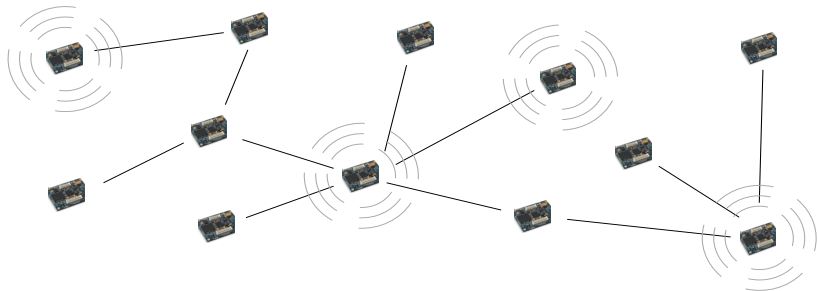


Algorithmen für Ad-hoc- und Sensornetze

VL 06 – Routing

Roman Prutkin | 12. Nov. 2015 (Version 1)

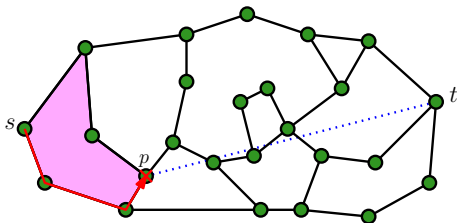
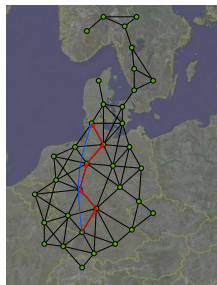
INSTITUT FÜR THEORETISCHE INFORMATIK - LEHRSTUHL FÜR ALGORITHMIK (PROF. WAGNER)



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

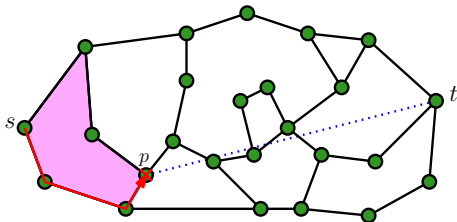
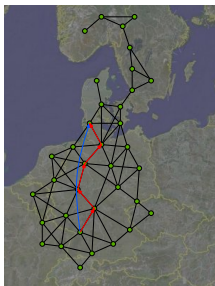
Müssen Koordinaten immer die tatsächl. Einbettung widerspiegeln?



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?



- 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 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 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 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)

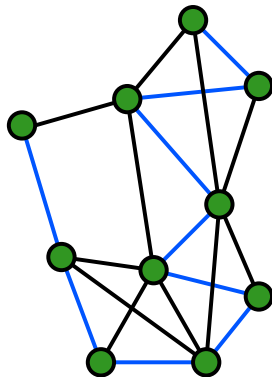
Satz

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

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

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

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



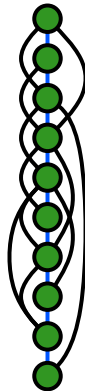
Satz

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

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

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

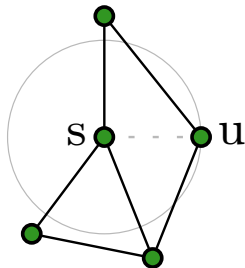
- Entspricht bereits Greedy-Einbettung in \mathbb{R}



Wann gibt es keine Greedy-Einbettungen?

Lemma

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



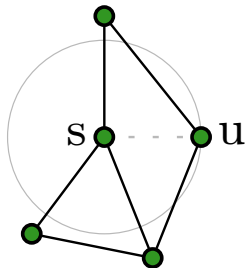
Wann gibt es keine Greedy-Einbettungen?

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.



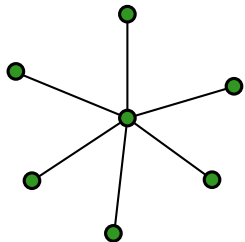
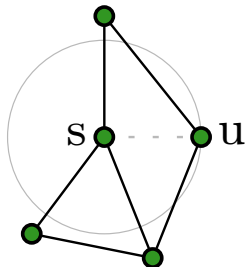
Wann gibt es keine Greedy-Einbettungen?

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.



Knobelaufgabe

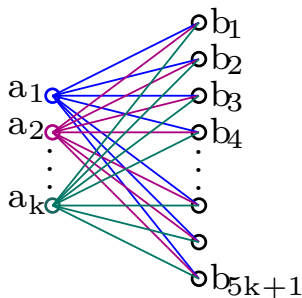
Hat $K_{1,6}$ eine Greedy-Einbettung?

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^* 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^* (und damit an allen a_j)
 - dichtester Knoten an b^- ist ein b_i
 - Widerspruch!

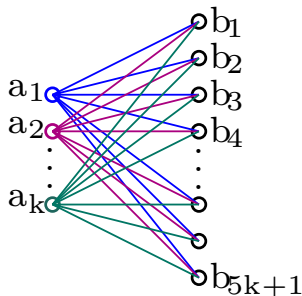


⇒ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*

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 $\leq 60^\circ$
 - 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!**



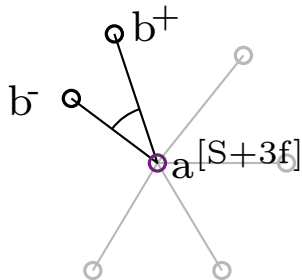
⇒ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*

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^* 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^* (und damit an allen a_j)
 - dichtester Knoten an b^+ ist ein b_i
 - **Widerspruch!**



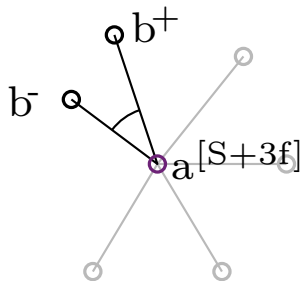
⇒ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*

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^* 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^* (und damit an allen a_j)
 - dichtester Knoten an b^+ ist ein b_i
 - **Widerspruch!**



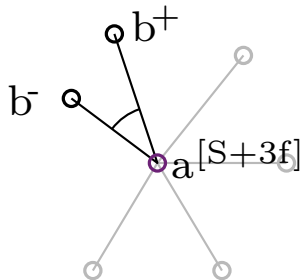
⇒ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*

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^* 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^* (und damit an allen a_j)
 - dichtester Knoten an b^+ ist ein b_i
 - **Widerspruch!**



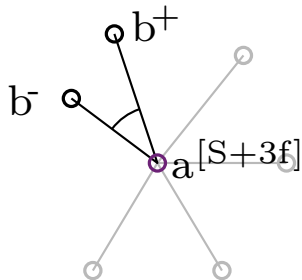
⇒ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*

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^* 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^* (und damit an allen a_j)
 - dichtester Knoten an b^+ ist ein b_i
 - Widerspruch!



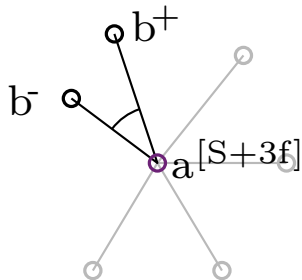
⇒ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*

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^* 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^* (und damit an allen a_j)
 - dichtester Knoten an b^+ ist ein b_i
 - **Widerspruch!**



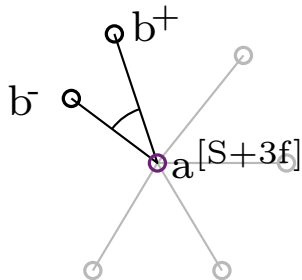
⇒ *Ein hoher Durchschnittsgrad garantiert keine Greedy-Einbettung.*

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^* 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^* (und damit an allen a_j)
 - dichtester Knoten an b^+ ist ein b_i
 - **Widerspruch!**



⇒ *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
 - $K_{3,16}, \dots$ 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
 - $K_{3,16}, \dots$ 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
 - $K_{3,16}, \dots$ sind nicht mehr planar.

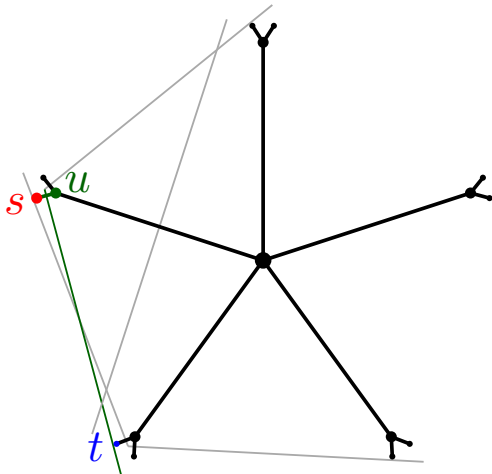
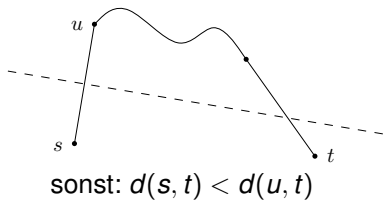
Inzwischen gelöst:

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.

Äquivalente Bedingung zu Greedy

Mittelsenkrechte an eine Kante schneidet keine andere Kante.



Äquivalente Bedingung zu Greedy

Mittelsenkrechte an eine Kante schneidet keine andere Kante.

Satz [Nöllenburg, Prutkin 2013]

Man kann in $O(n)$ entscheiden, ob ein Baum eine GE in \mathbb{R}^2 hat.

Äquivalente Bedingung zu Greedy

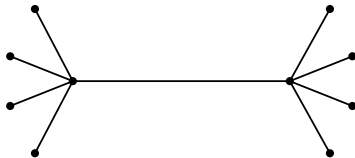
Mittelsenkrechte an eine Kante schneidet keine andere Kante.

Satz [Nöllenburg, Prutkin 2013]

Man kann in $O(n)$ entscheiden, ob ein Baum eine GE in \mathbb{R}^2 hat.

Knobelaufgabe

Hat folgender Baum eine GE?



Äquivalente Bedingung zu Greedy

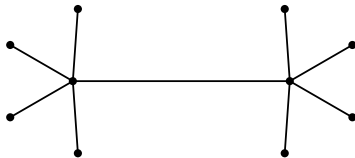
Mittelsenkrechte an eine Kante schneidet keine andere Kante.

Satz [Nöllenburg, Prutkin 2013]

Man kann in $O(n)$ entscheiden, ob ein Baum eine GE in \mathbb{R}^2 hat.

Knobelaufgabe

Hat folgender Baum eine GE?



Äquivalente Bedingung zu Greedy

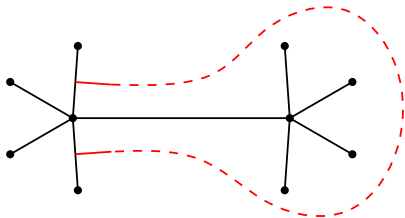
Mittelsenkrechte an eine Kante schneidet keine andere Kante.

Satz [Nöllenburg, Prutkin 2013]

Man kann in $O(n)$ entscheiden, ob ein Baum eine GE in \mathbb{R}^2 hat.

Knobelaufgabe

Hat folgender Baum eine GE?



Äquivalente Bedingung zu Greedy

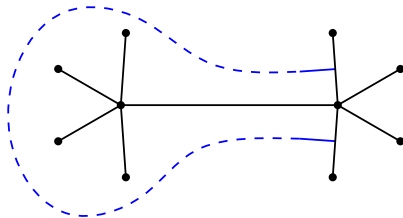
Mittelsenkrechte an eine Kante schneidet keine andere Kante.

Satz [Nöllenburg, Prutkin 2013]

Man kann in $O(n)$ entscheiden, ob ein Baum eine GE in \mathbb{R}^2 hat.

Knobelaufgabe

Hat folgender Baum eine GE?



Äquivalente Bedingung zu Greedy

Mittelsenkrechte an eine Kante schneidet keine andere Kante.

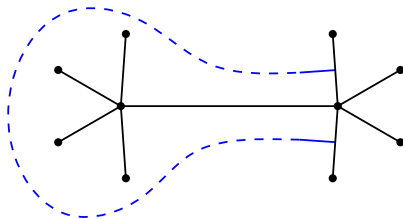
Satz [Nöllenburg, Prutkin 2013]

Man kann in $O(n)$ entscheiden, ob ein Baum eine GE in \mathbb{R}^2 hat.

Knobelaufgabe

Hat folgender Baum eine GE?

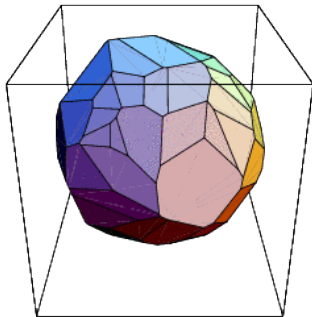
Nein



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.

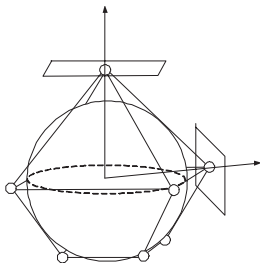
- Beweis:
 - Jeder dreifach zshg. planare Graph ist Kantengraph eines konvexen Polytops
 - sogar so, dass alle Kanten eine gemeinsamen Kugel (um 0) berühren
 - setze $d(t, v) = -\vec{t} \cdot \vec{v}$
- keine Metrik, schwer zu verteilen



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 $d(t, v) = -\vec{t} \cdot \vec{v}$
- keine Metrik, schwer zu verteilen



Tree-Routing

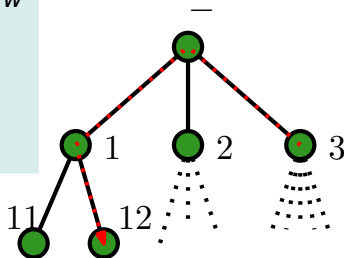
Vorbereitung

- 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?



Tree-Routing

Vorbereitung

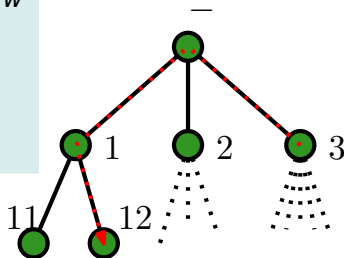
- Berechne aufspannenden Baum
- Wähle Wurzel w beliebig
- Identifiziere jeden Knoten mit Abstiegen von w

Routing

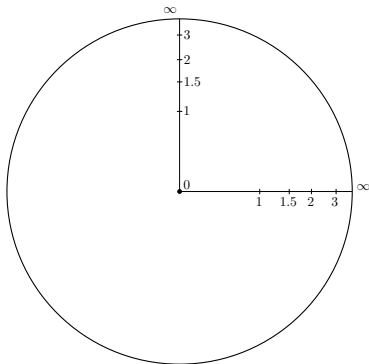
- ist t Nachfolger, steige in entsprechenden Teilbaum ab
- sonst steige auf

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

Nein! ($K_{1,6}$)

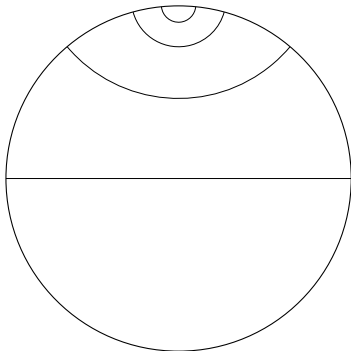


Poincaré-Disk-Modell der Hyperbolischen Ebene \mathbb{H}^2



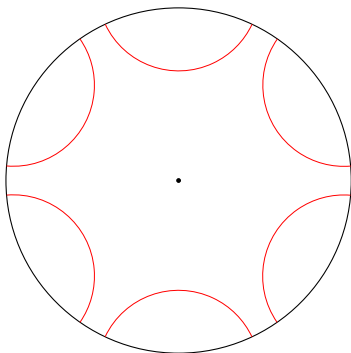
- Punkte liegen im Kreis mit Zentrum 0 und ∞ am Rand
- Der Raum wird in Richtung Rand immer dichter
- Geraden: Kreise, die den Rand orthogonal schneiden

Poincaré-Disk-Modell der Hyperbolischen Ebene \mathbb{H}^2



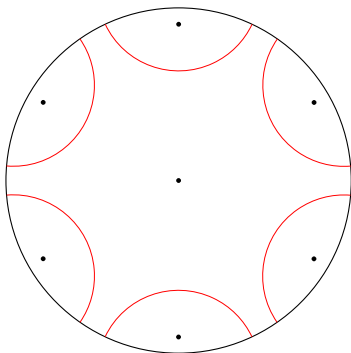
- Punkte liegen im Kreis mit Zentrum 0 und ∞ am Rand
- Der Raum wird in Richtung Rand immer dichter
- Geraden: Kreise, die den Rand orthogonal schneiden

Greedy Drawings in \mathbb{H}^2



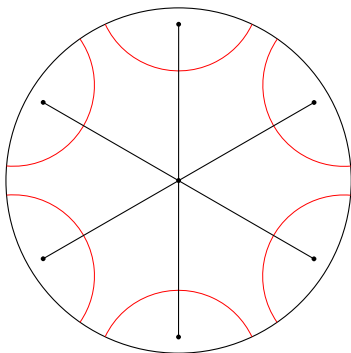
- Spiegelbilder von 0 an den Kreisbögen
- Kreisbögen sind die Mittelsenkrechten

Greedy Drawings in \mathbb{H}^2

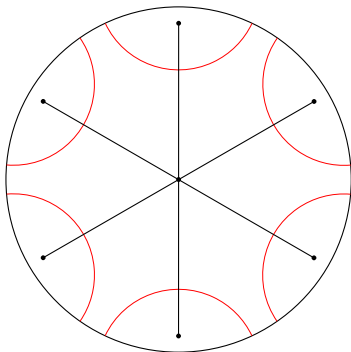


- Spiegelbilder von 0 an den Kreisbögen
- Kreisbögen sind die Mittelsenkrechten

Greedy Drawings in \mathbb{H}^2



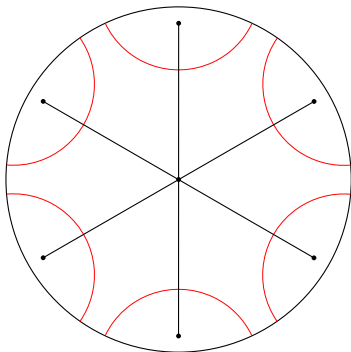
- Spiegelbilder von 0 an den Kreisbögen
- Kreisbögen sind die Mittelsenkrechten



- Spiegelbilder von 0 an den Kreisbögen
- Kreisbögen sind die Mittelsenkrechten

Äquivalente Bedingung zu Greedy, gilt auch in \mathbb{H}^2

Mittelsenkrechte an eine Kante schneidet keine andere Kante.

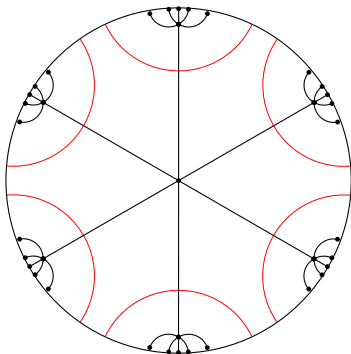


$K_{1,6}$ hat Greedy
Einbettung in \mathbb{H}^2

- Spiegelbilder von 0 an den Kreisbögen
- Kreisbögen sind die Mittelsenkrechten

Äquivalente Bedingung zu Greedy, gilt auch in \mathbb{H}^2

Mittelsenkrechte an eine Kante schneidet keine andere Kante.



$K_{1,6}$ hat Greedy
Einbettung in \mathbb{H}^2

alle anderen
Bäume auch

- Spiegelbilder von 0 an den Kreisbögen
- Kreisbögen sind die Mittelsenkrechten

Äquivalente Bedingung zu Greedy, gilt auch in \mathbb{H}^2

Mittelsenkrechte an eine Kante schneidet keine andere Kante.

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

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

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

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).

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).

- Bei gezielter Wahl



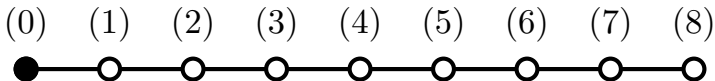
- Bei zufälliger Wahl



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



- 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



- 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)

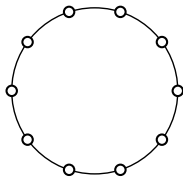


- Bei gezielter Wahl reicht 1 Ankerknoten für eindeutige Namen
 - Wähle Knoten am Ende des Pfades!
 - Routing dann klar
- Bei zufälliger Wahl reichen 2 Ankerknoten immer
 - Wenn zwei Knoten u, v dieselbe Koordinate $d_G(u, a_k) = d_G(v, a_k)$ haben, liegt a_k genau in der Mitte (und da liegt maximal 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

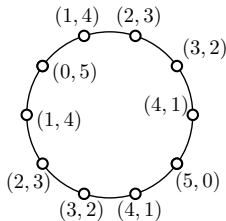
(2, 4) (1, 3) (0, 2) (1, 1) (2, 0) (3, 1) (4, 2) (5, 3) (6, 4)



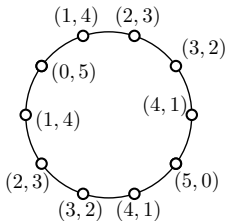
- Es reichen bei gezielter Wahl ?, bei zufälliger Wahl ? Knoten
 - (Fast) dasselbe Argument: Haben zwei Knoten dieselben Koordinaten, dann liegt der entsprechende Anker gleich weit von beiden entfernt
 - so ein Problem entsteht bei einem Knoten immer
 - bei zwei Knoten nur dann, wenn die Anker sich gegenüberliegen!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



- 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!)

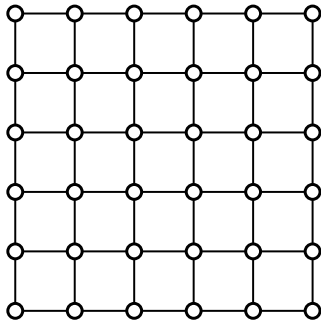


- Es reichen bei gezielter Wahl 2, bei zufälliger Wahl 3 Knoten
 - (Fast) dasselbe Argument: Haben zwei Knoten dieselben Koordinaten, dann liegt der entsprechende Anker gleich weit von beiden entfernt
 - so ein Problem entsteht bei einem Knoten immer
 - bei zwei Knoten nur dann, wenn die Anker sich gegenüberliegen!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



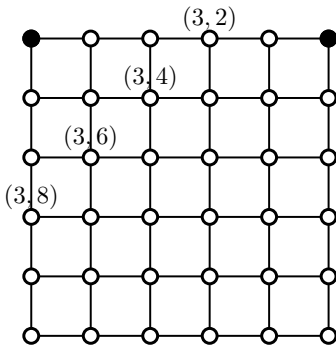
Quizfrage: Eindeutige Namen im Gitter

- 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)



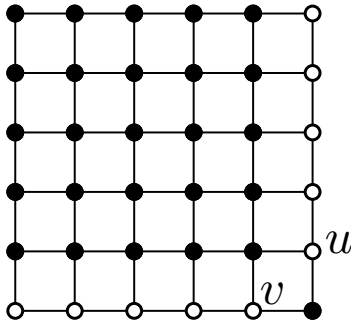
Quizfrage: Eindeutige Namen im Gitter

- 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*).

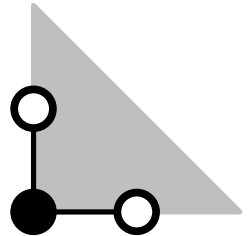
- Zwei benachbarte Blätter sind nur unterscheidbar, wenn einer der beiden Anker ist!
- ⇒ selbst bei gezielter Wahl braucht man die Hälfte der Blätter, also $\Theta(n)$ Anker!
- bei zufälliger Wahl schlimmstenfalls alle!

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!
- ⇒ 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!)

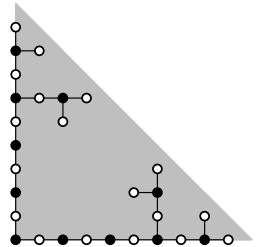


Wieviele Knoten braucht man in UDG?

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!
- ⇒ 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!)

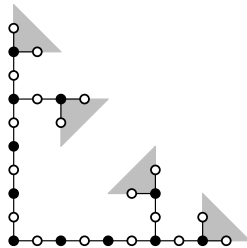


Wieviele Knoten braucht man in UDG?

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!
- ⇒ 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!)

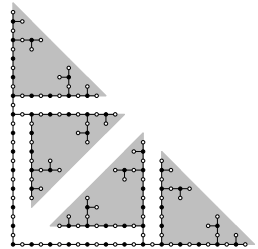


Wieviele Knoten braucht man in UDG?

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!
- ⇒ 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!)

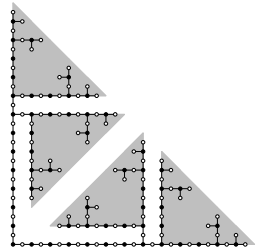


Wieviele Knoten braucht man in UDG?

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!
- ⇒ 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!)

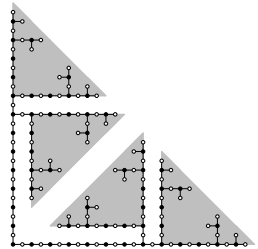


Wieviele Knoten braucht man in UDG?

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!
- ⇒ 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!)

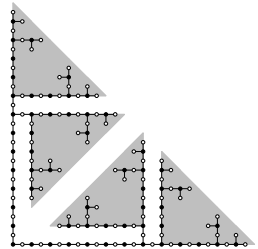


Wieviele Knoten braucht man in UDG?

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!
- ⇒ 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!)



- 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!

- 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

- 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 $d : \mathbb{Z}^k \rightarrow \mathbb{R}^+$ verbessert
- 3 wenn keine Verbesserung möglich, route in Richtung des dem Ziel dichtesten Ankers
 - Paket enthält bisher minimalen Abstand δ^-
 - wird unterschritten, wird wieder greedy geroutet
- 4 wenn das Paket den Anker erreicht, wird lokal geflutet
 - Radius ist bekannt!

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 Abstandsfunction $d : \mathbb{Z}^k \rightarrow \mathbb{R}^+$ verbessert
- 3 wenn keine Verbesserung möglich, route in Richtung des dem Ziel dichtesten Ankers
 - Paket enthält bisher minimalen Abstand δ^-
 - wird unterschritten, wird wieder greedy geroutet
- 4 wenn das Paket den Anker erreicht, wird lokal geflutet
 - Radius ist bekannt!

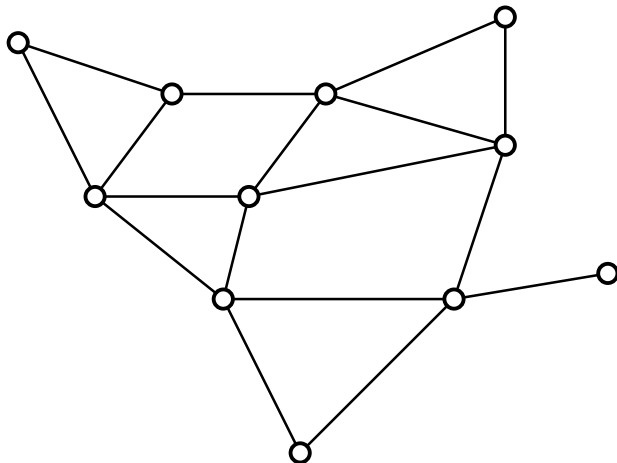
Beacon Vector Routing

- 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 $d : \mathbb{Z}^k \rightarrow \mathbb{R}^+$ verbessert
- 3 wenn keine Verbesserung möglich, route in Richtung des dem Ziel dichtesten Ankers
 - Paket enthält bisher minimalen Abstand δ^-
 - wird unterschritten, wird wieder greedy geroutet
- 4 wenn das Paket den Anker erreicht, wird lokal geflutet
 - Radius ist bekannt!

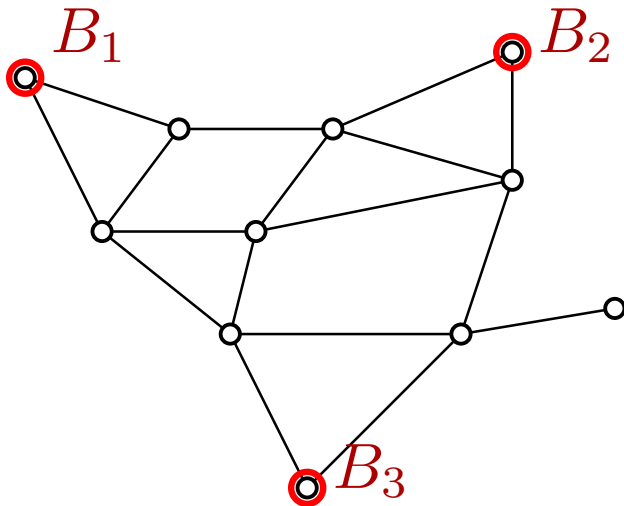
Beacon Vector Routing

- 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 $d : \mathbb{Z}^k \rightarrow \mathbb{R}^+$ verbessert
- 3 wenn keine Verbesserung möglich, route in Richtung des dem Ziel dichtesten Ankers
 - Paket enthält bisher minimalen Abstand δ^-
 - wird unterschritten, wird wieder greedy geroutet
- 4 wenn das Paket den Anker erreicht, wird lokal geflutet
 - Radius ist bekannt!

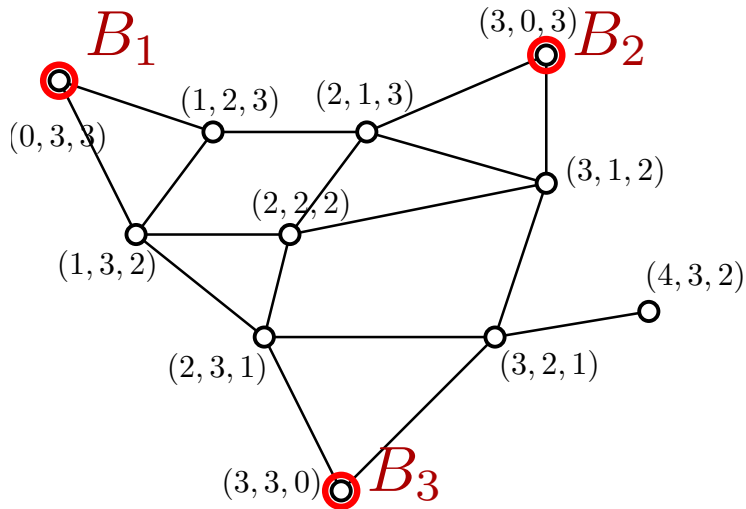
Ein einfaches Beispiel (L_1 -Metrik)



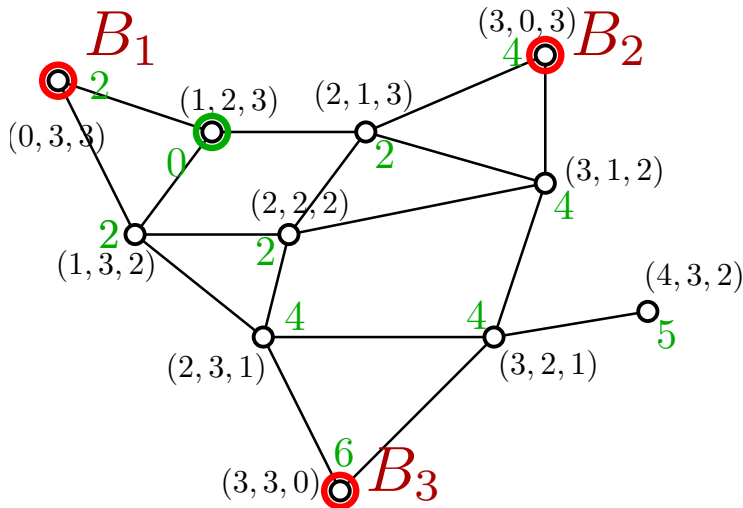
Ein einfaches Beispiel (L_1 -Metrik)



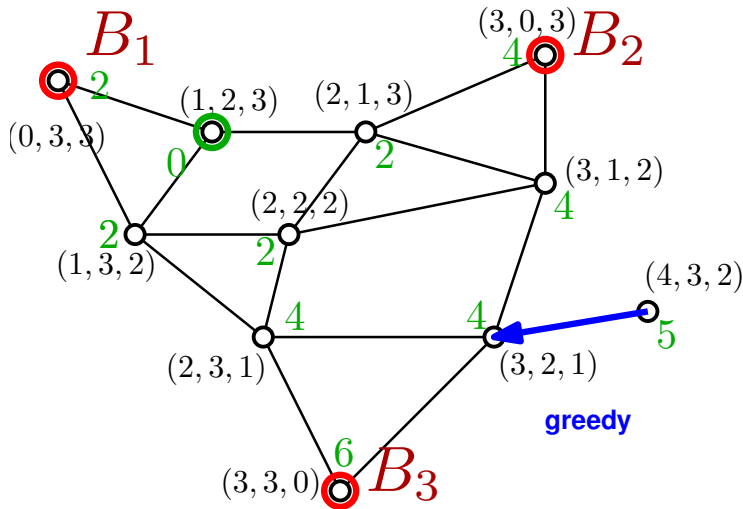
Ein einfaches Beispiel (L_1 -Metrik)



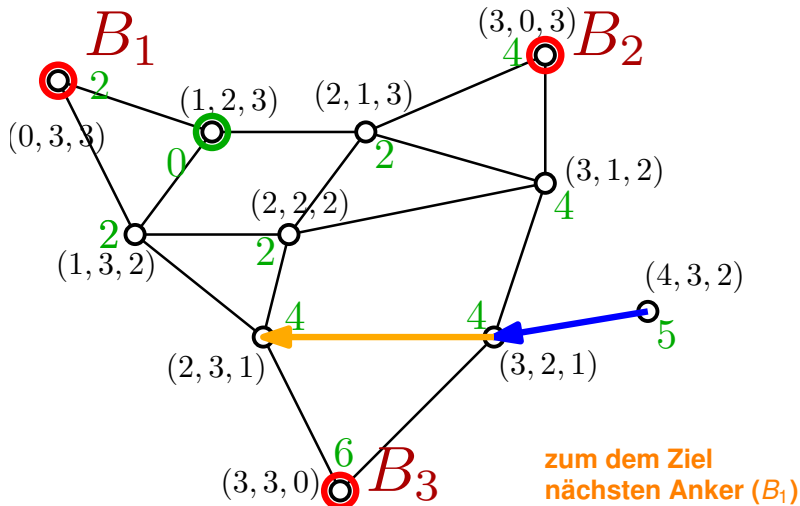
Ein einfaches Beispiel (L_1 -Metrik)



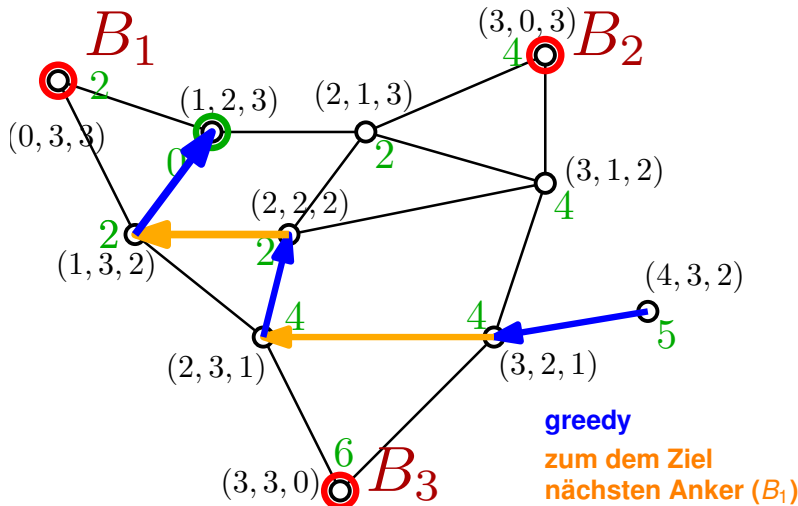
Ein einfaches Beispiel (L_1 -Metrik)



Ein einfaches Beispiel (L_1 -Metrik)



Ein einfaches Beispiel (L_1 -Metrik)



Beobachtung

Die Information von Landmarken, denen man näherkommen will:

$$d_G(B_i, t) < d_G(B_i, v),$$

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

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

verkleinere δ_+ möglichst stark, als Tie-Break: δ_-

⇒ Macht Fallback nicht unmöglich, aber unwahrscheinlicher...

Beobachtung

Die Information von Landmarken, denen man näherkommen will:

$$d_G(B_i, t) < d_G(B_i, v),$$

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

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

verkleinere δ_+ möglichst stark, als Tie-Break: δ_-

⇒ *Macht Fallback nicht unmöglich, aber unwahrscheinlicher...*

- Greedy-Einbettungen
- Pseudo-Geometrisches 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!

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!

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!

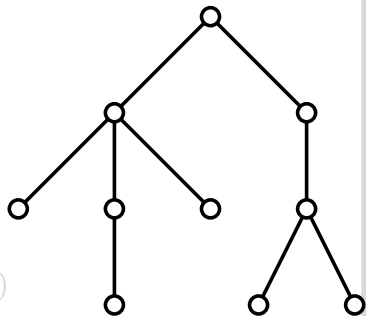
Als *kompakt* bezeichnet man alle Routingverfahren, die garantiert mit Routingtabellen mit $o(n)$ Bits lokalem Speicher auskommen.

- 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!

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?)

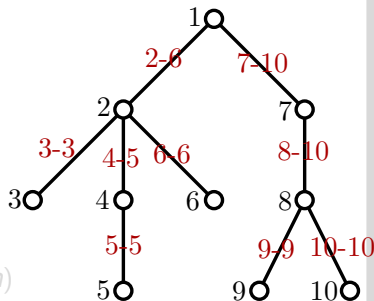


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist, wenn Knoten nicht umbenannt werden dürfen, 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?)

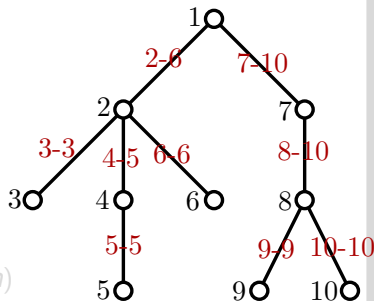


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist, wenn Knoten nicht umbenannt werden dürfen, 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?)

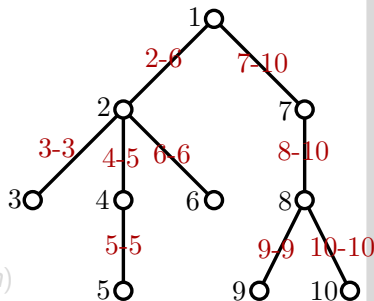


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist, wenn Knoten nicht umbenannt werden dürfen, 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?)

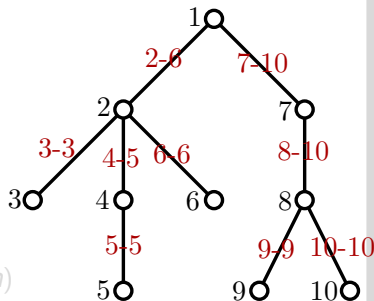


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist, wenn Knoten nicht umbenannt werden dürfen, 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?)

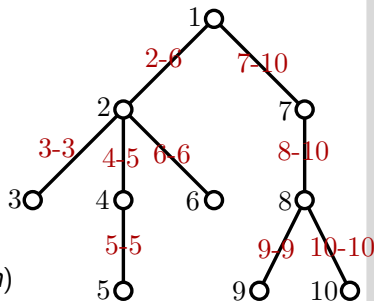


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist, wenn Knoten nicht umbenannt werden dürfen, 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?)

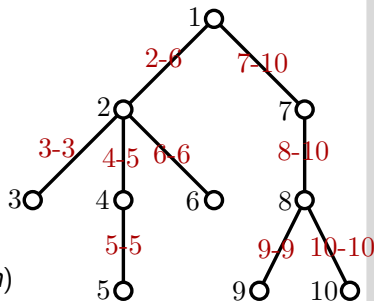


Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist, wenn Knoten nicht umbenannt werden dürfen, 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?)



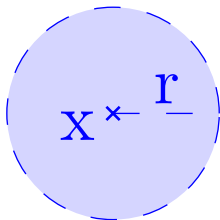
Klassisches Resultat (ohne Beweis)

In allgemeinen Graphen ist, wenn Knoten nicht umbenannt werden dürfen, mit $o(n)$ Bits kein Stretch < 3 für alle Knotenpaare möglich!

Definition

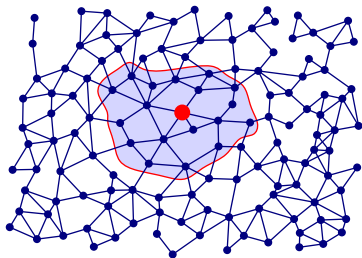
In einem metrischen Raum ist ein *Ball* $B(x, r)$ um ein Element x die Menge aller Elemente mit $d(x, y) \leq r$.

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



⇒ Kreise, Kugeln

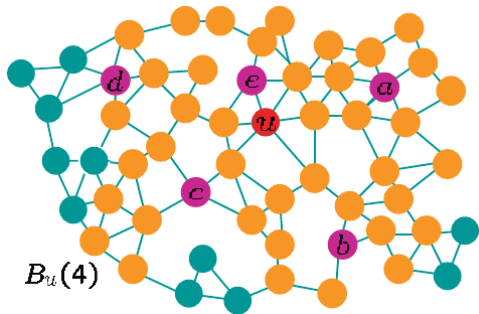
- Graph G mit hop-Abständen $d_G(u, v)$



⇒ r -hop-Nachbarschaften

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!

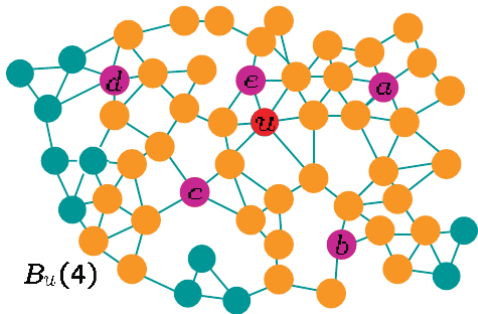
Knobelaufgabe

Gibt es UDG, die keine konstant beschränkte DD haben?

Bild: Roger Wattenhofer

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!

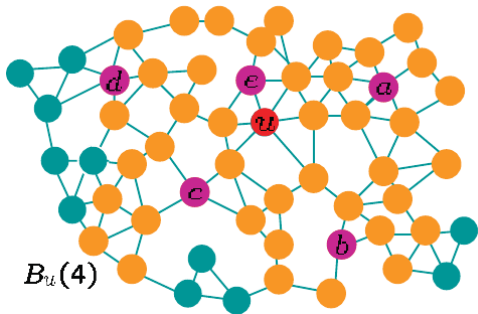
Knobelaufgabe

Gibt es UDG, die keine konstant beschränkte DD haben?

Bild: Roger Wattenhofer

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?

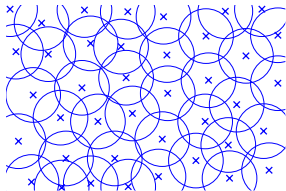
Bild: Roger Wattenhofer

Definition

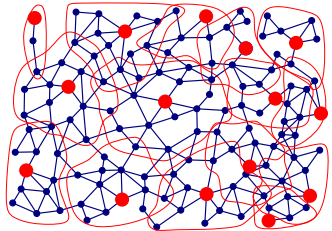
Ein ρ -Netz in einem metrischen Raum ist eine Teilmenge $U \subset V$ von Zentren mit

- 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



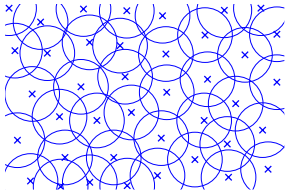
- Zentral leicht berechenbar: Entferne sukzessive Knoten u mitsamt ρ -Nachbarschaft aus G .

Definition

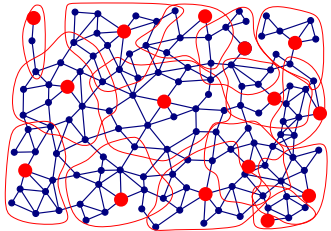
Ein ρ -Netz in einem metrischen Raum ist eine Teilmenge $U \subset V$ von Zentren mit

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



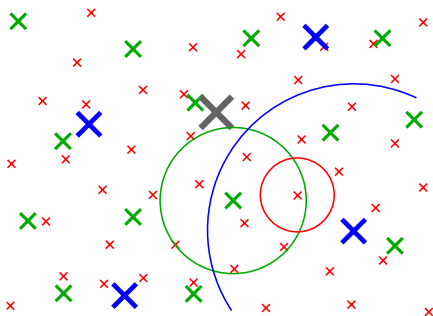
- Graphen



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

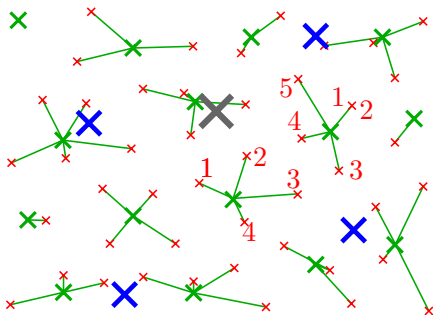
Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, \rho)$, wie hilft mir das?

- 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



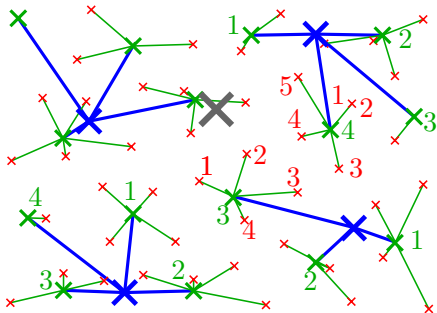
Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, \rho)$, wie hilft mir das?

- 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



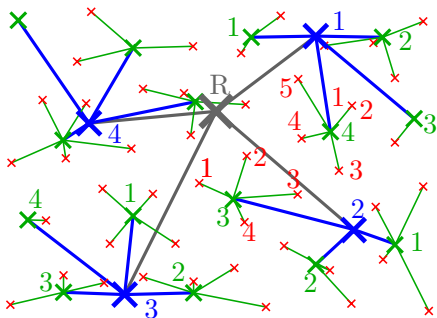
Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, \rho)$, wie hilft mir das?

- 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



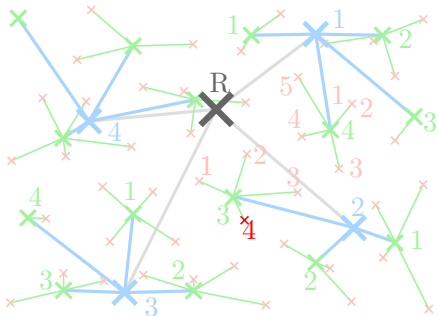
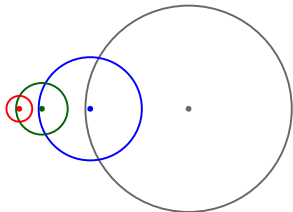
Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, \rho)$, wie hilft mir das?

- 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



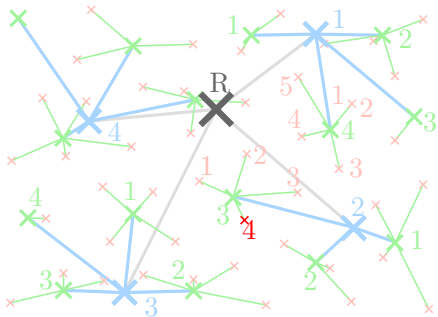
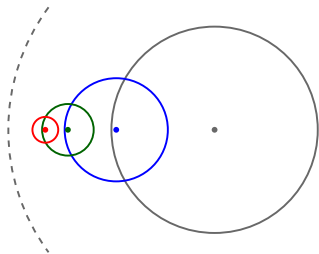
Anwendung für ρ -Netze: Routing

Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , 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?



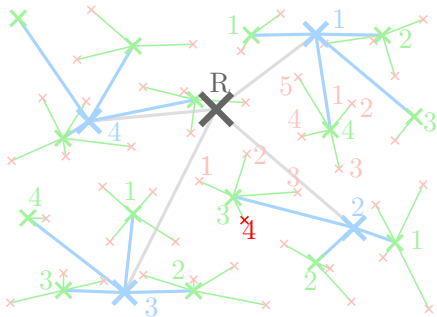
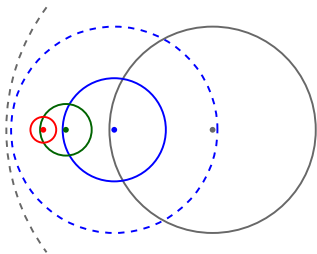
Anwendung für ρ -Netze: Routing

Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , 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?



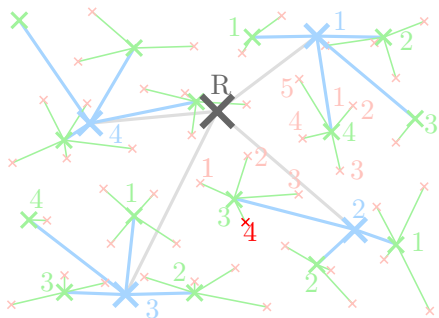
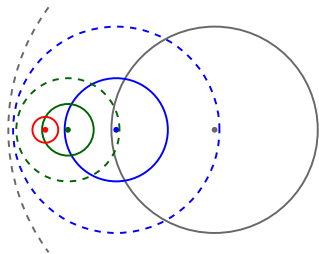
Anwendung für ρ -Netze: Routing

Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , 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?

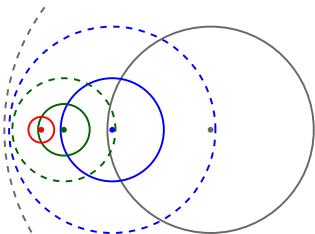


Anwendung für ρ -Netze: Routing

Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , 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?

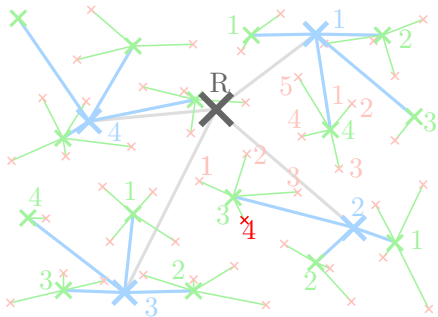


Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , eindeutige Adressen für jedes Zentrum und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, 2\rho)$, wie hilft mir das?



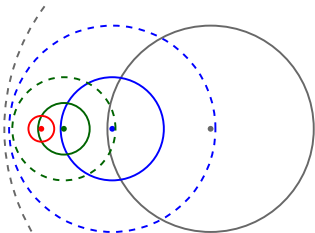
- Jeder Knoten speichert den Teilbaum aus Zentren, in deren verdoppeltem Radius er sich befindet

- Namensgebende Zentren werden markiert



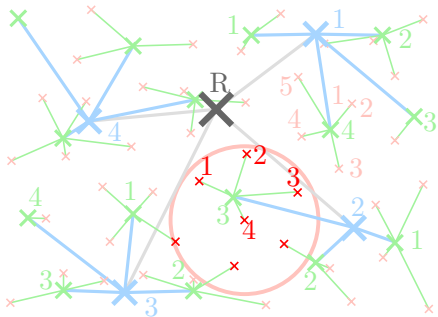
Anwendung für ρ -Netze: Routing

Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , 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?



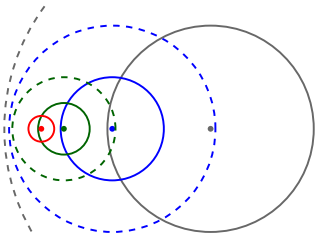
- Jeder Knoten speichert den Teilbaum aus Zentren, in deren verdoppeltem Radius er sich befindet

- Namensgebende Zentren werden markiert



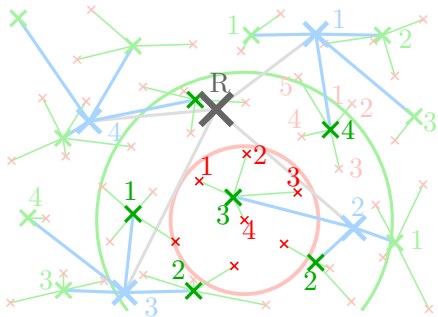
Anwendung für ρ -Netze: Routing

Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , 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?

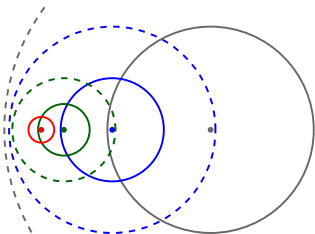


- Jeder Knoten speichert den Teilbaum aus Zentren, in deren verdoppeltem Radius er sich befindet

- Namensgebende Zentren werden markiert

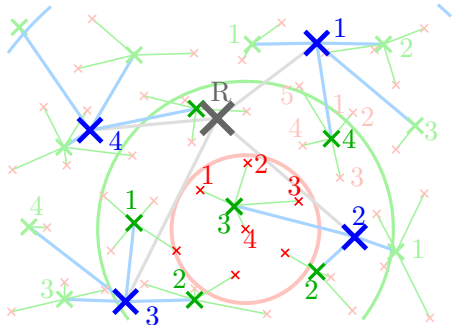


Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , eindeutige Adressen für jedes Zentrum und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, 2\rho)$, wie hilft mir das?

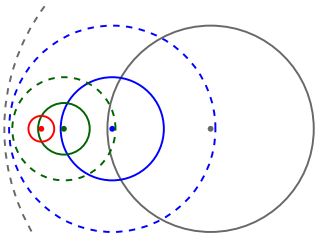


- Jeder Knoten speichert den Teilbaum aus Zentren, in deren verdoppeltem Radius er sich befindet

- Namensgebende Zentren werden markiert

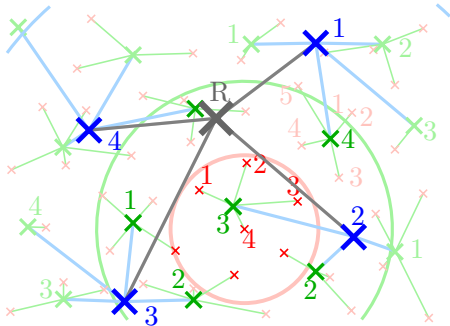


Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , eindeutige Adressen für jedes Zentrum und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, 2\rho)$, wie hilft mir das?



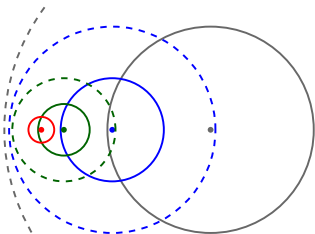
- Jeder Knoten speichert den Teilbaum aus Zentren, in deren verdoppeltem Radius er sich befindet

- Namensgebende Zentren werden markiert



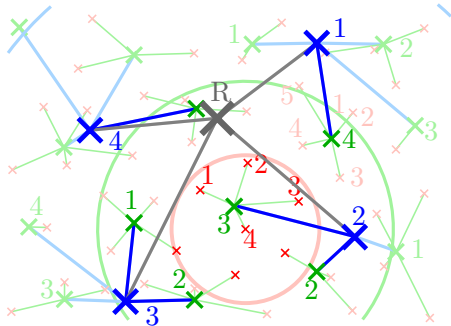
Anwendung für ρ -Netze: Routing

Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , 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?

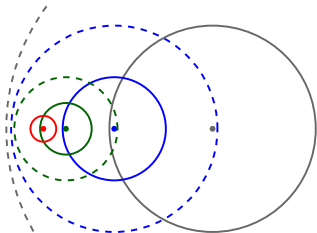


- Jeder Knoten speichert den Teilbaum aus Zentren, in deren verdoppeltem Radius er sich befindet

- Namensgebende Zentren werden markiert

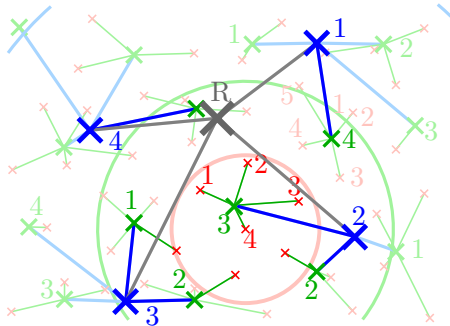


Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , eindeutige Adressen für jedes Zentrum und jedes x kennt jedes $u \in U_\rho$ mit $x \in B(u, 2\rho)$, wie hilft mir das?

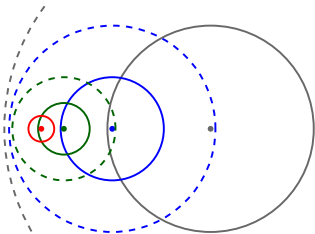


- Jeder Knoten speichert den Teilbaum aus Zentren, in deren verdoppeltem Radius er sich befindet

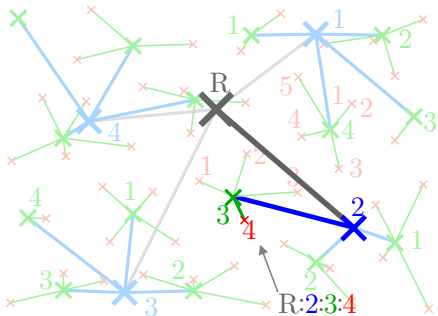
- Namensgebende Zentren werden markiert



Angenommen, ich habe ρ -Netze für $\rho = 1, 2, 4, \dots$, nämlich U_1, U_2, U_4, \dots , 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 speichert den Teilbaum aus Zentren, in deren verdoppeltem Radius er sich befindet
- Namensgebende Zentren werden markiert



Anwendung für ρ -Netze: Routing

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

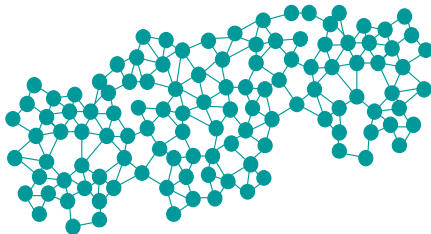


Bild: Roger Wattenhofer

Anwendung für ρ -Netze: Routing

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

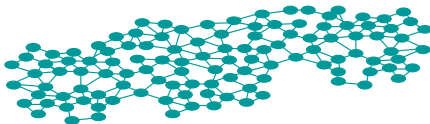


Bild: Roger Wattenhofer

Anwendung für ρ -Netze: Routing

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

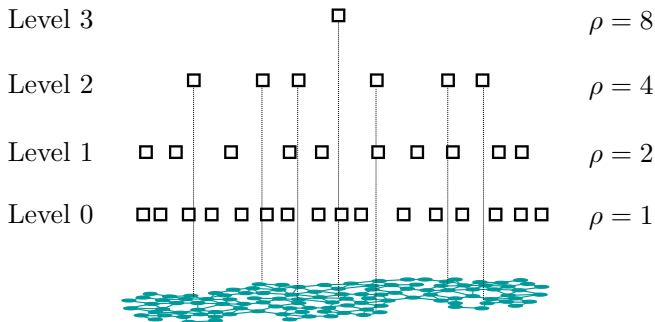


Bild: Roger Wattenhofer

Anwendung für ρ -Netze: Routing

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

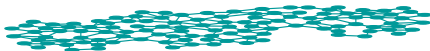
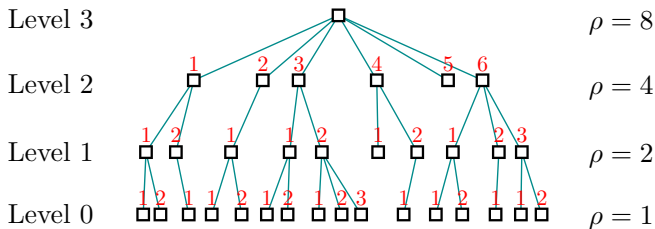


Bild: Roger Wattenhofer

- 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 (ohne Beweis); $\lceil \log D \rceil$ Ebenen
 - Kodiere nächsten Hop mit $\log \Delta$ Bits; Routingtabelle mit $O(\log \Delta \cdot \log D)$ reicht aus, um alle bekannten Zentren zu erreichen

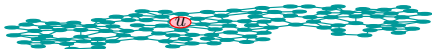
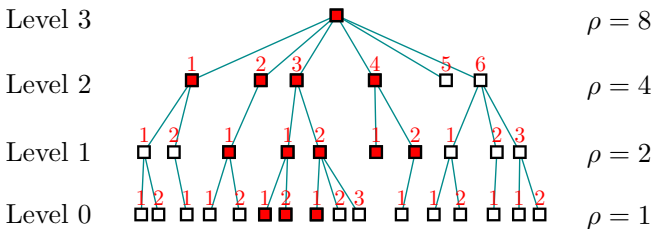


Bild: Roger Wattenhofer

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.)

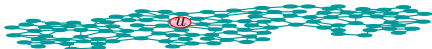
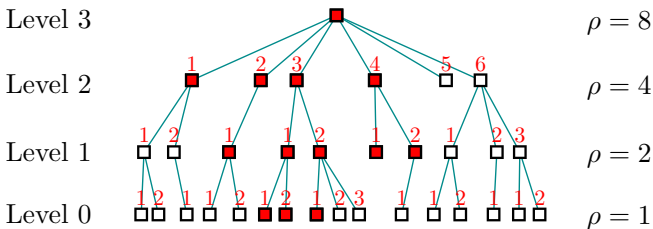


Bild: Roger Wattenhofer

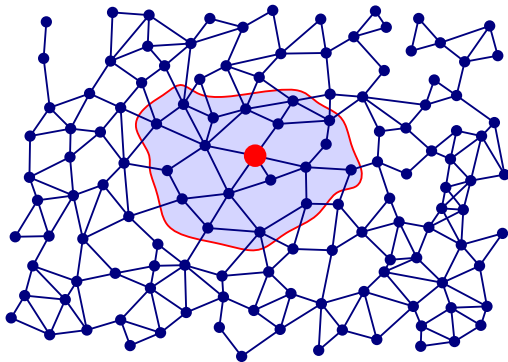
- Greedy-Einbettungen
 - Wann kann man einen Graphen so einbetten, dass klassisches Greedy-Routing immer funktioniert?
 - Vor allem theoretisch interessantes Problem
- Pseudo-Geometrisches Routing
 - Theorie: Auf Basis von Anker-Abständen zu routen ist nicht so leicht umzusetzen (viele Anker, Routing-Funktion?)
 - 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!

- 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 Sensor Networks*. 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
- 5 M. Nöllenburg, R. Prutkin: *Euclidean Greedy Drawings of Trees*. In: 21st European Symposium on Algorithms (ESA), 2013

Quizfrage

Hat jeder UDG konstant beschränkte Doubling Dimension?

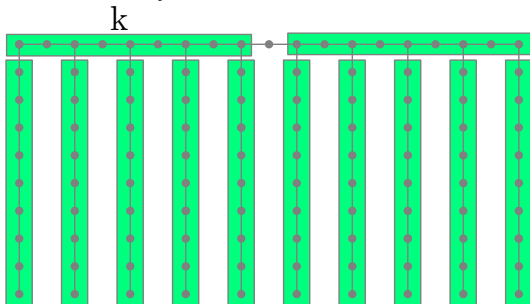
- Nein, alle Knoten liegen in Entfernung $2k$ vom blauen Knoten, aber rote Knoten haben disjunkte k -Nachbarschaften!



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!



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!

