |
|-----|---------------|----------------|-------------------|-------------------|--------------------|
| | time [h:m] | space [B/n] | #settled nodes | #relaxed edges | time [μ s] |
| 1.0 | 1:40 | 15.1 | 1 572 | 3 705 | 578 |
| 1.5 | 1:20 | 14.8 | 886 | 2 464 | 348 |
| 2.0 | 1:20 | 14.7 | 714 | 2 171 | 301 |
| 2.5 | 1:21 | 14.5 | 654 | 2 116 | 290 |
| 3.0 | 1:23 | 14.6 | 622 | 2 109 | 286 |

Beobachtung:

- Einfluss der Kontraktion begrenzt

Einfluss Flaggenberechnung

| ARC-FLAGS | | PREPRO | | QUERY | | |
|------------------------|------------|-------------|-------------|----------------|----------------|------------|
| core refinement levels | levels | time [h:m] | space [B/n] | #settled nodes | #relaxed edges | time [μs] |
| - | - | 0:16 | 12.8 | 204 518 | 960 653 | 76 640 |
| 5 | 5 | 0:24 | 13.2 | 23 313 | 70 225 | 6 021 |
| 5 | 4-5 | 0:24 | 13.2 | 6 583 | 23 038 | 1 843 |
| 5 | 3-5 | 0:25 | 13.3 | 2 394 | 11 547 | 856 |
| 5 | 2-5 | 0:27 | 13.6 | 1 350 | 8 721 | 611 |
| 5 | 1-5 | 0:29 | 13.7 | 1 127 | 8 091 | 553 |
| 4-5 | 4-5 | 0:30 | 13.7 | 6 186 | 18 170 | 1 626 |
| 4-5 | 3-5 | 0:30 | 13.7 | 2 042 | 6 683 | 648 |
| 4-5 | 2-5 | 0:31 | 13.7 | 993 | 3 883 | 405 |
| 4-5 | 1-5 | 0:34 | 13.7 | 784 | 3 338 | 355 |
| 3-5 | 3-5 | 0:35 | 13.8 | 1 974 | 5 962 | 615 |
| 3-5 | 2-5 | 0:37 | 14.2 | 933 | 3 161 | 371 |
| 3-5 | 1-5 | 0:39 | 14.2 | 729 | 2 629 | 323 |
| 2-5 | 2-5 | 0:44 | 14.3 | 900 | 2 862 | 354 |
| 2-5 | 1-5 | 0:46 | 14.3 | 696 | 2 335 | 305 |
| 1-5 | 1-5 | 0:54 | 14.5 | 684 | 2 236 | 300 |
| 0-5 | 0-5 | 1:21 | 14.5 | 654 | 2 116 | 290 |

Beobachtung:

- partielle Berechnung reicht aus
- Verfeinerung wichtiger als Berechnung
- Ungenauigkeit am Ende der Anfrage eher verzeihbar als am Anfang (Suchraum fächert sich auf)

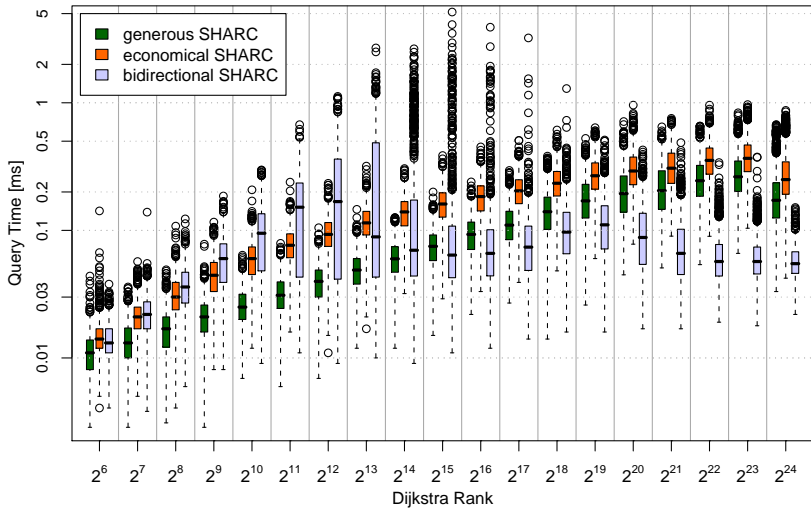
| c | SHARC | | | | stripped SHARC | | | |
|------|--------|-------|----------|------|----------------|-------|-----------|----------|
| | PREPRO | | QUERY | | PREPRO | | QUERY | |
| | time | space | #settled | time | time | space | #settled | time |
| | [h:m] | [B/n] | nodes | [ms] | [h:m] | [B/n] | nodes | [ms] |
| 0.50 | 7:32 | 13.8 | 10 876 | 3.38 | 7:48 | 7.8 | 14 697 | 4.76 |
| 0.75 | 4:29 | 14.3 | 5 420 | 1.99 | 4:43 | 7.7 | 62 303 | 26.03 |
| 1.00 | 1:45 | 15.1 | 1 997 | 0.90 | 2:03 | 7.5 | 1 891 320 | 1 096.48 |

Beobachtung:

- volles Entfernen nur bei niedriger Kontraktion

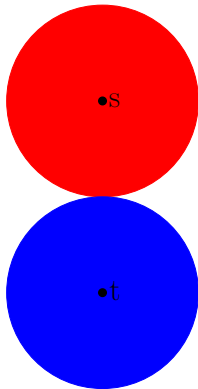
| | Europe | | | | USA | | | |
|---------------------|--------|-------|----------|------------|--------|-------|----------|------------|
| | PREPRO | | QUERY | | PREPRO | | QUERY | |
| | time | space | #settled | time | time | space | #settled | time |
| | [h:m] | [B/n] | nodes | [μ s] | [h:m] | [B/n] | nodes | [μ s] |
| generous SHARC | 1:21 | 14.5 | 654 | 290 | 0:58 | 18.1 | 865 | 376 |
| economical SHARC | 0:34 | 13.7 | 784 | 355 | 0:38 | 17.2 | 1 230 | 578 |
| stripped SHARC | 7:48 | 7.8 | 14 697 | 4 762 | 6:41 | 9.2 | 38 817 | 12 719 |
| bidirectional SHARC | 2:38 | 21.0 | 125 | 65 | 2:34 | 23.1 | 254 | 118 |

Lokale Anfragen

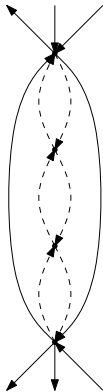


Landmarken

Bidirektionale
Suche



Kontraktion



Arc-Flags

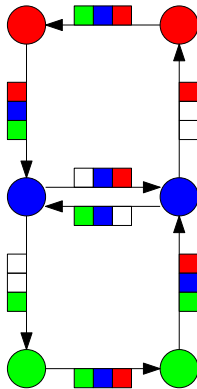


Table-
Lookups

Motivation:

- CHs sind ungerichtet
- weitere Beschleunigung von CH durch zielgerichtetes Verfahren

Hauptideen:

- Flaggen auf Suchgraphen berechnen
- nur auf oberen Teil der Hierarchie

Vorbereitung:

- CH-Vorbereitung
- partitioniere Suchgraphen
- berechne Flaggen für Suchgraphen

Anfragen:

- normale CH-query
- mit Flaggen pruning

Problem:

- mehr Randknoten durch Shortcuts
- sehr lange Vorberechnungszeit

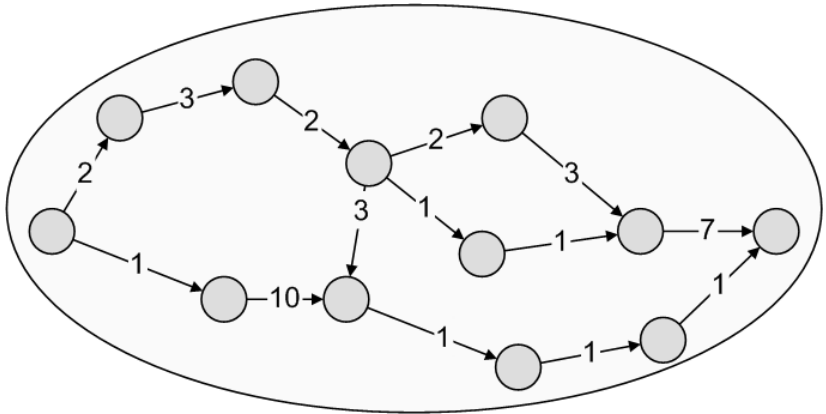
Vorbereitung:

- extrahiere Graphen H mit k wichtigsten Knoten
- führe Vorbereitung für H aus

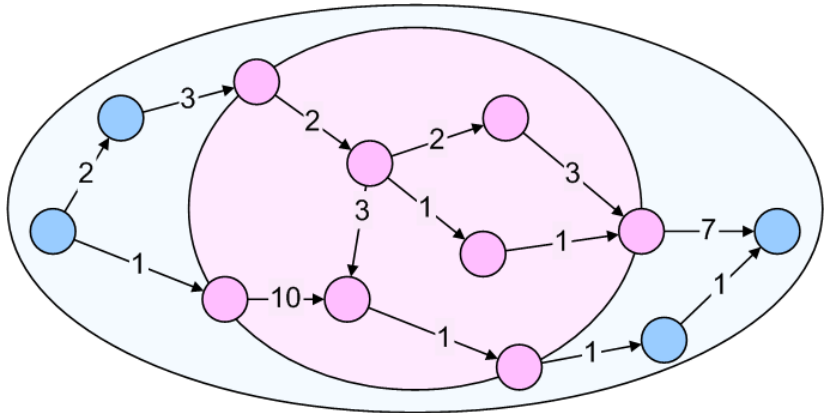
2-Phasen Anfragen:

- 1 normale CH-query
- 2 CH + Flaggen query

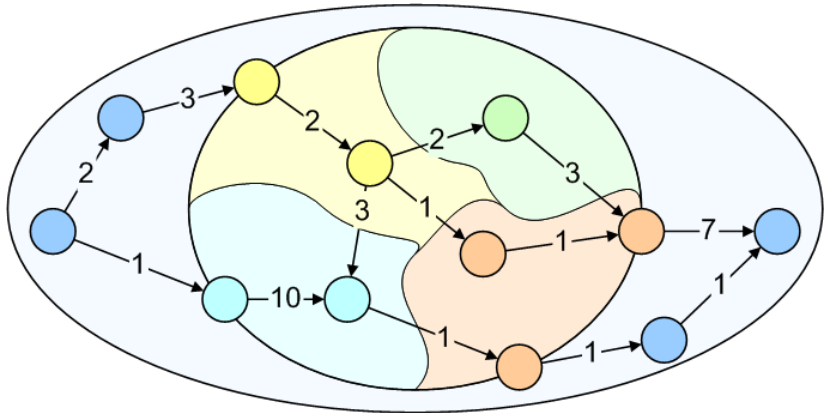
Beispiel



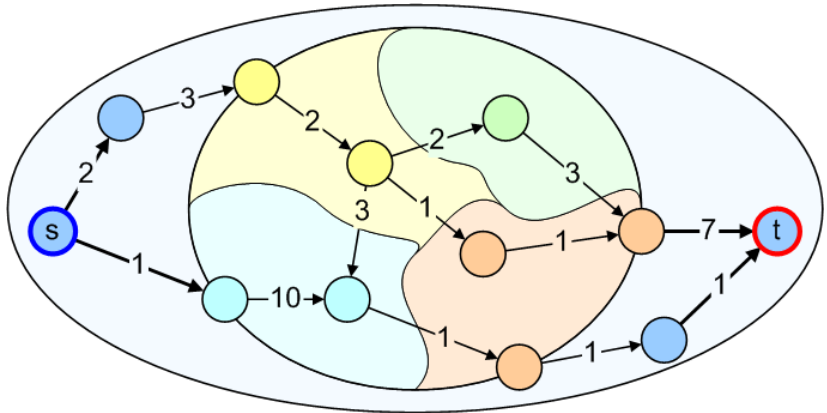
Beispiel



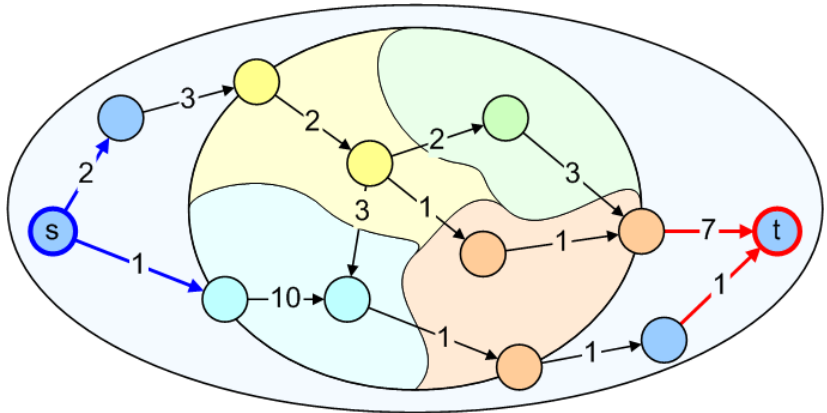
Beispiel



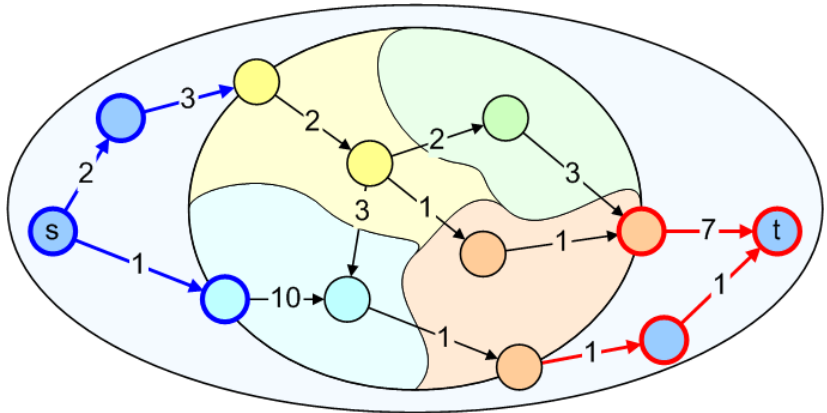
Beispiel



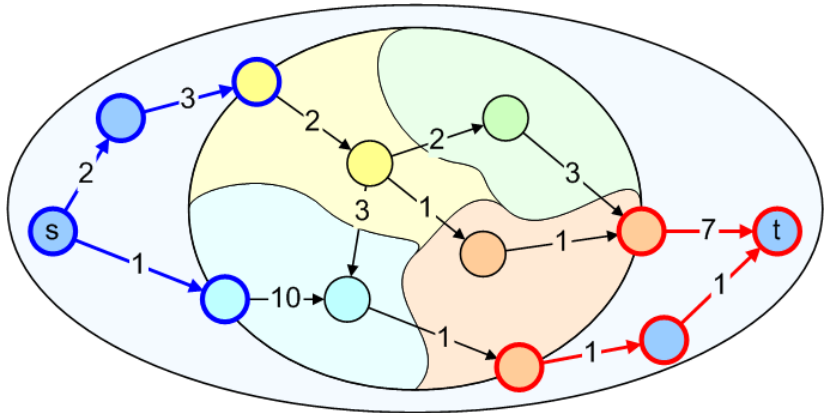
Beispiel



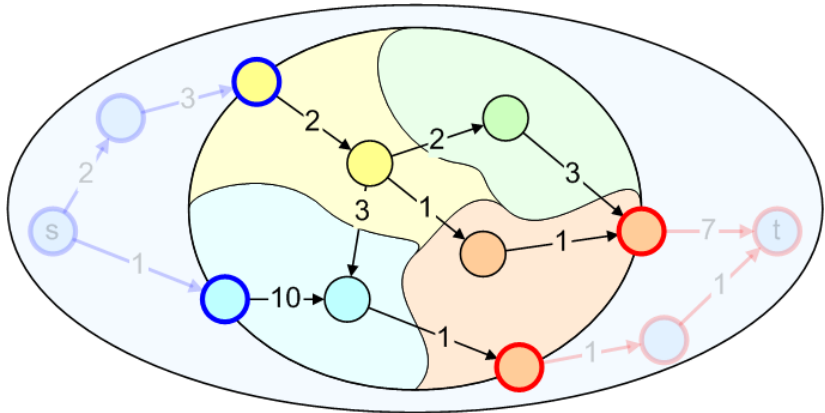
Beispiel



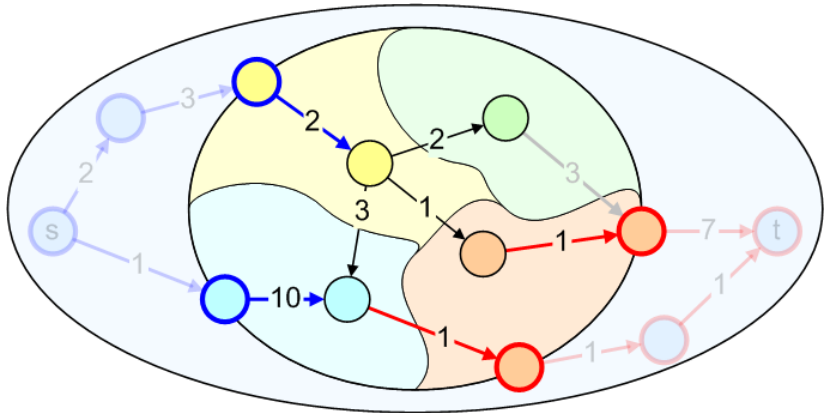
Beispiel



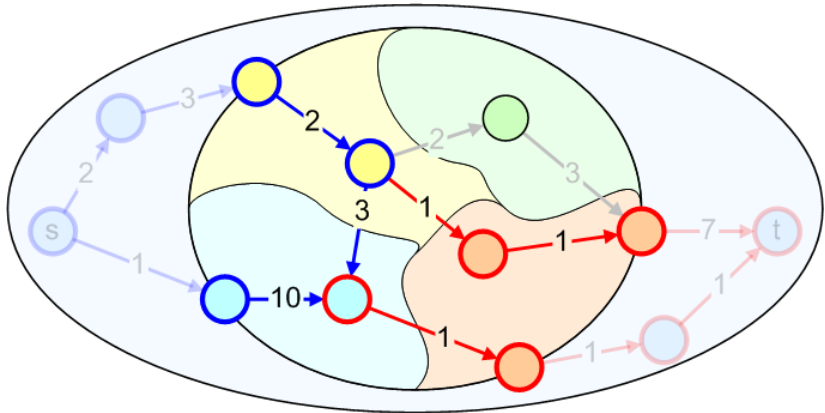
Beispiel



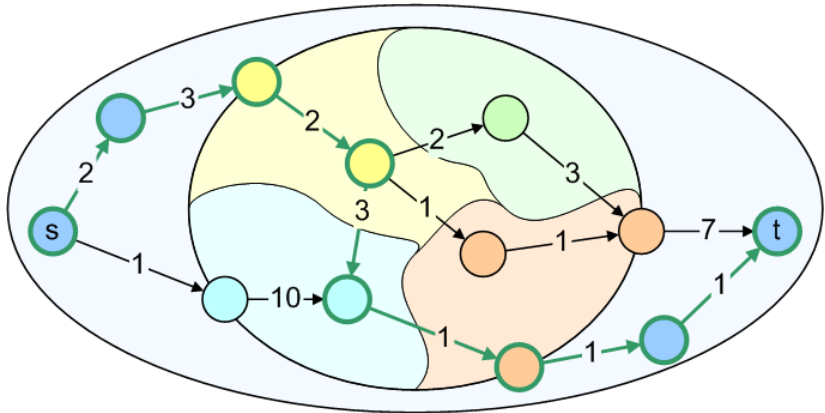
Beispiel



Beispiel



Beispiel



| size of Arc-Flags | | 0% | 0.5% | 1% | 2% | 5% | 10% | 20% |
|-------------------|-----------------|-------|------|------|------|------|------|------|
| Prepro. | [h:m] | 0:25 | 0:32 | 0:41 | 1:02 | 1:39 | 4:04 | 8:56 |
| | [B/n] | -2.7 | 0.0 | 1.9 | 4.9 | 12.1 | 22.2 | 39.5 |
| Query | # settled | 355 | 111 | 78 | 59 | 45 | 39 | 35 |
| | time [μ s] | 180.0 | 43.8 | 30.8 | 23.1 | 17.3 | 14.9 | 13.0 |

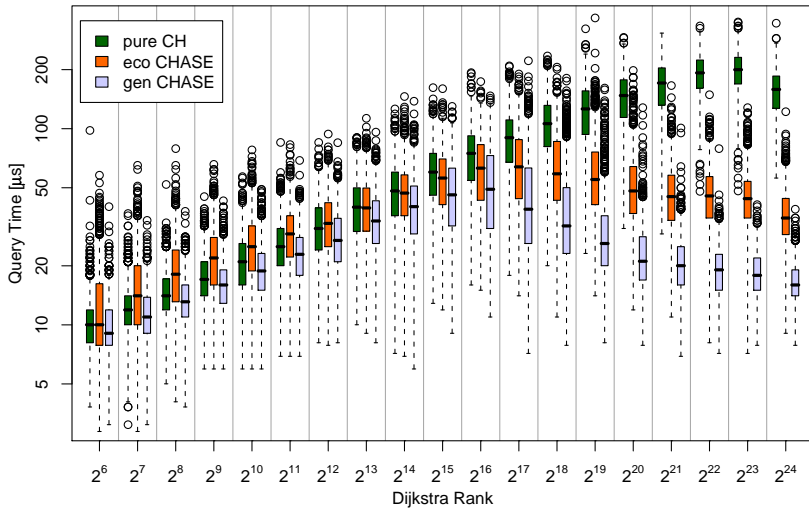
Beobachtungen:

- zusätzliche 7 Minuten Vorberechnung \Rightarrow zusätzlicher speedup von 4.1 (economical variant)
- eine Stunde \Rightarrow speedup von 10.4
- mehr als 5% lohnen sich nicht

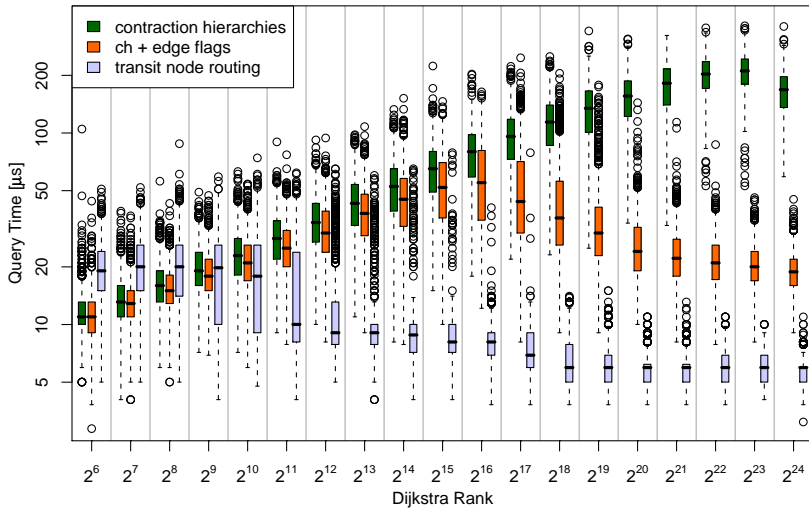
| | Preprocessing | | Query | |
|-------------------------|---------------|-------|----------|------------|
| | [h:m] | [B/n] | #settled | [μ s] |
| Contraction Hierarchies | 0:25 | -3 | 355 | 180 |
| bidirectional Arc-Flags | 17:08 | 19 | 2 369 | 1 600 |
| Transit-Node Routing | 1:52 | 204 | N/A | 3.4 |
| bidirectional SHARC | 3:32 | 21 | 125 | 65 |
| REAL-(64,16) | 2:21 | 32 | 679 | 1 100 |
| eco. CHASE | 0:32 | 0 | 111 | 43.8 |
| gen. CHASE | 1:39 | 12 | 45 | 17.3 |

Beobachtung:

- CHASE fast so schnell wie TNR
- aber weniger Vorberechnungsaufwand

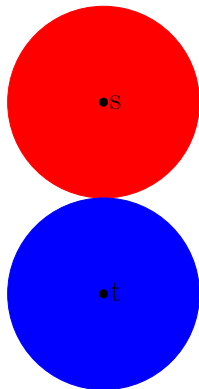


Lokale Anfragen



Landmarken

Bidirektionale Suche



Kontraktion

Arc-Flags

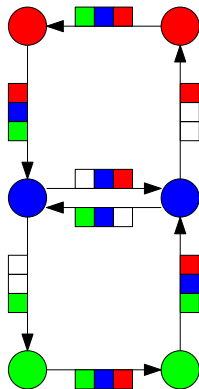
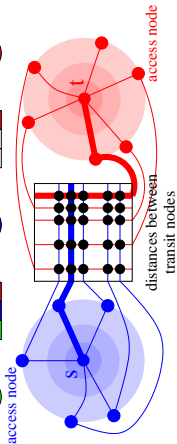


Table-
Lookups

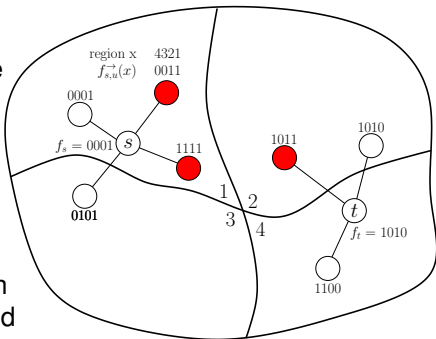


Beobachtung:

- Table-Lookups in Distanztabelle dauern "lange"

Idee:

- **partitioniere** Graphen induziert von Transit-Knoten
- berechne welche Transit-Knoten für welche Regionen wichtig sind



Anfragen:

- evaluiere nur Tabelleneinträge mit gesetzter Flagge

| | Preprocessing | | Query | |
|-------------------------|---------------|-------|----------|------------|
| | [h:m] | [B/n] | #settled | [μ s] |
| bidirectional Arc-Flags | 17:08 | 19 | 2 369 | 1 600 |
| Transit-Node Routing | 1:52 | 204 | N/A | 3.4 |
| gen. CHASE | 1:39 | 12 | 45 | 17.3 |
| Goal-Directed TNR | 3:49 | 321 | N/A | 1.9 |

Beobachtung:

- **milder** speedup gegenüber puren TNR
- aber: **> 3 Mio.** mal schneller als Dijkstra

Kombinationen

- Basismodule:
 - bidirektionale Suche
 - landmarken
 - reach
 - arc-flags
 - Kontraktion
 - Table-Lookups
- beste Kombinationen: hierarchisch + zielgerichtet
- CH + Arc-Flags sehr effizient
- TNR + Arc-Flags schnellstes Verfahren: $< 2 \mu\text{s}$ Anfragezeit

Literatur (SHARC und Kombinationen):

- Reinhard Bauer and Daniel Delling:
SHARC: Fast and Robust Unidirectional Routing
In: *ACM Journal of Experimental Algorithmics*, 14:2.4, 2009.
- Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner:
Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm
In: *ACM Journal of Experimental Algorithmics*, 15:2.3, 2010.

Username: `routePlanning`

Password: `ss10`