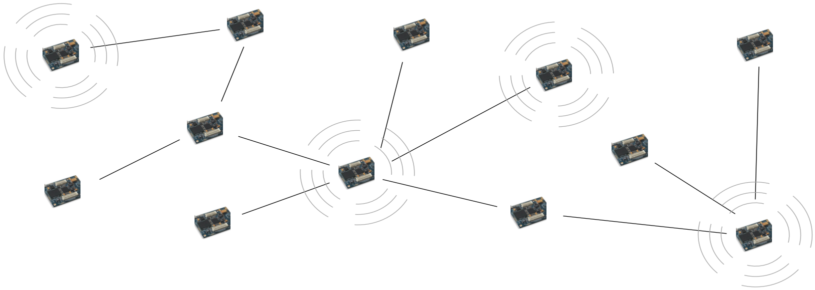
Algorithmen für Ad-hoc- und Sensornetze

VL 06 – Routing

Markus Völker | 30. Mai 2012 (Version 1)

INSTITUT FÜR THEORETISCHE INFORMATIK - LEHRSTUHL FÜR ALGORITHMIK (PROF. WAGNER)

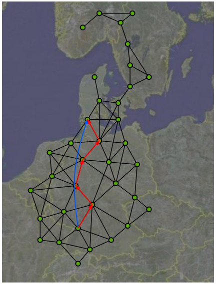


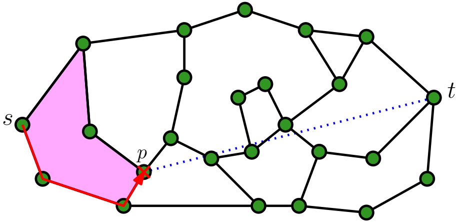
Rückblick VL02: Georouting

Kennt jeder Knoten seine Position und die seiner Nachbarn, kann man Pakete zu *Zielkoordinaten* routen.

- Greedy: schnell und einfach, keine garantierte Auslieferung
- Facettenrouting: Garantierte Auslieferung und Laufzeit

Müssen Koordinaten immer die tatsächl. Einbettung widerspiegeln?



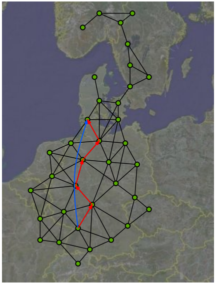


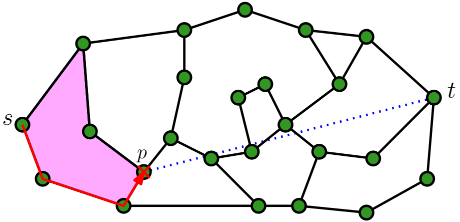
Rückblick VL02: Georouting

Kennt jeder Knoten seine Position und die seiner Nachbarn, kann man Pakete zu *Zielkoordinaten* routen.

- Greedy: schnell und einfach, keine garantierte Auslieferung
- Facettenrouting: Garantierte Auslieferung und Laufzeit

Müssen Koordinaten immer die tatsächl. Einbettung widerspiegeln?





Heute

- Greedy-Einbettungen in metrische Räume
  - Wenn Greedy Routing auf echten Koordinaten fehlschlagen kann, gibt es Koordinaten, auf denen es funktioniert?
  - zwei exotische Exemplare
- Pseudo-Geometrisches Routing
  - Greedy Routing ohne *metrische* Einbettung: Einfache Koordinaten, Routing ohne Tabellen
  - Theoretisches: Schranken
  - Praktisches: Beacon-Vector-Routing
- Compact Routing
  - Routing mit (kompakten) Routingtabellen
  - Ein neues Modell: Bounded Doubling Dimension

Heute

- Greedy-Einbettungen in metrische Räume
  - Wenn Greedy Routing auf echten Koordinaten fehlschlagen kann, gibt es Koordinaten, auf denen es funktioniert?
  - zwei exotische Exemplare
- Pseudo-Geometrisches Routing
  - Greedy Routing ohne *metrische* Einbettung: Einfache Koordinaten, Routing ohne Tabellen
  - Theoretisches: Schranken
  - Praktisches: Beacon-Vector-Routing
- Compact Routing
  - Routing mit (kompakten) Routingtabellen
  - Ein neues Modell: Bounded Doubling Dimension

Heute

- Greedy-Einbettungen in metrische Räume
  - Wenn Greedy Routing auf echten Koordinaten fehlschlagen kann, gibt es Koordinaten, auf denen es funktioniert?
  - zwei exotische Exemplare
- Pseudo-Geometrisches Routing
  - Greedy Routing ohne *metrische* Einbettung: Einfache Koordinaten, Routing ohne Tabellen
  - Theoretisches: Schranken
  - Praktisches: Beacon-Vector-Routing
- Compact Routing
  - Routing mit (kompakten) Routingtabellen
  - Ein neues Modell: Bounded Doubling Dimension

Greedy-Einbettungen

Greedy Routing

Weiterleiten der Nachricht an den Nachbarn, der dem Ziel am nächsten liegt (Greedy Routing) ist erfolgreich genau dann, wenn zu jedem Knotenpaar s, t ein Nachbar p von s existiert mit $d(p, t) < d(s, t)$.

Greedy-Einbettungen

Eine Einbettung in einen metrischen Raum, in der zu jedem Knotenpaar s, t ein Nachbar p von s existiert mit $d(p, t) < d(s, t)$ heißt *Greedy-Einbettung*. (Erstmal: metrisch=*euklidisch*)

Wann gibt es Greedy-Einbettungen?

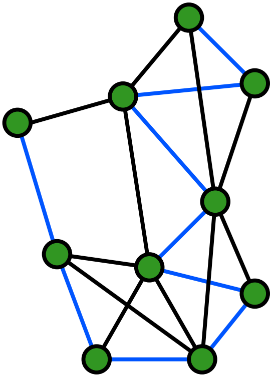
Satz

Enthält ein Graph einen hamiltonschen Pfad, gibt es eine Greedy-Einbettung in die Ebene.

- Beweis: Sei $v_1, v_2, \ldots$ ein hamiltonscher Pfad, dann ordne den Pfad auf einer Geraden an:

$$p(v_i) := (i, 0)$$

- Entspricht bereits Greedy-Einbettung in $\mathbb{R}$



Wann gibt es Greedy-Einbettungen?

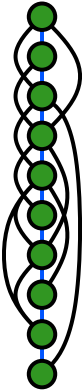
Satz

Enthält ein Graph einen hamiltonschen Pfad, gibt es eine Greedy-Einbettung in die Ebene.

- Beweis: Sei $v_1, v_2, \ldots$ ein hamiltonscher Pfad, dann ordne den Pfad auf einer Geraden an:

$$p(v_i) := (i, 0)$$

- Entspricht bereits Greedy-Einbettung in $\mathbb{R}$

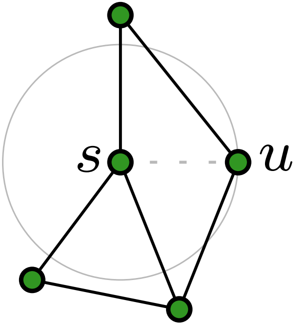


Wann gibt es keine Greedy-Einbettungen?

Lemma

In Greedy-Einbettungen ist jeder Knoten mit dem dichtesten Knoten verbunden.

- Beweis:
  - Sei ein Knoten s nicht mit dem dichtesten Knoten u verbunden.
  - Knoten u hat keinen Nachbarn, der dichter an s ist als er selbst.



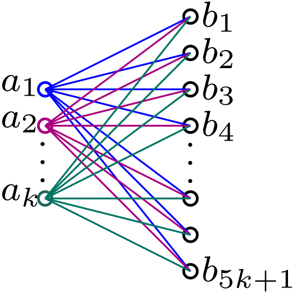
Wann gibt es keine Greedy-Einbettungen?

Satz

Die Graphen $K_{k,5k+1}$ haben keine Greedy-Einbettungen in der Ebene.

- Angenommen, ein $K_{k,5k+1}$ wäre so eingebettet:
  - Ordne jedes b_i dem dichtesten a_j zu
  - Einem a^* sind ≥ 6 b_i zugeordnet und
  - ein Winkel zwischen zwei b_i ist $\leq 60°$
  - nenne diese b^- (dichter) und b^+ (weiter weg)
  - b^+ ist dichter an b^- als an a^* (und damit an allen a_j)
  - dichtester Knoten an b^+ ist ein b_i
  - Widerspruch!

$\Rightarrow$ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*



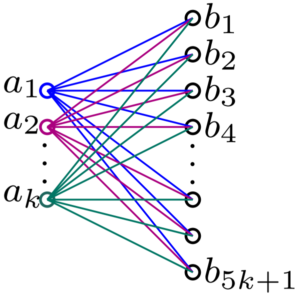
Wann gibt es keine Greedy-Einbettungen?

Satz

Die Graphen $K_{k,5k+1}$ haben keine Greedy-Einbettungen in der Ebene.

- Angenommen, ein $K_{k,5k+1}$ wäre so eingebettet:
  - Ordne jedes b_i dem dichtesten a_j zu
  - Einem $a^\star$ sind ≥ 6 b_i zugeordnet und
  - ein Winkel zwischen zwei b_i ist $\leq 60°$
  - nenne diese b^- (dichter) und b^+ (weiter weg)
  - b^+ ist dichter an b^- als an $a^\star$ (und damit an allen a_j)
  - dichtester Knoten an b^+ ist ein b_i
  - Widerspruch!

$\Rightarrow$ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*



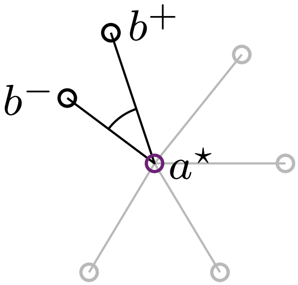
Wann gibt es keine Greedy-Einbettungen?

Satz

Die Graphen $K_{k,5k+1}$ haben keine Greedy-Einbettungen in der Ebene.

- Angenommen, ein $K_{k,5k+1}$ wäre so eingebettet:
  - Ordne jedes b_i dem dichtesten a_j zu
  - Einem $a^\star$ sind ≥ 6 b_i zugeordnet und
  - ein Winkel zwischen zwei b_i ist $\leq 60°$
  - nenne diese b^- (dichter) und b^+ (weiter weg)
  - b^+ ist dichter an b^- als an $a^\star$ (und damit an allen a_j)
  - dichtester Knoten an b^+ ist ein b_i
  - Widerspruch!

$\Rightarrow$ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*



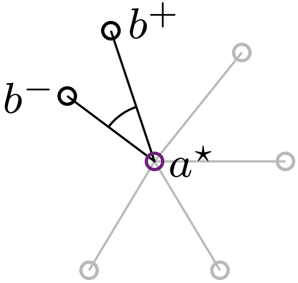
Wann gibt es keine Greedy-Einbettungen?

Satz

Die Graphen $K_{k,5k+1}$ haben keine Greedy-Einbettungen in der Ebene.

- Angenommen, ein $K_{k,5k+1}$ wäre so eingebettet:
  - Ordne jedes b_i dem dichtesten a_j zu
  - Einem $a^\star$ sind ≥ 6 b_i zugeordnet und
  - ein Winkel zwischen zwei b_i ist $\leq 60°$
  - nenne diese b^- (dichter) und b^+ (weiter weg)
  - b^+ ist dichter an b^- als an $a^\star$ (und damit an allen a_j)
  - dichtester Knoten an b^+ ist ein b_i
  - Widerspruch!

$\Rightarrow$ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*



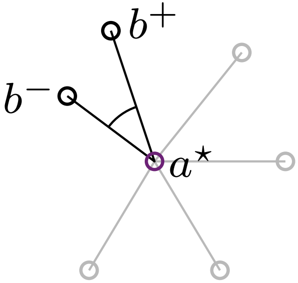
Wann gibt es keine Greedy-Einbettungen?

Satz

Die Graphen $K_{k,5k+1}$ haben keine Greedy-Einbettungen in der Ebene.

- Angenommen, ein $K_{k,5k+1}$ wäre so eingebettet:
  - Ordne jedes b_i dem dichtesten a_j zu
  - Einem $a^\star$ sind ≥ 6 b_i zugeordnet und
  - ein Winkel zwischen zwei b_i ist $\leq 60°$
  - nenne diese b^- (dichter) und b^+ (weiter weg)
  - b^+ ist dichter an b^- als an $a^\star$ (und damit an allen a_j)
  - dichtester Knoten an b^+ ist ein b_i
  - Widerspruch!

$\Rightarrow$ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*



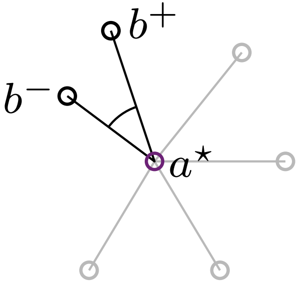
Wann gibt es keine Greedy-Einbettungen?

Satz

Die Graphen $K_{k,5k+1}$ haben keine Greedy-Einbettungen in der Ebene.

- Angenommen, ein $K_{k,5k+1}$ wäre so eingebettet:
  - Ordne jedes b_i dem dichtesten a_j zu
  - Einem $a^\star$ sind ≥ 6 b_i zugeordnet und
  - ein Winkel zwischen zwei b_i ist $\leq 60°$
  - nenne diese b^- (dichter) und b^+ (weiter weg)
  - b^+ ist dichter an b^- als an $a^\star$ (und damit an allen a_j)
  - dichtester Knoten an b^+ ist ein b_i
  - Widerspruch!

$\Rightarrow$ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*



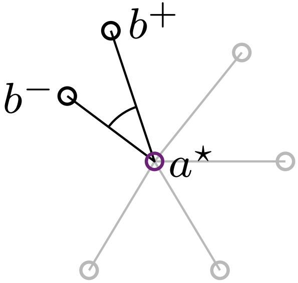
Wann gibt es keine Greedy-Einbettungen?

Satz

Die Graphen $K_{k,5k+1}$ haben keine Greedy-Einbettungen in der Ebene.

- Angenommen, ein $K_{k,5k+1}$ wäre so eingebettet:
  - Ordne jedes b_i dem dichtesten a_j zu
  - Einem $a^\star$ sind ≥ 6 b_i zugeordnet und
  - ein Winkel zwischen zwei b_i ist $\leq 60°$
  - nenne diese b^- (dichter) und b^+ (weiter weg)
  - b^+ ist dichter an b^- als an $a^\star$ (und damit an allen a_j)
  - dichtester Knoten an b^+ ist ein b_i
  - Widerspruch!

$\Rightarrow$ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*



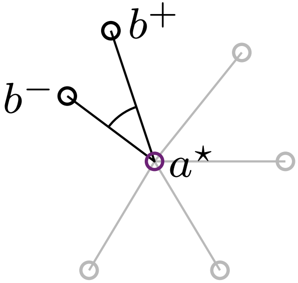
Wann gibt es keine Greedy-Einbettungen?

Satz

Die Graphen $K_{k,5k+1}$ haben keine Greedy-Einbettungen in der Ebene.

- Angenommen, ein $K_{k,5k+1}$ wäre so eingebettet:
  - Ordne jedes b_i dem dichtesten a_j zu
  - Einem $a^\star$ sind ≥ 6 b_i zugeordnet und
  - ein Winkel zwischen zwei b_i ist $\leq 60^\circ$
  - nenne diese b^- (dichter) und b^+ (weiter weg)
  - b^+ ist dichter an b^- als an $a^\star$ (und damit an allen a_j)
  - dichtester Knoten an b^+ ist ein b_i
  - Widerspruch!

$\Rightarrow$ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*



Dreifach zusammenhängende planare Graphen

Vermutung [Papadimitriou, Ratajczak, 2004]

Enthält ein Graph einen dreifach zusammenhängenden planaren Subgraphen, lässt er sich greedy in die Ebene einbetten.

- kein Widerspruch zur Vermutung:
  - $K_{1,6}$ und $K_{2,11}$ sind planar aber nicht dreifach zusammenhängend
  - $K_{3,16}, \ldots$ sind nicht mehr planar.

Inzwischen gelöst:

Satz [Leighton, Moitra, 2008]

Jeder 3-fach zusammenhängende Graph der keinen $K_{3,3}$ als Minor enthält ermöglicht eine Greedy-Einbettung in der euklidischen Ebene.

Dreifach zusammenhängende planare Graphen

Vermutung [Papadimitriou, Ratajczak, 2004]

Enthält ein Graph einen dreifach zusammenhängenden planaren Subgraphen, lässt er sich greedy in die Ebene einbetten.

- kein Widerspruch zur Vermutung:
  - $K_{1,6}$ und $K_{2,11}$ sind planar aber nicht dreifach zusammenhängend
  - $K_{3,16}, \ldots$ sind nicht mehr planar.

Inzwischen gelöst:

Satz [Leighton, Moitra, 2008]

Jeder 3-fach zusammenhängende Graph der keinen $K_{3,3}$ als Minor enthält ermöglicht eine Greedy-Einbettung in der euklidischen Ebene.

Dreifach zusammenhängende planare Graphen

Karlsruher Institut für Technologie

Vermutung [Papadimitriou, Ratajczak, 2004]

Enthält ein Graph einen dreifach zusammenhängenden planaren Subgraphen, lässt er sich greedy in die Ebene einbetten.

- kein Widerspruch zur Vermutung:
  - $K_{1,6}$ und $K_{2,11}$ sind planar aber nicht dreifach zusammenhängend
  - $K_{3,16}, \ldots$ sind nicht mehr planar.

Inzwischen gelöst:

Satz [Leighton, Moitra, 2008]

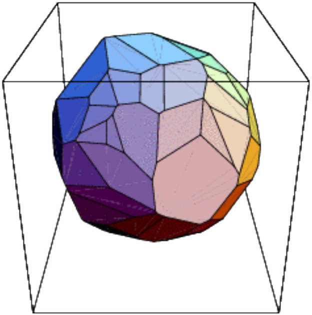
Jeder 3-fach zusammenhängende Graph der keinen $K_{3,3}$ als Minor enthält ermöglicht eine Greedy-Einbettung in der euklidischen Ebene.

Exot I: Einbettung in $\mathbb{R}^3$

Satz

Jeder dreifach zusammenhängender planare Graph lässt sich so in den $\mathbb{R}^3$ einbetten, dass Greedy Routing mit einer geeigneten Distanzfunktion immer erfolgreich ist.

- Beweis:
  - Jeder dreifach zshg. planare Graph ist Kantengraph eines konvexen Polytops
  - sogar so, dass alle Kanten eine gemeinsamen Kugel (um 0) berühren
  - setze $d(t, v) = -\overrightarrow{t} \cdot \overrightarrow{v}$
- keine Metrik, schwer zu verteilen

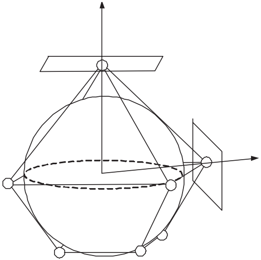


Exot I: Einbettung in $\mathbb{R}^3$

Satz

Jeder dreifach zusammenhängender planare Graph lässt sich so in den $\mathbb{R}^3$ einbetten, dass Greedy Routing mit einer geeigneten Distanzfunktion immer erfolgreich ist.

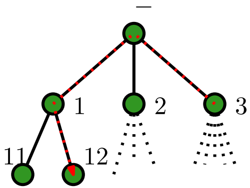
- Beweis:
  - Jeder dreifach zshg. planare Graph ist Kantengraph eines konvexen Polytops
  - sogar so, dass alle Kanten eine gemeinsamen Kugel (um 0) berühren
  - setze $d(t, v) = -\overrightarrow{t} \cdot \overrightarrow{v}$
- keine Metrik, schwer zu verteilen



Tree-Routing

- Vorberechnung
  - Berechne aufspannenden Baum
  - Wähle Wurzel *w* beliebig
  - Identifiziere jeden Knoten mit Abstiegen von *w*
- Routing
  - ist *t* Nachfolger, steige in entsprechenden Teilbaum ab
  - sonst steige auf

- Leicht zu verteilen
- Schlechter Worst-Case-Stretch: $\Theta(n)$
- Greedy-Routing im $\mathbb{R}^2$ möglich?

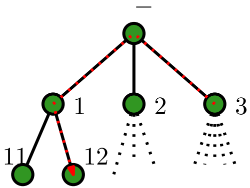


Exot II: Routing in der Hyperbolischen Ebene

Tree-Routing

- Vorberechnung
  - Berechne aufspannenden Baum
  - Wähle Wurzel w beliebig
  - Identifiziere jeden Knoten mit Abstiegen von w
- Routing
  - ist t Nachfolger, steige in entsprechenden Teilbaum ab
  - sonst steige auf

- Leicht zu verteilen
- Schlechter Worst-Case-Stretch: $\Theta(n)$
- Greedy-Routing im $\mathbb{R}^2$ möglich?
  Nein! ($K_{1,6}$)



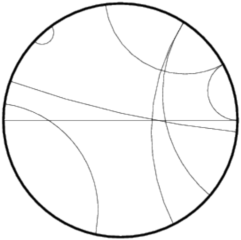
Tree-Routing in der Hyperbolischen Ebene

Poincaré-Modell der Hyperbolischen Ebene

$$H = \{x \in \mathbb{R}^2 : |x| < 1\}$$

$$d(u,v) = \operatorname{arccosh}\left(1 + \frac{2\|u-v\|}{(1-\|u\|)(1-\|v\|)}\right)$$

- Punkte im Einheitskreis
- Geraden: Kreise, die den Rand orthogonal schneiden
- parallele Geraden: besitzen keinen gemeinsamen Punkt



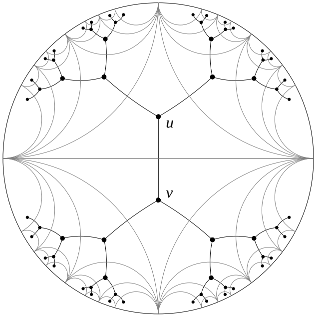
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Tree-Routing in der Hyperbolischen Ebene

Satz (nur zum Staunen!)

d-reguläre Bäume haben *Greedy-Einbettungen* in der Hyperbolischen Ebene.

- Damit haben alle zshg. Graphen *Greedy-Einbettungen*:
  - Jeder Baum ist Teil eines *d*-regulären Baumes
  - Jeder Graph ist ein Baum mit zusätzlichen Kanten



Überblick

- Greedy-Einbettungen
- Pseudo-Geometrisches Routing
- Compact Routing

Pseudo-geometrisches Routing

Die einfachsten Koordinaten, mit denen wir einen Knoten identifizieren können, sind Hop-Abstände zu ausgewählten Knoten!

- Wähle Ankerknoten $a_1, a_2, \ldots, a_k$,
- Jeder Ankerknoten flutet das Netz zur Abstandsmessung
- Knoten u hat Koordinaten $(d_G(u, a_1), \ldots d_G(u, a_k)) \in \mathbb{Z}^k$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Gute Ankerknoten

Eindeutige Koordinaten

Ankerknoten müssen garantieren, dass jeder Knoten *eindeutige* Koordinaten $(d_G(u, a_1), \ldots d_G(u, a_k))$ zugewiesen bekommt.

Routing

Ankerknoten sollen eine lokale Routing-Regel ermöglichen.

- Jeder Knoten entscheidet auf Basis seiner Koordinaten und der Koordinaten seiner Nachbarn (wie beim Georouting).

Gute Ankerknoten

Karlsruher Institut für Technologie

Eindeutige Koordinaten

Ankerknoten müssen garantieren, dass jeder Knoten *eindeutige* Koordinaten $(d_G(u, a_1), \dots d_G(u, a_k))$ zugewiesen bekommt.

Routing

Ankerknoten sollen eine lokale Routing-Regel ermöglichen.

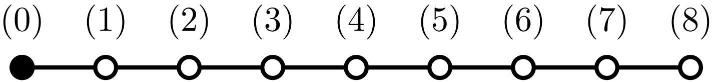
- Jeder Knoten entscheidet auf Basis seiner Koordinaten und der Koordinaten seiner Nachbarn (wie beim Georouting).

Warm-up: Pfade

- Bei gezielter Wahl
- Bei zufälliger Wahl
    - aus lokaler Information lassen sich Positionen der Anker ableiten
    - dann ist auch klar, in welche Richtung geroutet werden muss

Warm-up: Pfade

- Bei gezielter Wahl reicht 1 Ankerknoten für eindeutige Namen
  - Wähle Knoten am Ende des Pfades!
  - Routing dann klar
- Bei zufälliger Wahl
  - aus lokaler Information lassen sich Positionen der Anker ableiten
  - dann ist auch klar, in welche Richtung geroutet werden muss



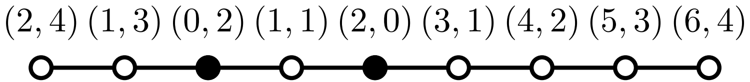
Warm-up: Pfade

- Bei gezielter Wahl reicht 1 Ankerknoten für eindeutige Namen
  - Wähle Knoten am Ende des Pfades!
  - Routing dann klar
- Bei zufälliger Wahl reichen 2 Ankerknoten immer
  - Wenn zwei Knoten u, v dieselbe Koordinate $d_G(u, a_k) = d_G(v, a_k)$ haben, liegt a_k genau in der Mitte (und da liegt maximal ein a_k).
  - lokales Routing auf kürzestem Weg möglich
    - aus lokaler Information lassen sich Positionen der Anker ableiten
    - dann ist auch klar, in welche Richtung geroutet werden muss

$$(2,4)\ (1,3)\ (0,2)\ (1,1)\ (2,0)\ (3,1)\ (4,2)\ (5,3)\ (6,4)$$

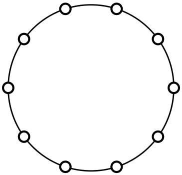
Warm-up: Pfade

- Bei gezielter Wahl reicht 1 Ankerknoten für eindeutige Namen
  - Wähle Knoten am Ende des Pfades!
  - Routing dann klar
- Bei zufälliger Wahl reichen 2 Ankerknoten immer
  - Wenn zwei Knoten u, v dieselbe Koordinate $d_G(u, a_k) = d_G(v, a_k)$ haben, liegt a_k genau in der Mitte (und da liegt maximal ein a_k).
  - lokales Routing auf kürzestem Weg möglich
    - aus lokaler Information lassen sich Positionen der Anker ableiten
    - dann ist auch klar, in welche Richtung geroutet werden muss



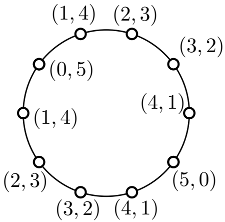
Zum Üben: Ringe

- Es reichen bei gezielter Wahl ?, bei zufälliger Wahl ? Knoten
  - (Fast) dasselbe Argument: Haben zwei Knoten dieselben Koordinaten, dann liegt der entsprechende Anker gleich weit von beiden entfernt
    - so ein Problem entsteht bei einem Knoten immer
    - bei zwei Knoten nur dann, wenn die Anker sich gegenüberliegen!
  - Lokales Routing wieder auf kürzestem Weg möglich!
    - (Alles andere wäre auch traurig!)



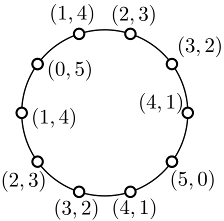
Zum Üben: Ringe

- Es reichen bei gezielter Wahl 2, bei zufälliger Wahl 3 Knoten
  - (Fast) dasselbe Argument: Haben zwei Knoten dieselben Koordinaten, dann liegt der entsprechende Anker gleich weit von beiden entfernt
    - so ein Problem entsteht bei einem Knoten immer
    - bei zwei Knoten nur dann, wenn die Anker sich gegenüberliegen!
  - Lokales Routing wieder auf kürzestem Weg möglich!
    - (Alles andere wäre auch traurig!)



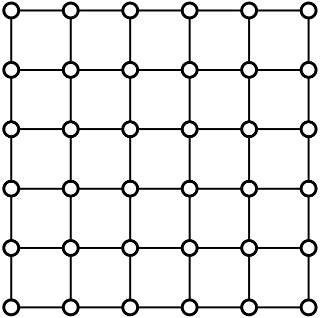
Zum Üben: Ringe

- Es reichen bei gezielter Wahl 2, bei zufälliger Wahl 3 Knoten
  - (Fast) dasselbe Argument: Haben zwei Knoten dieselben Koordinaten, dann liegt der entsprechende Anker gleich weit von beiden entfernt
    - so ein Problem entsteht bei einem Knoten immer
    - bei zwei Knoten nur dann, wenn die Anker sich gegenüberliegen!
  - Lokales Routing wieder auf kürzestem Weg möglich!
    - (Alles andere wäre auch traurig!)



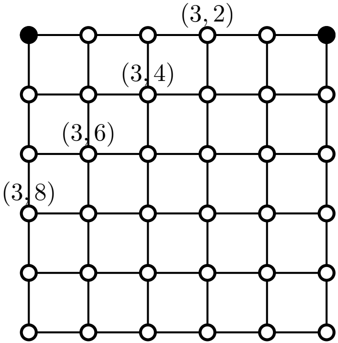
Quizfrage: Eindeutige Namen im Gitter

- Bei gezielter Wahl reichen
  - Knoten in den beiden oberen Ecken benennen eindeutig
  - lokales Routing wieder einfach!
- Erst $\Theta(n)$ zufällige Knoten lösen das Problem immer!
  - Knoten u, v haben identische Abstände zu allen Ankern!
  - (Das Routingproblem klammern wir mal aus)



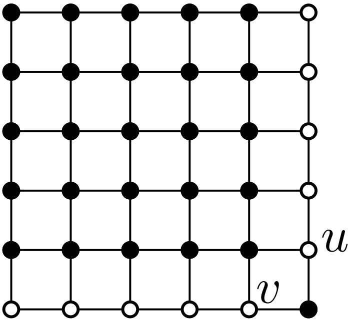
Quizfrage: Eindeutige Namen im Gitter

- Bei gezielter Wahl reichen 2 Anker
  - Knoten in den beiden oberen Ecken benennen eindeutig
  - lokales Routing wieder einfach!
- Erst $\Theta(n)$ zufällige Knoten lösen das Problem immer!
  - Knoten u, v haben identische Abstände zu allen Ankern!
  - (Das Routingproblem klammern wir mal aus)



Quizfrage: Eindeutige Namen im Gitter

- Bei gezielter Wahl reichen 2 Anker
  - Knoten in den beiden oberen Ecken benennen eindeutig
  - lokales Routing wieder einfach!
- Erst $\Theta(n)$ zufällige Knoten lösen das Problem immer!
  - Knoten u, v haben identische Abstände zu allen Ankern!
  - (Das Routingproblem klammern wir mal aus)



Wieviele Knoten braucht man in UDG?

Unit-Disk-Trees

Rekursiv definierte Bäume auf dem Gitter mit $\Theta(n)$ Blättern *(ohne Beweis)*.

- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!

$\Rightarrow$ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!

- bei zufälliger Wahl schlimmestenfalls alle!

Wieviele Knoten braucht man in UDG?

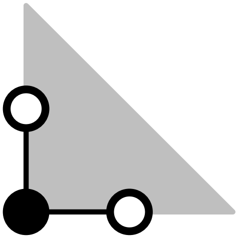
Unit-Disk-Trees

Rekursiv definierte Bäume auf dem Gitter mit $\Theta(n)$ Blättern *(ohne Beweis)*.

- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!

$\Rightarrow$ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!

- bei zufälliger Wahl schlimmstenfalls alle!
  - (dazu reicht schon ein kleiner Baum am Rand eines Netzes!)



Wieviele Knoten braucht man in UDG?

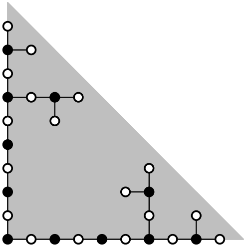
Unit-Disk-Trees

Rekursiv definierte Bäume auf dem Gitter mit $\Theta(n)$ Blättern *(ohne Beweis)*.

- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!

$\Rightarrow$ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!

- bei zufälliger Wahl schlimmstenfalls alle!
  - (dazu reicht schon ein kleiner Baum am Rand eines Netzes!)



Wieviele Knoten braucht man in UDG?

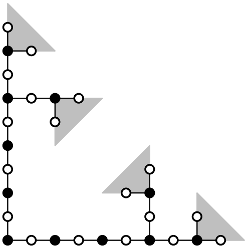
Unit-Disk-Trees

Rekursiv definierte Bäume auf dem Gitter mit $\Theta(n)$ Blättern *(ohne Beweis)*.

- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!

$\Rightarrow$ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!

- bei zufälliger Wahl schlimmestenfalls alle!
  - (dazu reicht schon ein kleiner Baum am Rand eines Netzes!)



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wieviele Knoten braucht man in UDG?

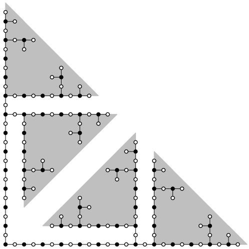
Unit-Disk-Trees

Rekursiv definierte Bäume auf dem Gitter mit $\Theta(n)$ Blättern *(ohne Beweis)*.

- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!

$\Rightarrow$ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!

- bei zufälliger Wahl schlimmestenfalls alle!
  - (dazu reicht schon ein kleiner Baum am Rand eines Netzes!)



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wieviele Knoten braucht man in UDG?

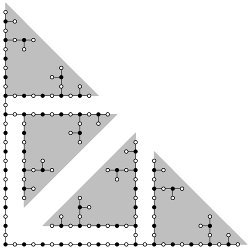
Unit-Disk-Trees

Rekursiv definierte Bäume auf dem Gitter mit $\Theta(n)$ Blättern *(ohne Beweis)*.

- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!

$\Rightarrow$ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!

- bei zufälliger Wahl schlimmestenfalls alle!
  - (dazu reicht schon ein kleiner Baum am Rand eines Netzes!)



Wieviele Knoten braucht man in UDG?

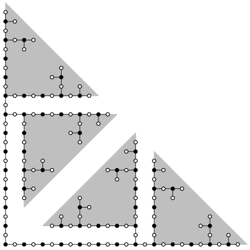
Unit-Disk-Trees

Rekursiv definierte Bäume auf dem Gitter mit $\Theta(n)$ Blättern *(ohne Beweis)*.

- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!

$\Rightarrow$ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!

- bei zufälliger Wahl schlimmestenfalls alle!
  - (dazu reicht schon ein kleiner Baum am Rand eines Netzes!)



Wieviele Knoten braucht man in UDG?

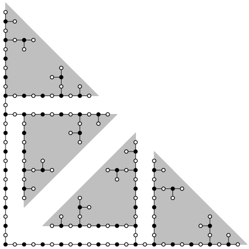
Unit-Disk-Trees

Rekursiv definierte Bäume auf dem Gitter mit $\Theta(n)$ Blättern *(ohne Beweis)*.

- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!

$\Rightarrow$ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!

- bei zufälliger Wahl schlimmestenfalls alle!
  - (dazu reicht schon ein kleiner Baum am Rand eines Netzes!)



Kurze Bilanz

- Gute Anker zu garantieren ist sehr schwer
  - bei zufälliger Wahl sind im schlechtesten Fall alle Knoten nötig, sobald die Instanz einen kleinen Baum enthält
  - UDT geben sogar Schranke bei gezielter Auswahl
- Widerspricht der Praxis!
  - Netze sind keine UDT (aber auch keine Gitter)
  - schon verhältnismäßig wenige Anker reichen meist aus, um fast alle Knoten eindeutig zu benennen!

Kurze Bilanz

- Gute Anker zu garantieren ist sehr schwer
  - bei zufälliger Wahl sind im schlechtesten Fall alle Knoten nötig, sobald die Instanz einen kleinen Baum enthält
  - UDT geben sogar Schranke bei gezielter Auswahl
- Widerspricht der Praxis!
  - Netze sind keine UDT (aber auch keine Gitter)
  - schon verhältnismäßig wenige Anker reichen meist aus, um fast alle Knoten eindeutig zu benennen!

Beacon Vector Routing

Beacon Vector Routing

1. r Anker a_i fluten das Netz (zufällig oder heuristisch gewählt)
   - Knoten *kennen* kompletten Abstandsvektor $(d_G(u, a_1), \ldots d_G(u, a_k))$ von sich und Nachbarn
   - Adressen bestehen aus *eindeutiger ID* des Knotens und Abständen zu den $k < r$ dichtesten Ankern (und deren IDs)
2. Knoten wählen falls möglich den Nachbarn, der metrische Abstandsfunktion $d : \mathbb{Z}^k \to \mathbb{R}^+$ verbessert
3. wenn keine Verbesserung möglich, route in Richtung des dem Ziel dichtesten Ankers
   - Paket enthält bisher minimalen Abstand δ^-
   - wird der unterschritten, wird wieder greedy geroutet
4. wenn das Paket den Anker erreicht, wird lokal geflutet
   - Radius ist bekannt!

Beacon Vector Routing

Beacon Vector Routing

1. r Anker a_i fluten das Netz (zufällig oder heuristisch gewählt)
   - Knoten *kennen* kompletten Abstandsvektor $(d_G(u, a_1), \dots d_G(u, a_k))$ von sich und Nachbarn
   - Adressen bestehen aus *eindeutiger ID* des Knotens und Abständen zu den $k < r$ dichtesten Ankern (und deren IDs)
2. Knoten wählen falls möglich den Nachbarn, der metrische Abstandsfunktion $d : \mathbb{Z}^k \to \mathbb{R}^+$ verbessert
3. wenn keine Verbesserung möglich, route in Richtung des dem Ziel dichtesten Ankers
   - Paket enthält bisher minimalen Abstand δ^-
   - wird der unterschritten, wird wieder greedy geroutet
4. wenn das Paket den Anker erreicht, wird lokal geflutet
   - Radius ist bekannt!

Beacon Vector Routing

Beacon Vector Routing

1. r Anker a_i fluten das Netz (zufällig oder heuristisch gewählt)
   - Knoten *kennen* kompletten Abstandsvektor $(d_G(u, a_1), \ldots d_G(u, a_k))$ von sich und Nachbarn
   - Adressen bestehen aus *eindeutiger ID* des Knotens und Abständen zu den $k < r$ dichtesten Ankern (und deren IDs)
2. Knoten wählen falls möglich den Nachbarn, der metrische Abstandsfunktion $d : \mathbb{Z}^k \to \mathbb{R}^+$ verbessert
3. wenn keine Verbesserung möglich, route in Richtung des dem Ziel dichtesten Ankers
   - Paket enthält bisher minimalen Abstand δ^-
   - wird der unterschritten, wird wieder greedy geroutet
4. wenn das Paket den Anker erreicht, wird lokal geflutet
   - Radius ist bekannt!

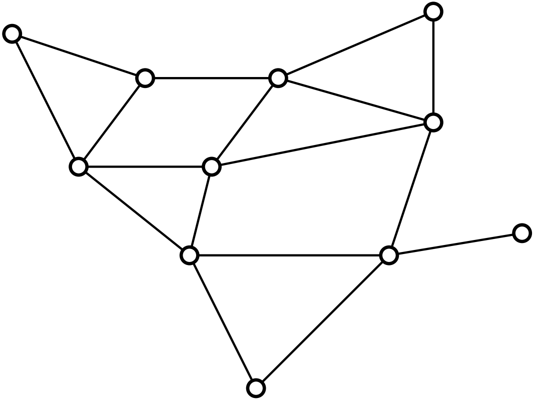
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beacon Vector Routing

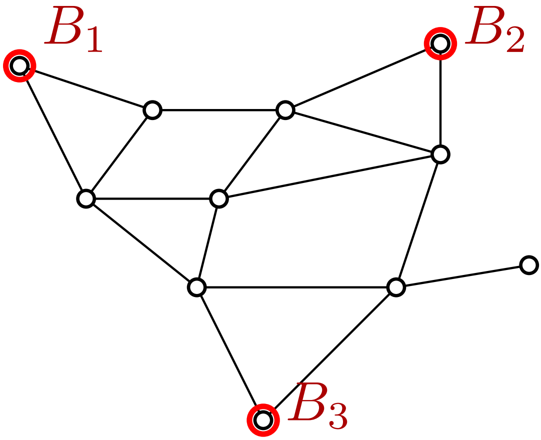
Beacon Vector Routing

1. r Anker a_i fluten das Netz (zufällig oder heuristisch gewählt)
   - Knoten *kennen* kompletten Abstandsvektor $(d_G(u, a_1), \ldots d_G(u, a_k))$ von sich und Nachbarn
   - Adressen bestehen aus *eindeutiger ID* des Knotens und Abständen zu den $k < r$ dichtesten Ankern (und deren IDs)
2. Knoten wählen falls möglich den Nachbarn, der metrische Abstandsfunktion $d : \mathbb{Z}^k \to \mathbb{R}^+$ verbessert
3. wenn keine Verbesserung möglich, route in Richtung des dem Ziel dichtesten Ankers
   - Paket enthält bisher minimalen Abstand δ^-
   - wird der unterschritten, wird wieder greedy geroutet
4. wenn das Paket den Anker erreicht, wird lokal geflutet
   - Radius ist bekannt!

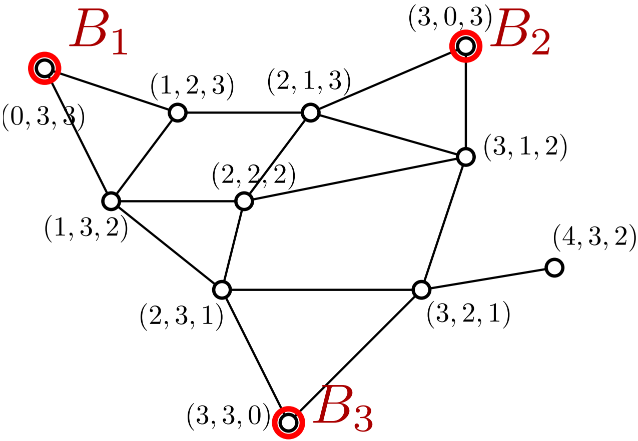
Ein einfaches Beispiel (L_1-Metrik)



Ein einfaches Beispiel (L_1-Metrik)

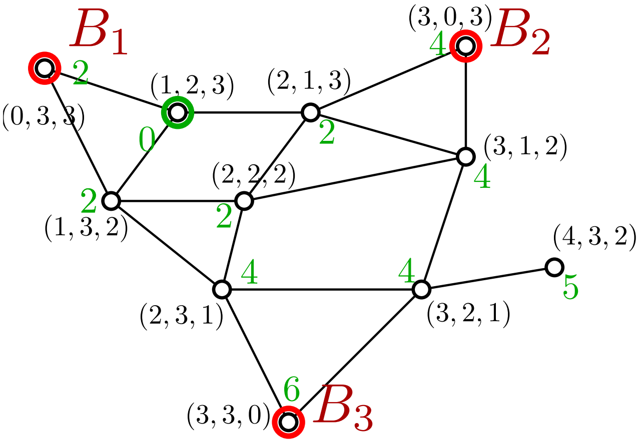


Ein einfaches Beispiel (L_1-Metrik)

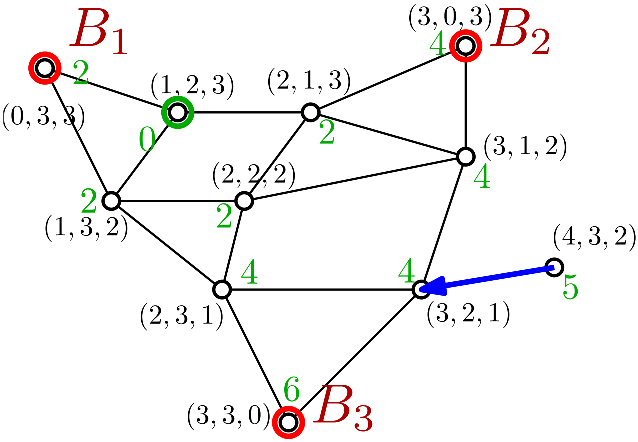


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

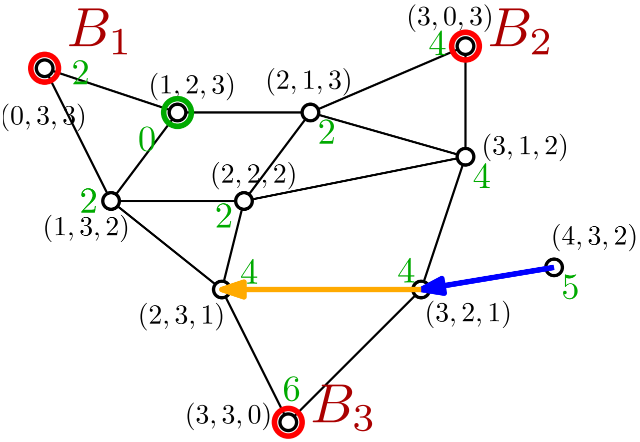
Ein einfaches Beispiel (L_1-Metrik)



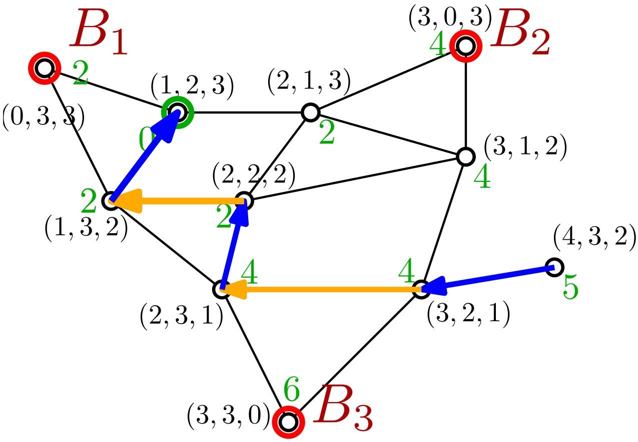
Ein einfaches Beispiel (L_1-Metrik)



Ein einfaches Beispiel (L_1-Metrik)



Ein einfaches Beispiel (L_1-Metrik)



Alternative Metriken

Beobachtung

Die Information von Landmarken, denen man näherkommen will, ist wertvoller als die Information von Landmarken, von denen man sich entfernen muss.

- Intuition: Es gibt weniger Wege zu einem Anker als von einem Anker weg

Alternative Metrik

$$d(v,t) = \sum_{i=1}^{k} C \max(v_i - t_i, 0) + \max(t_i - v_i, 0) \quad C > 1$$

$\Rightarrow$ *Macht Fallback nicht unmöglich, aber unwahrscheinlicher...*

Alternative Metriken

Beobachtung

Die Information von Landmarken, denen man näherkommen will, ist wertvoller als die Information von Landmarken, von denen man sich entfernen muss.

- Intuition: Es gibt weniger Wege zu einem Anker als von einem Anker weg

Alternative Metrik

$$d(v,t) = \sum_{i=1}^{k} C \max(v_i - t_i, 0) + \max(t_i - v_i, 0) \quad C > 1$$

$\Rightarrow$ *Macht Fallback nicht unmöglich, aber unwahrscheinlicher...*

Überblick

- Greedy-Einbettungen
- Pseudo-Geometrisches Routing
- Compact Routing

Compact Routing

Als *kompakt* bezeichnet man alle Routingverfahren, die garantiert mit Routingtabellen mit $o(n)$ Bits lokalem Speicher auskommen.

- Knoten entscheiden auf Basis von Zieladresse und Information im Speicher, an welchen Nachbarn sie ein Paket schicken
- Unterscheidung: labeled vs. name-independent
  - labeled: Knoten können IDs wählen
  - name-independent: Knoten müssen mit ursprünglichen IDs adressiert werden

Tradeoff lokaler Speicher / Routing Stretch!

Compact Routing

Als *kompakt* bezeichnet man alle Routingverfahren, die garantiert mit Routingtabellen mit $o(n)$ Bits lokalem Speicher auskommen.

- Knoten entscheiden auf Basis von Zieladresse und Information im Speicher, an welchen Nachbarn sie ein Paket schicken
- Unterscheidung: labeled vs. name-independent
  - labeled: Knoten können IDs wählen
  - name-independent: Knoten müssen mit ursprünglichen IDs adressiert werden

Tradeoff lokaler Speicher / Routing Stretch!

Compact Routing

Als *kompakt* bezeichnet man alle Routingverfahren, die garantiert mit Routingtabellen mit $o(n)$ Bits lokalem Speicher auskommen.

- Knoten entscheiden auf Basis von Zieladresse und Information im Speicher, an welchen Nachbarn sie ein Paket schicken
- Unterscheidung: labeled vs. name-independent
  - labeled: Knoten können IDs wählen
  - name-independent: Knoten müssen mit ursprünglichen IDs adressiert werden

Tradeoff lokaler Speicher / Routing Stretch!

Compact Routing

Als *kompakt* bezeichnet man alle Routingverfahren, die garantiert mit Routingtabellen mit $o(n)$ Bits lokalem Speicher auskommen.

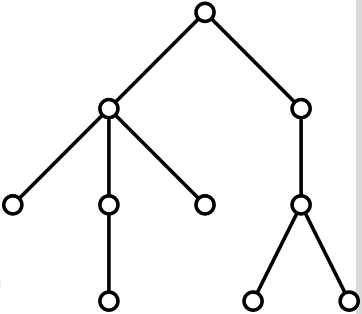
- Knoten entscheiden auf Basis von Zieladresse und Information im Speicher, an welchen Nachbarn sie ein Paket schicken
- Unterscheidung: labeled vs. name-independent
  - labeled: Knoten können IDs wählen
  - name-independent: Knoten müssen mit ursprünglichen IDs adressiert werden

Tradeoff lokaler Speicher / Routing Stretch!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Einfaches Beispiel: Intervall Routing

- beliebige Wurzel spannt Baum auf
- Knoten werden in Pre-order numeriert
- jeder Knoten merkt sich, welche IDs in welchem Teilbaum liegen
- alle anderen IDs werden Richtung Wurzel geroutet
- $O(deg(v) \log n)$ Bits reichen aus!
- aber: Wege können sich um Faktor $\Theta(n)$ verlängern! (Beispiel?)

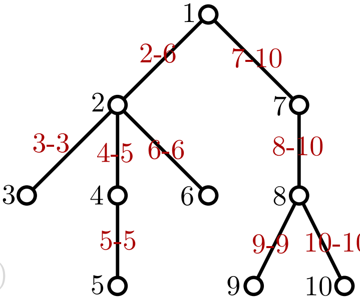


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist mit $o(n)$ Bits kein Stretch < 3 für alle Knotenpaare möglich!

Einfaches Beispiel: Intervall Routing

- beliebige Wurzel spannt Baum auf
- Knoten werden in Pre-order numeriert
- jeder Knoten merkt sich, welche IDs in welchem Teilbaum liegen
- alle anderen IDs werden Richtung Wurzel geroutet
- $O(deg(v) \log n)$ Bits reichen aus!
- aber: Wege können sich um Faktor $\Theta(n)$ verlängern! (Beispiel?)

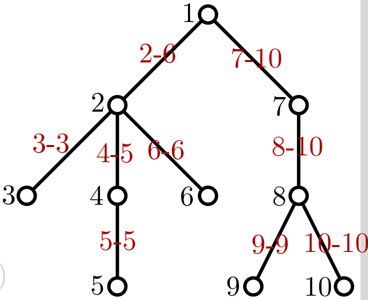


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist mit $o(n)$ Bits kein Stretch < 3 für alle Knotenpaare möglich!

Einfaches Beispiel: Intervall Routing

- beliebige Wurzel spannt Baum auf
- Knoten werden in Pre-order numeriert
- jeder Knoten merkt sich, welche IDs in welchem Teilbaum liegen
- alle anderen IDs werden Richtung Wurzel geroutet
- $O(deg(v) \log n)$ Bits reichen aus!
- aber: Wege können sich um Faktor $\Theta(n)$ verlängern! (Beispiel?)

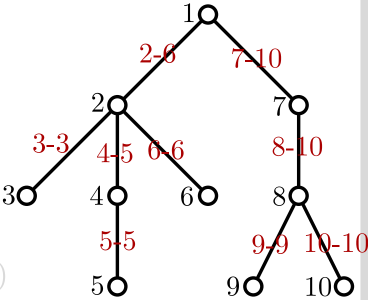


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist mit $o(n)$ Bits kein Stretch < 3 für alle Knotenpaare möglich!

Einfaches Beispiel: Intervall Routing

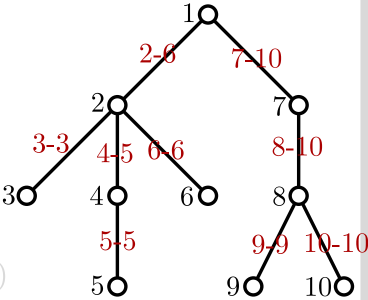
- beliebige Wurzel spannt Baum auf
- Knoten werden in Pre-order numeriert
- jeder Knoten merkt sich, welche IDs in welchem Teilbaum liegen
- alle anderen IDs werden Richtung Wurzel geroutet
- $O(deg(v) \log n)$ Bits reichen aus!
- aber: Wege können sich um Faktor $\Theta(n)$ verlängern! (Beispiel?)



Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist mit $o(n)$ Bits kein Stretch < 3 für alle Knotenpaare möglich!

- beliebige Wurzel spannt Baum auf
- Knoten werden in Pre-order numeriert
- jeder Knoten merkt sich, welche IDs in welchem Teilbaum liegen
- alle anderen IDs werden Richtung Wurzel geroutet
- $O(deg(v) \log n)$ Bits reichen aus!
- aber: Wege können sich um Faktor $\Theta(n)$ verlängern! (Beispiel?)

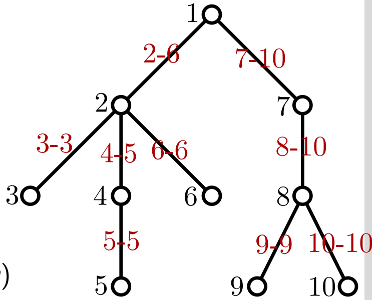


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist mit $o(n)$ Bits kein Stretch < 3 für alle Knotenpaare möglich!

Einfaches Beispiel: Intervall Routing

- beliebige Wurzel spannt Baum auf
- Knoten werden in Pre-order numeriert
- jeder Knoten merkt sich, welche IDs in welchem Teilbaum liegen
- alle anderen IDs werden Richtung Wurzel geroutet
- $O(deg(v) \log n)$ Bits reichen aus!
- aber: Wege können sich um Faktor $\Theta(n)$ verlängern! (Beispiel?)

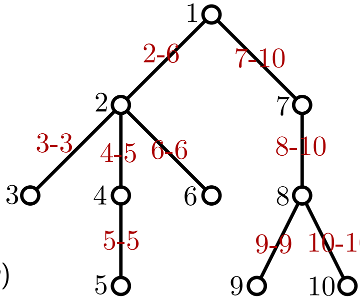


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist mit $o(n)$ Bits kein Stretch < 3 für alle Knotenpaare möglich!

Einfaches Beispiel: Intervall Routing

- beliebige Wurzel spannt Baum auf
- Knoten werden in Pre-order numeriert
- jeder Knoten merkt sich, welche IDs in welchem Teilbaum liegen
- alle anderen IDs werden Richtung Wurzel geroutet
- $O(deg(v) \log n)$ Bits reichen aus!
- aber: Wege können sich um Faktor $\Theta(n)$ verlängern! (Beispiel?)



Klassisches Resultat (ohne Beweis)

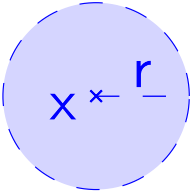
In allgemeinen Graphen ist mit $o(n)$ Bits kein Stretch < 3 für alle Knotenpaare möglich!

Kreise, Kugeln und Bälle

Definition

In einem metrischen Raum ist ein *Ball* $B(x, r)$ um ein Element x die Menge aller Elemente mit $d(x, y) \leq r$.

- $\mathbb{R}^2$, $\mathbb{R}^3$ mit euklidischen Abständen



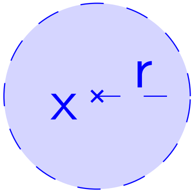
$\Rightarrow$ Kreise, Kugeln

Kreise, Kugeln und Bälle

Definition

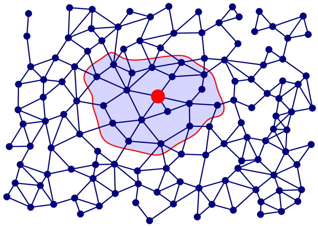
In einem metrischen Raum ist ein *Ball* $B(x, r)$ um ein Element x die Menge aller Elemente mit $d(x, y) \leq r$.

- $\mathbb{R}^2$, $\mathbb{R}^3$ mit euklidischen Abständen



$\Rightarrow$ Kreise, Kugeln

- Graph G mit hop-Abständen $d_G(u, v)$

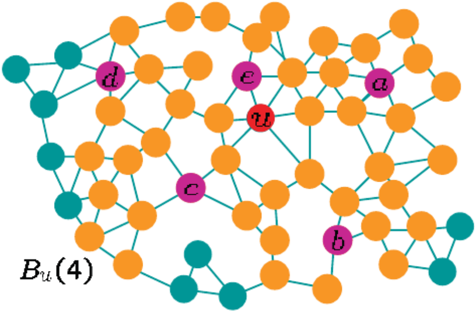


$\Rightarrow$ r-hop-Nachbarschaften

Doubling Dimension

Definition

Ein Graph hat „Doubling Dimension" α, wenn sich die k-hop-Nachbarschaft jedes Knotens mit maximal 2^{α} $k/2$-hop-Nachbarschaften von anderen Knoten abdecken lässt.



Ab jetzt gehen wir von einer konstant beschränkten DD aus!

Knobelaufgabe

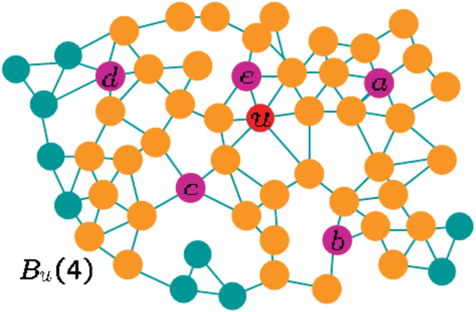
Gibt es UDG, die keine konstant beschränkte DD haben?

Bild: Roger Wattenhofer

Doubling Dimension

Definition

Ein Graph hat „Doubling Dimension“ α, wenn sich die k-hop-Nachbarschaft jedes Knotens mit maximal 2^α $k/2$-hop-Nachbarschaften von anderen Knoten abdecken lässt.



Ab jetzt gehen wir von einer konstant beschränkten DD aus!

Knobelaufgabe

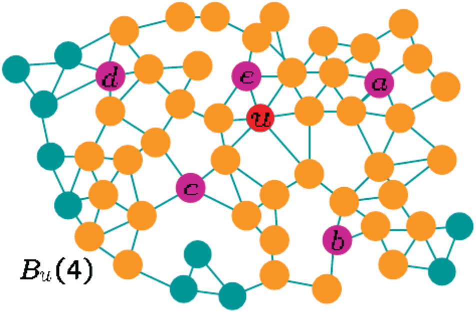
Gibt es UDG, die keine konstant beschränkte DD haben?

Bild: Roger Wattenhofer

Doubling Dimension

Definition

Ein Graph hat „Doubling Dimension“ α, wenn sich die k-hop-Nachbarschaft jedes Knotens mit maximal 2^α $k/2$-hop-Nachbarschaften von anderen Knoten abdecken lässt.



Ab jetzt gehen wir von einer konstant beschränkten DD aus!

Knobelaufgabe

Gibt es UDG, die keine konstant beschränkte DD haben?

Bild: Roger Wattenhofer

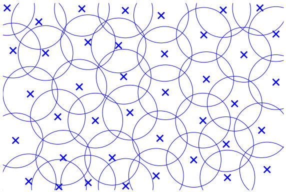
ρ-Netze

Definition

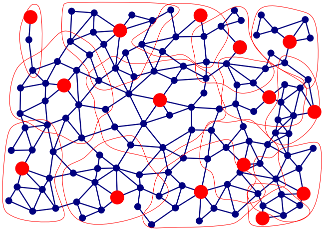
Ein ρ-Netz in einem metrischen Raum ist eine Teilmenge $U \subset V$ von *Zentren* mit

- für jedes $v \in V$ gibt es ein $u \in U$ mit $v \in B(u, \rho)$
- für alle $u, u' \in U$ gilt: $u' \notin B(u, \rho)$

- euklid. $\mathbb{R}^2$, $\mathbb{R}^3$



- Graphen



- Zentral leicht berechenbar: Entferne sukzessive Knoten u mitsamt ρ-Nachbarschaft aus G.

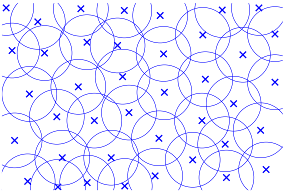
ρ-Netze

Definition

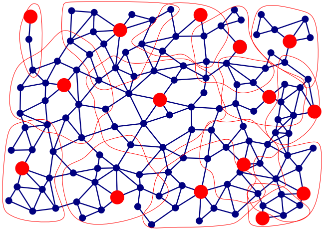
Ein ρ-Netz in einem metrischen Raum ist eine Teilmenge $U \subset V$ von *Zentren* mit

- für jedes $v \in V$ gibt es ein $u \in U$ mit $v \in B(u, \rho)$
- für alle $u, u' \in U$ gilt: $u' \notin B(u, \rho)$

- euklid. $\mathbb{R}^2$, $\mathbb{R}^3$



- Graphen

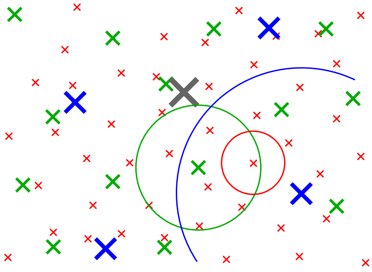


- Zentral leicht berechenbar: Entferne sukzessive Knoten u mitsamt ρ-Nachbarschaft aus G.

Anwendung für ρ-Netze: Routing

Angenommen, ich habe ρ-Netze für $\rho = 1, 2, 4, \ldots$, nämlich $U_1, U_2, U_4, \ldots$ und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, \rho)$, wie hilft mir das?

- Jedes $u' \in U_\rho$ kennt dichtestes $u \in U_{2\rho}$ ($u' \in B(u, 2\rho)$!)
  $\Rightarrow$ u' wird Kind von u!
- Jedes Zentrum numeriert Kinder
  $\Rightarrow$ Eindeutige Adressen aller Zentren!
  $\Rightarrow$ ID aus $O(\log \Delta \cdot \log D)$ Bits

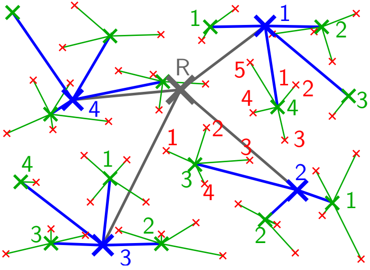


Anwendung für ρ-Netze: Routing

Angenommen, ich habe ρ-Netze für $\rho = 1, 2, 4, \ldots$, nämlich $U_1, U_2, U_4, \ldots$ und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, \rho)$, wie hilft mir das?

- Jedes $u' \in U_\rho$ kennt dichtestes $u \in U_{2\rho}$ ($u' \in B(u, 2\rho)$!)
  $\Rightarrow$ u' wird Kind von u!

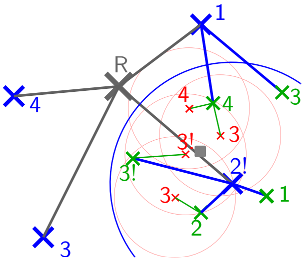
- Jedes Zentrum numeriert Kinder
  $\Rightarrow$ Eindeutige Adressen aller Zentren!
  $\Rightarrow$ ID aus $O(\log \Delta \cdot \log D)$ Bits



Anwendung für ρ-Netze: Routing

Angenommen, ich habe ρ-Netze für $\rho = 1, 2, 4, \ldots$, nämlich $U_1, U_2, U_4, \ldots$, eindeutige Adressen für jedes Zentrum und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, 2\rho)$, wie hilft mir das?

- Jeder Knoten bezeichnet sich durch den Baum von Zentren, in deren *verdoppeltem Radius* er sich befindet
  - Namensgebende Zentren werden markiert



Anwendung für ρ-Netze: Routing

ρ-Netze für $\rho = 1, 2, 4, 8, 2^{\lceil \log D \rceil}$

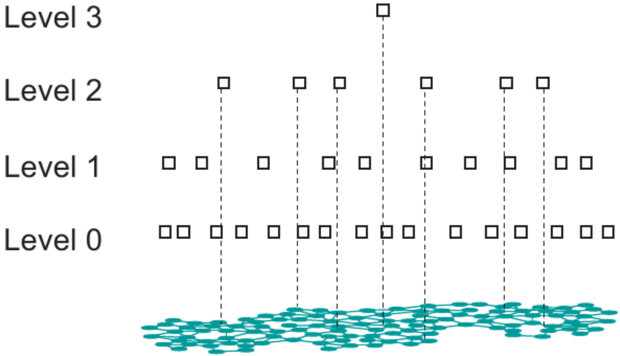


Bild: Roger Wattenhofer

Adressen und Baumstruktur

- Jeder Knoten aus einem ρ-Netz flutet seine 2ρ-Nachbarschaft
- Knoten merken sich alle ankommenden Nachrichten
  - Dank Doubling Dimension α sind das nur $2^{2\alpha}$ (also konstant viele) pro Ebene (nicht ganz trivial)

  $\Rightarrow$ Routingtabelle mit $O(\log deg(v) \cdot \log D)$ reicht aus, um alle bekannten Zentren zu erreichen

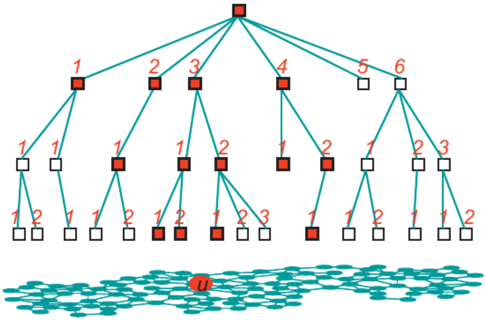


Bild: Roger Wattenhofer

Routing

Um ein Paket zu einem Knoten zu routen, wird es zum niedrigsten bekannten Zentrum geschickt, das in der Adresse des Zielknotens auftaucht.

- Bsp: u schickt ein Paket an $R:2:1:1$ direkt an $R:2:1$
- Jedes Level-i-Zentrum kennt alle adressierbaren Level-$i-1$-Zentren
- Stretch ist konstant, kann bis auf $1-\epsilon$ reduziert werden (o.B.)

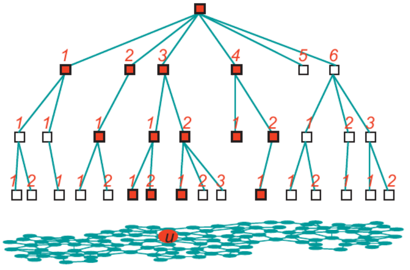


Bild: Roger Wattenhofer

Zum Mitnehmen

- Greedy-Einbettungen
  - Wann kann man einen Graphen so einbetten, dass klassisches Greedy-Routing immer funktioniert?
  - Vor allem theoretisch interessantes Problem
- Pseudo-Geometrisches Routing
  - Theorie: Auf Basis von Anker-Abständen zu routen ist nicht so leicht umzusetzen (viele Anker, Routing-Funktion?)
  - Praxis: Beacon-Vector-Routing sehr einfach und erfolgreich
- Compact Routing
  - klassische Untersuchung des Tradeoffs zwischen Stretch und lokalem Speicher
  - Neue Graphenmodelle lassen bessere Lösungen zu!

Literatur

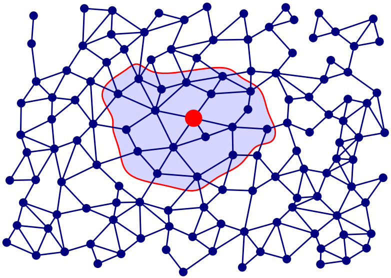
1. C. H. Papadimitriou, D. Ratajczak: *On a conjecture related to geometric routing*. In: Theor. Comput. Sci., 344(1):3–14,2005
2. M. Wattenhofer, R. Wattenhofer, P. Widmayer: *Geometric Routing without Geometry*. In: 12th Colloquium on Structural Information and Communication Complexity (SIROCCO), 2005
3. R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, I. Stoica: *Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets*. In: Proceedings of the Second USENIX Symposium on Networked Systems Design and Implementation (NSDI), 329–342, 2005
4. R. Flury, R. Wattenhofer: *Routing, Anycast, and Multicast for Mesh and Sensor Networks*. In: 26th Annual IEEE Conference on Computer Communications (INFOCOM), 2007

Doubling Dimension in UDG

Quizfrage

Hat jeder UDG konstant beschränkte Doubling Dimension?

- Nein, alle Knoten liegen in Entfernung $2k$ vom blauen Knoten, aber rote Knoten haben disjunkte k-Nachbarschaften!

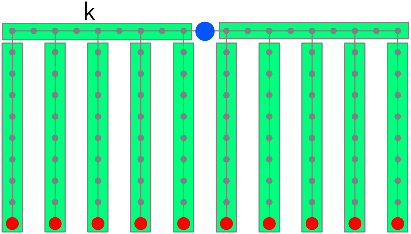


Doubling Dimension in UDG

Quizfrage

Hat jeder UDG konstant beschränkte Doubling Dimension?

- Nein, alle Knoten liegen in Entfernung $2k$ vom blauen Knoten, aber rote Knoten haben disjunkte k-Nachbarschaften!



$$\sum = (k+3) \cdot k + 1 \in \Theta(k^2)$$