Karlsruher Institut für Technologie

Algorithmen für Routenplanung

5. Sitzung, Sommersemester 2013

Julian Dibbelt | 6. Mai 2013

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

Letztes Mal: ALT und Arc-Flags

Wie Suche zielgerichtet machen?

- Nichtbeachten von Kanten oder Knoten die in die “falsche” Richtung führen
- Reihenfolge in der Knoten besucht werden ändern

Jetzt: ersteres

Arc-Flags

Karlsruher Institut für Technologie

Idee:

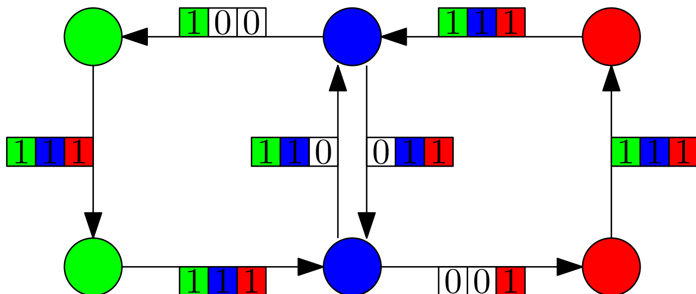
- partitioniere Graphen in k Zellen
- hänge Label mit k Bits an jede Kante
- gibt an, ob e für Ziele in Zielzelle T benötigt wird

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Randknotenbäume

Beobachtung: Man muss durch Randknoten in die Zelle

- setze Intra-Zellen Kanten auf `true`
- einen Rückwärts Dijkstra-Baum pro Randknoten b
- setze Flagge $AF_{region(b)}(e) = \texttt{true}$ wenn e Baumkante des Baums von b ist



Zentralisierte Bäume

KIT Karlsruher Institut für Technologie

verallgemeinere Dijkstra:

- für jede Region r (mit Randknotenmenge B_r)
- hänge Label $l(u)$ der Größe $|B_r|$ an jeden Knoten u
- speichert den Abstand $d(u, b)$ für alle $b \in B_r$
- triviale Intialisierung der Randknoten, füge alle in Queue ein
- nehme Knoten u aus der Queue (key: $\min\{l(u)\}$)
- relaxiere Kanten (u, v):
  - erzeuge Label l durch $l(u) + \text{len}(u, v)$
  - checke ob l das Label $l(v)$ verbessert
- breche ab, wenn keine Verbesserungen mehr möglich sind
- setze Flagge auf `true` wenn $d(u, b) + \text{len}(u, v) = d(v, b)$ gilt

Zentralisierte Bäume

Karlsruher Institut für Technologie

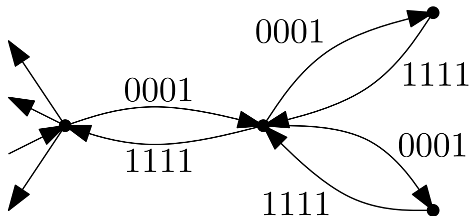
Bemerkungen:

- Knoten können mehrfach besucht werden (label-correcting)
- Prioritäten der Knoten beliebig wählbar
  - z.B. Minimum über alle Einträge von u
- hoher Speicherverbrauch durch n Label der Größe $|B_r|$
  - 18 Mio. Knoten und 1000 Randknoten $\Rightarrow$ 72 GB
  - teile Arbeit in mehrere Schritte mit maximal 100 Randknoten
  - meist Beschleunigung um Faktor 4 gegenüber Randknotenansatz

Automatisches Setzen von Flaggen

Beobachtung:

- Für manche Kanten kann man die Flaggen automatisch setzen



Angehangene Bäume:

- Kanten zur Wurzel hin haben alle Flaggen gesetzt
- Kanten von Wurzel weg haben nur eine Flagge gesetzt
- Also: Entferne Bäume vor Arcs-Flag-Berechnung aus Graph
- Knotenzahl verringert sich um 1 Drittel

Anmerkung: Alle Knoten eines angehangenen Baumes müssen zur gleichen Zelle wie dessen Wurzel gehören.

Partitionierung

Karlsruher Institut für Technologie

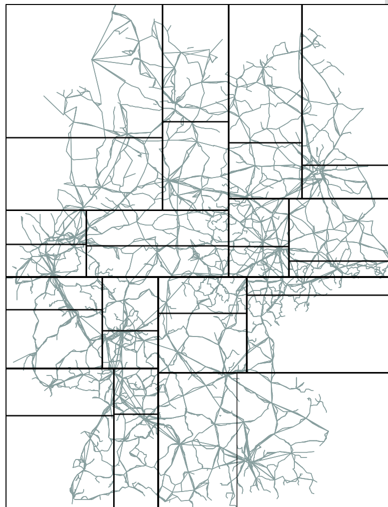
Anforderungen:

- balanciert

mögliche Partitionen:

- Gitter
- Quad-Baum
  - iterativ in 4 Zellen unterteilen
- kd-Baum
  - Verallgemeinerung von Quad-Baum

weitere Anforderungen?



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

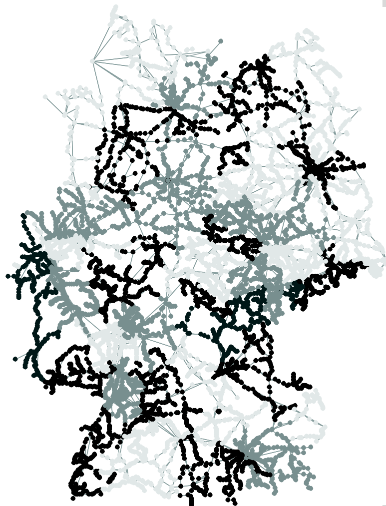
Partitionierung

Anforderungen:

- ausbalanciert
- wenige Randknoten
- zusammenhängend

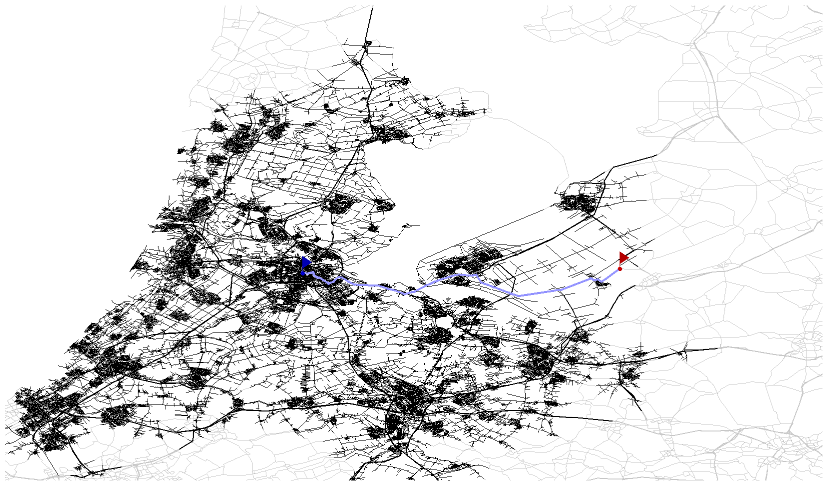
Black-Box Partitionierer (METIS, ...):

- benutzen keine Einbettung
- oft: teilen rekursiv Graphen in k Teile mit kleinem Schnitt
- bringen gute Lösungen für Arc-Flags



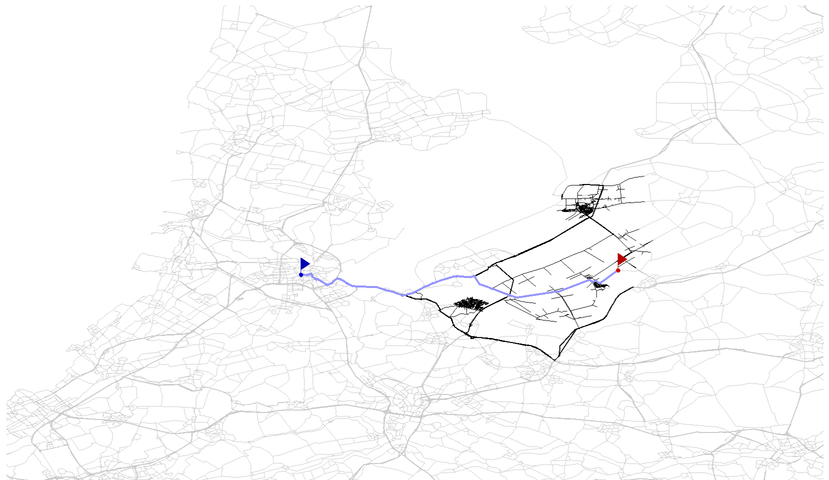
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Suchraum



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Suchraum



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Diskussion

Karlsruher Institut für Technologie

Vorteile:

- einfacher Anfrage Algorithmus
- leicht on-top auf bestehende Systeme zu setzen
- ausreichende Beschleunigung

Nachteile:

- hohe Vorberechnungsdauer
- weitere?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

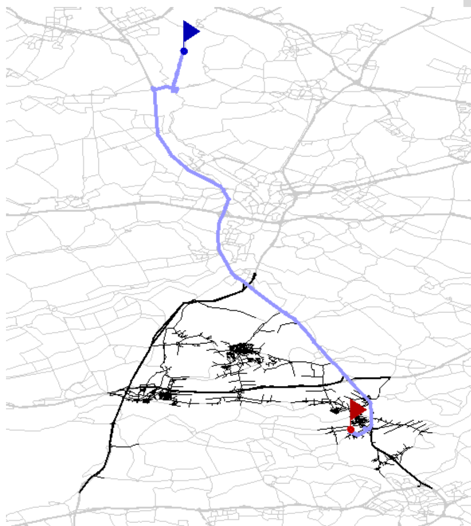
Coning-Effekt

Beobachtung:

- lange Zeit nur eine Kante wichtig
- daher immer nur ein Knoten in der Queue
- aber: je näher an der Zelle, desto mehr Kanten werden wichtig
- Suche fächert sich auf
- in Zelle werden dann alle Kanten relaxiert

Zwei Ansätze:

- bidirektionale Flags
- multi-level Flags



Bidirektionale Arc-Flags

Vorberechung:

- Vorwärts- und Rückwärtsflaggen
- Rückwärtsflaggen können analog für eingehende Kanten berechnet werden
- Vorberechungszeit in gerichteten Graphen erhöht sich um Faktor 2

Anfrage:

- bidirektional:
  - Vorwärtssuche relaxiert nur Kanten mit Flagge für T
  - Rückwärtssuche nur Kanten mit Flaggen für S
- normales Stopp-Kriterium von bidirektionalem Dijkstra

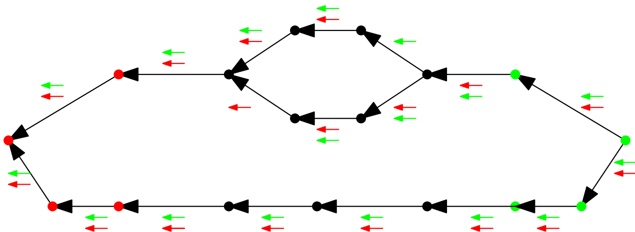
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bidirektionale Arc-Flags

Karlsruher Institut für Technologie

Problem:

- Eindeutigkeit der Wege
- eventuell nicht korrekt



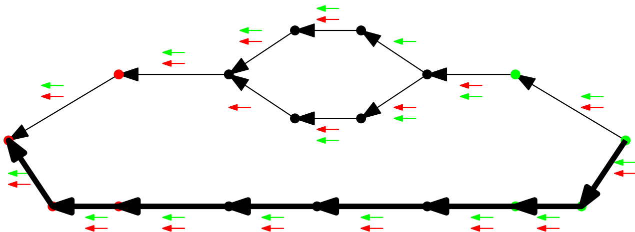
Lösung:

- kommt in Straßengraphen kaum vor
- daher öffne Flaggen für alle möglichen Wege

Bidirektionale Arc-Flags

Problem:

- Eindeutigkeit der Wege
- eventuell nicht korrekt



Lösung:

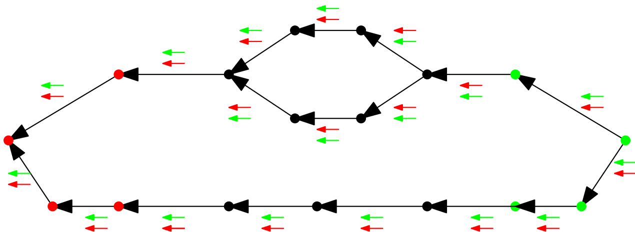
- kommt in Straßengraphen kaum vor
- daher öffne Flaggen für alle möglichen Wege

Bidirektionale Arc-Flags

Karlsruher Institut für Technologie

Problem:

- Eindeutigkeit der Wege
- eventuell nicht korrekt



Lösung:

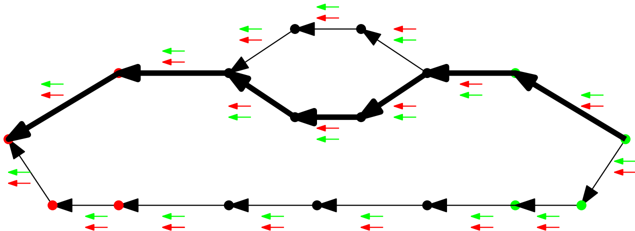
- kommt in Straßengraphen kaum vor
- daher öffne Flaggen für alle möglichen Wege

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bidirektionale Arc-Flags

Problem:

- Eindeutigkeit der Wege
- eventuell nicht korrekt



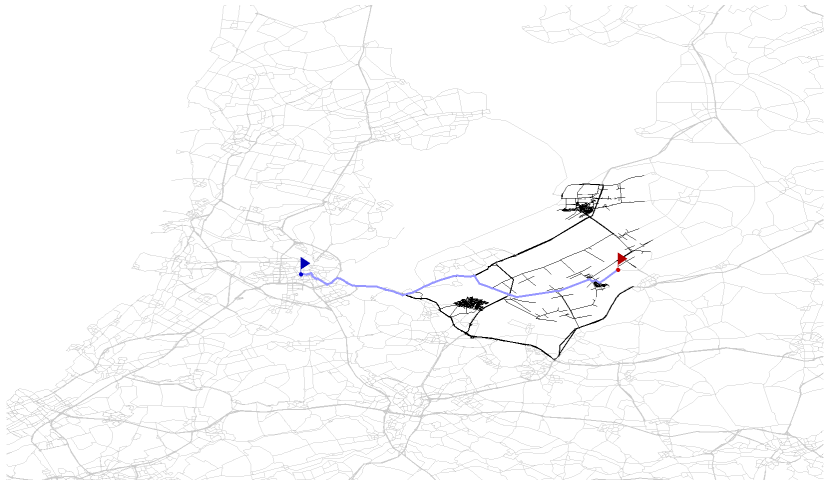
Lösung:

- kommt in Straßengraphen kaum vor
- daher öffne Flaggen für alle möglichen Wege

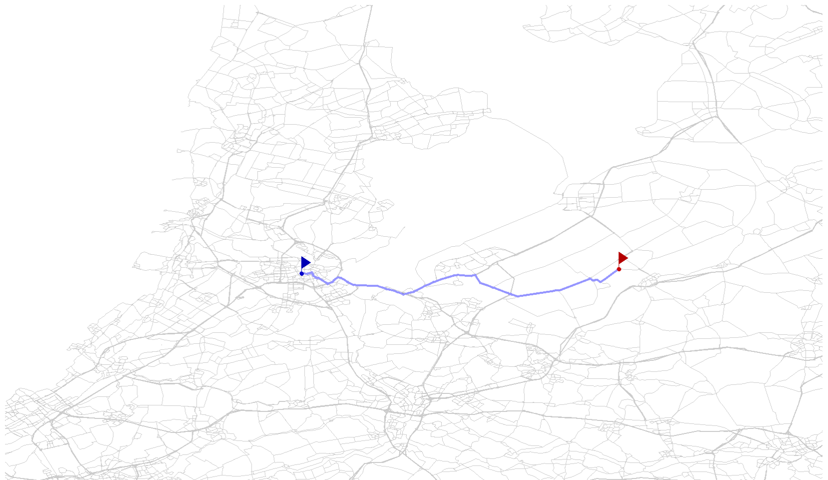
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Suchraum

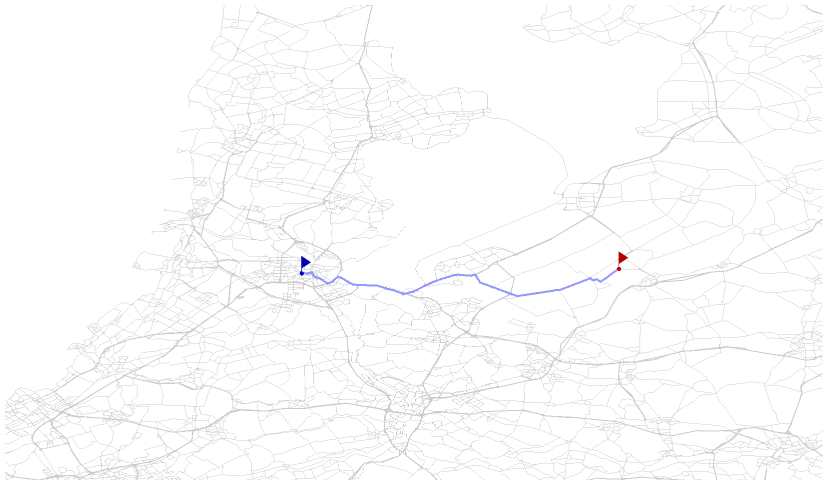
Karlsruher Institut für Technologie



Suchraum



Suchraum



Nachteile?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Arc-Flags

Problem:

- Anfragen in einer Zelle ohne Beschleunigung
- viele Real-Welt Anfragen sind lokal

Multi-Level Arc-Flags:

- Multi-Level Partition
- berechne partielle Flaggen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Multi-Level Arc-Flags

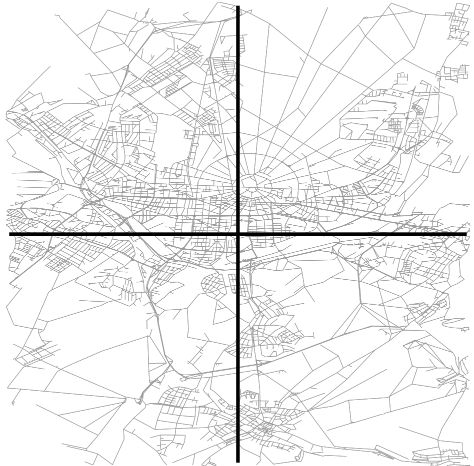
Karlsruher Institut für Technologie

Problem:

- Anfragen in einer Zelle ohne Beschleunigung
- viele Real-Welt Anfragen sind lokal

Multi-Level Arc-Flags:

- Multi-Level Partition
- berechne partielle Flaggen



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

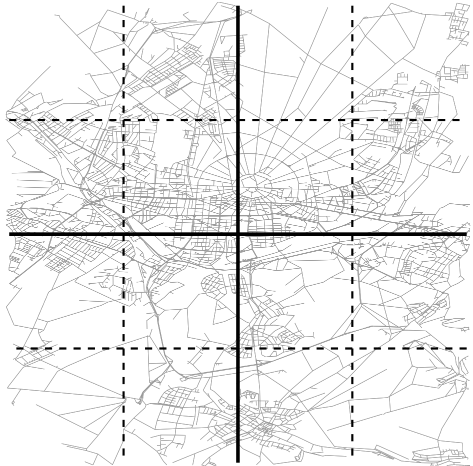
Multi-Level Arc-Flags

Problem:

- Anfragen in einer Zelle ohne Beschleunigung
- viele Real-Welt Anfragen sind lokal

Multi-Level Arc-Flags:

- Multi-Level Partition
- berechne partielle Flaggen



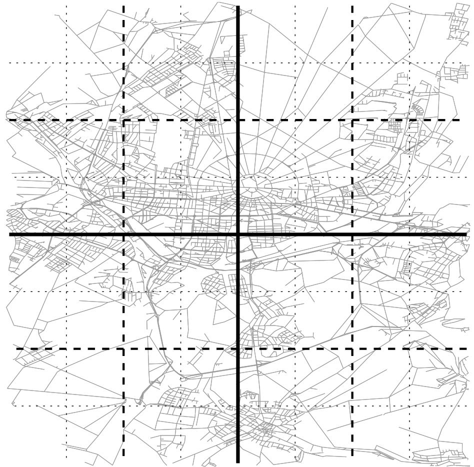
Multi-Level Arc-Flags

Problem:

- Anfragen in einer Zelle ohne Beschleunigung
- viele Real-Welt Anfragen sind lokal

Multi-Level Arc-Flags:

- Multi-Level Partition
- berechne partielle Flaggen

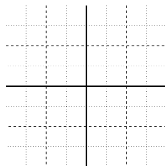


Multi-Level Arc-Flags

Karlsruher Institut für Technologie

Vorberechnung:

- oberster Level wie gehabt
- auf unteren Leveln:
  - von jedem Randknoten führe Dijkstra aus, bis alle Knoten der Superzelle abgearbeitet worden sind
  - setze Flaggen für alle Kanten (u, v) für die u in Superzelle
  - Hinweis: es reicht nicht, nur den Subgraphen der Superzelle zu betrachten (Übungsaufgabe)

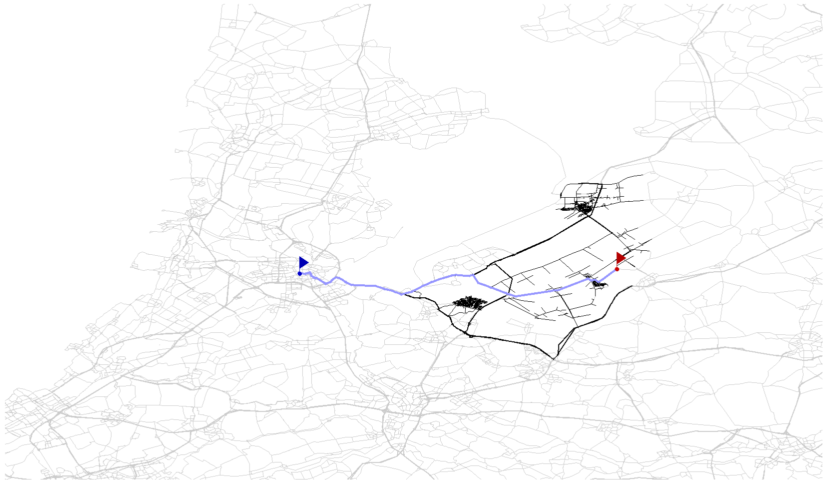


Anfragen:

- bestimme gemeinsamen Level l von u und t
- werte Flaggen auf dem Level l aus

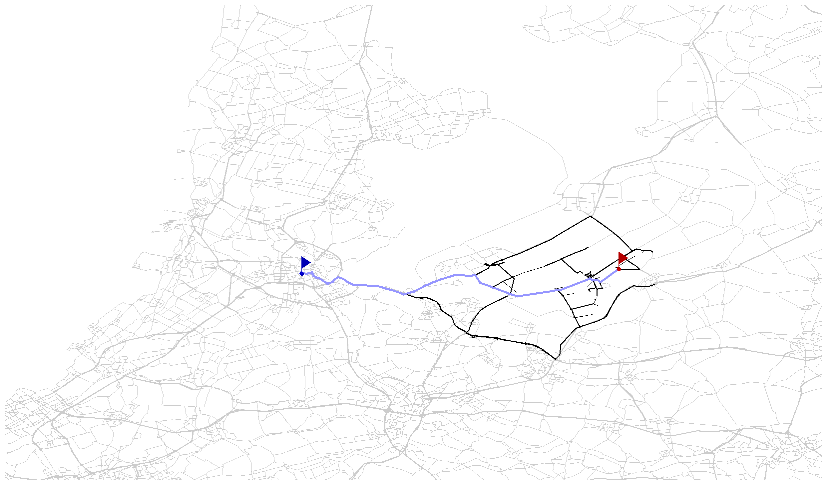
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Suchraum



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Suchraum

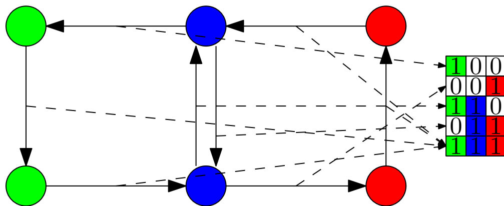


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Datenstruktur

Effizient Flaggen speichern?

- pro Kante eine Flagge
- Beobachtung: Anzahl Kombinationen begrenzt



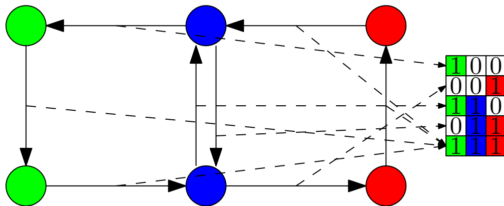
Idee:

- speichere Flaggen in Matrix
- Zeiger von Kanten auf die Matrix
- verringert Speicherverbrauch um einen Faktor 5

Flaggenkompression

Beobachtung:

- kippen eines Bits von 1 auf 0 verboten
- kippen eines Bits von 0 auf 1 erlaubt (weiterhin korrekt, eventuell langsamer)



Idee:

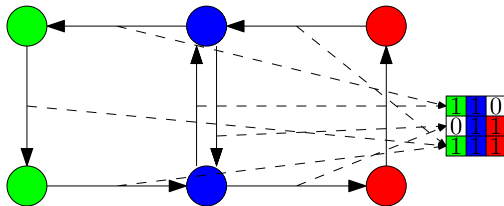
- verringere Anzahl eindeutiger Arc-Flags durch kippen
- dadurch Kompression der Matrix
- finde “gutes” Mapping

Flaggenkompression

KIT
Karlsruher Institut für Technologie

Beobachtung:

- kippen eines Bits von 1 auf 0 verboten
- kippen eines Bits von 0 auf 1 erlaubt (weiterhin korrekt, eventuell langsamer)



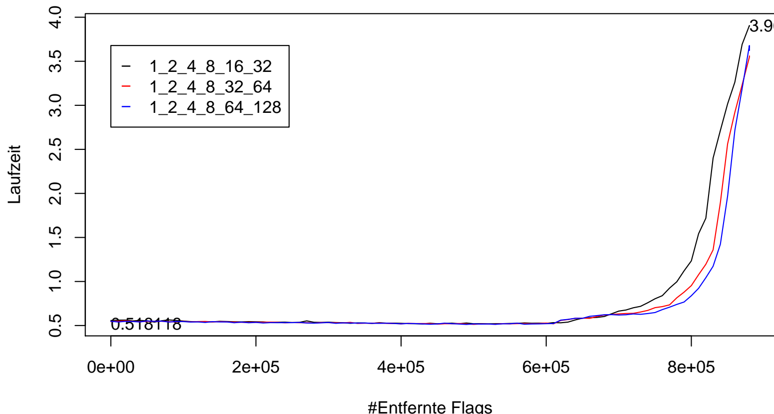
Idee:

- verringere Anzahl eindeutiger Arc-Flags durch kippen
- dadurch Kompression der Matrix
- finde "gutes" Mapping

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Flaggenkompression

(Multi-Level, unidirektional)



Beobachtungen

Karlsruher Institut für Technologie

- kaum Verlust bis zu 60% entfernte Flaggen
- geringer Verlust bis zu 80% entfernte Flaggen
- kippen von niedrig-leveligen Flaggen billiger

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zufallsanfragen: Anzahl Regionen

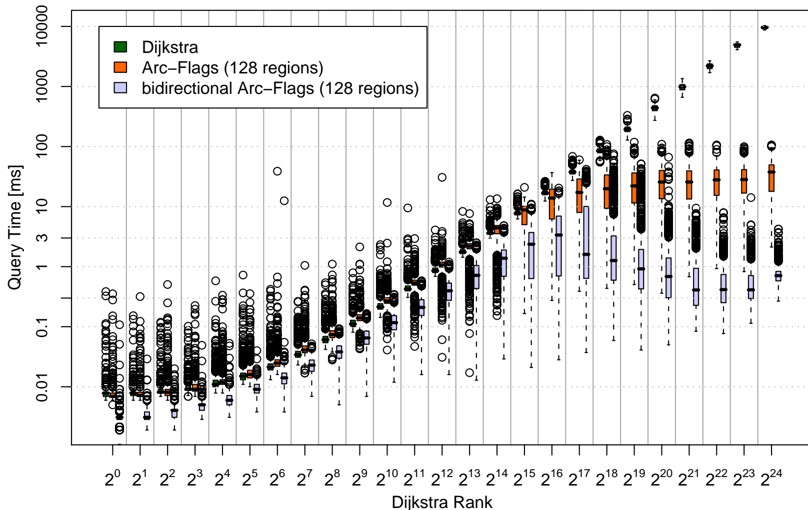
	Prepro		Query		
regions	time [min]	space [B/n]	# settled nodes	time [ms]	spd up
0	0	0	9 114 385	5 591.6	1.0
200	1 028	19	2 369	1.6	3 494.8
400	1 366	20	1 868	1.2	4 659.7
600	1 723	21	1 700	1.1	5 083.3
800	1 892	23	1 642	1.4	3 994.0
1000	2 156	25	1 593	1.1	5 083.3

Beobachtungen:

- lange Vorberechnung
- hohe Beschleunigung
- geringer Speicherverbrauch
- mehr als 200 Regionen lohnt sich nicht

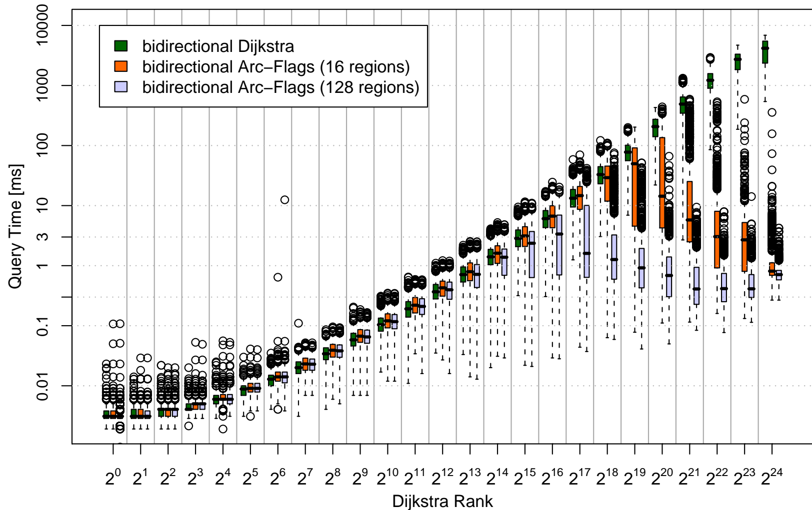
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Anfragen Arc-Flags I



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Anfragen Arc-Flags I



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beobachtungen

Karlsruher Institut für Technologie

- birektionale Arc-Flags deutlich schneller als unidirektionale
- gegenüber Dijkstra nur Beschleunigung für weite Anfragen
- 128 Regionen deutlich besser 16 (bei 2^{24} nahezu gleich auf)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Übersicht: bisherige Techniken

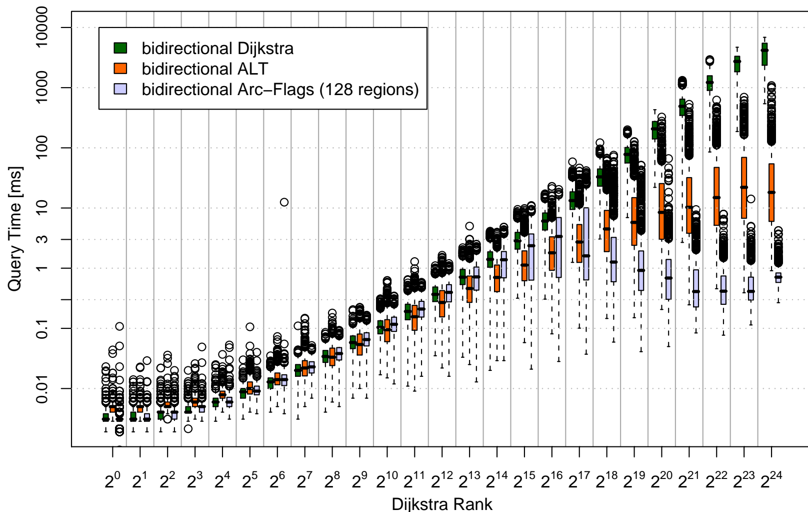
	Vorberechnung		Anfrage		
	Zeit [h:m]	Platz [byte/n]	Such raum	Zeit [ms]	Beschl.
Dijkstra	0:00	0	9 114 385	5 591.6	1.0
Bi-Dijkstra	0:00	0	4 764 110	2 713.2	2.1
Uni-ALT-16	1:25	128	815 639	327.6	17.1
Uni-ALT-64	1:08	512	604 968	288.5	19.4
Bi-ALT-16	1:25	128	74 669	53.6	104.3
Bi-ALT-64	1:08	512	25 324	19.6	285.3
Uni Arc-Flags (128)	8:34	10	92 545	31.9	175.3
Bi Arc-Flags (128)	17:08	20	2 764	0.8	6 988.1

Beobachtung:

- ALT: deutlich unterlegen bei Anfragen und Platzverbrauch
- Arc-Flags: deutlich längere Vorberechnungszeiten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Anfragen Vergleich



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Zusammenfassung

Karlsruher Institut für Technologie

Arc-Flags

- Teile Graphen in k Regionen
- Flaggen zeigen an, ob Kante wichtig für Zielregion ist
- Einfacher Anfrage-Algorithmus
- Vorberechnung
  - Kürzeste-Wege-Baum von jedem Randknoten
  - Manche Flaggen können automatisch gesetzt werden
- Bidirektional und multi-level Erweiterungen
- Beschleunigung von bis zu 7000
- Nahe Anfragen nicht schneller als Dijkstra
- Vorberechnung dauert allerdings (immer noch) ca. 1 Tag

Literatur

Karlsruher Institut für Technologie

Literatur (Arc-Flags):

- Moritz Hilger and Ekkehard Köhler and Rolf H. Möhring and Heiko Schilling:
  Fast Point-to-Point Shortest Path Computations with Arc-Flags
  In: *Shortest Paths: Ninth DIMACS Implementation Challenge, 2009*

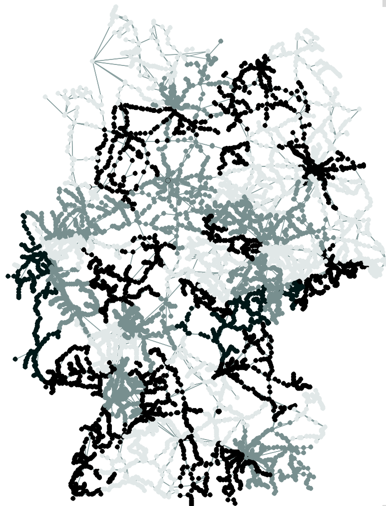
Partitionierung

Anforderungen:

- ausbalanciert
- wenige Randknoten
- zusammenhängend

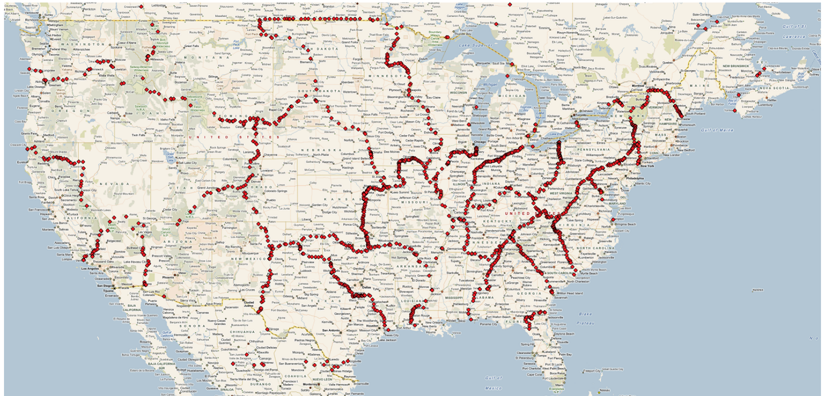
Black-Box Partitionierer:

- benutzen keine Einbettung
- oft: teilen rekursiv Graphen in k Teile mit kleinem Schnitt
- lassen sich auf eine Vielzahl Graphklassen anwenden
- heute (Exkurs): spezielle Strassengraph-Partitionierer



Graph Partitionierung

- Informell: teile Graphen in lose verbundene Regionen (Zellen).



Graph Partitionierung

- Formale definition:
  - Eingabe: ungerichteter graph $G = (V, E)$
  - Ausgabe: Partition von V in Zellen $V_1, V_2, \ldots, V_k$
  - Ziel: minimale Anzahl Kanten zwischen Zellen
- Standard Variante: $|V_i| \leq U$ für festes U:
  - Anzahl Zellen kann variieren ($\geq \lceil n/U \rceil$).
- Balancierte Variante: k Zellen und Unausgeglichenheit ϵ:
  - genau k Zellen (können unzusammenhängend sein) Zellen, Grösse $\leq (1 + \epsilon)\lceil n/U \rceil$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Graph Partitionierung

Karlsruher Institut für Technologie

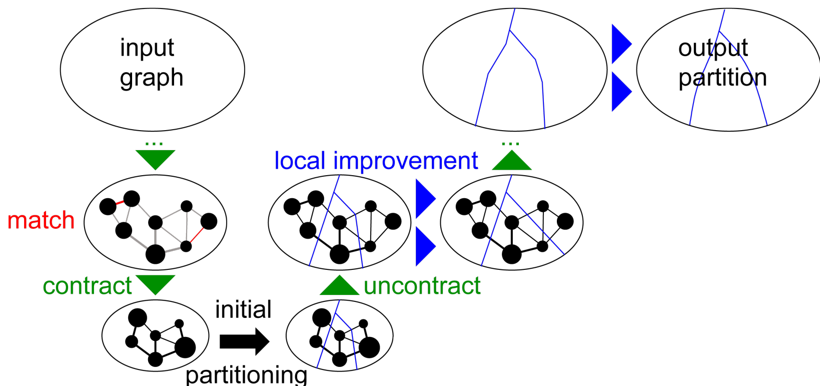
- Formale definition:
  - Eingabe: ungerichteter graph $G = (V, E)$
  - Ausgabe: Partition von V in Zellen $V_1, V_2, \ldots, V_k$
  - Ziel: minimale Anzahl Kanten zwischen Zellen
- Standard Variante: $|V_i| \leq U$ für festes U:
  - Anzahl Zellen kann variieren ($\geq \lceil n/U \rceil$).
- Balancierte Variante: k Zellen und Unausgeglichenheit ϵ:
  - genau k Zellen (können unzusammenhängend sein) Zellen, Grösse $\leq (1 + \epsilon)\lceil n/U \rceil$.

beides NP schwer $\Rightarrow$ benutze Heuristiken

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

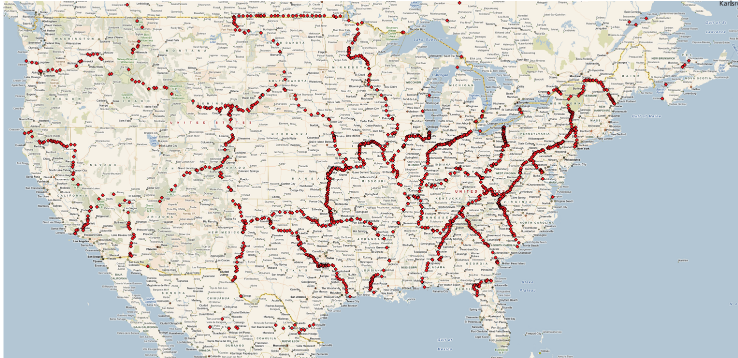
Generelle Verfahren

- METIS [KK99]
- SCOTCH [PR96]
- DiBaP [MMS09]
- Kappa [HSS10], KaSPar [OS10], Kaffpa [SS11], KaffpaE [SS12]



Natural Cuts

Karlsruher Institut für Technologie



Strassengraphen: dichte Regionen (Gitter) abwechselnd mit natürlichen Schnitten

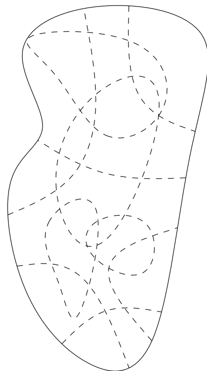
Flüsse, Berge, Wüsten, Wälder, Parks, Grenzen, Autobahnen, ...

PUNCH: Partitioner Using Natural-Cut Heuristics

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Übersicht

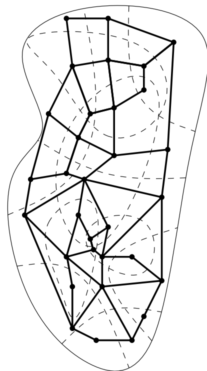
1. Ausdünnung:
   entspricht “match + contract” auf Folie 34
   - finde natürliche Schnitte (auf richtiger Skala)
   - behalte Schnittkanten, kontrahiere alle anderen
2. Zusammensetzen:
   - partitioniere (kleineren) kontrahierten graph
     “initial partitioning”
   - greedy + lokale Suche [+ Kombinationen]
     “uncontract + local improvements”



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Übersicht

1. Ausdünnung:
   entspricht “match + contract” auf Folie 34
   - finde natürliche Schnitte (auf richtiger Skala)
   - behalte Schnittkanten, kontrahiere alle anderen



2. Zusammensetzen:
   - partitioniere (kleineren) kontrahierten graph
     “initial partitioning”
   - greedy + lokale Suche [+ Kombinationen]
     “uncontract + local improvements”

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Schnitte: Definitionen

Karlsruher Institut für Technologie

Schnitt

- Partition des Graphen in 2 Teile (V_1, V_2)
- Grösse: Anzahl Schnittkanten $(|S|)$

Minimaler *s*-*t* Schnitt

- entferne minimale Anzahl Kanten, so dass *s* und *t* im Graphen nicht mehr verbunden sind
- kann in maximalen Fluss überführt werden
- in Polynomialzeit zu berechnen

Dünnster Schnitt

- Schnitt mit $|S| / \min\{|V_1|, |V_2|\}$ minimal
- NP schwer

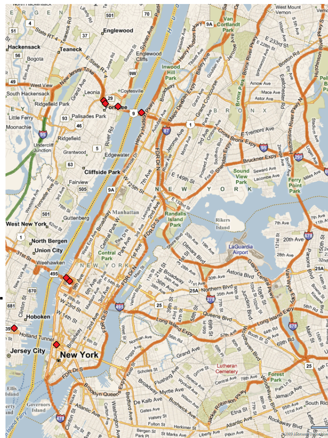
Exact Graph Bisection

- Schnitt mit $|S|$ minimal und $|V_1|, |V_2| < \lceil |V|/2 \rceil$
- NP schwer

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Finde Natürliche Schnitte

- Wir brauchen dünne Schnitte zwischen dichten Regionen:
- Dünnster Schnitt?
  - Zu aufwendig.
- Berechne minimalen *s*-*t* Schnitt (für zufälliges *s*, *t*)?
  - Meist trivial: Knotengrade sind klein.
- Wir brauchen was anderes:
  - *s*-*t* Schnitte zwischen Regionen



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Finde Natürliche Schnitte

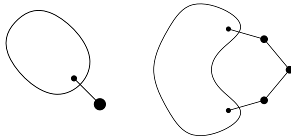
KIT Karlsruher Institut für Technologie

1. Wähle ein Zentrum v.
2. Breitensuche der Grösse U um v:
   - Erste $U/10$ Knoten: Kern
   - Verbliebene Knoten in der Queue: Ring
3. Finde minimum Kern/Ring Schnitt:
   - standard s–t minimaler Schnitt.
4. Wiederhole für verschiedene "zufällige" v:
   - bis jeder Knoten in ≥ 2 Kernen war

v

Berechene kleine Schnitte extra:

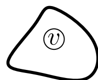
- identifiziere 1- and 2-Schnitte
- reduziert Strassengraph um Faktor 2
- beschleunigt Schnittfindung



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Finde Natürliche Schnitte

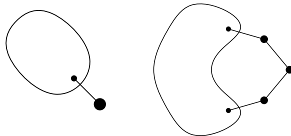
1. Wähle ein Zentrum v.
2. Breitensuche der Grösse U um v:
   - Erste $U/10$ Knoten: Kern
   - Verbliebene Knoten in der Queue: Ring
3. Finde minimum Kern/Ring Schnitt:
   - standard s–t minimaler Schnitt.
4. Wiederhole für verschiedene "zufällige" v:
   - bis jeder Knoten in ≥ 2 Kernen war



$U/10$ nodes

Berechene kleine Schnitte extra:

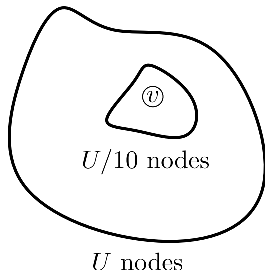
- identifiziere 1- and 2-Schnitte
- reduziert Strassengraph um Faktor 2
- beschleunigt Schnittfindung



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

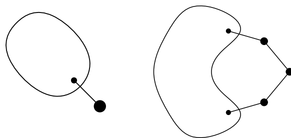
Finde Natürliche Schnitte

1. Wähle ein Zentrum v.
2. Breitensuche der Grösse U um v:
   - Erste $U/10$ Knoten: Kern
   - Verbliebene Knoten in der Queue: Ring
3. Finde minimum Kern/Ring Schnitt:
   - standard s–t minimaler Schnitt.
4. Wiederhole für verschiedene “zufällige” v:
   - bis jeder Knoten in ≥ 2 Kernen war



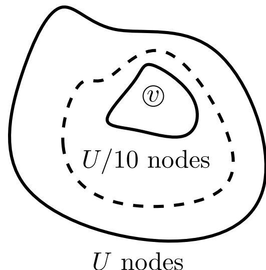
Berechene kleine Schnitte extra:

- identifiziere 1- and 2-Schnitte
- reduziert Strassengraph um Faktor 2
- beschleunigt Schnittfindung



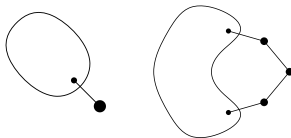
Finde Natürliche Schnitte

1. Wähle ein Zentrum v.
2. Breitensuche der Grösse U um v:
   - Erste $U/10$ Knoten: Kern
   - Verbliebene Knoten in der Queue: Ring
3. Finde minimum Kern/Ring Schnitt:
   - standard s–t minimaler Schnitt.
4. Wiederhole für verschiedene “zufällige” v:
   - bis jeder Knoten in ≥ 2 Kernen war



Berechene kleine Schnitte extra:

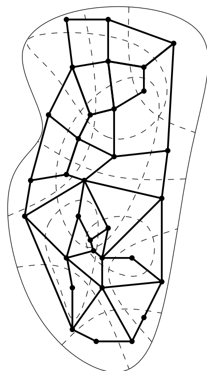
- identifiziere 1- and 2-Schnitte
- reduziert Strassengraph um Faktor 2
- beschleunigt Schnittfindung



Eigenschaften der Ausdünung

1. viele Kanten werden nie geschnitten
2. Schnittkanten partitionieren den Graphen in Fragmente
3. Fragmentgrösse $\leq U$ (meist viel kleiner)

- Generiere Fragmentgraphen:
  - Fragment $\rightarrow$ gewichteter Knoten
  - benachbarte Fragmente $\rightarrow$ gewichtete Kante



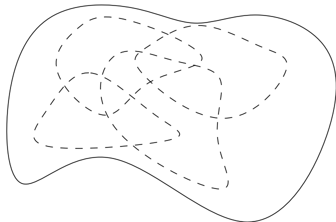
U	fragments	frag size
4 096	605 864	30
65 536	104 410	173
1 048 576	10 045	1 793

(Europe: 18M nodes)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

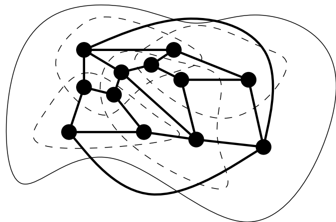
Konstruktionsphase

- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:



Konstruktionsphase

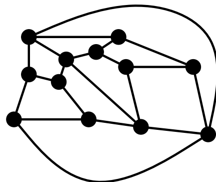
- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

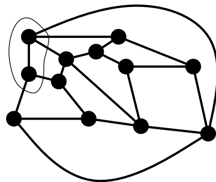
Konstruktionsphase

- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;
  - kombiniere adjazente Zellen;



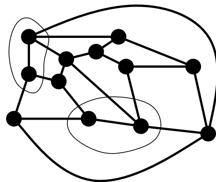
Konstruktionsphase

- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;
  - kombiniere adjazente Zellen;



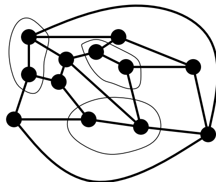
Konstruktionsphase

- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;
  - kombiniere adjazente Zellen;



Konstruktionsphase

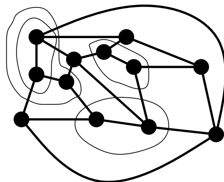
- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;
  - kombiniere adjazente Zellen;



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

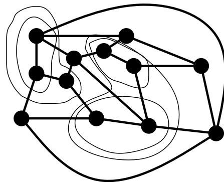
Konstruktionsphase

- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;
  - kombiniere adjazente Zellen;



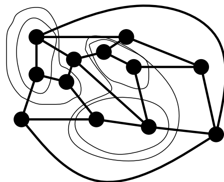
Konstruktionsphase

- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;
  - kombiniere adjazente Zellen;
  - stoppe wenn maximal.



Konstruktionsphase

- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;
  - kombiniere adjazente Zellen;
  - stoppe wenn maximal.

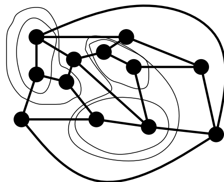


- Zufällig greedy:
  - füge Fragmente zusammen, die stärker verbunden...
  - ...im Verhältnis zu ihrer Grösse (= # Knoten, die sie repräsentieren).

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktionsphase

- Fragmentgröße deutlich unterhalb von U, Schnitt unnötig groß
- Finde bessere Partition durch Zusammenfassen von Fragmenten
- Algorithmus:
  - starte mit isolierten Fragmenten;
  - kombiniere adjazente Zellen;
  - stoppe wenn maximal.



- Zufällig greedy:
  - füge Fragmente zusammen, die stärker verbunden...
  - ...im Verhältnis zu ihrer Grösse (= # Knoten, die sie repräsentieren).

Ergebnis okay, aber es geht besser.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Suche

(Lokale Reoptimierung auf teilweise entpacktem Graphen)

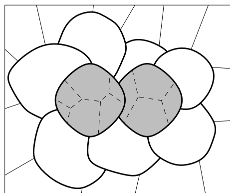
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Suche

Karlsruher Institut für Technologie

(Lokale Reoptimierung auf teilweise entpacktem Graphen)

- Für paarweise benachbarte Zellen:
  - Zerteilung in Fragments;
  - lass konstruktiven, randomisierten Algorithmus auf Subproblem laufen;
  - behalte Lösung wenn besser.

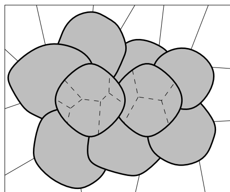


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Suche

(Lokale Reoptimierung auf teilweise entpacktem Graphen)

- Für paarweise benachbarte Zellen:
  - Zerteilung in Fragments;
  - lass konstruktiven, randomisierten Algorithmus auf Subproblem laufen;
  - behalte Lösung wenn besser.



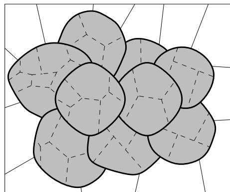
- Variante benutzt auch Nachbarzellen:
  - mehr Flexibilität;
  - beste Ergebnisse (standard).

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Suche

(Lokale Reoptimierung auf teilweise entpacktem Graphen)

- Für paarweise benachbarte Zellen:
  - Zerteilung in Fragments;
  - lass konstruktiven, randomisierten Algorithmus auf Subproblem laufen;
  - behalte Lösung wenn besser.



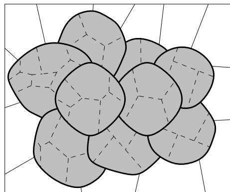
- Variante benutzt auch Nachbarzellen:
  - mehr Flexibilität;
  - beste Ergebnisse (standard).
- Nachbarzellen können auch in Fragmente zerteilt werden:
  - Subprobleme zu gross;
  - schlechtere Ergebnisse.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Lokale Suche

(Lokale Reoptimierung auf teilweise entpacktem Graphen)

- Für paarweise benachbarte Zellen:
  - Zerteilung in Fragments;
  - lass konstruktiven, randomisierten Algorithmus auf Subproblem laufen;
  - behalte Lösung wenn besser.

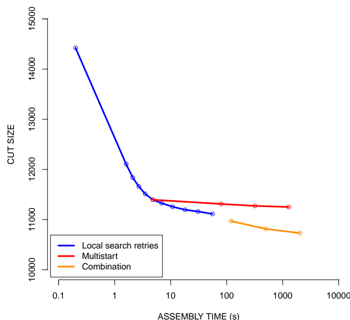


- Variante benutzt auch Nachbarzellen:
  - mehr Flexibilität;
  - beste Ergebnisse (standard).
- Nachbarzellen können auch in Fragmente zerteilt werden:
  - Subprobleme zu gross;
  - schlechtere Ergebnisse.

Evaluiere jedes Subproblem mehrmals (mit Zufall).

Weitere Verbesserungen

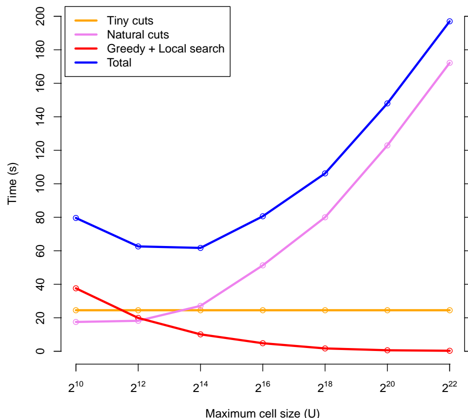
- Mehrfacher Test für jedes Paar
  - lokale Suche hat Zufallskomponenten
- Multistart:
  - konstruktiv+lokale Suche;
  - behalte beste Lösung.
- Kombination:
  - kombiniere manche Lösungen;
  - merge + lokale Suche.



(Europe, $U = 2^{16}$)

längere Laufzeit $\rightarrow$ bessere Lösungen

Laufzeit



Europe (18M vertices), 12 cores

Flaschenhalse: Aufbau für kleine U, Ausdünnung für grosse U

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Qualität

Karlsruher Institut für Technologie

U	A	B	$B/\sqrt{U}$	$B/\sqrt[3]{U}$
1 024	895	16.8	0.52	1.66
4 096	3 602	27.6	0.43	1.73
16 384	14 437	45.6	0.36	1.80
65 536	57 376	72.7	0.28	1.80
262 144	222 626	103.7	0.20	1.62
1 048 576	826 166	134.3	0.13	1.32
4 194 304	3 105 245	127.9	0.06	0.79

(Europe, 16 retries, no multistart/combination)

U: maximal erlaubte Zellengrösse

A: durchschn. Zellengrösse der Lösungen

B: durchschn. Randknoten pro Zelle

Strassengraphen haben sehr kleine Separatoren

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Balancierte Partitionen

Karlsruher Institut für Technologie

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

1. benutze standard PUNCH
   mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen,
   verteile den Rest (Multistart)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Balancierte Partitionen

Karlsruher Institut für Technologie

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

1. benutze standard PUNCH
   mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen,
   verteile den Rest (Multistart)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

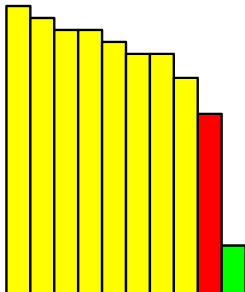
Balancierte Partitionen

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

1. benutze standard PUNCH
   mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen,
   verteile den Rest (Multistart)



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

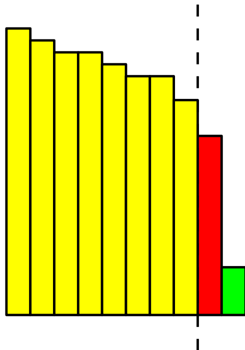
Balancierte Partitionen

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

1. benutze standard PUNCH mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen, verteile den Rest (Multistart)



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

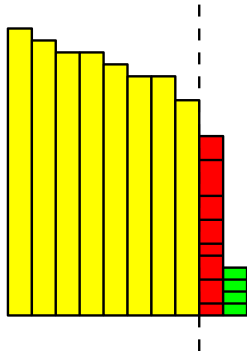
Balancierte Partitionen

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

1. benutze standard PUNCH
   mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen,
   verteile den Rest (Multistart)



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Balancierte Partitionen

Karlsruher Institut für Technologie

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

1. benutze standard PUNCH
   mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen,
   verteile den Rest (Multistart)

Balancierte Partitionen

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

1. benutze standard PUNCH mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen, verteile den Rest (Multistart)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

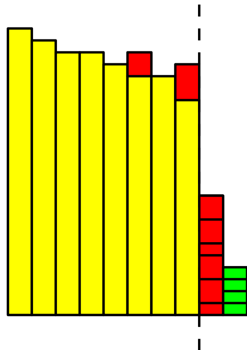
Balancierte Partitionen

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

1. benutze standard PUNCH mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen, verteile den Rest (Multistart)



Balancierte Partitionen

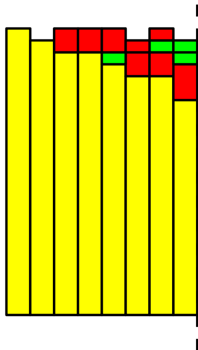
Karlsruher Institut für Technologie

Andere Verfahren lösen balancierte Variante:

- finde k Zellen mit Grösse $\leq (1+\epsilon)\lceil n/U \rceil$.

Erweiterung von PUNCH:

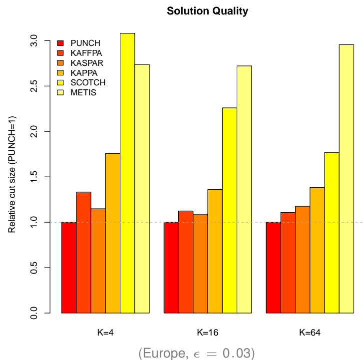
1. benutze standard PUNCH mit $U = (1+\epsilon)\lceil n/U \rceil$;
2. wähle k Basis Zellen, verteile den Rest (Multistart)



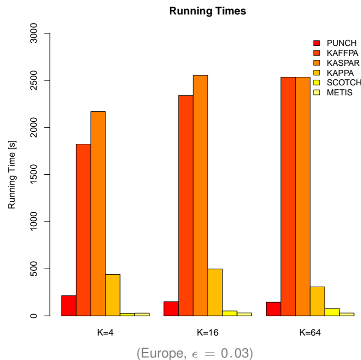
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Balancierte Partitionen

PUNCH: bessere Lösungen... ...in vernünftiger Zeit.



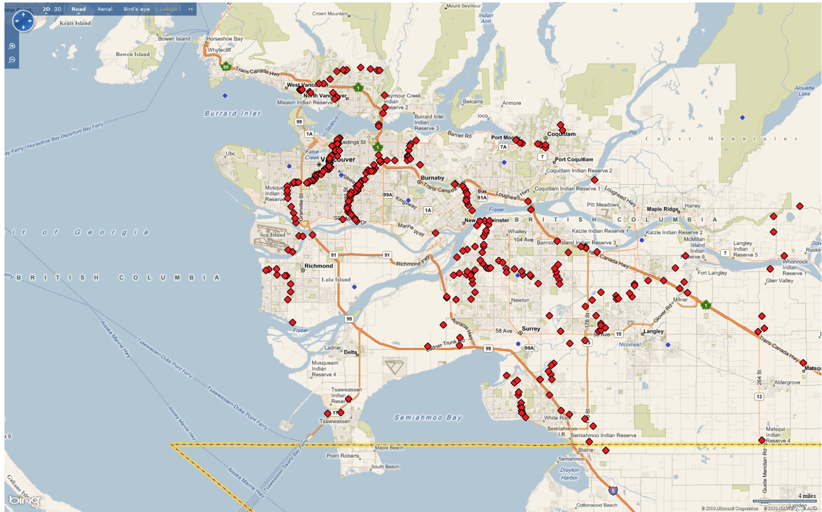
(Europe, $\epsilon = 0.03$)



(Europe, $\epsilon = 0.03$)

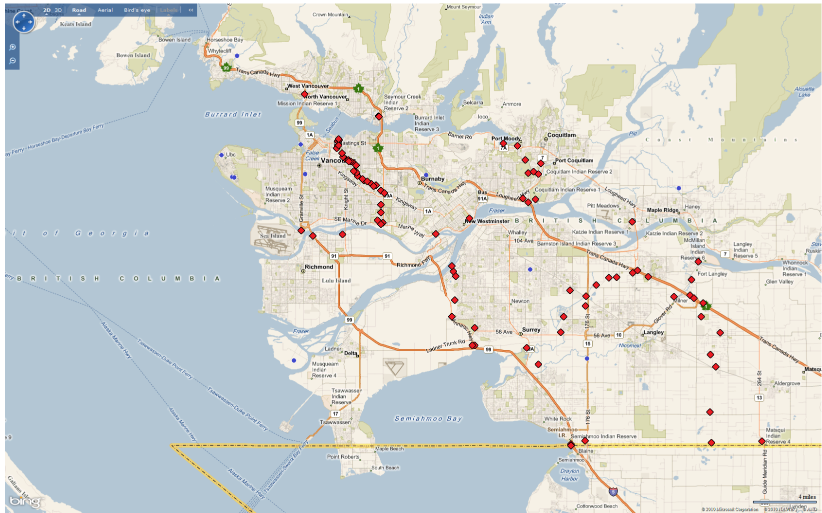
Vancouver mit METIS

Karlsruher Institut für Technologie



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

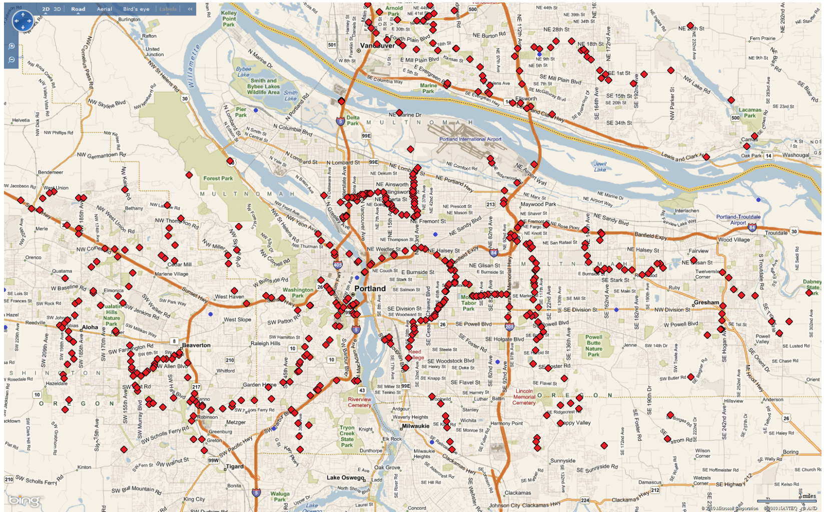
Vancouver mit PUNCH



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

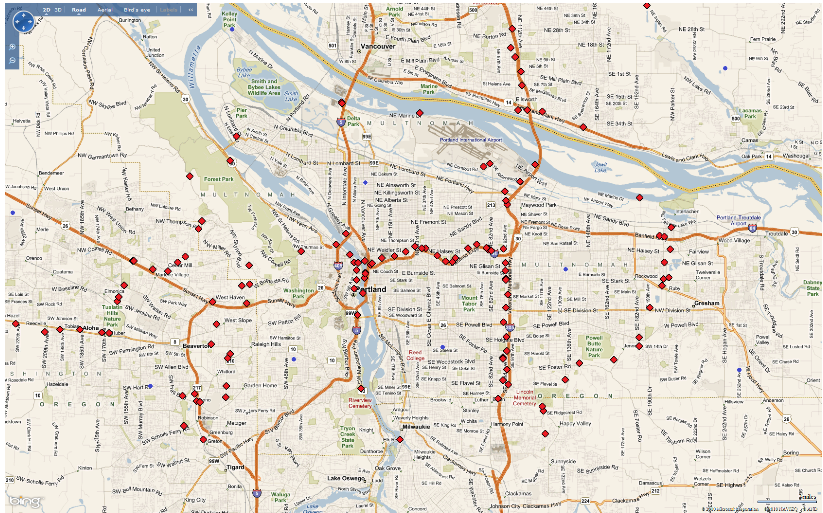
Portland mit METIS

Karlsruher Institut für Technologie



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

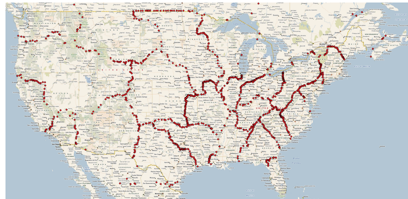
Portland mit PUNCH



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bemerkungen

- reduziert Randknoten gegenüber METIS um mehr als Faktor 2
- Beschleunigung von Arc-Flags Vorberechnung um Faktor 2
- auch multi-level Partitionen berechnenbar
  top-down liefert beste Ergebnisse
- dabei Vorteile gegenüber METIS sogar grösser
- Wie weit vom Optimum entfernt?
- Wird für die Bing Routing Engine benutzt.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Literatur

Karlsruher Institut für Technologie

Literatur (Partitionierung):

- George Karypis, Vipin Kumar:
  A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs
  In: *SIAM Journal on Scientific Computing, 1998*
- Peter Sanders, Christian Schulz:
  Distributed Evolutionary Graph Partitioning
  In: *ALENEX, 2012*
- Daniel Delling, Andrew Goldberg, Ilya Razenshteyn, Renato Werneck:
  Graph Partitioning with Natural Cuts
  In: *IPDPS, 2011*

Hierarchische Techniken

Karlsruher Institut für Technologie

Bisher: zielgerichtete Suche
Nächste Idee: Nutze "Hierarchie" im Graphen

in etwa: Seiten-, Haupt-, Land-, Bundesstraßen, Autobahnen

Ideensammlung:

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

Jetzt: letzteres

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

Wie V_L wählen?

Overlay Graph

Karlsruher Institut für Technologie

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

Wie V_L wählen?

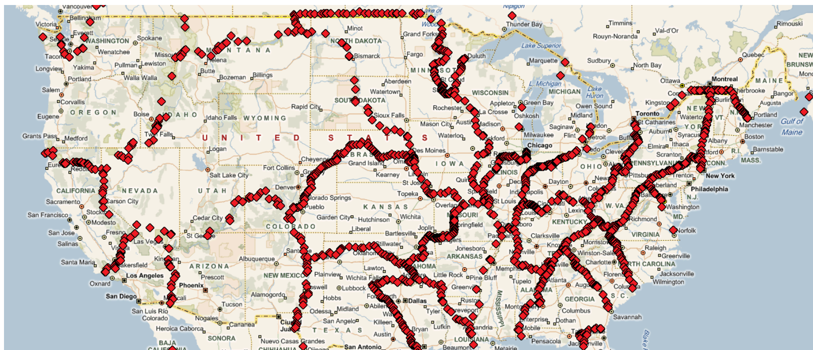
- Heute: Randknoten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Partitionierung

Karlsruher Institut für Technologie

Letztes Mal: Strassengraphen haben natürliche Schnitte



- Jeder Pfad durch eine Zelle betritt/verlässt die Zelle durch einen Randknoten

$\Rightarrow$ Minimiere # Schnittkanten mit Zellgrösse $\leq U$ (Eingabeparameter)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Karlsruher Institut für Technologie

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

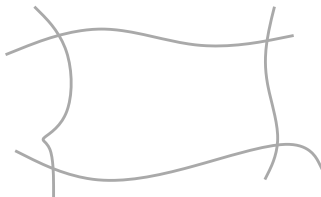
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

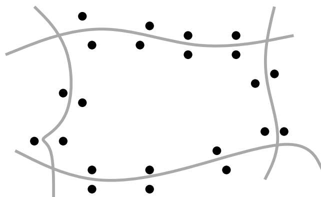
Ausnutzung der Partition

Karlsruher Institut für Technologie

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

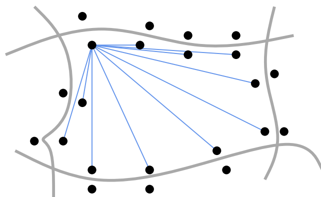


Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

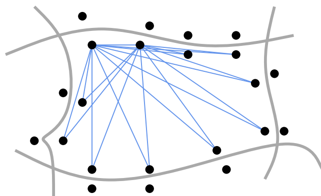


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Karlsruher Institut für Technologie

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

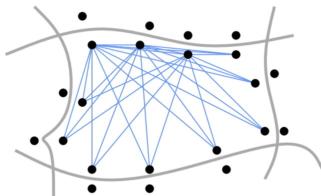
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

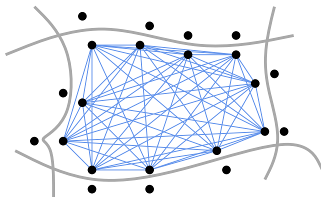
Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

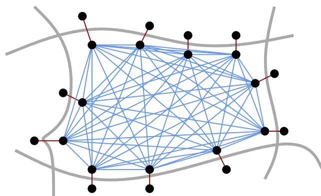


Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



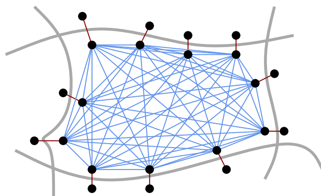
Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

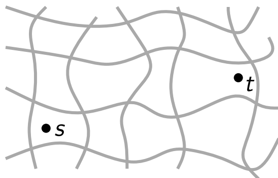
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Suchgraph:

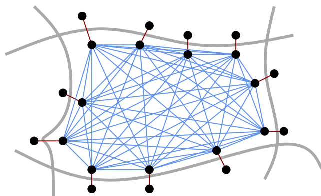
- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

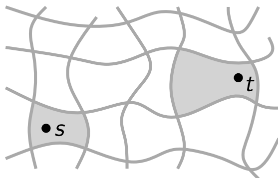
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





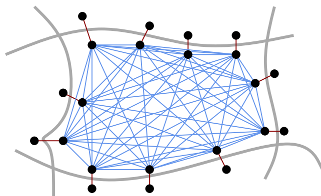
Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

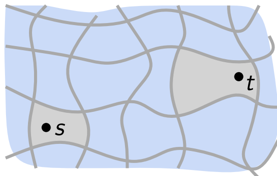
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Suchgraph:

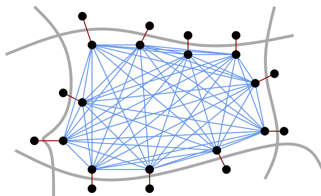
- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

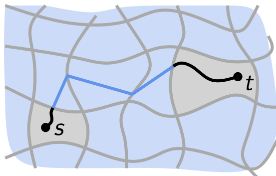
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



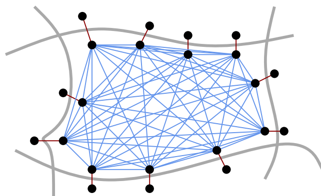


Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

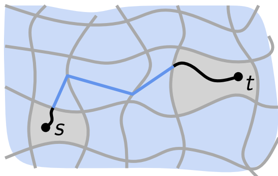
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*



Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten



Suchgraph:

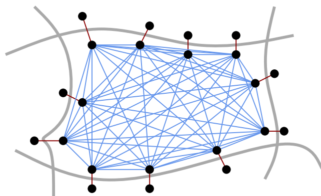
- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

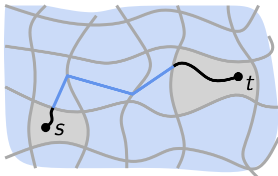
Ausnutzung der Partition

Idee: Berechne Distanzen zwischen Randknoten *in jeder Zelle*

Overlay Graph:

- Randknoten
- Cliquen in jeder Zelle
- Schnittkanten





Suchgraph:

- Start- und Zielzelle...
- ...plus Overlaygraph.
- (bidirektionaler) Dijkstra

Example

2^{15} Knoten pro Zelle, 626 Zellen $\Rightarrow$ 34 k Knoten im Overlaygraphen

Ergebnisse

Karlsruher Institut für Technologie

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Ergebnisse

Karlsruher Institut für Technologie

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Beispiel:

	Metric Customization		Queries	
metric	time [s]	space [MB]	scans	time [ms]
Travel time	20	10	45134	10
Distance	20	10	47127	11

(partition: $\leq 2^{14}$ nodes/cell)

Ergebnisse

Karlsruher Institut für Technologie

Worst-Case:

- Kanten scans: $O(\sum \text{cliques} + 2 \cdot \text{cell size})$.
  Grösse des Overlaygraphen is metrikunabhängig

Beispiel:

	Metric Customization		Queries	
metric	time [s]	space [MB]	scans	time [ms]
Travel time	20	10	45134	10
Distance	20	10	47127	11

(partition: $\leq 2^{14}$ nodes/cell)

West Europa (18 M nodes, 42 M edges)
Intel Core-i7 920 (four cores at 2.67 GHz)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Nächste Termine

Karlsruher Institut für Technologie

Mittwoch, 8.5.2013

Montag, 13.5.2013

Institut für Theoretische Informatik
Lehrstuhl Algorithmik