Teil III: Routing - Inhalt I

Teil III: Routing - Inhalt II

Weitere Fragestellungen

- Kosten
- Multicast
- Datenaggregation
- Datenzentrisches Routing

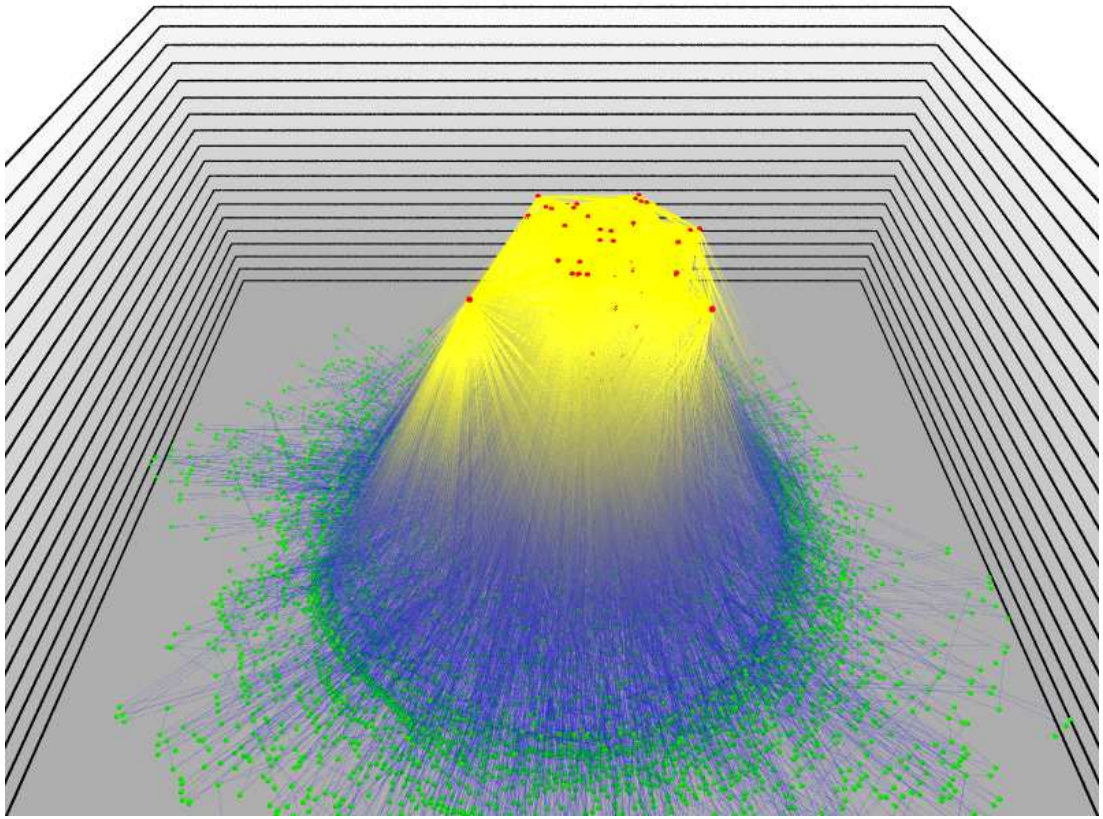
Routing

- Gegeben: Graph $G = (V, E)$, Quellknoten s_i, Zielknoten t_i
- Gesucht: Verteilter Algorithmus um Nachrichten von s_i nach t_i zu senden („Routing Protocol“)
- Bewertung von Routen ($=$ Wege in G)
  - Anzahl der verwendeten Kanten
  - Datendurchsatz auf der Route
  - Kosten für die Provider
  - ...
- Oft zwei Stufen
  - Vorberechnung („proaktiv“) von Informationen an Knoten
    - Problematisch bei mobilen Funknetzen: Topologieänderungen, Energieverbrauch, ...
  - Versenden von s_i nach t_i gemäß diesen Informationen

Routing im Internet

- Autonome Systeme (AS)
- Router zwischen Autonomen Systemen
  - Border Gateway Protocol (BGP)
- Router innerhalb eines Autonomen Systems
  - „Distance-Vector Protocols“, z.B. RIP
  - „Link-State Protocols“, z.B. OSPF
- Clients mit Verbindung zu einem Router („Default Gateway“)

Border Gateway Protocol: Top-Level Routing



AS-Graph

Routing Information Protocol (RIP)
Intra-AS Routing

- Basiert auf Bellman-Ford Algorithmus für kürzeste Wege:

1. $d_s := 0$, $d_v := \infty$ für alle $v \in V \setminus \{s\}$
2. Vorgänger vor(v) :=null für alle $v \in V$
3. für $i := 1$ bis $|V|$
4. für jede Kante $e = \{u, v\}$
5. falls $d_v > d_u + w_e$
6. $d_v := d_u + w_e$
7. vor$(v) := u$
8. (teste auf negative Kreise)

RIP: Verteilter Bellman-Ford Algorithmus (DBF)

- Knoten (Router) speichern Routing Tabelle
  - Für jede Zieladresse wird Distanz und Vorgänger gespeichert „Distance Vector“
  - Route Aggregation: Zusammenfassen von Subnetzen
- Berechnung der Routing Tabellen
  - Verteilter „all-pairs“ Bellman-Ford Algorithmus
  - Weglänge z.B. Anzahl Kanten („Hop Count“)
  - Periodische (asynchrone) Kommunikation (alle 30 Sekunden)
- Finden der Route zu Zieladresse
  - Sende Nachricht zu in der Routing Tabelle gespeichertem Vorgänger

RIP: Verträglichkeit mit Topologieänderungen

- Kein Abbruch
- Falls $\text{vor}(v) = u$ wird immer $d_v := d_u + w_e$ gesetzt, selbst wenn $d_v \leq d_u + w_e$
- Time-out: Nach 180 Sekunden ohne Nachricht von u
  - setze für alle Einträge mit $\text{vor}(v) = u$ die Distanzen auf ∞

Open Shortest Path First (OSPF)

- Gebräuchlichstes Inter-AS Routingprotokoll
- Knoten (Router) speichern den Kommunikationsgraph
  - „Link-State“
  - Verteilter Aufbau des Graphen durch „advertise“ Nachrichten
- Jeder Router berechnet einen Kürzeste-Wege Baum
  - Algorithmus von Dijkstra
- Hierarchisch
  - AS kann in Gebiete und Backbone aufgeteilt werden
  - Jeder Knoten speichert nur gebietsinterne Graphstruktur

Routing in Sensor und Ad-hoc Netzen

- Energie ist knapp
  - Ständiges „Flooding“ z.B. von Distanzvektoren ungeeignet
- Mobilität, Kommunikation über Funk
  - Topologie kann sich oft ändern
  - Routing-Tabellen werden ungültig und werden zu langsam aktualisiert
- Begrenzte Ressourcen
  - Speichern von kompletten Routing-Informationen evtl. nicht möglich
- Adressierung von Anfragen an „das Netz“
  - Punkt zu Punkt Kommunikation ungeeignet

Routing-Protokolle für Ad-hoc Netze

- Proaktiv: CGSR, DBF, DSDV, HSLS, HSR, LCA, MMRP, OLSR, STAR, TBRPF, WRP, ...
- Reaktiv („On-demand“): ARA, ABR, AODV, BSR, CHAMP, DSR, DSRFLOW, DNVR DYMO, FORP, GB, LBR, LMR, LUNAR, MOR, MPRDV, RDMAR, SSR, TORA, PLBR, ...
- Geo-Routing: Face, GOAFR, GOAFR+, ...
- Hierarchisch: BRP, CBRP, CEDAR, DART, DDR, FSR, GSR, HARP, HSR, IARP, IERP, LANMAR, OORP, ZRP, ...
- Geographisch: DREAM, GLS(Grid), LAR, GPSAL, ZHLS, GPSR, ...
- Energiebewußt: ISAIAH, PARO, EADSR, PAMAS, ...
- Multicast: ABAM, ADMR, AMRIS, DCMP, AMRoute, CAMP, CBM, DDM, FGMP, LAM DSR-MB, MAODV, MCEDAR, MZR, ODMRP, SRMP, LBM, GeoGRID, GeoTORA, MRGR, ...
- Datenbasiert: Directed Diffusion, ...

Link Reversal Routing „Gafni-Bertsekas Algorithmus“

- Sehr ähnliche Vorgehensweise im „Temporally-Ordered Routing Algorithm“ (TORA)
  - Reaktives Protokoll
  - Routing-Information wird nur bei Bedarf berechnet
- Gegeben: Graph G, Startknoten s und Zielknoten t
- Aufbau einer gerichteten azyklischen Version von G
  - Wird hier auch als gegeben vorausgesetzt
- Aktualisieren der Richtungen bei Änderungen im Graphen
  - Änderungen sind Einfügen oder Löschen von Kanten
  - Aktualisierung durch Umkehren einiger Kanten („link reversal“)

Gerichteter azyklischer Graph (DAG)

Definition 1

Ein gerichteter Graph G ohne gerichteten Zykel heißt Dag („directed acyclic graph“). Weiter heißt der Graph t-zielorientiert genau dann wenn t der einziger Knoten ohne ausgehende Kante ist.

Topologische Nummerierung

Definition 2

Eine Abbildung $t : V \to \mathbb{N}$ mit $t(u) \neq t(v)$ für $u \neq v$ heißt topologische Nummerierung von G falls

$$(u, v) \in E \Rightarrow t(u) > t(v).$$

Die sich ergebende absteigende Anordnung der Knoten heißt topologische Anordnung.

Lemma 3

Ein gerichteter Graph ist genau dann ein Dag wenn es eine topologische Anordnung der Knoten gibt.

Berechnen topologischer Nummerierungen

- Sei D ein gerichteter azyklischer Graph
  - Tiefensuche in D
  - Nummer eines Knotens:
    Letzter Bearbeitungszeitpunkt während der Tiefensuche

Berechnen eines t-zielorientierten Dags

- Sei G ein ungerichteter Graph und $A = (v_1, \ldots v_n)$ eine beliebige Anordnung der Knoten
- Breitensuche in G ausgehend vom Ziel t
- $l(u)$ sei der Level von $u \in V$ im Breitensuchbaum
- Orientiere eine Kante zwischen zwei Knoten: (u, v)
  - Zeigt vom größeren zum kleineren Level ($l(u) > l(v)$), oder
  - ist gemäß A gerichtet ($u = v_i, v = v_j, i < j$)

Routing mit DAGs

- Der Graph muss t-zielorientiert sein
  - Das Ziel t ist einziger Knoten ohne ausgehende Kante
- Routing von s nach t gemäß der Richtungen
  - Versende Nachricht über eine ausgehende Kante
  - Kein „kürzeste Wege Routing“
  - Mehrere Wege möglich
- Nachricht kommt bei t an weil
  - G keine Kreise enthält und
  - G zielorientiert ist.

Aktualisieren eines desorientierten Dags
Spezialfall „Full Reversal“

Verteilter Algorithmus

1. Falls der Knoten nicht Zielknoten ist
2. Falls der Knoten keine ausgehende Kante hat
3. Kehre die Richtungen aller eingehenden Kanten um

Aktualisieren eines desorientierten Dags
Höhenwerte für Knoten

- Die Knoten seien $v_1, \ldots, v_{n+1} = t$
- Sei $\mathcal{H}$ (Höhe) eine abzählbar unendliche Menge mit
  - einer totalen Ordnung $<$
  - einer Unterteilung in unendliche Teilmengen $\mathcal{H}_1, \ldots \mathcal{H}_n$
  - $\mathcal{H}_i$ ist nicht beschränkt bzgl. $<$
  - $\mathcal{H}_i$ ist eine abelsche Gruppe bzgl. einer Addition $+$
- $X := \mathcal{H}_1 \times \mathcal{H}_2 \times \ldots \times \mathcal{H}_n$
- $x = (a_1, \ldots, a_n) \in X$ definiert eine Orientierung der Kanten:
  - $a_i > a_j \Rightarrow$ Kante ist gerichtet von i nach j
    (Kanten von „oben“ nach „unten“ gerichtet)
  - Der so gerichtete Graph ist azyklisch
    (x liefert topologische Anordnung)

Aktualisieren eines desorientierten Dags
Zu aktualisierende Knoten

- $S(x) := \{i \mid \{i, n+1\} \notin E$ und
  $a_i < a_j$ für alle j mit $\{i, j\} \in E, 1 \leq i \leq n\}$
- Der x-orientierte Graph ist zielorientiert g.d.w. $S(x)$ leer
- $S(x)$ ist Menge der Knoten ohne ausgehende Kanten

Aktualisieren eines desorientierten Dags
Algorithmus

Algorithmus

- Gegeben x_0 (top. Nummerierung des ursprünglichen Dags)
- Berechne iterativ $x_{k+1} \in \mathcal{A}(x_k)$
- $\mathcal{A}(x_k)$ ist eine Teilmenge von X

Bemerkung

- Es wird verteilte Berechnung modelliert
- Mehrere Möglichkeiten aus x_k neue Höhen x_{k+1} zu berechnen
  - Reihenfolge, in der die Knoten in $S(x_k)$ abgearbeitet werden
  - Parallelität
- $\mathcal{A}(x_k)$ enthält alle möglichen verteilten Berechnungsschritte

Aktualisieren eines desorientierten Dags
Festlegung des Algorithmus

Voraussetzungen an $\mathcal{A}$

1. Es gibt Funktionen $g_i : X \to \mathcal{H}_i$, so dass entweder
   - $\mathcal{A}(x) = \{x\}$ falls $S(x)$ leer, ansonsten
   - $\mathcal{A}(x) = \{\overline{x} \mid \overline{x} \neq x$ und entweder $\overline{a}_i = a_i$ oder $\overline{a}_i = g(a_i)\}$
2. Es gilt
   - $g_i(x) > a_i$ falls $i \in S(x)$, und $g_i(x) = a_i$ sonst
   - $g_i(x)$ hängt nur von a_j ab für die $\{i, j\} \in E$
3. Sei $(x^k)_{k\in\mathbb{N}}$ unendliche Folge, Knoten i unendlich oft in $S(x^k)$

   - Dann ist

$$a_i^0 + \sum_{r=0}^{k} (g_i(x^r) - a_i^r)$$

   nicht beschränkt in $\mathcal{H}_i$

Beispiel: Full Reversal

- Definition der Höhenwerte $\mathcal{H}_i$ und der Funktionen g_i
  - $\mathcal{H} = \mathbb{N} \times \{1, \ldots, n\}$, lexikographisch sortiert
  - $(\alpha_i, i) + (\alpha'_i, i) := (\alpha_i + \alpha'_i, i)$
  - Sei $x = ((\alpha_1, 1), \ldots, (\alpha_n, n))$:

$$g_i(x) = \begin{cases} (\max\{\alpha_j \mid \{i,j\} \in E\} + 1, i) & \text{falls } i \in S(x) \\ (\alpha_i, i) & \text{falls } i \notin S(x) \end{cases}$$

- Es gelten die Voraussetzungen 1-3

Korrektheit

Satz 4

Sei G zusammenhängend. Für jede Folge $(x_k)_{k\in\mathbb{N}}$, die vom Algorithmus generiert wird, existiert ein Index $\overline{k}$, so dass $S(x_k)$ leer ist für alle $k \geq \overline{k}$.

Bemerkung

Der Algorithmus konvergiert also nach endlich vielen Schritten in einem $(n+1)$-zielorientierten Dag. ($S(x_k)$ ist leer für $k \geq \overline{k}$.)

Eindeutigkeit

Satz 5

Sei G zusammenhängend und $x_0 \in X$. Dann gibt es ein $x^* \in X$ (das nur von G und x_0 abhängt), so dass es für jede Folge $(x_k)_{k\in\mathbb{N}}$, die von obigem Algorithmus berechnet wird, ein $\overline{k} \in \mathbb{N}$ gibt mit $x_k = x^*$ für alle $k \geq \overline{k}$.

(ohne Beweis)

Stabilität

Lemma 6

Sei G zusammenhängend und $(x^k)_{k\in\mathbb{N}}$ eine Folge, die vom Algorithmus generiert wurde. Dann gilt für jeden Knoten i $(1 \leq i \leq n)$, für den im x^0-orientierten Graph ein Weg $(i, j_1, j_2, \ldots, j_m, n+1)$ existiert (d.h., $a_i^0 > a_{j_1}$, $a_{j_1}^0 > a_{j_2}^0, \ldots$):

$$a_i^0 = a_i^k \text{ für alle } k > 0.$$

Laufzeit

Lemma 7

Sei b die Anzahl der „schlechten“ Knoten in x_0, d.h. die Knoten ohne Weg zum Ziel t. Dann endet der Full-Reversal Algorithmus nach $O(b^2)$ Berechnungsschritten.

(ohne Beweis)

Lemma 8

Sei $b \in \mathbb{N}$ gegeben. Es gibt Graphen und Ausgangshöhen x_o mit b „schlechten“ Knoten, für die jeder Link-Reversal Algorithmus $\Omega(b^2)$ Berechnungsschritte ausführt.

(ohne Beweis)

Link Reversal Routing
Literatur

- Eli M. Gafni und Dimitri P. Bertsekas: Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications*, Com-29:1, 1981.

  `http://web.mit.edu/dimitrib/www/Gafni_Loopfree.pdf`

- C. Busch, S. Surapaneni und S. Tirthapura: Analysis of Link Reversal Routing Algorithms for Mobile Ad Hoc Networks. *Proc. ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 210-219, 2003.

  `http://www.cs.rpi.edu/~buschc/papers/reversal.pdf`

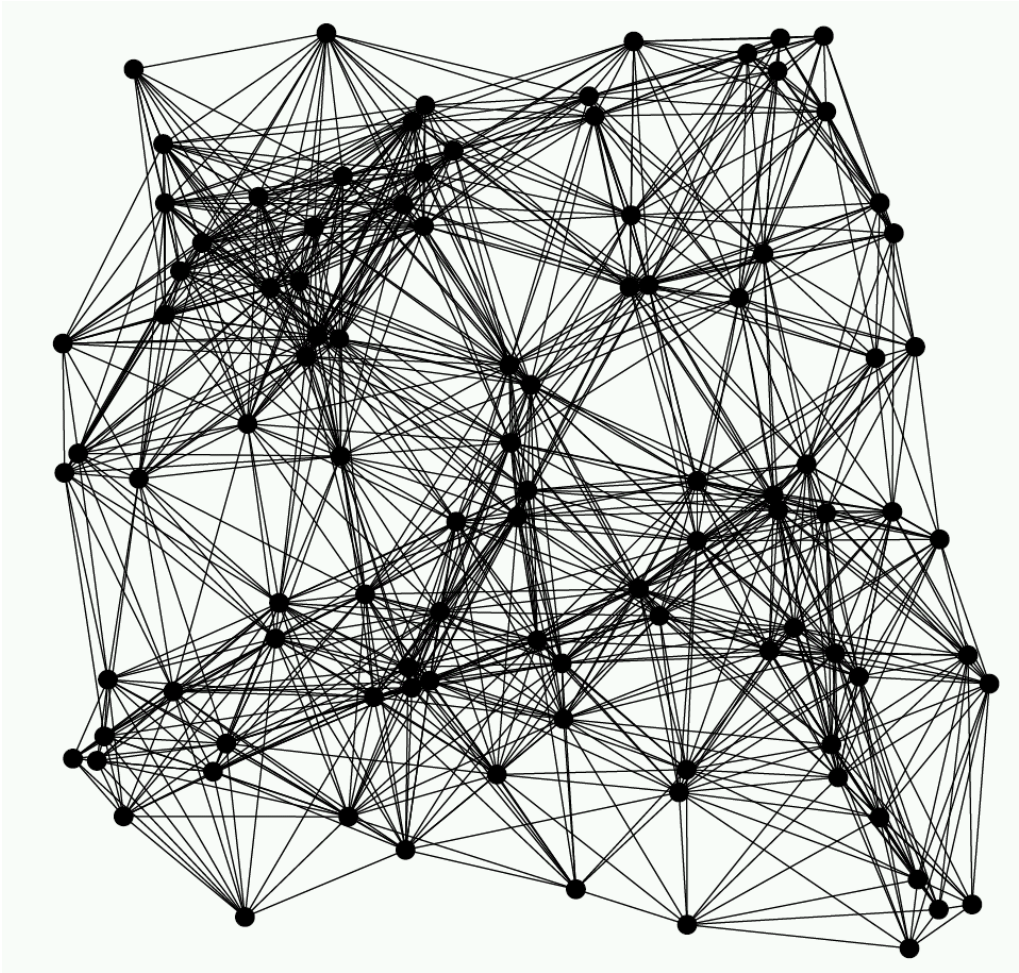
Geometric Routing
Literatur

- E. Kranakis, H. Singh und Jorge Urrutia: Compass Routing on Geometric Networks. *Canadian Conference on Computational Geometry*, 51–54, August 1999.
- Fabian Kuhn, Roger Wattenhofer und Aaron Zollinger: Asymptotically Optimal Geometric Mobile Ad-Hoc Routing. *Dial-M'02*, September 2002.

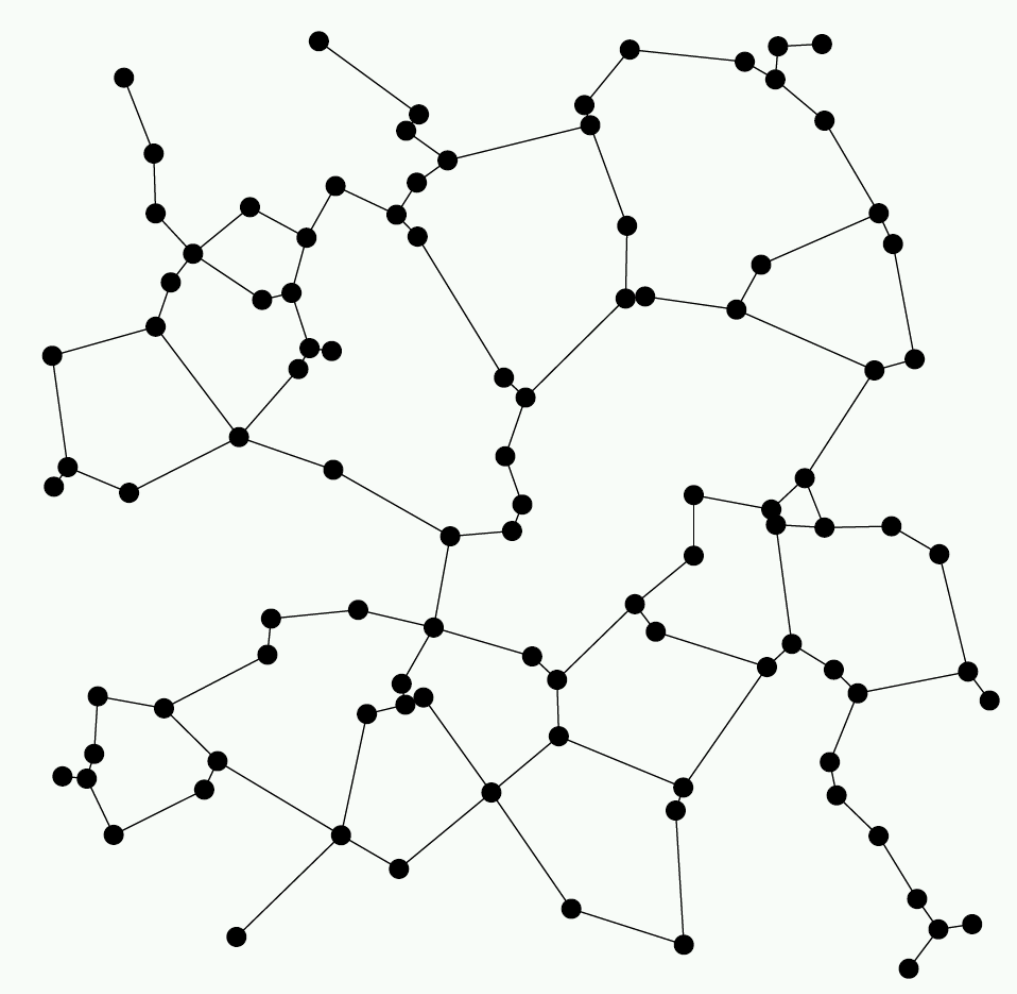
Voraussetzungen

- Kommunikationsgraph ist zshgd. Unit Disk Graph
- Einbettung in die Ebene ($\rightarrow$ Geometrisches Netzwerk)
  - Jeder Knoten kennt seine Koordinaten
  - Bezeichne euklidischen Abstand zwischen u und v als $d(u, v)$
- Koordinaten des Zielknotens t sind dem Startknoten s bekannt
- Algorithmen sind verteilt und lokal
- Algorithmen arbeiten auf planarem zshgd. Teilgraph
  - Schnitt mit Gabriel Graph
  - Schnitt mit Relative Neighbourhood Graph
  - ...

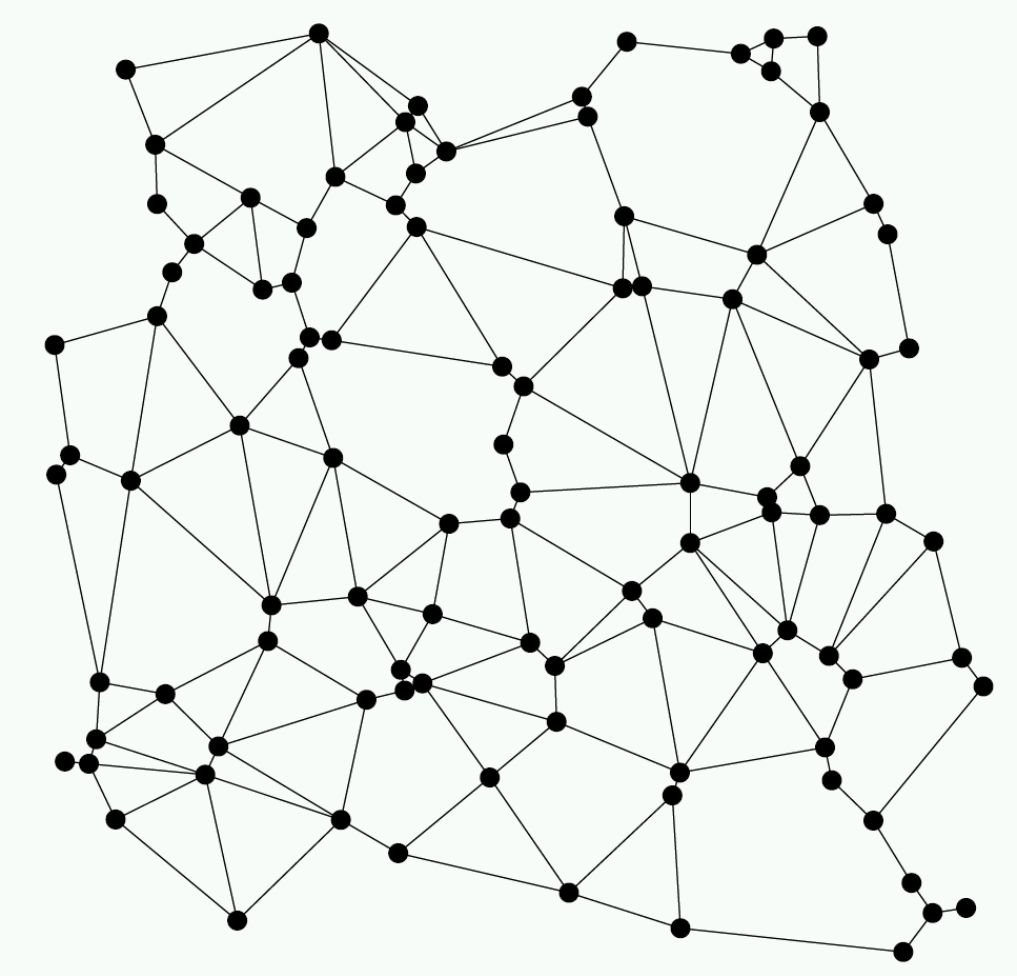
Unit Disk Graph (UDG)



UDG ∩ Relative Neighbourhood Graph (RNG)



UDG ∩ Gabriel Graph (GG)



Kosten

- Verschiedene Kostenfunktionen abhängig vom euklidischen Abstand d:
  - Konstant: $c_k(d) := 1$
  - Euklidisch: $c_d(d) := d$
  - Energie: $c_E(d) := d^2$ (evtl. auch höhere Exponenten)
- Kosten einer Kante: $c_x(u, v) := c_x(d(u, v))$
- Kosten des Algorithmus: Summe der Kantengewichte über alle gesendeten Nachrichten
- Kosten der Route: Summe aller Kanten auf der Route

$\Omega(1)$-Modell

Definition 9

Im $\Omega(1)$-Modell wird vorausgesetzt, dass es eine Konstante d_0 gibt, so dass je zwei Knoten mindestens Abstand d_0 haben.

Lemma 10

Für Unit Disk Graphen im $\Omega(1)$-Modell sind die drei Kostenmodelle Konstant, Euklidisch und Energie eines Weges $p = (e_1, \ldots, e_k)$ bis auf einen konstanten Faktor gleich. (Beweis s. Übung)

Gabriel Graph: Kürzeste Wege im $\Omega(1)$-Modell

Lemma 11

*Im $\Omega(1)$ Modell ist ein kürzester Weg im Schnitt des Unit Disk Graph mit dem Gabriel Graph höchstens um einen konstanten Faktor länger ist als ein kürzester Weg im Unit Disk Graph, für alle drei oben genannten Kostenmodelle. (*Beweis s. Übung)

Bemerkung

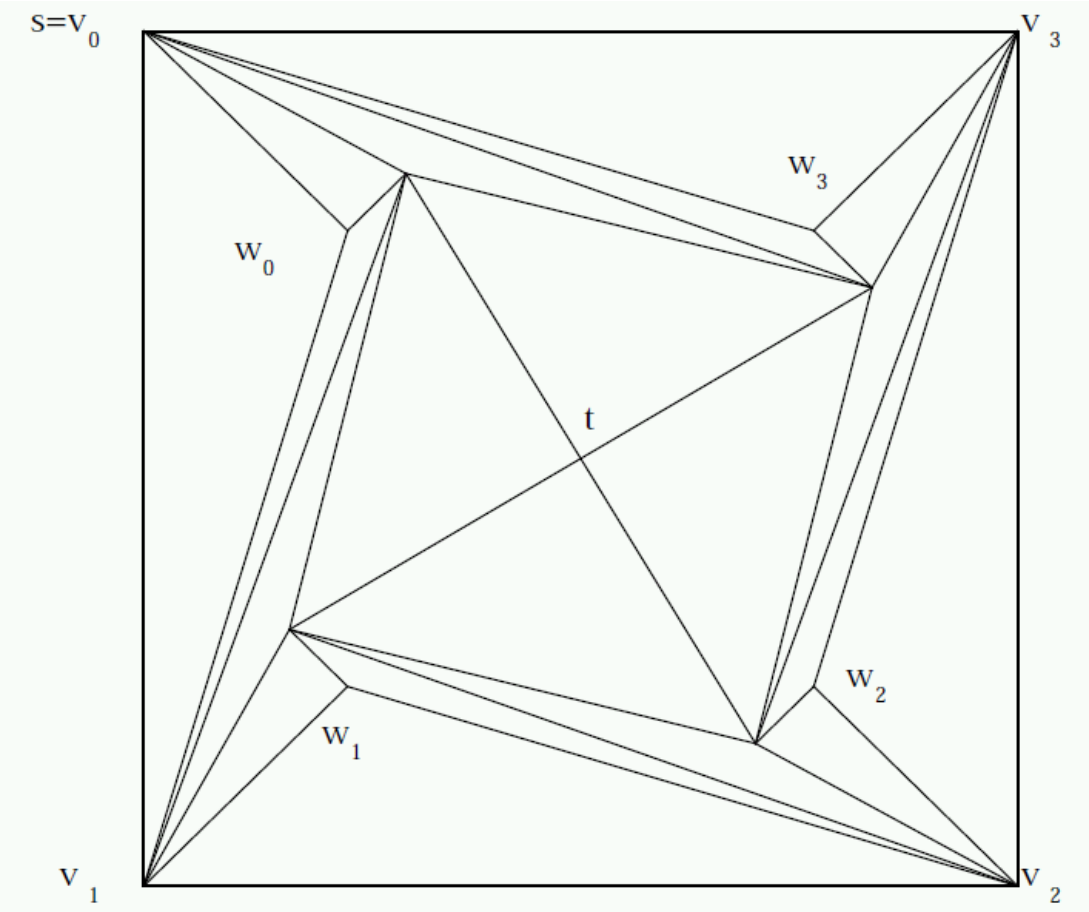
Teilgraphen mit dieser Eigenschaft nennt man Spanner. Im Allgemeinen, also ohne die Voraussetzung eines $\Omega(1)$ Modells, ist der Schnitt eines UDG mit dem Gabriel Graph kein Spanner bzgl. der konstanten und euklidischen Kosten.

Greedy Compass Routing

Algorithmus CR (für jeden Knoten v)

1. Sei e die Kante, deren Richtung am wenigsten von $\overline{vt}$ abweicht
2. Sende Nachricht über e

Greedy Compass Routing
Nachricht muss nicht immer ankommen

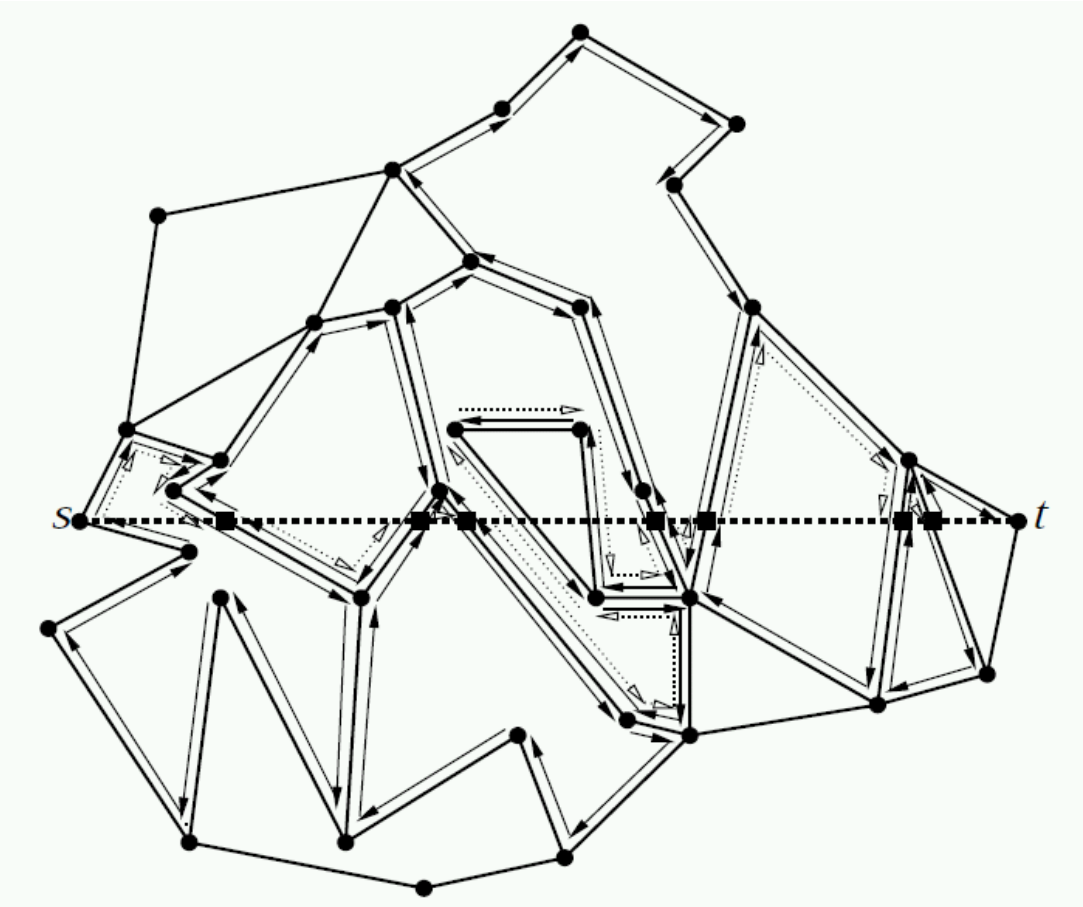


Face Routing

Algorithmus FR

1. Sei f die Facette, die $\overline{st}$ bei s schneidet
2. Solange t noch nicht erreicht wurde
3. Traversiere die Kanten von f
4. Merke die Kante e_f, deren Schnitt p mit $\overline{st}$ am nächsten an t ist
5. Traversiere die Kanten von f bis e_f erreicht ist
6. Sei f die Facette, die $\overline{st}$ bei p schneidet

Face Routing – Beispiel



Face Routing – Korrektheit & Kosten

Lemma 12

Beim Face Routing Algorithmus wird das Ziel t stets erreicht. Dabei werden höchstens $O(n)$ Kanten traversiert.

Diskussion

- Greedy Routing
  - Keine Garantie, dass Route gefunden wird
  - Effizient, falls Route gefunden wird
- Face Routing
  - Route wird garantiert gefunden
  - Alle Kanten auf Facetten, die $\overline{st}$ schneidet, werden traversiert
- Verbessere Face Routing
  - Traversiere weniger Kanten
- Betrachte allgemeine UDG (ohne $\Omega(1)$-Modell Einschränkung)
- Kombiniere Greedy- und Face Routing

Bounded Face Routing
Voraussetzungen

- Sei p^* ein kürzester s-t-Weg mit Kosten $c_d(p^*)$
- Sei $\widehat{c_d}$ obere Schranke für kürzesten s-t-Weg, also $c_d(p^*) \leq \widehat{c_d}$
- Wir nehmen an, $\widehat{c_d}$ sei dem Routing-Algorithmus bekannt
- $\mathcal{E}$ sei Ellipse mit Brennpunkten s und t
  - $u \in \mathcal{E}$ g.d.w. $d(s,u) + d(u,t) \leq \widehat{c_d}$

$\Rightarrow$ Jeder Weg über Punkt außerhalb von $\mathcal{E}$ ist länger als $\widehat{c_d}$

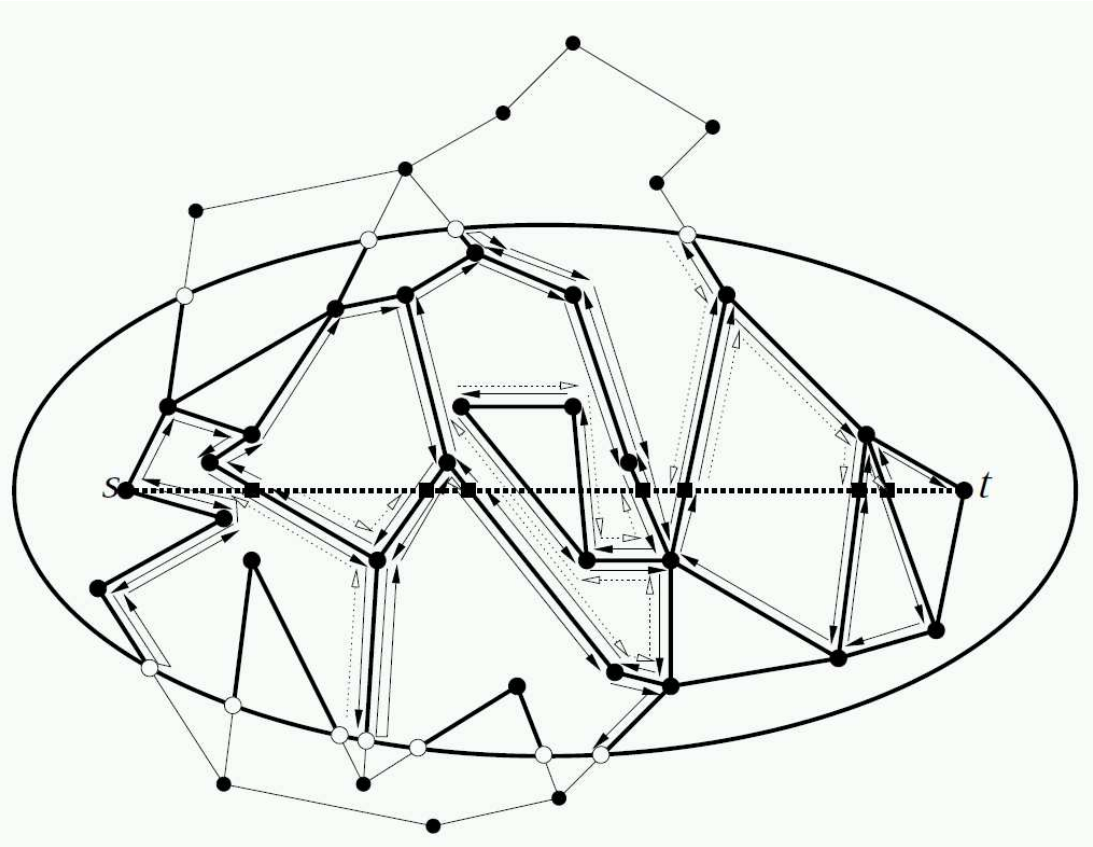
$\Rightarrow$ p^* ist komplett in $\mathcal{E}$ enthalten

Bounded Face Routing Algorithmus

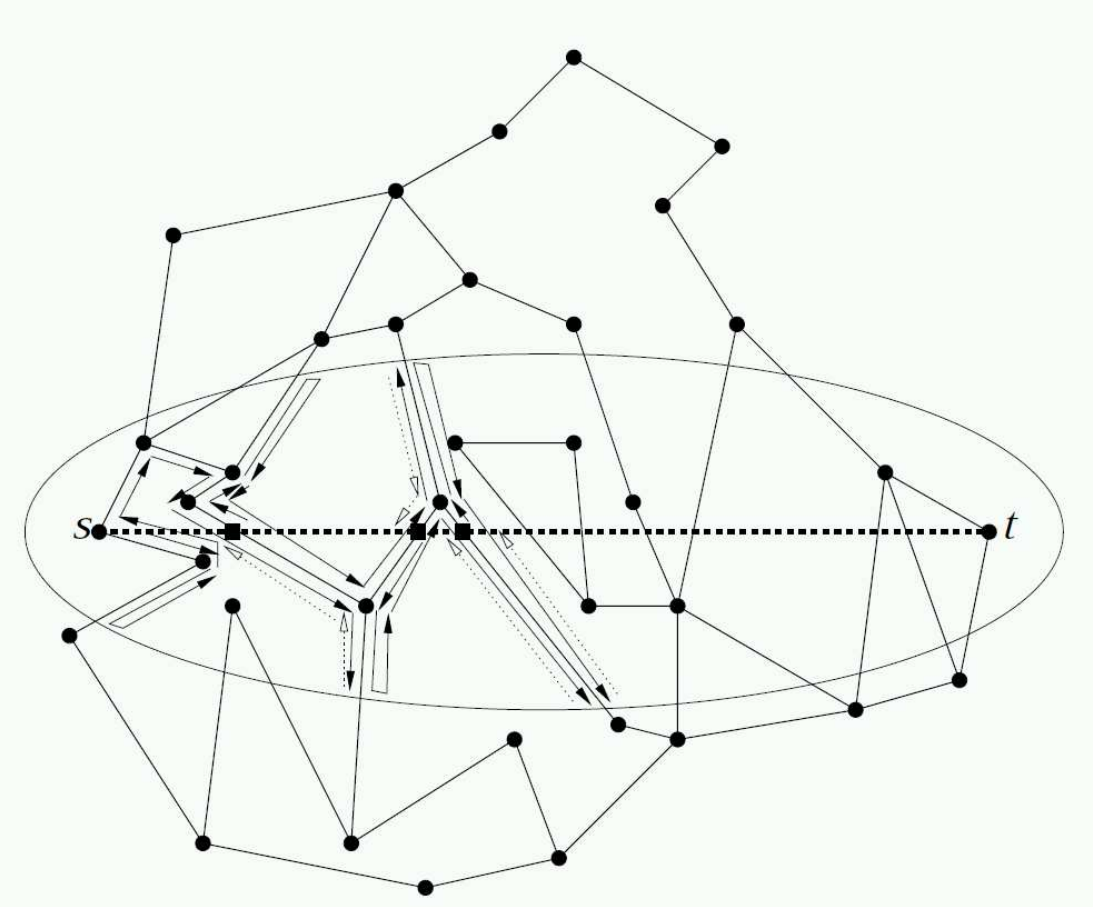
Algorithmus BFR($\widehat{c_d}$)

- Wie Face Routing mit folgenden Änderungen:
  - Traversiere die Facette nur innerhalb von $\mathcal{E}$
  - Beim ersten Verlassen von $\mathcal{E}$ traversiere die Facette in umgekehrter Richtung
  - Beim zweiten Verlassen von $\mathcal{E}$
    - Falls kein besserer Schnittpunkt p als im vorigen Schritt gefunden kehre zurück zu s

Boudend Face Routing
Beispiel 1: Nachricht erreicht t



Boudend Face Routing
Beispiel 2: Nachricht kehrt zu s zurück



Bounded Face Routing - Korrektheit & Kosten

Lemma 13

Sei G ein Graph im $\Omega(1)$-Modell. Falls $c_d(p^*) \leq \widehat{c_d}$, so erreicht eine Nachricht beim Bounded Face Routing immer das Ziel t. Wenn das Ziel nicht erreicht wird, so kehrt die Nachricht zu s zurück. In jedem Fall werden höchstens $O(\widehat{c_d}^2)$ Kanten traversiert.

Adaptive Face Routing

Algorithmus AFR

1. Setze $\widetilde{c_d} := 2d(s,t)$
2. Führe BFR($\widetilde{c_d}$) aus
3. Falls t noch nicht erreicht wurde
4. Setze $\widetilde{c_d} := 2\widetilde{c_d}$
5. Gehe zu 1.

Adaptive Face Routing – Korrektheit & Kosten

Lemma 14

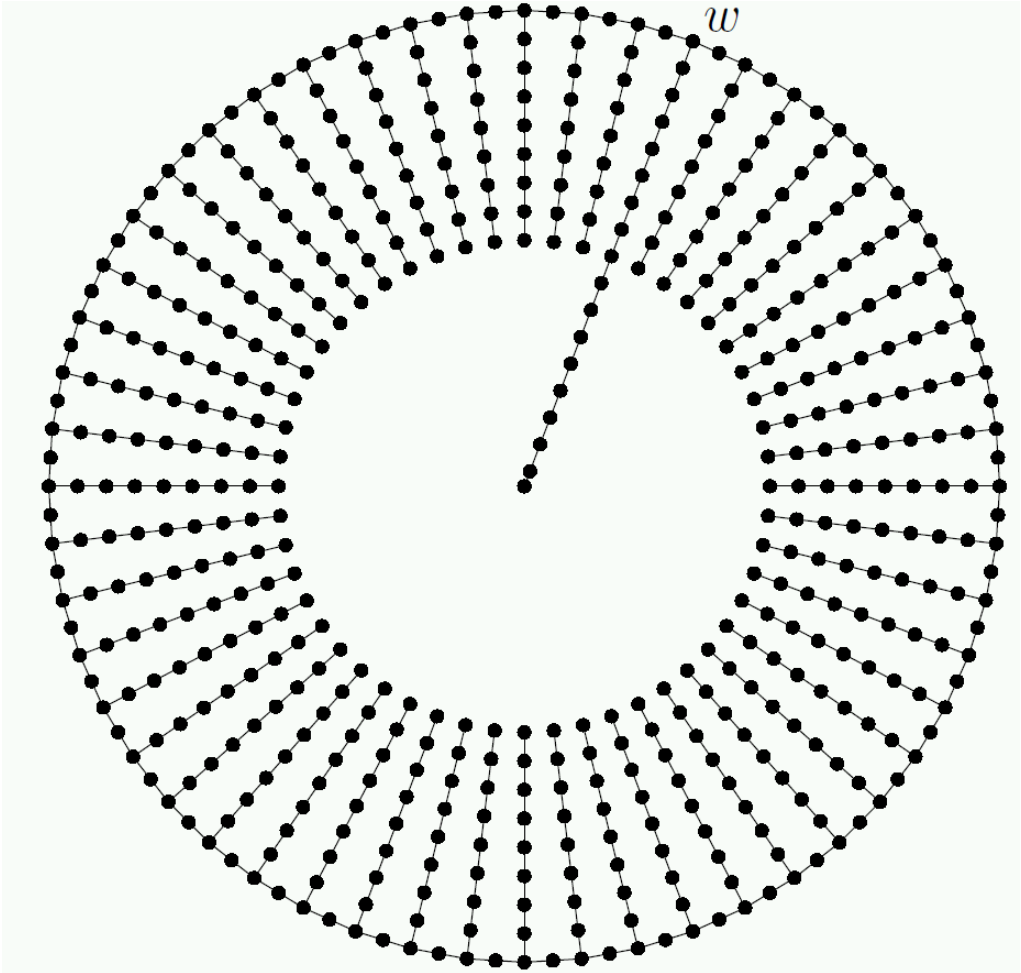
Im $\Omega(1)$-Modell erreicht der Face Routing Algorithmus stests das Ziel t und traversiert höchstens $O(c_d^2(p^*))$ Kanten.

Untere Schranke für geometrisches Routing
Lemma

Lemma 15

Seien c die Kosten einer optimalen Route. Dann hat jeder verteilte geometrische Routing Algorithmus im Worst-case Kosten von $\Omega(c^2)$ für jedes der drei betrachteten Kostenmodelle.

Untere Schranke für geometrisches Routing
Beweisskizze



Adaptive Face Routing – Ergebnis

Satz 16

Seien c die Kosten einer optimalen Route. In zusammenhängenden planaren Teilgraphen eines Unit Disk Graphen im $\Omega(1)$-Modell ist Adaptive Face Routing stets erfolgreich für die drei betrachteten Kostenfunktionen c_k, c_d und c_E mit Worst-Case Kosten von $\Theta(c^2)$.

Verallgemeinerung des $\Omega(1)$-Modells I

Definition 17

Sei G ein Unit Disk Graph. Falls der Grad jedes Knoten durch eine vorgegebene Zahl k beschränkt ist, heißt G Bounded Degree Unit Disk Graph (BDUDG) der Ordnung k.

Verallgemeinerung des $\Omega(1)$-Modells II

Lemma 18

Sei ein Unit Disk Graph G im $\Omega(1)$-Modell gegeben mit Konstante d_0. Dann ist G ein Bounded Degree Unit Disk Graph der Ordnung $4/d_0^2 + 4/d_0 + 1$.

AFR auf Bounded Degree Unit Disk Graphen

Satz 19

Seien c die Kosten einer optimalen Route. In zusammenhängenden planaren Teilgraphen eines Bounded Degree Unit Disk Graphen ist Adaptive Face Routing stets erfolgreich. Für die drei betrachteten Kostenfunktionen c_k, c_d und c_E sind im Worst-Case die Kosten des Algorithmus in $\Theta(c^2)$.

AFR auf Bounded Degree Unit Disk Graphen Beweisidee

- Kostenfunktionen auf s-t-Wegen mit $d(s,t) > 1$ sind asymptotisch äquivalent auf BDUDG
  - Ersetzt Kostenäquivalenz im $\Omega(1)$-Modell
- In Kreis mit Radius 1 sind höchstens $k+1$ Knoten
  - Ergibt ähnliches Flächenargument wie im $\Omega(1)$-Modell

Linear beschränkte Kostenfunktionen

Definition 20

Eine Kostenfunktion $c(d)$ heißt linear beschränkt falls

$$\lim_{d \to 0} \frac{c(d)}{d} > 0.$$

Nicht linear beschränkte Kostenfunktionen heißen superlinear.

Bemerkung

Die konstante und euklidische Kostenfunktionen sind linear beschränkt, die Energiekosten sind superlinear.

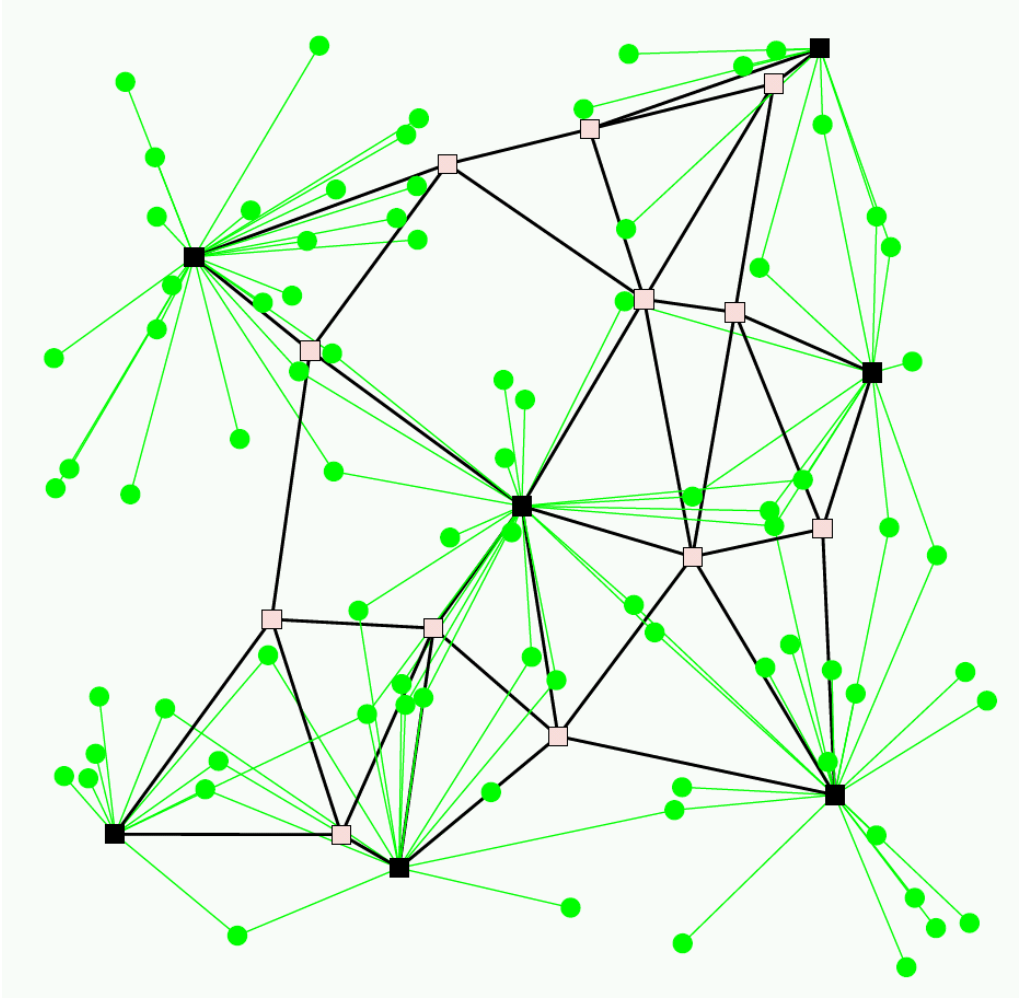
Allgemeine Unit Disk Graphen

- Verwende einen „Backbone“ Graph welcher
  - Bounded Degree Unit Disk Graph,
  - Spanner bzgl. linear beschränkten Kostenfunktionen und
  - Connected Dominating Set ist.

Algorithmus AFR-Backbone

1. Falls s kein Dominator schicke Nachricht an Dominator
2. Führe AFR auf planarem Teilgraph des Backbone aus
3. Falls Dominator v mit $d(v,t) \leq 1$ erreicht ist
4. Schicke Nachricht direkt zu t

AFR-Backbone – Beispiel



AFR mit Backbone auf allgemeinen UDG

Satz 21

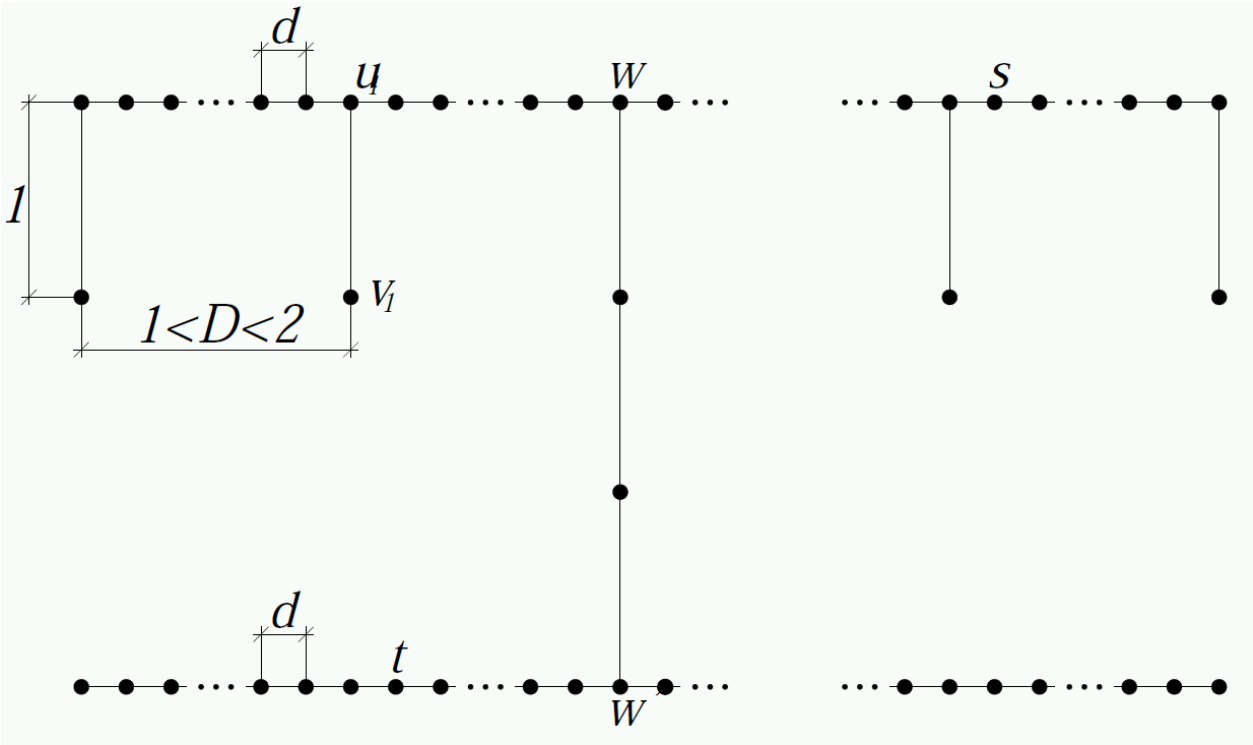
Seien c die Kosten einer optimalen Route. In zusammenhängenden Unit Disk Graphen ist der Algorithmus AFR-Backbone stets erfolgreich. Für linear beschränkte Kostenfunktionen sind im Worst-Case die Kosten des Algorithmus in $\Theta(c^2)$.

Superlineare Kostenfunktionen
Satz

Satz 22

Sei p^* die optimale Route bzgl. einer superlinearen Kostenfunktion $c(\cdot)$. Dann gibt es keinen lokalen geometrischen Algorithmus, dessen Kosten beschränkt sind durch eine Funktion von $c(p^*)$.

Superlineare Kostenfunktionen
Beweisskizze



Kostenvorteil in Funknetzen

- Bei einmaligem Senden alle Nachbarn erreicht
- Bisherige Kostenmodelle
  - Summiere über die Kommunikationskosten pro Kanten
- Kostenmodell für Funknetze
  - Summiere über die maximalen Kommunikationskosten pro Knoten

Multicast

- Sende Daten von einem Startknoten zu mehreren Zielknoten
- Gegeben
  - Graph $G = (V, E)$
  - Kantengewichte c
  - Startknoten s, Zielknoten $U \subset V$
- Problem
  - Gesucht: Baum minimalen Gewichts, der s und alle Knoten U enthält
  - Solch ein Baum heißt Steinerbaum
  - Das Steinerbaumproblem ist $\mathcal{NP}$-schwer
  - Approximationsalgorithmen, z.B. ähnlich wie die MST-Algorithmen

Multicast – Andere Kosten

- Beispiel eines Kostenmodells in Funknetzen:
  - Betrachte gerichtete Version eines Multicastbaum T
    - Wurzel s
    - Kanten zeigen von der Wurzel weg
  - Minimiere

$$\sum_{u \in T} \max_{e=(u,v) \in T} c(e)$$

Datenaggregation

- Empfange Daten an Senke s von mehreren Quellen $U \subset V$
- Zwischenknoten können Informationen verarbeiten, z.B.
  - Duplikate unterdrücken oder
  - Sensordaten auswerten
- „Umgekehrtes“ Multicast-Problem
- Gesucht
  - Baum mit Wurzel s, der alle Knoten U enthält
  - Kanten sind von Blättern zur Wurzel gerichtet

Kosten

- Sei $d(e)$ die Anzahl der Quellen im Teilbaum „unter“ e
- Modelliere Datenaggregation durch Funktion $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$
- Kosten eines Baumes T:

$$\sum_{e \in T} c(e) \cdot f(d(e))$$

- Beispiele
  - Unterdrückung von Duplikaten: $f(d(e))$ konstant für $d(e) \geq 1$
  - Keine Datenaggregation: $f(d(e)) := d(e)$
  - Bei Informationsverarbeitung an Knoten liegt f typischerweise zwischen diesen Extremfällen

Fragestellungen

- f ist als gegeben vorausgesetzt
  - Approximationsalgorithmus mit konstantem Faktor
  - Steinerbaumproblem ist Spezialfall (bei konstantem f)
- Bestimme Lösung, die gut ist „für alle $f \in \mathcal{F}$“
  - $\mathcal{F}$ sind konkave, nicht fallende Funktionen
  - Minimiere $\max_{f \in \mathcal{F}} c_{\mathcal{A}}(f)/c_{\mathrm{OPT}}(f)$
  - $O(\log k)$ Approximationsalgorithmus (bei k Quellen)

Problemstellung

- Bisherige Routingalgorithmen
  - Punkt zu Punkt Kommunikation
  - Zieladresse bekannt
- Datenzentrisches Routing: Szenario
  - An einem (oder mehreren) Knoten im Sensornetz wird Anfrage gestellt
  - Spezifikation der angeforderten Daten, z.B.
    - Bewegung in einem Gebiet X (Angabe der Koordinaten)
  - Broadcast dieser Anfrage (evtl. geographisch eingeschränkt)
  - Sensorknoten, die Informationen bzgl. der Anfrage haben, senden diese zurück zur Quelle

Beispiel: Directed Diffusion

- Kombination von
  - Datenzentrischem (und evtl. geographischem) Routing
    - Broadcast der Anfrage
    - Evtl. mittels geometrischem Routing
  - Datenaggregation
  - Dynamischem Routing
- Konzept des „Reinforcement“
  - Bestimme einen „besten“ Pfad zur Übermittlung von Sensordaten für eine Anfrage
  - Robust bzgl. dynamischen Änderungen im Sensornetz