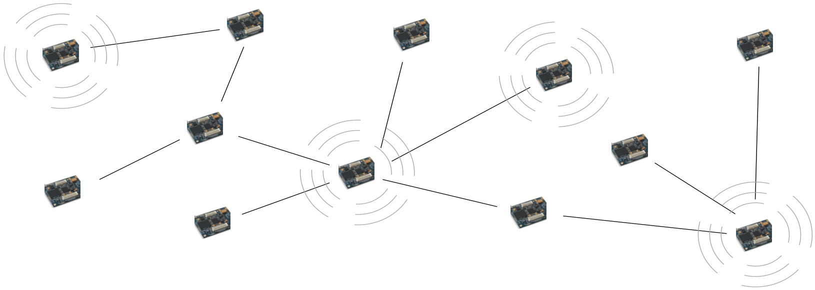
Karlsruher Institut für Technologie

Algorithmen für Ad-hoc- und Sensornetze

VL 05 – Lokalisierung und virtuelle Koordinaten

Fabian Fuchs | 5. Nov. 2015 (Version 1)

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



Thema heute: Lokalisierung

Karlsruher Institut für Technologie

Warum wir Knotenpositionen brauchen

- Zuordnung der Daten (was tun, wenn's brennt?)
- Georouting
- Topologiekontrolle
- ...

Was tun, wenn (alle/einige/viele) Knoten ihre Positionen nicht kennen, weil

- GPS/Galileo nicht zur Verfügung steht
  - ist schwer, groß, teuer (verglichen mit SmartDust)
  - kostet Energie
  - funktioniert nicht überall (drinnen, im Wald)
  - ist nicht besonders genau
- jedem Knoten die Position einprogrammieren nicht geht

Thema heute: Lokalisierung

Warum wir Knotenpositionen brauchen

- Zuordnung der Daten (was tun, wenn's brennt?)
- Georouting
- Topologiekontrolle
- ...

Was tun, wenn (alle/einige/viele) Knoten ihre Positionen nicht kennen, weil

- GPS/Galileo nicht zur Verfügung steht
  - ist schwer, groß, teuer (verglichen mit SmartDust)
  - kostet Energie
  - funktioniert nicht überall (drinnen, im Wald)
  - ist nicht besonders genau
- jedem Knoten die Position einprogrammieren nicht geht

Überblick

- Überblick: Entfernungs- und Richtungsschätzung
- Lokalisierung mit Ankerknoten
- Ankerfreie Lokalisierung: Virtuelle Koordinaten
  - Komplexitätsbetrachtungen
  - Heuristiken zur ankerfreien Lokalisierung

Was man alles messen kann

- Entfernungen
  - Zusammenhang im Kommunikationsgraphen
    - sehr grobe Information über benachbarte Lage von Knoten
  - RSSI: Received Signal Strength Indicator
    - Sendestärke bekannt $\Rightarrow$ Distanz lässt sich *theoretisch* schätzen
    - in der *Praxis* recht ungenau
  - ToA: Time of Arrival
    - Bestimmung von Signallaufzeiten (roundtrip time)
  - TDoA: Time Difference of Arrival
    - Laufzeitvergleiche zweier Signale (z. B. Radio / Ultraschall)
- Richtungen
  - AoA: Angle of Arrival
    - Phasenverschiebung in Antennenarrays
    - gerichtete Antennen
  - Zusatzhardware (Laser etc.)

Was man alles messen kann

- Entfernungen
  - Zusammenhang im Kommunikationsgraphen
    - sehr grobe Information über benachbarte Lage von Knoten
  - RSSI: Received Signal Strength Indicator
    - Sendestärke bekannt $\Rightarrow$ Distanz lässt sich *theoretisch* schätzen
    - in der *Praxis* recht ungenau
  - ToA: Time of Arrival
    - Bestimmung von Signallaufzeiten (roundtrip time)
  - TDoA: Time Difference of Arrival
    - Laufzeitvergleiche zweier Signale (z. B. Radio / Ultraschall)

- Richtungen
  - AoA: Angle of Arrival
    - Phasenverschiebung in Antennenarrays
    - gerichtete Antennen
  - Zusatzhardware (Laser etc.)

Überblick

- Überblick: Entfernungs- und Richtungsschätzung
- Lokalisierung mit Ankerknoten
- Ankerfreie Lokalisierung: Virtuelle Koordinaten
  - Komplexitätsbetrachtungen
  - Heuristiken zur ankerfreien Lokalisierung

Lokalisierung mit Ankerknoten

Wenn GPS/Einprogrammieren so teuer ist, reicht es vielleicht, wenn *einige* Knoten, sogenannte *Ankerknoten*, ihre Position kennen?

Wenn jeder Knoten genug Anker „sieht“:

- *Trilateration* bei bekannten Enfernungen
  - Drei Anker für eindeutige Lokalisierung
- *Triangulation* bei bekannten Winkeln
  - Zwei Anker für eindeutige Lokalisierung
- Bei gestörten Werten und/oder mehr Ankern
  - Least Squares Verfahren etc. → Schätztheorie

Was, wenn wir nur wenige Anker haben? Einige Knoten haben dann vielleicht keine Anker in ihrer Nachbarschaft!

Lokalisierung mit Ankerknoten

Karlsruher Institut für Technologie

Wenn GPS/Einprogrammieren so teuer ist, reicht es vielleicht, wenn *einige* Knoten, sogenannte *Ankerknoten*, ihre Position kennen?

Wenn jeder Knoten genug Anker „sieht“:

- *Trilateration* bei bekannten Enfernungen
  - Drei Anker für eindeutige Lokalisierung
- *Triangulation* bei bekannten Winkeln
  - Zwei Anker für eindeutige Lokalisierung
- Bei gestörten Werten und/oder mehr Ankern
  - Least Squares Verfahren etc. $\rightarrow$ Schätztheorie

Was, wenn wir nur wenige Anker haben? Einige Knoten haben dann vielleicht keine Anker in ihrer Nachbarschaft!

Lokalisierung mit Ankerknoten

Wenn GPS/Einprogrammieren so teuer ist, reicht es vielleicht, wenn *einige* Knoten, sogenannte *Ankerknoten*, ihre Position kennen?

Wenn jeder Knoten genug Anker „sieht“:

- *Trilateration* bei bekannten Enfernungen
  - Drei Anker für eindeutige Lokalisierung
- *Triangulation* bei bekannten Winkeln
  - Zwei Anker für eindeutige Lokalisierung
- Bei gestörten Werten und/oder mehr Ankern
  - Least Squares Verfahren etc. → Schätztheorie

Was, wenn wir nur wenige Anker haben? Einige Knoten haben dann vielleicht keine Anker in ihrer Nachbarschaft!

Lokalisierung mit Ankerknoten

Karlsruher Institut für Technologie

Wenn GPS/Einprogrammieren so teuer ist, reicht es vielleicht, wenn *einige* Knoten, sogenannte *Ankerknoten*, ihre Position kennen?

Wenn jeder Knoten genug Anker „sieht“:

- *Trilateration* bei bekannten Enfernungen
  - Drei Anker für eindeutige Lokalisierung
- *Triangulation* bei bekannten Winkeln
  - Zwei Anker für eindeutige Lokalisierung
- Bei gestörten Werten und/oder mehr Ankern
  - Least Squares Verfahren etc. $\rightarrow$ Schätztheorie

Was, wenn wir nur wenige Anker haben? Einige Knoten haben dann vielleicht keine Anker in ihrer Nachbarschaft!

Lokalisierung mit wenigen Ankern

Typisches Vorgehen

Knoten schätzen Abstände/Richtungen zu Ankern über mehrere Hops und wenden *dann* Triangulation/Trilateration an.

- Furchtbar viele Verfahren und Heuristiken
- unklare Problemstellung
  - Dichte, Verteilung von Ankerknoten
- Was kann man aus algorithmischer Sicht beitragen?

Im folgenden nehmen wir immer an, dass wir keine weiteren Messungen haben, sondern nur den Verbindungsgraphen und das UDG-Modell.

Algorithmus HOP

Idee

In großen Unit-Disk-Graphen könnte die Länge eines kürzesten Weges stark genug mit der Entfernung korrellieren!

1. Berechne Hop-Distanz zu Ankerknoten
   - Broadcasts von Ankerknoten genügen
2. Wähle Position so, dass Abstände der Hop-Distanz bestmöglich entsprechen

Wie gut funktioniert ein solcher Ansatz?

Algorithmus HOP

Idee

In großen Unit-Disk-Graphen könnte die Länge eines kürzesten Weges stark genug mit der Entfernung korrellieren!

1. Berechne Hop-Distanz zu Ankerknoten
   - Broadcasts von Ankerknoten genügen
2. Wähle Position so, dass Abstände der Hop-Distanz bestmöglich entsprechen

Wie gut funktioniert ein solcher Ansatz?

Kompetitivität: Vorbereitung

Problem: UDG-Lokalisierung anhand von Ankerknoten

Gegeben: Unit-Disk-Graph $G = (V, E)$ mit entsprechender Einbettung $\mathbf{p} : V \to \mathbb{R}^2$, Ankerknoten $V_\mathrm{A} \subset V$ mit bekannten Positionen

Gesucht: Knotenpositionen $\widetilde{\mathbf{p}} : V \to \mathbb{R}^2$ mit geringen Fehlern

$$\mathrm{Err}(v) := \|\mathbf{p}(v) - \widetilde{\mathbf{p}}(v)\| \quad .$$

In die Berechnung der $\widetilde{\mathbf{p}}(v)$ dürfen keine $\mathbf{p}(v)$ für ein $v \notin V_A$ einfließen!

Optimalität, Kompetitivität

Definition: Optimaler Algorithmus

Ein *optimaler Algorithmus* wählt zu einem Graphen G und Ankerpositionen die Knotenpositionen, die den maximalen Fehler über alle möglichen Einbettungen minimieren.

- Das muss nicht leicht sein, ist aber ein guter Vergleich:
  - Kein Algorithmus ist im worst-case besser!

Definition: Kompetitivität

Ein Algorithmus $\mathtt{ALG}$ heißt *c-kompetitiv*, wenn immer gilt:

$$\mathrm{Err}_{\mathtt{ALG}}(v) \leq c \cdot \mathrm{Err}_{\mathtt{OPT}}(v) + k$$

für alle $v \in V$ und ein $k \in \mathbb{R}$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Optimalität, Kompetitivität

Definition: Optimaler Algorithmus

Ein *optimaler Algorithmus* wählt zu einem Graphen G und Ankerpositionen die Knotenpositionen, die den maximalen Fehler über alle möglichen Einbettungen minimieren.

- Das muss nicht leicht sein, ist aber ein guter Vergleich:
  - Kein Algorithmus ist im worst-case besser!

Definition: Kompetitivität

Ein Algorithmus $\texttt{ALG}$ heißt *c-kompetitiv*, wenn immer gilt:

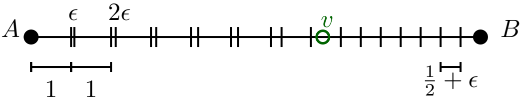
$$\mathrm{Err}_{\texttt{ALG}}(v) \leq c \cdot \mathrm{Err}_{\texttt{OPT}}(v) + k$$

für alle $v \in V$ und ein $k \in \mathbb{R}$.

Kompetitivität von HOP

Satz

HOP ist bereits in einer Dimension nicht kompetitiv.

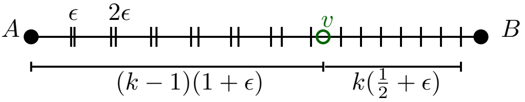


- Für bel. k setze Anker $A = 0$, $B = (k-1)(1+\epsilon) + k(\frac{1}{2}+\epsilon)$
- Jeder Algo, der nur Hops zählt, setzt v bestenfalls in die Mitte!
  - macht ein Algo es besser, drehen wir die Instanz um!
  - Fehler in $\Theta(k)$!
- Ist das bereits optimal?

Kompetitivität von HOP

Satz

HOP ist bereits in einer Dimension nicht kompetitiv.

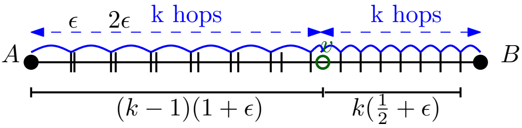


- Für bel. k setze Anker $A = 0$, $B = (k-1)(1+\epsilon) + k(\frac{1}{2}+\epsilon)$
- Jeder Algo, der nur Hops zählt, setzt v bestenfalls in die Mitte!
  - macht ein Algo es besser, drehen wir die Instanz um!
  - Fehler in $\Theta(k)$!
- Ist das bereits optimal?

Kompetitivität von HOP

Satz

HOP ist bereits in einer Dimension nicht kompetitiv.



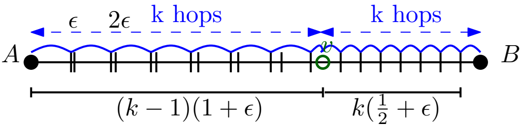
- Für bel. k setze Anker $A = 0$, $B = (k-1)(1+\epsilon) + k(\frac{1}{2} + \epsilon)$
- Jeder Algo, der nur Hops zählt, setzt v bestenfalls in die Mitte!
  - macht ein Algo es besser, drehen wir die Instanz um!
  - Fehler in $\Theta(k)$!
- Ist das bereits optimal?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kompetitivität von HOP

Satz

HOP ist bereits in einer Dimension nicht kompetitiv.

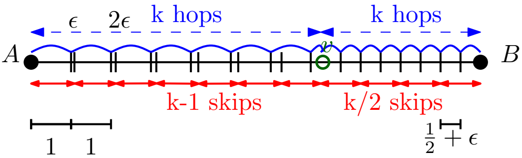


- Für bel. k setze Anker $A = 0$, $B = (k-1)(1+\epsilon) + k(\frac{1}{2}+\epsilon)$
- Jeder Algo, der nur Hops zählt, setzt v bestenfalls in die Mitte!
  - macht ein Algo es besser, drehen wir die Instanz um!
  - Fehler in $\Theta(k)$!
- Ist das bereits optimal?

Kompetitivität von HOP

Satz

HOP ist bereits in einer Dimension nicht kompetitiv.



Definition: Skip

Für einen Graph $G = (V, E)$ bilden zwei Knoten $u, w \in V$ einen *Skip*, falls $\{u, w\} \notin E$ und $\exists v$, so dass $\{u, v\}, \{v, w\} \in E$.

- *Skips* belegen Mindestabstände!
  - In diesem Fall grenzen sie die Position bis auf $O(k\epsilon)$ ein!

Kompetitivität von HOP

Karlsruher Institut für Technologie

Definition: Skip-Pfad

Ein Pfad $A = v_0, u_1, v_1, \ldots, u_s, v_s = v$ ist ein *Skip-Pfad* der Länge s zwischen A und v, wenn wenn

- $d_G(A, v_i) < d_G(A, v_{i+1})$
- $d_G(v_i, v_{i+1}) > 1$ (also $(v_i, v_{i+1}) \notin E$)

Die Länge eines längsten solchen Pfades ist die Skip-Entfernung von v zu A.

- *Hops* sind obere Schranke an Entfernung zum Anker.
- *Skips* sind untere Schranke!

Wie berechnet man Hop- und Skip-Distanzen?

Kompetitivität von HOP

Definition: Skip-Pfad

Ein Pfad $A = v_0, u_1, v_1, \ldots, u_s, v_s = v$ ist ein *Skip-Pfad* der Länge s zwischen A und v, wenn wenn

- $d_G(A, v_i) < d_G(A, v_{i+1})$
- $d_G(v_i, v_{i+1}) > 1$ (also $(v_i, v_{i+1}) \notin E$)

Die Länge eines längsten solchen Pfades ist die Skip-Entfernung von v zu A.

- *Hops* sind obere Schranke an Entfernung zum Anker.
- *Skips* sind untere Schranke!

Wie berechnet man Hop- und Skip-Distanzen?

Erinnerung: Hop-Berechnung

Karlsruher Institut für Technologie

Wie berechnet man *Hop-Distanzen*?

- jeder Knoten v startet mit $h_v = \infty$.
- hört ein Knoten v von einem Nachbarn u mit $h_u + 1 < h_v$, verringert er h_v zu $h_v = h_u + 1$.
- wenn ein Knoten sein h_v verringert, teilt er das allen Nachbarn mit.
- Startschuss: Ankerknoten verringert seinen Abstand auf $h_A = 0$.

Wenn man *hops* zu mehreren Ankern berechnen will, muss der jeweilige Anker Teil der Nachricht sein!

> *Ich, v habe meinen hop-Abstand zu Anker A auf h_v reduziert.*

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Erinnerung: Hop-Berechnung

Wie berechnet man *Hop-Distanzen*?

- jeder Knoten v startet mit $h_v = \infty$.
- hört ein Knoten v von einem Nachbarn u mit $h_u + 1 < h_v$, verringert er h_v zu $h_v = h_u + 1$.
- wenn ein Knoten sein h_v verringert, teilt er das allen Nachbarn mit.
- Startschuss: Ankerknoten verringert seinen Abstand auf $h_A = 0$.

Wenn man *hops* zu mehreren Ankern berechnen will, muss der jeweilige Anker Teil der Nachricht sein!

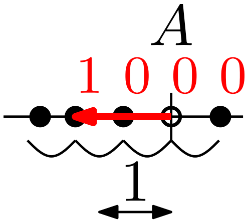
> *Ich, v habe meinen hop-Abstand zu Anker A auf h_v reduziert.*

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wie berechnet man *Skips*?

- jeder Knoten v startet mit $s_v = -1$.
- hört ein Knoten v von einem Nicht-Nachbarn u mit $s_u + 1 > s_v$, erhöht er s_v zu $s_v = s_u + 1$.
- wenn ein Knoten sein s_v erhöht, teilt er das allen Zwei-hop-Nachbarn mit.
- Startschuss: Jeder Nachbar v des Ankerknoten und A selbst erhöht seinen Abstand auf $s_v = 0$.

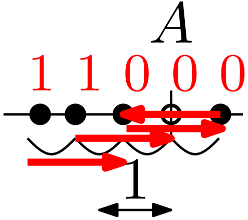
- Wie verhindert man „Aufschaukeln“?
- Beleg für Erhöhung sind nur Nicht-Nachbarn u, die echt zwischen v und Anker A liegen
- Dann liegt ein Nachbar mit niedrigerem Hop-Abstand dazwischen!



Wie berechnet man *Skips*?

- jeder Knoten v startet mit $s_v = -1$.
- hört ein Knoten v von einem Nicht-Nachbarn u mit $s_u + 1 > s_v$, erhöht er s_v zu $s_v = s_u + 1$.
- wenn ein Knoten sein s_v erhöht, teilt er das allen Zwei-hop-Nachbarn mit.
- Startschuss: Jeder Nachbar v des Ankerknoten und A selbst erhöht seinen Abstand auf $s_v = 0$.

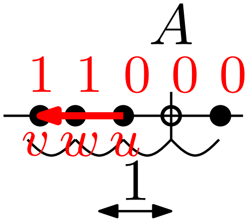
- Wie verhindert man „Aufschaukeln“?
- Beleg für Erhöhung sind nur Nicht-Nachbarn u, die echt zwischen v und Anker A liegen
- Dann liegt ein Nachbar mit niedrigerem Hop-Abstand dazwischen!



Wie berechnet man *Skips*?

- jeder Knoten v startet mit $s_v = -1$.
- hört ein Knoten v von einem Nicht-Nachbarn u mit $s_u + 1 > s_v$, *und zwar über einen Nachbarn* $w \neq A$ *mit* $h_w < h_v$, erhöht er s_v zu $s_v = s_u + 1$.
- wenn ein Knoten sein s_v erhöht, teilt er das allen Zwei-hop-Nachbarn mit.
- Startschuss: Jeder Nachbar v des Ankerknoten und A selbst erhöht seinen Abstand auf $s_v = 0$.

- Wie verhindert man „Aufschaukeln"?
- Beleg für Erhöhung sind nur Nicht-Nachbarn u, die echt zwischen v und Anker A liegen
- Dann liegt ein Nachbar mit niedrigerem Hop-Abstand dazwischen!



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Algorithmus HS (Hop & Skip)

1. Berechne Hop-Distanz zu allen Ankerknoten
2. Berechne Skip-Distanz zu allen Ankerknoten
3. Schneide Intervalle aus Skip- und Hop-Distanz für alle Anker
4. Wähle Punkt, der über das verbleibende Intervall den möglichen Fehler minimiert ($s < \mathrm{dist}_E(A, v) \leq h$)

Satz (ohne Beweis)

HS ist 1-kompetitiv in einer Dimension. [O'Dell, Wattenhofer, 2005].

- Man kann zeigen, dass wirklich jede Position im Schnitt der Intervalle angenommen werden kann

In zwei Dimensionen: Fehlende Kompetitivität von HOP bleibt, aber von HS bleibt nur die Idee, auch untere Schranken für die Länge von Pfaden zu verwenden.

Algorithmus HS (Hop & Skip)

1. Berechne Hop-Distanz zu allen Ankerknoten
2. Berechne Skip-Distanz zu allen Ankerknoten
3. Schneide Intervalle aus Skip- und Hop-Distanz für alle Anker
4. Wähle Punkt, der über das verbleibende Intervall den möglichen Fehler minimiert ($s < \mathrm{dist}_E(A, v) \leq h$)

Satz (ohne Beweis)

HS ist 1-kompetitiv in einer Dimension. [O'Dell, Wattenhofer, 2005].

- Man kann zeigen, dass wirklich jede Position im Schnitt der Intervalle angenommen werden kann

In zwei Dimensionen: Fehlende Kompetitivität von HOP bleibt, aber von HS bleibt nur die Idee, auch untere Schranken für die Länge von Pfaden zu verwenden.

Überblick

Karlsruher Institut für Technologie

- Überblick: Entfernungs- und Richtungsschätzung
- Lokalisierung mit Ankerknoten
- Ankerfreie Lokalisierung: Virtuelle Koordinaten
  - Komplexitätsbetrachtungen
  - Heuristiken zur ankerfreien Lokalisierung

Ankerfreie Lokalisierung

Karlsruher Institut für Technologie

Viele Anwendungen funktionieren mit *plausiblen* Koordinaten so gut wie mit echten (Georouting, ...).

- Ohne Ankerknoten kann es keine echte Lokalisierung geben
- Freiheitsgrade je nach Eingabe
  - Verschiebungen (immer)
  - Drehungen (fehlende absolute Richtungsinformation)
  - Spiegelungen (fehlende Richtungsinformation)
  - Skalierungen (fehlende Entfernungsinformation)
    - schon das Wissen, es mit UDG/qUDG zu tun zu haben, trägt Entfernungsinformation!
- Vorsicht: *plausible Koordinaten können auch völlig von der Realität abweichen!*
- Trotzdem: Oft sind plausible Koordinaten besser als keine!

Unit-Disk-Graph Einbettung

Wenn wir davon ausgehen, dass ein Graph ein Unit-Disk-Graph ist, wie schwer ist es dann, eine Einbettung zu finden, die das belegt?

(Erinnerung: Eigenschaft UDG zu sein ist unabhängig von der Einbettung!)

Mindestens so schwer, wie zu entscheiden, ob ein Graph ein Unit-Disk-Graph ist!

- Angenommen wir haben Algorithmus, um UDGs einzubetten
  - wende Algo auf irgendeinen Graphen an und teste, ob Ausgabe den Graphen als UDG einbettet
    - (das sind höchstens zusätzliche $O(n^2)$ Operationen)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Unit-Disk-Graph Einbettung

Wenn wir davon ausgehen, dass ein Graph ein Unit-Disk-Graph ist, wie schwer ist es dann, eine Einbettung zu finden, die das belegt?

(Erinnerung: Eigenschaft UDG zu sein ist unabhängig von der Einbettung!)

Mindestens so schwer, wie zu entscheiden, ob ein Graph ein Unit-Disk-Graph ist!

- Angenommen wir haben Algorithmus, um UDGs einzubetten
  - wende Algo auf irgendeinen Graphen an und teste, ob Ausgabe den Graphen als UDG einbettet
    - (das sind höchstens zusätzliche $O(n^2)$ Operationen)

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Unit-Disk-Graph Erkennung

Problem: UDG-Erkennung

Entscheide, ob ein gegebener Graph $G = (V, E)$ ein Unit-Disk-Graph ist.

Satz

Unit-Disk-Graph-Erkennung ist NP-schwer.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Erinnerung: Reduktion

Beweis der NP-Schwere von $\mathcal{A}$ durch Reduktion

1. Nimm bekanntes NP-schweres Problem $\mathcal{B}$.
2. Zeige, dass es Abbildung von Instanzen von $\mathcal{B}$ auf Instanzen von $\mathcal{A}$ gibt, die
   - in Polynomialzeit berechenbar ist,
   - Ja/Nein-Instanzen auf Ja/Nein-Instanzen abbildet

- Dann muss unser Problem auch NP-schwer sein
  - Mit einem Algo für unser Problem plus Polynomialzeit kann man ein NP-schweres Problem lösen

  $\Rightarrow$ Dann kann man jedes NP-schwere Problem damit lösen

Erinnerung: Erfüllbarkeitsproblem

Karlsruher Institut für Technologie

3SATISFIABILITY

Es ist NP-schwer, zu entscheiden, ob eine Formel in Konjunktiver Normalform (KNF) erfüllbar ist, auch unter der Einschränkung, dass

- jede Klausel nur drei Literale enthält
- jedes Literal in maximal drei Klauseln erscheint
  (diese Forderung gehört nicht zum üblichen 3-SAT Problem!)

- Bsp: $(\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_4)$

Wie bilden wir eine solche Formel auf einen Graphen ab, der genau dann ein UDG ist, wenn die Formel erfüllbar ist?

Erinnerung: Erfüllbarkeitsproblem

3SATISFIABILITY

Es ist NP-schwer, zu entscheiden, ob eine Formel in Konjunktiver Normalform (KNF) erfüllbar ist, auch unter der Einschränkung, dass

- jede Klausel nur drei Literale enthält
- jedes Literal in maximal drei Klauseln erscheint
  (diese Forderung gehört nicht zum üblichen 3-SAT Problem!)

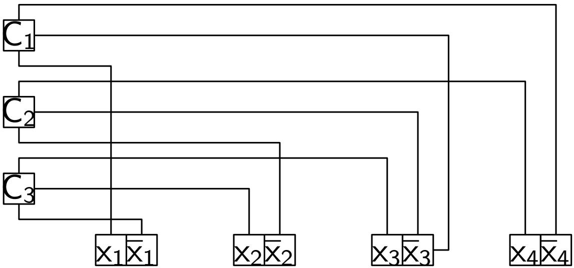
- Bsp: $(\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_4)$

Wie bilden wir eine solche Formel auf einen Graphen ab, der genau dann ein UDG ist, wenn die Formel erfüllbar ist?

Schritt 1: Orientierbarer Graph

Karlsruher Institut für Technologie

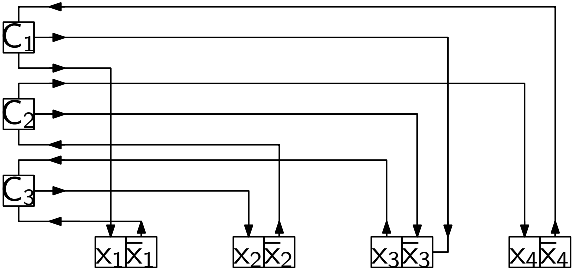
$(\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_4)$



- Formel ist erfüllbar $\Leftrightarrow$ es gibt Kantenrichtungen mit
  - für jede Klausel C gilt outdeg$(x) \geq 1$
  - für jede Variable x gilt: indeg$(x) = 0$ oder indeg$(\overline{x}) = 0$

Schritt 1: Orientierbarer Graph

$$(\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_4)$$

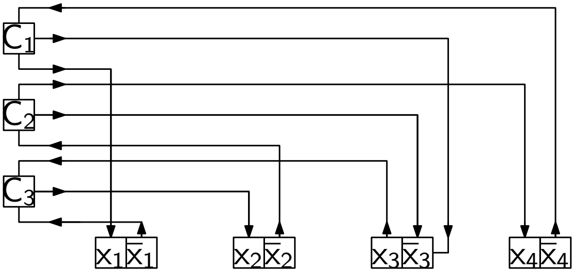


- Formel ist erfüllbar $\Leftrightarrow$ es gibt Kantenrichtungen mit
  - für jede Klausel C gilt outdeg$(x) \geq 1$
  - für jede Variable x gilt: indeg$(x) = 0$ oder indeg$(\overline{x}) = 0$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Schritt 2: Gadgets

$(\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee \overline{x}_3 \vee \overline{x}_4)$



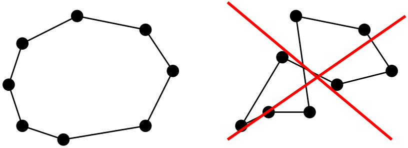
- Klassischer Gadgetbeweis: Baue Struktur nach aus
  - Variablen, Klauseln, Drähten und Kreuzungen
  - so dass gültige Einbettungen genau gültigen Orientierungen entsprechen

Einbettungen von Kreisen

Karlsruher Institut für Technologie

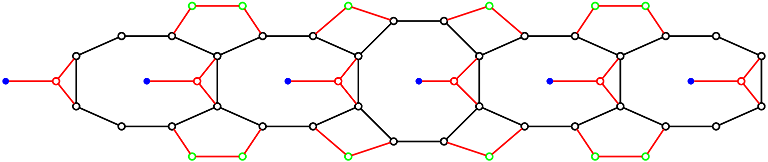
Lemma

Enthält ein Graph einen knoteninduzierten Kreis, muss der in jeder Einbettung als Unit-Disk-Graph planar eingebettet werden.

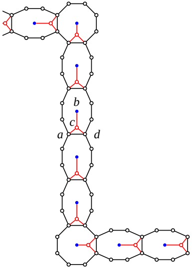


- knoteninduzierter Kreis: Menge von Knoten, die als Kreis verbunden sind *(keine weiteren Kanten!)*
- sollte uns bekannt vorkommen!

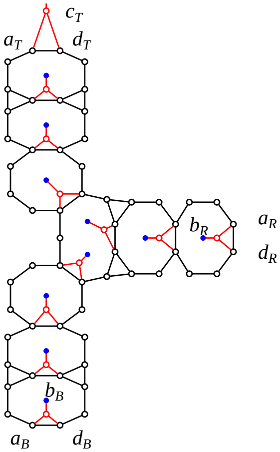
Schritt 2.1: Drähte



- Kreise als Käfige und Stabilisatoren
  - in Käfigen nur sehr begrenzt Platz!
  - je zwei Käfige durch zwei adjazente Gelenkknoten verbunden
  - Stabilisatoren auf folgenden Folien weggelassen
- jeder Käfig kann nur einen blauen Knoten aufnehmen
  - zeigt Richtung des Drahtes an

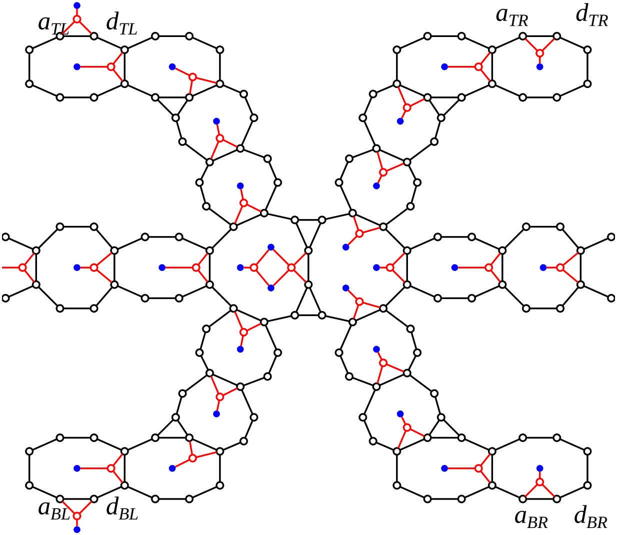


Schritt 2.2: Klauseln

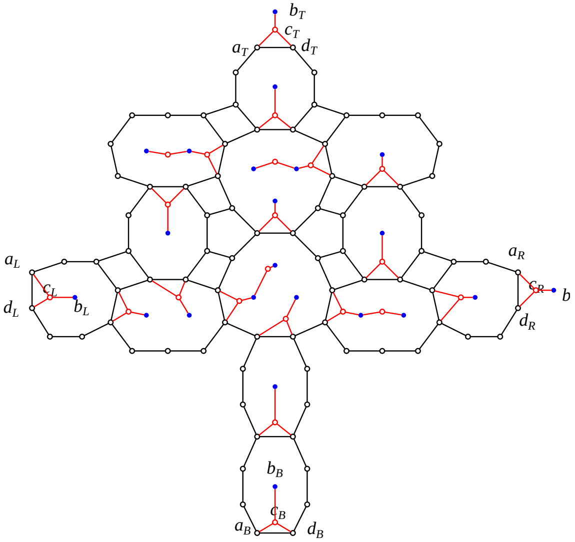


- Im wesentlichen ein Käfig, der 2 blaue Knoten aufnehmen kann
  $\Rightarrow$ Mind. einer der 3 ausgehenden Drähte ist auswärts gerichtet!

Schritt 2.3: Variablen



Schritt 2.4: Kreuzungen



Schwere UDG-Erkennung

Satz

Unit-Disk-Graph-Erkennung ist NP-schwer.

- Zu 3SAT-Formel konstruiere Graphen G wie beschrieben
- Eine erfüllende Belegung der Variablen verrät uns eine eine Einbettung als UDG
  - Knotenpositionen übernehmen wir dann aus unseren Zeichnungen der Gadgets
  - nur die Einbettung der roten Elemente passen wir an
- Eine Einbettung als UDG verrät uns eine erfüllende Belegung der Formel
  - Die exakten Knotenpositionen sind dann nicht wichtig
  - kombinatorische Einbettung der roten Elemente reicht!

Unit-Disk-Graph-Approximation

Wenn es schwer ist, einen Graphen als UDG einzubetten, kommt man dann wenigstens „dicht“ heran? Definiere *Qualität* einer Einbettung $\mathbf{p}$ als

$$q(\mathbf{p}) := \frac{\max_{\{u,v\} \in E} \|\mathbf{p}(u) - \mathbf{p}(v)\|}{\min_{\{u,v\} \notin E} \|\mathbf{p}(u) - \mathbf{p}(v)\|}$$

- $q(\mathbf{p}) \leq 1$: Einbettung belegt UDG, sonst nur $1/q(\mathbf{p})$-Quasi-Unit-Disk-Graph!

Man kann den Beweis noch so „aufbohren“, dass er folgende Reduktion liefert:

- Formel erfüllbar $\Rightarrow$ Graph einbettbar mit $q < 1$
- Formel nicht erfüllbar $\Rightarrow$ Graph nicht einbettbar mit $q < \sqrt{2}$.

Was genau bedeutet das?

- Es ist NP-schwer, einen Unit-Disk-Graphen zu erkennen
  - also gibt es keinen Polynomialzeitalgorithmus, der UDG entsprechend einbettet.

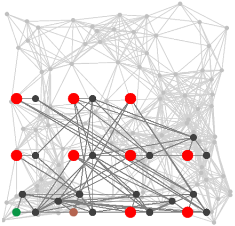
- Es ist auch schwer, zu erkennen, ob man einen Graphen mit Qualität $< \sqrt{2}$ einbetten kann
  - Also sind auch $1/\sqrt{2}$-Quasi-Unit-Disk-Graphen schwer zu erkennen
  - *Damit ist auch UDG-Approximation schwer!*

Beste bekannte Approximation

Satz (ohne Beweis)

Es gibt einen Algorithmus mit polynomieller Laufzeit, der mit hoher Wahrscheinlichkeit eine Einbettung mit Qualität in $O(\log^{2.5} n \sqrt{\log \log n})$ berechnet.

- Mit hoher W'keit: $p > 1 - 1/n$
- Ganz grobe Skizze:
  1. LP-Lösung liefert echte Metrik auf Knoten (Definitheit, Dreiecksungleichung)
  2. Knoten werden geschickt in $\mathbb{R}^n$ eingebettet
  3. Einbettung wird zufällig auf den $\mathbb{R}^2$ projiziert
  4. Ergebnis wird nach festen Regeln verfeinert (z.B. Erzwingen von Minimaldistanzen)
- Gleich mehrere schwere Geschütze!



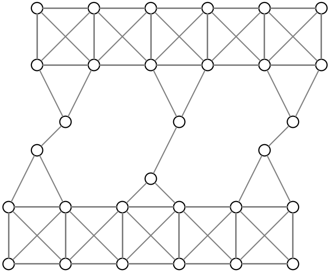
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Noch ein schweres Problem

Karlsruher Institut für Technologie

Satz (ohne Beweis)

Es ist NP-schwer, zu erkennen, ob man einen Graphen mit vorgegebenen Kantenlängen einbetten kann. Das gilt auch noch, wenn man sich auf UDGs einschränkt.



Weitere Ergebnissen

- Wenn man Richtungen und Entfernungen kennt, ist Lokalisierung leicht (klar)
  - schon beliebig kleine Fehler machen das Problem schwer
  - es macht keinen Unterschied, ob man sich auf UDGs einschränkt
- Wenn man nur Richtungen kennt, kann man in Polynomialzeit eine entsprechende Einbettung finden
  - wenn man aber nach einer entsprechenden UDG-Einbettung fragt, wird's schwer!

Das Finden plausibler Koordinaten ist schon in den meisten idealisierten Fällen schwer!

- *Heuristiken* für dichte Graphen gehen davon aus, dass plausible Lösungen „ziemlich“ eindeutig sind!

Weitere Ergebnissen

- Wenn man Richtungen und Entfernungen kennt, ist Lokalisierung leicht (klar)
  - schon beliebig kleine Fehler machen das Problem schwer
  - es macht keinen Unterschied, ob man sich auf UDGs einschränkt
- Wenn man nur Richtungen kennt, kann man in Polynomialzeit eine entsprechende Einbettung finden
  - wenn man aber nach einer entsprechenden UDG-Einbettung fragt, wird's schwer!

Das Finden plausibler Koordinaten ist schon in den meisten idealisierten Fällen schwer!

- *Heuristiken* für dichte Graphen gehen davon aus, dass plausible Lösungen „ziemlich" eindeutig sind!

Weitere Ergebnissen

- Wenn man Richtungen und Entfernungen kennt, ist Lokalisierung leicht (klar)
  - schon beliebig kleine Fehler machen das Problem schwer
  - es macht keinen Unterschied, ob man sich auf UDGs einschränkt
- Wenn man nur Richtungen kennt, kann man in Polynomialzeit eine entsprechende Einbettung finden
  - wenn man aber nach einer entsprechenden UDG-Einbettung fragt, wird's schwer!

Das Finden plausibler Koordinaten ist schon in den meisten idealisierten Fällen schwer!

- *Heuristiken* für dichte Graphen gehen davon aus, dass plausible Lösungen „ziemlich“ eindeutig sind!

Eine Heuristik: AFL

(Anchor-Free Localization)

Grundidee

Kräftebasierte Verfahren minimieren lokalen Stress iterativ

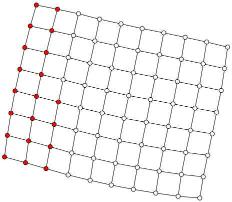
$$\sum_{\{u,v \in E\}} (\|u,v\| - \ell_{uv})^2$$

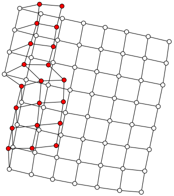
Problem

Lokale Optima können Überlappungen aufweisen!

Lösung?

Finde überlappungsfreie Initiallösung!





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

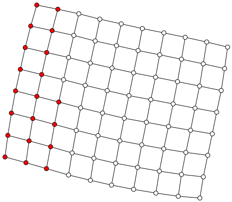
Eine Heuristik: AFL

(Anchor-Free Localization)

Grundidee

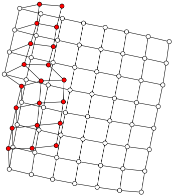
Kräftebasierte Verfahren minimieren lokalen Stress iterativ

$$\sum_{\{u,v \in E\}} (\|u,v\| - \ell_{uv})^2$$



Problem

Lokale Optima können Überlappungen aufweisen!



Lösung?

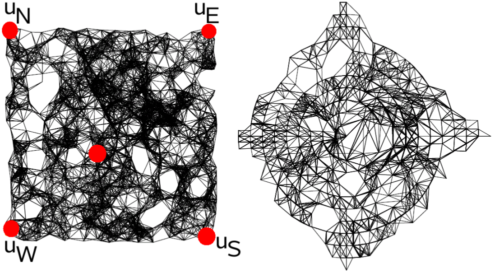
Finde überlappungsfreie Initiallösung!

Eine Heuristik: AFL (Initiallösung)

Karlsruher Institut für Technologie

1. Finde vier extremale und einen zentralen Knoten
   - U_N, U_W, U_S, U_E, U_C
2. Bestimme Hop-Entfernungen zu diesen Knoten
3. Bette Knoten anhand von Polarkoordinaten ein

$$\rho_v = d_G(v, u_C) \quad \theta_v = \arctan \frac{d_G(v, U_N) - d_G(v, U_S)}{d_G(v, U_W) - d_G(v, U_E)}$$

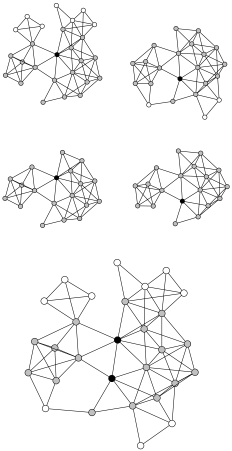


Funktioniert nicht so gut, wenn das Gebiet komplexer wird!

Weitere heuristische Ideen

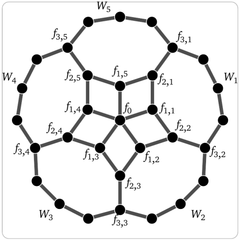
- Dichte Netze erlauben gute lokale Lösungen
  - viele lokale Lösungen zu berechnen skaliert gut!
  - setze Lösungen iterativ oder hierarchisch zusammen

- Erkennen von inneren Knoten und Randknoten
  - Ausnutzen von lokalen Strukturen
  - Ausnutzen von statistischen Eigenschaften



Weitere heuristische Ideen

- Dichte Netze erlauben gute lokale Lösungen
  - viele lokale Lösungen zu berechnen skaliert gut!
  - setze Lösungen iterativ oder hierarchisch zusammen
- Erkennen von inneren Knoten und Randknoten
  - Ausnutzen von lokalen Strukturen
  - Ausnutzen von statistischen Eigenschaften



Zusammenfassung

- Ankerbasierte Lokalisierung
  - fast ausschließlich Schätztheorie oder Heuristiken
  - kaum algorithmische Analyse (wegen unklarer Parameter?)
  - *Ausnahme:* Hop/HS zu eindimensionaler Lokalisierung

- Ankerfreie Lokalisierung
  - klarere algorithmische Probleme
  - vor allem Negativergebnisse
  - Heuristiken für dichte Graphen, aber ohne Garantien

Literatur

1. R. O'Dell, R. Wattenhofer: *Theoretical aspects of connectivity-based multi-hop positioning*. In: Theoretical Computer Science 344:1, pp. 47-68, 2005
2. H. Breu, D.G. Kirkpatrick: *Unit Disk Graph Recognition is NP-Hard*. In: Computational Geometry. Theory and Applications 9, 1993
3. F. Kuhn, T. Moscibroda, R. Wattenhofer: *Unit Disk Graph Approximation*. In: ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), 2004