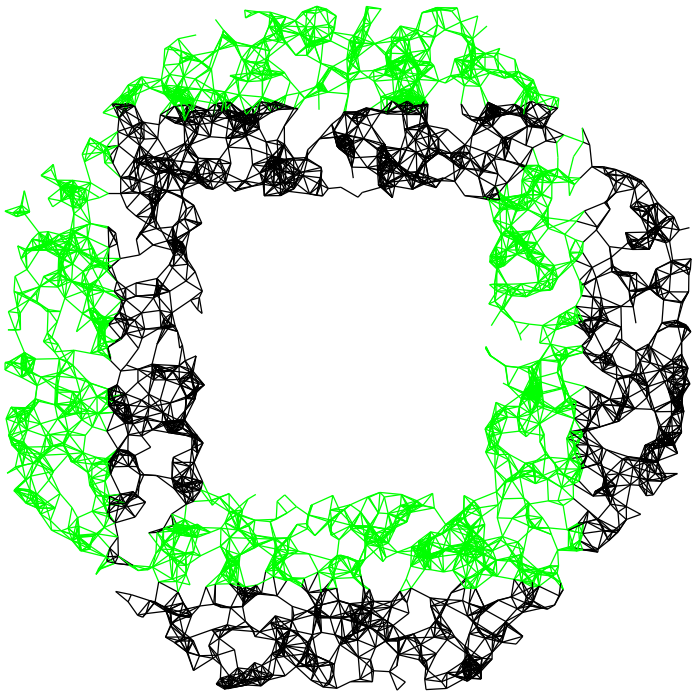
Richtungsbasierte Lokalisierung von Sensornetzwerken

Bastian Katz

Diplomarbeit



Betreut durch:
Prof. Dr. Dorothea Wagner
Dipl.-Math. Marco Gaertler

Institut für Theoretische Informatik
Universität Karlsruhe (TH)

Danksagung

Ich möchte meiner Betreuerin, Prof. Dr. Dorothea Wagner, herzlich danken für die Möglichkeit, mit dieser Arbeit in dieses interessante Forschungsgebiet einsteigen zu können, und allen Mitarbeitern ihres Lehrstuhls und der Gruppe von Dr. Alexander Wolff für die offene Atmosphäre. Ganz besonders bedanke ich mich bei meinem Betreuer Marco Gaertler, der mich während der Themensuche und dieser Arbeit geduldig begleitete, mir viel Zeit opferte und bei allen Fragen helfend zur Verfügung stand.

Unendlich dankbar bin ich meiner Freundin Isabelle Jellen dafür, dass sie mich in der ganzen Zeit ertrug und mir so vieles abnahm, während ich diese Arbeit schrieb. Sie half mir an unzähligen Stellen durch ihr aufmerksames Zuhören, meine Gedanken zu sortieren, und baute mich auf, wenn mich der Mut verließ.

Ich widme diese Arbeit meinen Eltern, Hertje und Michael Katz, auf deren Unterstützung ich mein ganzes Leben fest vertrauen konnte.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 30. Januar 2006

Inhaltsverzeichnis

1. Einleitung

Der Fortschritt in der Entwicklung von immer höher integrierter Hardware und drahtlosen Kommunikationsverfahren hat in den letzten Jahren einen verhältnismäßig neuartigen Ansatz zur sensorischen Überwachung von Umweltbedingungen oder räumlichen Phänomenen geschaffen. Es ist möglich geworden, kleine und vor allem kostengünstige Geräte zu entwickeln, die Sensorik, begrenzte Rechenleistung und die Möglichkeit zur drahtlosen Kommunikation in sich vereinen und es so ermöglichen, engmaschige Sensornetzwerke durch das Ausbringen von Hunderten oder bis zu Zehntausenden von Sensoreinheiten ohne vorher festgelegte Infrastruktur zu schaffen.

Eines der grundlegenden Probleme, die ein Netzwerk von solchen autonomen Sensorknoten zu bewältigen hat, ist die Rekonstruktion der Netztopologie; zum einen erleichtert eine geometrische Vorstellung des Netzwerkes andere Aufgaben, wie zum Beispiel das Routing von Anfragen und Meldungen über auftretende Ereignisse, zum anderen lassen sich kaum Szenarien vorstellen, die für Messungen *keine* Kenntnis des Ortes voraussetzen. Gerade Kleinstsensoren, die in großer Stückzahl und mit niedrigen Kosten ein dichtes Netz aufspannen sollen, können zur Lösung dieses Problems aber nur sehr wenig beisteuern – zum Beispiel lässt sich kaum jeder Sensor mit GPS ausrüsten. Trotzdem gibt es eine Vielzahl von Möglichkeiten, welche Informationen über die Topologie Sensoren ermitteln können. Gemeinsam ist allen, dass das Problem, die Topologie zu rekonstruieren, sehr eng mit *Realisierungsproblemen* aus der Graphentheorie verwandt ist. In der Praxis werden in der Mehrzahl Entferungsmessungen zwischen den Knoten verwendet; diese Entscheidung ist vor allem in der leichteren hardwareseitigen Umsetzung begründet. Sie ist aber keineswegs intuitiv: Der Mensch orientiert sich in seiner Umwelt sehr viel stärker durch das Erfassen von Richtungen zu anderen Objekten als durch Entfernungsmessungen. Die menschliche Sensorik ist auch viel präziser im Schätzen von Richtungen. Das betrifft sowohl die Augen, die mit einer sehr hohen Winkelauflösung Objekte in ihrer Richtung sehr gut orten, die Entfernung aber nur bei sehr nahen Objekten präzise schätzen kann, als auch das Gehör, mit dem man zwar leidlich die Richtung eines Geräusches abschätzen kann, aber kaum dessen Entfernung, selbst wenn man wüsste, wie laut die Schallquelle war. Auch in großem Maßstab setzte der Mensch zur Ortsbestimmung, z. B. in der Navigation, traditionell auf Richtungsmessungen.

Diese beiden Varianten sind aber keineswegs ebenbürtig; von der notwendigen Sensorik einmal abgesehen ist die Richtungsmessung aus theoretischer Sicht der Entfernungsmessung zum Zwecke der Topologierekonstruktion überlegen. Diese Tatsache ist in bisherige Arbeiten zur Lokalisation von Sensornetzwerken allerdings kaum eingeflossen,

auch wenn (z. B. in [BGJ05]) die Idee der richtungsbasierten Lokalisierung durchaus aufgegriffen wurde.

Struktur dieser Arbeit

Kapitel 2 Im folgenden Kapitel werde ich zunächst eine sehr kurze Übersicht über Ad-hoc-Netzwerke, ihre Einsatzgebiete und verwendete Techniken geben. Von besonderem Interesse werden hierbei die Randbedingungen sein, die es beim Algorithmenentwurf zu bedenken gibt, und die Hilfsmittel, derer sich Sensoren bedienen können, um Informationen über die Netzwerktopologie zu ermitteln.

Kapitel 3 Um Ad-hoc-Netzwerke und die zu untersuchenden Probleme modellieren zu können, werden in diesem Kapitel einige Grundlagen und Vereinbarungen zur Graphentheorie aufgeführt. Darüber hinaus werden einige algorithmische Grundlagen wiederholt, die in späteren Kapiteln vorausgesetzt werden.

Kapitel 4 Da das Problem der Sensorlokalisierung bereits von vielen verschiedenen Seiten angegangen wurde, wird in diesem Kapitel eine Übersicht über bestehende Modellierungen gegeben, zusammen mit bekannten Ergebnissen und Herangehensweisen.

Kapitel 5 In der entfernungsbasierten Sensorlokalisierung finden sich vielfach Verweise auf die *Rigidity Theory*, die aus der Untersuchung der Starrheit von Gerüsten heraus auch Aussagen über die Eindeutigkeit von Einbettungen unter bestimmten Nebenbedingungen hervorgebracht hat. Dabei ist es bisher kaum mehr als eine Randbemerkung, wieviel schwächer die Anforderungen zur Eindeutigkeit und wie fundamental einfacher das Auffinden verträglicher Einbettungen bei bekannten Kantenrichtungen als bei bekannten Kantenlängen sind. Dieses Kapitel beleuchtet diese Ergebnisse zusammen mit einigen vorbereitenden Aussagen.

Kapitel 6 Sensornetzwerke mit bekannten Kommunikationsrichtungen lassen schon bei verhältnismäßig geringer Dichte eine exakte Rekonstruktion *weiter Teile* des Netzes zu. In diesem Kapitel wird ein Algorithmus beschrieben, der *maximale* Teilnetze dieser Art schnell identifiziert und eine Rekonstruktion berechnet. Zusätzlich wird eine Rekonstruktion des gesamten Sensornetzwerkes berechnet, die die tatsächliche Topologie in realistischen Szenarien sehr gut annähert.

Kapitel 7 In diesem Kapitel wird die Vorgehensweise des vorangegangenen Kapitels auf Fälle übertragen, in denen die Eingabe fehlerbehaftet ist, d. h. in denen die Richtungsmessungen bestimmten Fehlern unterworfen sind. Es zeigt sich, dass sich unter Ausnutzung von Redundanz und lokaler Optimierung das bekannte Verfahren auch für diesen Fall prinzipiell gut eignet.

Kapitel 8 Alle vorgestellten Verfahren sind für diese Arbeit implementiert worden. Dieses Kapitel führt verwendete Software und einige Randbemerkungen zur Implementierung auf.

Kapitel 9 Abschließend werden die wesentlichen Ergebnisse dieser Arbeit noch einmal kurz zusammengefasst. Viele Punkte, die im Rahmen dieser Arbeit angeschnitten wurden, konnten nicht erschöpfend behandelt werden oder legen eine weitere Untersuchung nahe. Dieses Kapitel führt solche Punkte auf und fasst erste Ansätze zusammen.

Anhang A Im Anhang dieser Arbeit befinden zum einen noch ergänzende Tests, zum anderen werde Nebenrechnungen ohne tiefere Bedeutung für den Fluss der Arbeit aufgeführt.

2. Ad-hoc-Sensornetzwerke

Wie eingangs erwähnt, hat die Hardwareentwicklung durch ihre immer weiter gehende Miniaturisierung drahtlos kommunizierende Sensoreinheiten möglich gemacht, die die Möglichkeiten zur Überwachung der Umwelt revolutionieren, sei es bei der Beobachtung von waldbrandgefährdeten Gebieten, sei es im militärischen Einsatz zur Beobachtung von Truppenbewegungen, aber auch in viel kleinerem Maßstab bis hin zum sogenannten *Smart Dust* zur unauffälligen Überwachung von Räumen mit nur wenige Millimeter großen Einheiten. Verwendete Sensoreinheiten bestehen typischerweise aus Sensorhardware, Prozessor, Speicher, einer Energieversorgung und einem drahtlosen Kommunikationssystem. Einen knappen Einstieg bietet [TM03]. Viele dieser Sensoren sollen zusammen folgende Aufgaben erledigen:

Selbstorganisation Die Knoten müssen ihre Infrastruktur selbst verwalten, das heißt insbesondere, dass sie die für das Routen von Informationen und/oder Anfragen nötige Infrastruktur selbst aufbauen müssen.

Datenaggregation und -verarbeitung Es sollen nicht alle anfallenden Daten (die jeder der Knoten zu jedem Zeitpunkt misst) permanent durch das Netz propagiert und überall zur Verfügung gestellt werden; vielmehr sollen die Sensoren Daten sammeln, vorverarbeiten, und gegebenenfalls bewerten und filtern. So lassen sich gezielt echte Ereignisse erkennen und diese übermitteln bzw. von außen kommende Anfragen mit aussagekräftigen Daten beantworten.

Dabei ergeben sich eine ganze Reihe von Randbedingungen, die vor allem im Entwurf von Algorithmen beachtet werden sollten:

Beschränkte lokale Kapazitäten Einzelne Sensorknoten sind in ihren Fähigkeiten sehr beschränkt. Das betrifft alle Aspekte, also die geringe Rechenleistung, einen sehr kleinen Speicher und – vor allem wegen eines sehr eingeschränkten Energiebudgets und der angestrebten geringen Größe – nur eine geringe Reichweite der Kommunikation. Es ist deshalb „teuer“ oder unmöglich, große Berechnungen in einzelnen Knoten stattfinden zu lassen oder über weite Strecken, also über viele Zwischenstationen, zu kommunizieren.

Hohe Ausfallwahrscheinlichkeit Sensorknoten sind anders als große Rechner sehr viel anfälliger gegenüber Ausfällen, sei es herstellungsbedingt oder aufgrund zu befürchtender physischer Einwirkung. Sensornetzwerke sollten deshalb möglichst redundant arbeiten und solche Ausfälle verkraften.

Veränderung der Topologie Abhängig vom Einsatzgebiet kann sich die Topologie des Netzes permanent ändern, weil sich Sensoreinheiten bewegen.

2.1. Messen topologischer Informationen

In dieser Arbeit ist besonders die Fähigkeit der Sensornetzwerke zur Lokalisation, zur Rekonstruktion der räumlichen Struktur des Netzes von Interesse. Diese Aufgabe lässt sich erleichtern, wenn man Sensorknoten verwendet, die neben der eigentlichen Nutzsensorik (Temperaturfühler o. ä.) auch Sensorik zur Unterstützung dieser Aufgabe mitbringen. Diese lassen sich nach der Art der gelieferten Information gliedern (siehe auch [NN03b], [SHS01]):

Absolute Positionierung Es gibt zwei Wege, Knoten in die Lage zu versetzen, ihre absolute Position zu bestimmen; der erste besteht darin, die Position beim Ausbringen einzuprogrammieren und zu garantieren, dass sich Sensoren nicht bewegen. Der zweite ist ein Ausrüsten der Sensoreinheiten mit GPS-Empfängern. Beide Varianten kommen höchstens für einige wenige Sensoreinheiten in Betracht, denn das Einprogrammieren der Position ist für Tausende von Knoten schlicht nicht zu leisten, die Ausrüstung mit GPS-Empfängern kaum zu bezahlen. Für Überwachung sehr kleiner Gebiete mit Kleinstsensoren lassen sich GPS-Empfänger weder mit der Größenvorstellung für Sensoreinheiten noch mit der notwendigen Genauigkeit vereinen. Gibt es in einem Sensornetzwerk Knoten mit bekannter Position, spricht man auch von Ankerknoten (*anchors*).

Absolute Orientierung Es ist verhältnismäßig leicht, Knoten mit einem digitalen Kompass zu versehen. Solche Kompasse erreichen leicht eine Auflösung von unter einem Grad.

Entfernungsmessung Zur Ermittelung von Entfernungen miteinander kommunizierender Sensoren gibt es eine Reihe von Techniken, die auf unterschiedlichen Prinzipien basieren:

- Durch Messung der Signalstärke (RSSI, *Received Signal Strength Indicator*) lässt sich bei einheitlicher oder übermittelter Sendestärke die Entferung abschätzen.
- Durch Zeitmessung lassen sich Entfernungen abschätzen, wenn Knoten entweder *zwei* Signale mit verschiedenen Laufzeiten, z. B. akustische und elektromagnetische Impulse, gleichzeitig abschicken, die von Empfängern mit von der Entfernung abhängigen Laufzeiten empfangen werden (ToA, *Time of Arrival*) oder, indem ein Signale vom jeweiligen Empfänger „reflektiert“ werden (TDoA, *Time Difference of Arrival*).

Von diesen Techniken ist RSSI die am längsten eingesetzte, zumal sie wenig Anforderungen stellt; einen Vergleich von RSSI und ToA bzw. verschiedener Implementierungen von TDoA geben [SHS01] und [SGA+03].

Richtungsmessungen Richtungen zu umliegenden Knoten lassen sich zum Beispiel durch mehrere Empfänger auf Grundlage von Phasendifferenzen ermitteln; in [PMBT01] wird eine Implementierung mit fünf Ultraschallempfängern beschrieben, die durch Phasenunterschiede Laufzeitverschiebungen messen und so Richtungen zur Schallquelle mit Abweichungen von wenigen Grad bestimmen. Eine denkbare Alternative hierzu wären gerichtete Antennen, in jedem Fall werden diese Verfahren als AoA, *Angle of Arrival* zusammengefasst.

Diese Arbeit beschäftigt sich im Kern vor allem mit Richtungsmessungen ohne Ankerknoten mit exakten Positionen. Dabei ist außerhalb der rein theoretischen Betrachtung wie bei allen Verfahren nicht zu vernachlässigen, dass alle Messungen fehlerbehaftet sind. Die zwei wichtigsten Fehlerquellen sind die folgenden:

Externe Störungen Die Peilungen zu anderen Sensoren sowie das Zustandekommen von Kommunikation ist leicht zu beeinflussen durch das Relief des Gebietes, in dem die Sensoren ausgebracht sind, sowie durch Objekte, die die Signale behindern, reflektieren o. ä.

Interne Ungenauigkeiten Je nach Messverfahren ist die *Auflösung* erfasster Größen begrenzt, und selbst bei idealisierten Messverfahren ist spätestens die Weiterverarbeitung der Signale in digitaler Form der Diskretisierung unterworfen.

Die Stärke der Fehler hängt dabei massiv von der Umgebung und der verwendeten Technik ab, die wiederum auch abhängt vom Einsatzzweck, zum Beispiel der geforderten Reichweite der Knoten.

3. Vorbereitendes

Es ist mehr als naheliegend, Sensornetzwerke durch Graphen zu abstrahieren. Diese Graphen zeichnet aus, dass sie eine *natürliche* Einbettung in den $\mathbb{R}^2$ oder $\mathbb{R}^3$ haben, und das Lokalisierungsproblem übersetzt sich in das Problem, diese Einbettung bis auf unumgängliche Freiheitsgrade zu rekonstruieren. Im folgenden werde ich einige grundlegende Definitionen und Notationen vorstellen. Sie lehnen sich an [Jun99] an.

3.1. Graphen

Ein *einfacher, ungerichter* Graph $G = (V, E)$ besteht aus einer nichtleeren, endlichen Menge V, den *Knoten*, und einer Menge von ungeordneten Knotenpaaren $E \subseteq \binom{V}{2} := \{\{u, v\} \in V \times V \mid u \neq v\}$, den *Kanten*. Ein Graph heißt *vollständig*, wenn $E = \binom{V}{2}$.

Wenn nicht anders angegeben, gehe ich davon aus, dass $V = \{1, 2, \ldots, n\}$, und an einigen Stellen werde ich eine Kante $\{i, j\} \in E$ mit $i < j$ bezeichnen als $(i, j) \in E$ und so den Kanten willkürlich eine Richtung zuweisen. Sofern der Zusammenhang klar ist, wird $n := |V|$ und $m := |E|$ verwendet. Es bezeichne K_n den vollständigen Graphen mit n Knoten.

Definition 1 (Nachbarschaft, Grad) *Zu einem Graphen $G = (V, E)$ und einem Knoten $v \in V$ bezeichne* $\mathrm{Nb}(v) := \{u \in V \mid \{u, v\} \in E\}$ *die* (direkte) Nachbarschaft *und* $\mathrm{Nb}^2(v) := \{u \in V \setminus (\mathrm{Nb}(v) \cup v) \mid \exists w \in \mathrm{Nb}(v) : u \in \mathrm{Nb}(w)\}$ *die* indirekte Nachbarschaft*. Es bezeichne $deg(v) := |\mathrm{Nb}(v)|$ den Grad eines Knotens und $\Delta(G) := \max_{v \in V} deg(v)$ den maximalen Knotengrad in einem Graphen (kurz auch Δ, wenn der Graph selbstverständlich ist).*

Definition 2 (Subgraphen) *Ein Graph $S = (V_S, E_S)$ heißt Subgraph eines Graphen $G = (V, E)$, wenn $V_S \subseteq V$ und $E_S \subseteq E$. Knoten- bzw. Kantenteilmengen $V' \subset V$ bzw. $E' \subset E$ induzieren Subgraphen durch*

$$\begin{aligned} G[V'] &:= (V', E[V'] := \{\{v, w\} \in E \mid v, w \in V'\}) \\ G[E'] &:= (V[E'] := \{v \in V \mid \exists \{v, w\} \in E\}, E') \quad . \end{aligned}$$

Ein Subgraph S eines Graphen G heißt dementsprechend knoteninduziert *(kanteninduziert), wenn es eine Knotenteilmenge V' (Kantenteilmenge E') gibt, so dass $S = G[V']$ ($S = G[E']$).*

Man kann sich leicht überlegen, dass beides nicht selbstverständlich ist, ein Subgraph allerdings nur dann nicht kanteninduziert ist, wenn er einen Knoten mit dem Grad 0 enthält. Da ich im weiteren, sofern nicht anders erwähnt, davon ausgehe, dass betrachtete Graphen und Subgraphen zusammenhängend sind (s.u.), werden nur die knoteninduzierten Subgraphen speziell interessieren. Ist ein Subgraph mit k Knoten vollständig, spricht man von einer Clique der Größe k (oder auch k-Clique).

Definition 3 (Bipartite Graphen) *Lässt sich die Knotenmenge V eines Graphen $G = (V, E)$ so als $V = L \dot\cup R$ aufteilen, dass jede Kante genau einen Knoten aus L mit einem Knoten aus R verbindet ($E \subseteq \{\{l, r\} \mid l \in L, r \in R\}$), spricht man von* bipartiten Graphen

Definition 4 (Pfad, Zusammenhang) *Ein Pfad ist eine Knotenfolge $(v_1, \ldots, v_k)$ so, dass für $i = 1, \ldots, k-1$ gilt, dass $\{v_i, v_{i+1}\} \in E$. Ein solcher Pfad* verbindet v_1 *und v_k. Er heißt* einfach, *wenn er nur paarweise verschiedene Knoten enthält. Ein Graph heißt* zusammenhängend, *wenn es für alle $v \neq u \in V$ einen Pfad gibt, der v und u verbindet. Er heißt d-fach knotenzusammenhängend, wenn nach Wegnahme jeder Knotenteilmenge V' mit $|V'| = d$ der Restgraph zusammenhängend ist:*

$$\forall V' \subseteq V : |V'| \leq d \Longrightarrow G[V \setminus V'] \text{ ist zusammenhängend}$$

Ein Spannbaum *ist eine minimale Kantenteilmenge $T \subseteq E$ derart, dass $G[T]$ zusammenhängend ist.*

3.2. Einbettungen

Eine Einbettung eines Graphen $G = (V, E)$ ist eine Zuordnung $\mathbf{p} : V \to \mathbb{R}^d$. Sie weist jedem Knoten eine *Position* $\mathbf{p}(i) = (\mathbf{p}(i)_1, \ldots, \mathbf{p}(i)_d) \in \mathbb{R}^d$ zu. Verkürzt schreibe ich auch $\mathbf{p}_i := \mathbf{p}(i)$, sofern die einzelnen Koordinaten nicht interessieren, und $\mathbf{p}(i)_x = \mathbf{p}(i)_1$, $\mathbf{p}(i)_y = \mathbf{p}(i)_2$ und $\mathbf{p}(i)_z = \mathbf{p}(i)_3$, wenn $d \leq 3$. Für $d = 2$ bezeichne $\mathbf{p}^\perp$ eine um 90° gedrehte Einbettung, d. h. eine Einbettung, bei der $\mathbf{p}^\perp(i) = (-\mathbf{p}(i)_y, \mathbf{p}(i)_x)$ Eine Einbettung mit $\mathbf{p}_i = \mathbf{p}_j$ für alle $i, j \in V$ heißt trivial. Gegebenenfalls fasse ich eine Einbettung $\mathbf{p} : V \to \mathbb{R}^d$ als Punkt im $\mathbb{R}^{d \cdot |V|}$ auf als $\mathbf{p} \equiv (\mathbf{p}(1)_1, \ldots, \mathbf{p}(1)_d, \ldots \mathbf{p}(n)_1, \ldots, \mathbf{p}(n)_d)$.

Zwei Einbettungen $\mathbf{p}$ und $\mathbf{q}$ heißen *kongruent*, wenn es eine Isometrie T des $\mathbb{R}^d$ gibt mit $T(\mathbf{p}_i) = \mathbf{q}_i$ für alle $i \in V$. Sie heißen *ähnlich*, wenn es ein $c \in \mathbb{R}$ gibt, so dass $c\mathbf{p}$ und $\mathbf{q}$ kongruent sind.

Eine Graph $G = (V, E)$ zusammen mit einer Einbettung $\mathbf{p}$ wird als *Gerüst* $G(\mathbf{p})$ bezeichnet. Ist $d = 2$, spricht man von einem *ebenen* Gerüst. Soll eine ebene Einbettung zusätzlich auch noch den Knoten Richtungen zuweisen, notieren ich die Zuweisung von Richtungen an die Knoten als Ausrichtung $\mathbf{p}^\uparrow : V \to [0; 2\pi)$[1].

[1] Auch diese Ausrichtung ließe sich natürlich auf beliebige Dimensionen verallgemeinern; dafür, dass

Längenzuweisung Eine Längenzuweisung ist eine Zuweisung $\ell : E \to \mathbb{R}_0^+$ von Längen an die Kanten eines Graphen. Es heißt für einen Graphen $G = (V, E)$ eine Einbettung $\mathbf{p}$ mit einer Längenzuweisung ℓ verträglich, wenn $|\mathbf{p}_i - \mathbf{p}_j| = \ell(\{i, j\})$, darüber hinaus heißt eine Längenzuweisung ℓ *realisierbar* oder *konsistent*, wenn es eine verträgliche Einbettung gibt. Entsprechend induzieren ein Graph und eine Einbettung immer eine Längenzuweisung an die Kanten $\ell_{G,\mathbf{p}}$ mit $\ell_{G,\mathbf{p}}(i, j) := |\mathbf{p}_i - \mathbf{p}_j|$ Wir bezeichnen alle für G mit $\ell_{G,\mathbf{p}}$ verträglichen Einbettungen als $B(G, \mathbf{p})$; das sind genau alle Einbettungen, für die sich dieselben Kantenlängen ergeben wie für $\mathbf{p}$.

Richtungszuweisung Eine (globale) Richtungszuweisung ist eine Zuweisung $\alpha : E \to \mathbb{D} \cup \mathbf{0}$ (für die Einheitskugel $\mathbb{D} := \{\mathbf{x} \in \mathbb{R} \mid |x| = 1\}$) von Richtungen an die Kanten eines Graphen und es heißt für einen Graphen G eine Einbettung $\mathbf{p}$ mit einer Richtungszuweisung α *verträglich*, wenn für alle $(i, j) \in E$ gilt, dass $\mathbf{p}_j - \mathbf{p}_i = c_{ij} \cdot \alpha(i, j)$ für geeignete $c_{ij} \in \mathbb{R}^+$, und zu einer Richtungszuweisung α *parallel*, wenn nur gilt, dass $\mathbf{p}_j - \mathbf{p}_i = c_{ij} \cdot \alpha(i, j)$ für geeignete $c_{ij} \in \mathbb{R}$. Eine Richtungszuweisung heißt *realisierbar* oder *konsistent*, wenn es eine verträgliche Einbettung gibt. Analog zu den Längen induzieren Graph und Einbettung immer auch Kantenrichtungen

$$\forall (i, j) \in E : \quad \alpha_{G,\mathbf{p}}(i, j) := \begin{cases} \frac{\mathbf{p}_j - \mathbf{p}_i}{|\mathbf{p}_j - \mathbf{p}_i|} & : \text{ wenn } \mathbf{p}_j - \mathbf{p}_i \neq \mathbf{0} \\ \mathbf{0} & : \text{ sonst} \end{cases} ,$$

und es bezeichne alle für G mit $\alpha_{G,\mathbf{p}}$ verträglichen Einbettungen als $D(G, \mathbf{p})$, die parallelen Einbettungen als $P(G, \mathbf{p})$ (siehe Abbildung 3.1). Man beachte, dass nach diesen Definitionen eine Einbettung verträglich ist, wenn alle Kanten in die vorgegebene Richtung zeigen (und genau dann Länge 0 haben, wenn die vorgegebene Richtung $\mathbf{0}$ ist), aber parallel bereits, wenn jede Kante mit einer vorgegebenen Richtung $\mathbf{0}$ degeneriert ist und jede andere Kante entweder degeneriert oder parallel zur vorgegebenen Richtung. Offensichtlich sollte sein, dass $D(G, \mathbf{p}) \subsetneq P(G, \mathbf{p})$ für jede nichttriviale Einbettung $\mathbf{p}$ eines Graphen G.

Lokale Richtungszuweisung Eine Variante der Richtungszuweisung in der Ebene ist eine lokale Richtungszuweisung $\omega : E \to (\mathbb{D} \cup \mathbf{0})^2$, die jeder Kante eines Graphen $G = (V, E)$ zwei lokale Richtungen relativ zu einer Ausrichtung der Endknoten zuweist. Dabei bezeichne $\omega(i, j)$ die Richtung der Kante $\{i, j\}$ relativ zur Ausrichtung des Knotens i. Eine Einbettung mit Knotenausrichtung ist verträglich bzw. parallel, wenn für alle $(i, j) \in E$ gilt[2]

$$\begin{aligned} \mathbf{p}_j - \mathbf{p}_i &= c_{ij} \cdot R(\mathbf{p}_i^{\uparrow}) \cdot \omega(i, j) \\ \mathbf{p}_i - \mathbf{p}_j &= c_{ij} \cdot R(\mathbf{p}_j^{\uparrow}) \cdot \omega(j, i) \ . \end{aligned}$$

bis auf Randbemerkungen aber nur der ebene Fall benötigt wird, wäre das unverhältnismäßig viel Aufwand

[2]Hier und später bezeichne $R(\cdot)$ die zweidimensionale Rotationsmatrix

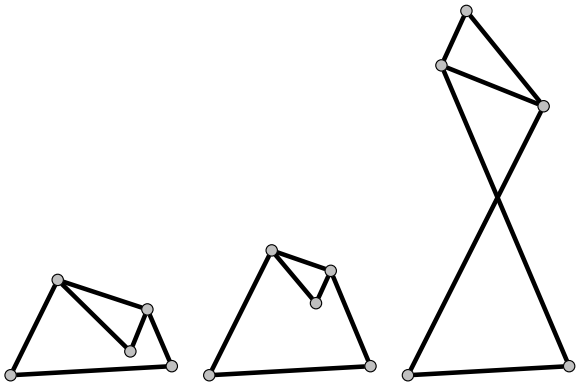


Abbildung 3.1.: Ein eingebetteter Graph (links) mit einer verträglichen Einbettung (mitte) und einer nur parallelen Einbettung (rechts)

Wieder bedeute Konsistenz, dass es eine verträgliche Einbettung gibt. Die durch eine Einbettung induzierte lokale Richtungszuweisung ist gegeben durch

$$\omega_{G,\mathbf{p}}(i,j) = \begin{cases} R(-\mathbf{p}_i^{\uparrow}) \cdot \frac{\mathbf{p}_j - \mathbf{p}_i}{|\mathbf{p}_j - \mathbf{p}_i|} & : \text{ wenn } \mathbf{p}_j - \mathbf{p}_i \neq \mathbf{0} \\ (\mathbf{0}, \mathbf{0}) & : \text{ sonst} \end{cases} .$$

Die Suche nach verträglichen Einbettungen lokaler Richtungszuweisungen unterscheidet sich nur marginal von der bezüglich globaler Richtungszuweisungen, sofern es einen Spannbaum aus Kanten mit $\omega(i,j) \neq \mathbf{0}$ gibt; zu gegebener lokaler Richtungszuweisung lässt sich ausgehend von einer willkürlichen Ausrichtung *eines* Knotens durch einfache Tiefensuche *jedem* Knoten eine Ausrichtung zuweisen, weil für die Baumkante $\{i,j\}$ und bekannter Ausrichtung von i oder j die Ausrichtung des anderen Knotens eindeutig bestimmt ist durch

$$\pi - \mathbf{p}_j^{\uparrow} + \arg\omega(j,i) = -\mathbf{p}_i^{\uparrow} + \arg\omega(i,j) \ . \tag{3.1}$$

Danach ist die lokale Winkelzuweisung entweder schnell als inkonsistent zu überführen, wenn für irgendeine Kante $\{i,j\} \in E$ Gleichung 3.1 *nicht* gilt, oder aber die Frage nach der Realisierbarkeit bzw. das Auffinden verträglicher Einbettungen reduziert sich auf den globalen Fall mit

$$\alpha(i,j) = R(\mathbf{p}_i^{\uparrow}) \cdot \omega(i,j) \text{ für alle } (i,j) \in E.$$

Der Sinn dieser lokalen Formulierung liegt zum einen darin, dass sie unter Umständen die Eingabe eines Lokalisierungsproblems sehr viel genauer abbildet – tatsächlich wird jede Verbindungsrichtung von beiden beteiligten Sensoren lokal gemessen – und dass zum anderen spätestens dann, wenn Richtungszuweisungen Fehlern unterworfen sind, der Unterschied sehr wohl zum Tragen kommt.

3.3. Datenstrukturen

Insbesondere bei der Analyse der Algorithmen werden einige grundlegende Kenntnisse üblicher Datenstrukturen vorausgesetzt. Sie können in [CLR90] nachgeschlagen werden. Drei wesentliche Überlegungen seien trotzdem hier vorangestellt.

Listen Sofern in den vorgestellten Algorithmen Listen verwendet werden, wird von einer Implementierung *doppelt verketteter Listen* ausgegangen (siehe Abbildung 3.2), in der Einträge aus Vorwärts- und Rückwärtszeigern und jeweils einer Referenz auf ein Nutzdatum besteht.

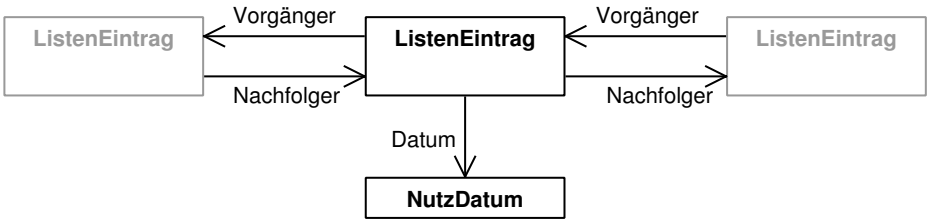


Abbildung 3.2.: Doppelt verkettete Liste

Dieser Listentyp ist eine der bekanntesten Datenstrukturen; der Aufwand für die grundlegenden Operationen, das Einfügen und das Entfernen von Elementen, wird üblicherweise als konstant angegeben. Das Entfernen eines *Nutzdatums* ohne Kenntnis des Listeneintrags ist allerdings in konstanter Zeit nicht möglich, weil dazu die Kenntnis des Listeneintrags mit Zugriff auf Vorwärts- und Rückwärtszeiger notwendig ist, sofern die Liste in der Größe nicht beschränkt ist und damit ein Traversieren in konstanter Zeit ermöglicht. Dieses Problem ist keineswegs trivial; in der Tat stellen gängige Implementierungen[3] solche Listen als Containertyp *ohne* die Möglichkeit, auf die Listeneinträge zuzugreifen. Dieses Problem lässt sich auf zwei Arten umgehen, zum einen durch eine Implementierung, in der der Nutzdatentyp mit dem Typ für die Listeneinträge zusammenfällt und somit selbst Vorwärts- und Rückwärtszeiger auf vorangehende und nachfolgende Nutzdaten speichert, oder, indem der Nutzdatentyp zusätzlich einen (Rück-)zeiger auf den Listeneintrag enthält, der es in der Liste repräsentiert – eine Listenimplementierung vorausgesetzt, die Zugriff auf die Listen*einträge* ermöglichen[4]. Abbildung 3.3 zeigt diese beiden Lösungswege.

Beide Varianten haben gemeinsam, dass Daten nur in vorher festgelegten Listen aufgeführt werden können, insbesondere nur in konstant vielen. Ohne darauf speziell hinzuweisen wurde für alle Algorithmen in den Kapiteln 6 und 7 darauf geachtet, dass

[3] zum Beispiel die aktuelle *Java 2 Platform 5.0*
[4] wie die verwendete Implementierung der yFiles, siehe Kapitel 8

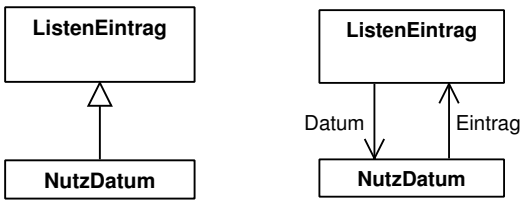


Abbildung 3.3.: Das gezielte Entfernen von Nutzdaten aus Listen in konstanter Zeit ist nur dann möglich, wenn die Typen für Listeneinträge und Nutzdaten zusammenfallen (links) oder sich gegenseitig referenzieren (rechts).

alle verwendeten Daten nur in konstant vielen, vorher bekannten Listen aufgeführt werden, die nicht ohnehin in der Länge beschränkt sind.

Graphen Zur Repräsentation von Graphen gibt es zwei wichtige Anmerkungen. Zum einen gehe ich im folgenden immer davon aus, dass es möglich ist, Daten zu Kanten und zu Knoten parasitär zu speichern, solange das nur konstant viele Daten betrifft. Das heißt, dass z. B. für den Zugriff auf Markierungen von Knoten kein zusätzlicher Aufwand anfällt. Zum anderen gibt es zwei grundverschiedene Arten, Graphen zu repräsentieren, die sich darin unterscheiden, wie sie die Adjazenzen der Knoten organisieren:

Matrixbasierte Graphenstrukturen Eine sehr einfache Datenstruktur besteht vor allem aus einer $n \times n$-Adjazenzmatrix mit 0/1-Einträgen, wobei der Eintrag in Zeile i und Spalte j genau dann 1 ist, wenn der repräsentierte Graph eine Kante $\{i, j\}$ enthält. Die Vorteile dieser Repräsentation liegen auf der Hand: Es kann jeweils in konstanter Zeit überprüft werden, ob eine Kante zwischen zwei Knoten existiert, genauso schnell können Kanten eingefügt oder gelöscht werden. Das Einfügen und Löschen von Knoten hingegen erfordert allein deshalb $\Omega(n)$ Schritte, weil dafür alle möglichen Kanten gelöscht werden müssen bzw. weil n Einträge initialisiert werden müssen, auch wenn man durch geschickte Implementierung verhindert, ständig die gesamte Matrix umkopieren zu müssen. Je dünner Graphen sind, desto mehr fallen noch andere Nachteile ins Gewicht: Der Speicherplatz ist auf $\Theta(n^2)$ festgelegt, auch für $m \in \mathcal{O}(n)$, und das Iterieren über alle zu einem Knoten inzidenten Kanten oder über die Nachbarn eines Knotens benötigt ein Durchlaufen von n Einträgen der Matrix, selbst wenn sich die Größe der Nachbarschaft enger beschränken lässt.

Listenbasierte Graphenstrukturen Wann immer im weiteren Graphen repräsentiert werden müssen, wird deshalb von einer *listenbasierten* Struktur ausgegangen, in der die zu einem Knoten inzidenten Kanten in einer Liste aufgeführt werden. Das hat

natürlich zur Folge, dass die Überprüfung, ob eine Kante existiert, bis zu $|\mathrm{Nb}(v)|$ Schritte dauern kann (das Löschen und Einfügen ohne vorherige Überprüfung, ob die Kante schon existiert, lässt sich bei oben beschriebenen Listen in konstanter Zeit durchführen). Die wesentlichen Vorteile liegen aber darin, dass in $\Theta(|\mathrm{Nb}(v)|)$ über die Nachbarschaft eines Knotens iteriert werden kann, und dass zu jedem Knoten auch nur ein Speicherbedarf in $\Theta(|\mathrm{Nb}(v)|)$ anfällt, also insgesamt ein Speicherbedarf in $\Theta(n+m)$.

4. Problemstellung, Modelle und Schranken

Wie angekündigt, werden im weiteren Sensornetzwerke stets als Graphen, d. h. Sensoren als Knoten und bestehende Verbindungen als Kanten, mit einer tatsächlichen Einbettung modelliert sein. Als Lokalisierungsproblem wird dann stets das Problem bezeichnet, diese Einbettung zu rekonstruieren oder anzunähern. Diese Probleme sind in jedem Fall eng verwandt mit schon länger untersuchten Fragestellungen aus der Visualisierung von Graphen (mit zusätzlichen Bedingungen), und ich werde auf ältere Ergebnisse aus diesem Bereich verweisen. Die verschiedenen Ausprägungen erhält das Problem durch die Art der zur Verfügung stehenden Informationen, die sich ganz grob in zwei Klassen einteilen lassen:

Strukturelle Annahmen In diese Klasse fallen Annahmen darüber, dass in der Realität auftretende Sensornetzwerke in ihrer Struktur nicht unabhängig sind von der Lage der Knoten. Klassisches Beispiel sind die *Unit-Disk-Graphen*, die ich in Abschnitt 4.1.1 behandeln werde.

Topologische Informationen Hier stehen zusätzliche Informationen über die tatsächliche Einbettung zur Verfügung. Diese reichen von der Kenntnis absoluter Positionen einzelner Knoten hin zu relativen Informationen wie den lokalen Peilungen von kommunizierenden Knoten. Diese Informationen korrespondieren mit zusätzlicher Sensorik wie in Kapitel 2 beschrieben.

Je nach Ausprägung lassen sich verschiedene Aspekte untersuchen, sei es theoretisch oder empirisch, und verwendete Verfahren daraufhin analysieren:

Eindeutigkeit Was ist maximal erreichbar, d. h. welche Freiheitsgrade sind überhaupt durch die zur Verfügung stehenden Informationen zu eliminieren? Welche Voraussetzungen müssen Netzwerke dafür haben, bzw. wie groß sind Teilnetzwerke, die diese Voraussetzungen erfüllen?

Qualitätsbegriff Wenn exakte Lösungen wegen vager, fehlerbehafteter oder schlicht unzureichender Eingaben oder Annahmen nicht zu erreichen sind – woran lässt sich die Qualität einer Rekonstruktion messen? Welche Qualität lässt sich für Verfahren garantieren? Welche Qualität wird in realistischen Szenarien erreicht?

Komplexität Wie komplex sind die Probleme, welche Laufzeit haben verwendete Verfahren?

In weiteren Verlauf diese Kapitels werde ich mich darauf konzentrieren, eine Übersicht darüber zu geben, welche Ausprägungen mit welchen Ergebnissen bisher behandelt worden sind und die dafür und für das folgende notwendige Formalisierungen einführen. Um dabei nicht zu verwirren, werde ich dabei die Notationen behutsam vereinheitlichen.

Unabhängig von der Art der Eingabe, die zur Rekonstruktion zur Verfügung steht, lässt sich die Qualität einer Rekonstruktion durch einen Vergleich der tatsächlichen Einbettung $\mathbf{p}$ mit der rekonstruierten Einbettung $\hat{\mathbf{p}}$ messen. Ein naheliegendes und dem Ziel der Lokalisierung entsprechendes Maß ist der Mittelwert des euklidischen Abstandes aller Knoten (siehe auch [MLRT04]),

$$Q_{\mathrm{GTE}}(\mathbf{p}, \hat{\mathbf{p}}) := \sum_{v \in V} \frac{|\mathbf{p}_v - \hat{\mathbf{p}}_v|}{n} \quad .$$

Je nach Art der Eingabe lässt eine Rekonstruktion allerdings bestimmte Freiheitsgrade, zum Beispiel Drehungen und Verschiebungen, wenn die Eingabe keinerlei absolute Orientierung oder Position umfasst, oder auch die Skalierung, wenn die Eingabe unabhängig von der Skalierung der tatsächlichen Einbettung ist. Für die Evaluation in den späteren Kapiteln werde ich deshalb eine leicht nachvollziehbare Variante verwenden, die immer noch ein intuitives Qualitätsmaß für die Rekonstruktion der Netztopologie bildet und bei der das Minimum von Q_{GTE} über diese Freiheitsgrade gebildet wird durch

$$Q_{\overline{\mathrm{GTE}}}(\mathbf{p}, \hat{\mathbf{p}}) := \min_{\gamma \in [0,\pi), \mathbf{x} \in \mathbb{R}^2, c \in \mathbb{R}^+} \sum_{v \in V} \frac{|\mathbf{p}_v - c \cdot R(\gamma) \cdot \hat{\mathbf{p}}_v + \mathbf{x}|}{n} \quad .$$

Die Optimierung über diese Freiheitsgrade ist nicht leicht; sofern dieses Maß in der Evaluation benutzt wird, ist eine obere Schranke angegeben, die dadurch gebildet wird, dass zuerst über die optimale Skalierung und Drehung einiger möglichst langer Kanten gemittelt wird und anschließend über die optimale Verschiebung aller Knoten. Entsprechend kann dieses Maß dann nur als Beleg einer gewissen Güte herangezogen werden, nicht als Beleg mißglückter Rekonstruktion.

Fällt Skalierung als Freiheitsgrad aus, werden in der Literatur mitunter auch die paarweisen Abstände vergleichen und somit die Optimierung über Drehungen und Verschiebungen eingespart durch

$$Q_{\mathrm{PAIR}}(\mathbf{p}, \hat{\mathbf{p}}) := \sqrt{\sum_{i,j \in V} \frac{\left(|\mathbf{p}_v - \mathbf{p}_v| - |\hat{\mathbf{p}}_v - \hat{\mathbf{p}}_v|\right)^2}{n \cdot (n-1)/2}} \quad .$$

Hier als quadratisches Mittel, weil die Kantenlängen in beide Richtungen abweichen können, d. h. zu kurz oder zu lang sein können. Abhängig von der Ausprägung der Eingabe lassen sich noch viele andere Qualitätsmaße für die Evaluation oder als Grundlage für Aussagen zur Approximierbarkeit denken; einige wichtige werden in den jeweiligen Abschnitten eingeführt.

4.1. Strukturelle Annahmen

Einer der grundlegenden Ansätze für Lokalisationsverfahren ist die Unterstellung, dass es bestimmte Korrellationen zwischen der Struktur von Sensornetzwerken und ihrer Einbettung gibt.

4.1.1. Unit-Disk-Graphen

Das wohl bekannteste Beispiel hierfür ist die Annahme, dass in Sensornetzwerken die graphentheoretische Nachbarschaft immer auch innerhalb einer gewissen Umgebung eingebettet sein muss und dass umgekehrt Knoten innerhalb dieser Umgebung immer auch der Nachbarschaft angehören müssen. Dies entspricht der Überlegung, dass es für Sensornetzwerke in der Realität so etwas wie einen Kommunikationsradius gibt, über den hinaus ein Sensor keine Verbindung aufbauen kann, und der idealisierten Annahme, dass in diesem Radius Verbindungen auch zwingend zustande kommen. Graphen, die sich so einbetten lassen, heißen *Unit-Disk-Graphen*; Abbildung 4.1 zeigt einen kleinen solchen Graphen.

Definition 5 (Unit-Disk-Graphen) *Ein Graph $G = (V, E)$ heißt* Unit-Disk-Graph *(UDG), wenn es eine Einbettung $\mathbf{p}$ in die Ebene gibt, so dass*

$$\forall v, u \in V, v \neq u : \{u, v\} \in E \iff |\mathbf{p}_u - \mathbf{p}_v| \leq 1 \ .$$

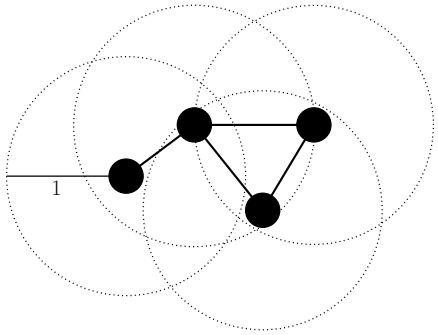


Abbildung 4.1.: Ein Unit-Disk-Graph. Knoten sind genau dann verbunden, wenn ihr Abstand kleiner oder gleich 1 ist.

Die Unterstellung besteht also darin, dass alle Sensornetzwerke, normiert man den Kommunikationsradius zu 1, Unit-Disk-Graphen sind und ihre tatsächliche Einbettung dies belegt. Diese Modellierung hat eine ganze Reihe von Ergebnissen hervorgebracht; allen voran die Aussage, dass die Entscheidung, ob ein gegebener Graph ein Unit-Disk-Graph ist, $\mathcal{NP}$-schwer ist ([BK93], eine Reduktion von SATISFIABILITY auf UNITDISKGRAPHRECOGNITION). Damit gibt es natürlich auch keine Hoffnung[1] auf

[1] hier und bei ähnlichen Formulierungen ergänze man immer „sofern nicht $\mathcal{P} = \mathcal{NP}$“

einen Algorithmus, der zu einem beliebigen Unit-Disk-Graphen in Polynomialzeit eine Einbettung findet, die diese Eigenschaft belegt. Ein neueres Ergebnis aus [KMW04] misst die Qualität einer Einbettung $\mathbf{p}$ eines Unit-Disk-Graphen $G = (V, E)$ durch einen Approximationsfaktor $Q_{\mathrm{UDG}}(\mathbf{p})$ mit

$$Q_{\mathrm{UDG}}(\mathbf{p}) := \frac{\max_{\{u,v\}\in E} |\mathbf{p}_u - \mathbf{p}_v|}{\min_{\{u,v\}\notin E} |\mathbf{p}_u - \mathbf{p}_v|}$$

und zeigt in Anlehnung an den Beweis aus [BK93], dass es sogar $\mathcal{NP}$-schwer ist, zu entscheiden, ob ein Graph eine Einbettung $\mathbf{p}$ mit $Q_{\mathrm{UDG}}(\mathbf{p}) \leq \sqrt{3/2} - \epsilon$ hat, wobei ϵ für große n gegen 0 strebt. Auf der anderen Seite beschreibt [MOWW04] einen Algorithmus, der für Unit-Disk-Graphen in polynomieller Zeit eine Einbettung $\mathbf{p}$ berechnet, so dass $Q_{\mathrm{UDG}}(\mathbf{p}) \in \mathcal{O}(\log^{2.5} n\sqrt{\log\log n})$.

Neben diesen vor allem theoretischen Ergebnissen gibt es eine ganze Reihe von Lokalisationsalgorithmen, die für Unit-Disk-Graphen ohne den Blick auf Approximationsgarantien versuchen, Einbettungen auf Basis der Graphenstruktur zu rekonstruieren, allerdings überwiegend ausgehend von Ankerknoten, das heißt wenigen Knoten, deren exakte Position bekannt ist, oft sogar durch verteilte Verfahren wie APS (*Ad-hoc Positioning System*, [NN03a]); hier wird der graphentheoretische Abstand zu Ankerknoten zur Abschätzung der Entfernungen herangezogen.

Interessant ist aus algorithmischer Sicht für diesen Fall auch die Herangehensweise von [BW04]; die Autoren beschäftigen sich sehr grundlegend mit der Frage, wie gut Lokalisation nur durch Kenntnis graphentheoretischer Abstände zu Ankerknoten überhaupt sein kann, und zeigen schon für eindimensionale Fälle, dass so ein Vorgehen nicht kompetitiv sein kann gegenüber einer vollständigen Nutzung der Struktur.

4.1.2. Quasi-Unit-Disk-Graphen

Die sehr enge Idealisierung der Unit-Disk-Graphen und vor allem die Ergebnisse zur Approximierbarkeit führten in letzter Zeit zu einer Verallgemeinerung des Modells der Unit-Disk-Graphen, bei dem nicht mehr *ein* Kommunikationsradius vorausgesetzt wird, bis zu dem Kommunikation stattfinden kann *und* muss, sondern zwei Radien, einer für die maximale Reichweite, einer für eine Zone, in der Kommunikation garantiert wird ([KWZ03]).

Definition 6 (d-Quasi-Unit-Disk-Graphen) *Ein Graph $G = (V, E)$ heißt* d-Quasi-Unit-Disk-Graph *(d-QUDG) für ein $0 < d \leq 1$, wenn es eine Einbettung $\mathbf{p}$ in die Ebene gibt, so dass*

$$\begin{aligned} \forall v, u \in V, v \neq u : \{u, v\} \in E &\implies |\mathbf{p}_u - \mathbf{p}_v| \leq 1 \\ \forall v, u \in V, v \neq u : \{u, v\} \in E &\impliedby |\mathbf{p}_u - \mathbf{p}_v| \leq d \ . \end{aligned}$$

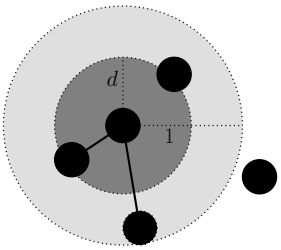


Abbildung 4.2.: Verbindungen in einem d-Quasi-Unit-Disk-Graphen müssen zwischen Knoten bestehen, die einander dichter sind als d. Knoten, die weiter voneinander entfernt sind als 1, dürfen nicht verbunden sein. Für den Bereich dazwischen wird keine Aussage getroffen.

Man beachte, dass dieses Modell keinerlei Aussagen über die Notwendigkeit oder Unmöglichkeit von Kanten zwischen Knoten mit Abständen in der „Grauzone“ $(d; 1]$ trifft (siehe Abbildung 4.2).

Da alle Unit-Disk-Graphen insbesondere auch Quasi-Unit-Disk-Graphen sind, übersetzen sich die Aussagen zu deren Nichtapproximierbarkeit direkt in die Aussage, dass es zumindest für $d \geq 1/\sqrt{2}$ $\mathcal{NP}$-schwer ist, zu entscheiden, ob ein Graph ein d-Quasi-Unit-Disk-Graph ist (ebenfalls [MOWW04]).

Dieses Modell wird vor allem in Kombination mit zusätzlichen Informationen über die Topologie aufgegriffen und ist auch relativ neu, insofern gibt es keine bekannten Lokalisierungsalgorithmen, die *ausschließlich* auf Quasi-Unit-Disk-Graphen aufbauen.

Algorithmus 1 : Generiere d-Quasi-Unit-Disk-Graph mit n Knoten in $A \subseteq \mathbb{R}^2$

für $i = 1$ **bis** n **tue**
  Füge Knoten v_i ein, wähle $\mathbf{p}_{v_i}$ zufällig gleichverteilt aus A
für alle $v_i \neq v_j$ **tue**
  wenn $|\mathbf{p}_{v_i} - \mathbf{p}_{v_j}| \leq d$ **dann**
    Füge Kante $\{v_i, v_j\}$ ein
  sonst wenn $|\mathbf{p}_{v_i} - \mathbf{p}_{v_j}| \leq 1$ **dann**
    Füge Kante $\{v_i, v_j\}$ mit Wahrscheinlichkeit $\frac{1-|\mathbf{p}_{v_i}-\mathbf{p}_{v_j}|}{1-d}$ein

Sofern für die Evaluation *zufällige* d-Quasi-Unit-Disk-Graphen herangezogen werden, sind diese durch Algorithmus 1 generiert worden[2]. Dabei nimmt die Wahrscheinlichkeit einer Kante von der Entfernung d bis zur Entfernung 1 linear ab (siehe Abbildung 4.3), wie auch zum Beispiel in [BGJ05], auch wenn dort die Knotenpositionen nicht gleichverteilt gewählt werden. Sofern auf die *Knotendichte* generierter Graphen bezug genommen wird, ist damit die Anzahl der Knoten pro Flächeneinheit $n/|A|$ gemeint; sofern nicht anders angegeben, ist $d = 1/2$ und die Fläche A quadratisch.

[2]bzw. durch eine im Ergebnis äquivalente Art und Weise

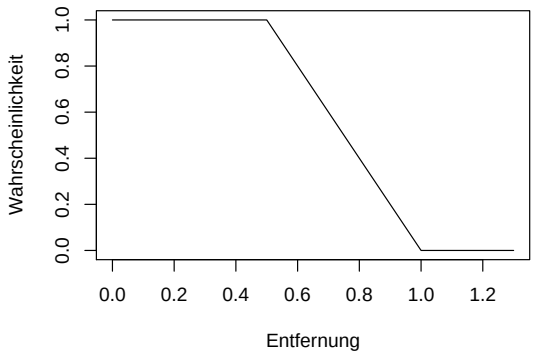


Abbildung 4.3.: Bei den generierten Quasi-Unit-Disk-Graphen nimmt die Verbindungswahrscheinlichkeit von $d = 1/2$ bis 1 linear ab.

4.1.3. Kritik an strukturellen Annahmen

Trotz der sehr naheliegenden Intuition und der beliebig wählbaren Freiheit, die d-Quasi-Unit-Disk-Graphen bieten, darf man sich nicht darüber hinwegtäuschen, dass besonders die Forderung, einen Radius angeben zu können, unterhalb dessen Kommunikation garantiert werden kann, von einer sehr mathematische Sichtweise geprägt ist. Alle Verfahren, die wesentlich auf der tatsächlichen Einbettbarkeit eines Kommunikationsgraphen als Quasi-Unit-Disk-Graph aufbaut, können in der Realität an simplen Hindernissen, die auch nahe Knoten an der Kommunikation hindern, scheitern. Darüber hinaus enthält das Modell implizit die Annahme, dass der Raum isotrop ist, und nicht etwa die Umweltbedingungen so schwanken, dass die Reichweiten sich innerhalb des Gebietes, in dem die Sensoren ausgebracht sind, signifikant voneinander abweichen.

4.2. Topologische Informationen

Neben den strukturellen Annahmen, die gewissermaßen kostenlos zur Verfügung stehen, wurde in Abschnitt 2.1 Sensorik beschrieben, über die Sensoren zusätzliche Informationen über die Lage ihrer Nachbarn ermitteln können.

4.2.1. Entfernungen

Die in der Lokalisation am meisten verwendete Zusatzinformation ist mit Sicherheit die Länge der Kanten, d. h. zur Rekonstruktion einer Einbettung steht die induzierte Längenzuweisung zur Verfügung. Die Beliebtheit dieser Ausprägung des Lokalisie-

rungsproblems hat zwei Ursachen: Zum einen sind zumindest vage Entfernungsinformationen verhältnismäßig billig, was den technischen Aufwand angeht (z. B. RSSI), zum anderen sind Algorithmen zur Realisierung von Graphen zu gegebenen Kantenlängen schon lange untersucht, sowohl in Hinblick auf Probleme der Graphenvisualisierung, als auch in Hinblick auf die Statik von Gerüsten. Obwohl der Fokus dieser Arbeit nicht auf Entfernungs- sondern auf Richtungsmessungen als Grundlage für Lokalisierung liegen soll, werden ich das Thema immer im Vergleich mit Entfernungsmessungen betrachten; das ist gewissermaßen das Modell, an dem man sich messen lassen muss, und es gibt hier bereits mehr Ansätze, von denen man lernen kann.

Definition 7 (Längenrealisierungsproblem) *Gegeben einen zusammenhängenden Graphen $G = (V, E)$ und eine konsistente Längenzuweisung ℓ, finde eine Einbettung* $\mathbf{p}$*, so dass* $\mathbf{p}$ *für G mit ℓ verträglich ist.*

Auch die Schwere dieses Problems ist schon lange bekannt: Saxe hat in [Sax79] gezeigt, dass es $\mathcal{NP}$-schwer ist, zu einem Graphen und einer Längenzuweisung zu entscheiden, ob es eine verträgliche Einbettung gibt. Wir werden im Rahmen von Kapitel 5 noch darauf eingehen, welche Fragestellungen aus kombinatorischer Sicht angegangen wurden, wie sich zum Beispiel Graphen charakterisieren lassen, für die die induzierte Längenzuweisung einer allgemeinen Einbettung diese *eindeutig* festlegt.

Rekonstruktionen lassen sich zwar danach bewerten, inwieweit eine Längenzuweisung eingehalten wurde, z. B.

$$Q_{\text{DIST}}(\mathbf{p}, \hat{\mathbf{p}}) = \sqrt{\sum_{\{u,v\}\in E} \frac{||\mathbf{p}_v - \hat{\mathbf{p}}_v| - \ell(e)|^2}{n}} \ ,$$

man sollte sich aber vor Augen halten, dass selbst vollständig eingehaltene Kantenlängen mitunter mehrere völlig verschiedene Einbettungen zulassen.

Algorithmisch wurde das Problem vor allem dadurch angegangen, ausgehend von Knoten, die ein Dreieck bilden, neue Knoten durch Trilateration anzubinden. Dieses Verfahren lässt sich verhältnismäßig leicht verteilt implementieren und – um Fehler zu korrigieren – auch iterativ immer weiter verbessern ([SRB01]). Es fußt auf der Beobachtung, dass so aufgebaute Graphen (*trilateration graphs*, [EGW+04]) in ihrer Einbettung eindeutig bestimmt sind, wenn sich die Knoten in allgemeiner Lage befinden (Abbildung 4.4). Dass dies nur eine Klassen von Graphen ist, für die das gilt, ist einer der Punkte, die in Kapitel 5 vertieft werden.

Verbindung mit (Quasi-)Unit-Disk-Graphen

Gerade in neueren Arbeiten wird das Problem der entfernungsbasierten Lokalisierung oft kombiniert mit der Annahme, dass sich Sensornetze als (Quasi-)Unit-Disk-Graphen beschreiben lassen. Dieser Verknüpfung entsprang unter anderem eine Untersuchung,

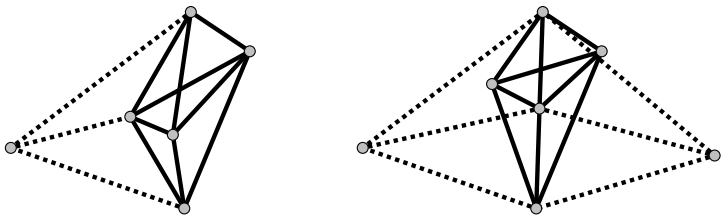


Abbildung 4.4.: Befinden sich die Knoten in allgemeiner Lage (links), reichen drei Knoten mit bekannter Einbettung, um einen Knoten anzubinden. Liegen die Einbettungen der Knoten auf einer Geraden (rechts), gibt es zwei Möglichkeiten.

ab welcher Dichte zufällige Sensornetzwerke, modelliert als Unit-Disk-Graphen mit hoher Wahrscheinlichkeit durch iterative Trilateration komplett abgedeckt werden können (ebenfalls [EGW+04]). Diese iterative Trilateration liefert als verteiltes Protokoll, eine Knotendichte von $\Omega(\log n)$ vorausgesetzt, in einer Laufzeit von $\mathcal{O}(n)$ mit hoher Wahrscheinlichkeit eine Lokalisation aller Knoten.

Für Graphen solcher Struktur ist eine eindeutige Rekonstruktion bei Knoten in allgemeiner Lage natürlich möglich; die Kombination aus Längenrealisierungsproblem und Unit-Disk-Graph-Realisierungsproblem ist trotzdem ohne diese Voraussetzungen $\mathcal{NP}$-schwer, wie [AGY04] durch eine Reduktion von CircuitSatisfiability auf Unit-DiskGraphReconstruction zeigt.

Ein Aspekt, der bei vielen dieser Verfahren verhältnismäßig wenig beachtet wird, ist der, dass die entfernungsbasierte Lokalisation von einem Knoten durch drei Ankerknoten sehr anfällig gegenüber Fehlern ist (siehe Abbildung 4.5). [MLRT04] schlägt deshalb ein robustes, verteiltes Verfahren vor, das auf der Auslegung von *stabilen* Vierecken (4-Cliquen) basiert. Dies erhöht zwar die Anforderungen an den Grad der Vernetzung bzw. verringert die erwartete Abdeckung, vermeidet aber fundamentale Fehlentscheidungen in der Auslegung neuer Knoten.

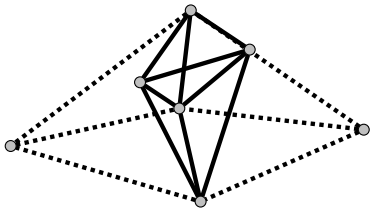


Abbildung 4.5.: Bei auch nur schwach fehlerhafter Eingabe müssen die Einbettungen der drei relevanten Knoten nicht unbedingt *genau* auf einer Geraden liegen, damit man nicht mehr zwischen zwei grundverschiedenen Einbettungen des nächsten Knotens unterscheiden kann.

Ein komplett anderer Ansatz stammt von den kräftebasierten Verfahren zur Visuali-

sierung von Graphen ab (zu diesen siehe [BETT99]); ausgehend von einer Initiallösung werden hier üblicherweise durch anziehende und abstoßende Kräfte zwischen verbundenen bzw. nichtverbundenen Knoten gute Layouts erzeugt. [PBDT03] adaptiert dieses Verfahren dazu, dass Kräfte nur so wirken, dass Kanten in die richtige Länge gezogen bzw. gestaucht werden. Gerade bei einem solchen Vorgehen ist das Finden einer guten Initiallösung, d. h. einer Lösung, die die Topologie bereits grundlegend annähert, wesentlich. Was es auf jeden Fall zu verhindern gilt, sind Faltungen des Graphen, die sich durch lokale Kräfte nur selten beheben lassen. Die Autoren lösen dieses Problem durch ein Verfahren, das im wesentlichen auf der Auswahl von fünf geschickt gewählten Knoten (einer im Zentrum und vier am Rand „in allen Himmelsrichtungen“) und der Auslegung der anderen Knoten auf Basis der graphentheoretischen Abstände zu diesen fünf ausgewählten Knoten aufbaut. Dieses Verfahren lässt sich vollständig verteilt umsetzen. In [GK05] wird darauf aufbauend vorgeschlagen, stattdessen ein initiales Layout durch spektrale Einbettung (auch eine Anleihe aus dem Bereich des Graphenzeichnens) zu ermitteln. Auch dieses Verfahren ist so entworfen, dass es sich verteilt implementieren lässt.

4.2.2. Richtungen

Der Verwendung von Kantenlängen gegenüber steht die Verwendung von Kantenrichtungen. Wie schon in Kapitel 3 erwähnt, ist es dabei letztlich irrelevant, ob man von lokalen oder globalen Richtungen für die Kanten ausgeht, zumindest, solange man ungestörten Eingaben betrachtet.

Definition 8 (Richtungsrealisierungsproblem) *Gegeben einen Graphen G und eine konsistente Richtungszuweisung α, finde eine Einbettung $\mathbf{p}$, so dass $\mathbf{p}$ für G mit α verträglich ist.*

Bemerkung 1 *Für das Richtungsrealisierungsproblem lässt sich O.B.d.A. annehmen, dass $\alpha(i,j) \neq \mathbf{0}$ für alle $\{i,j\} \in E$, ansonsten lässt sich die Kante i,j kontrahieren, da für alle mit ℓ verträglichen Einbettungen $\mathbf{p}_i = \mathbf{p}_j$ sein muss.*

Richtungen lassen sich in naheliegender Weise zur sukzessiven Triangulation nutzen; diese Nutzung wird auch (z. B. in [NN03b]) parallel zur Trilateration durch Entfernungsmessungen aufgeführt. Insgesamt wurden richtungsbasierte Verfahren jedoch weitaus weniger untersucht, obwohl sich, von der hardwareseitigen Implementierung einmal abgesehen, viele Gründe für solche Verfahren finden lassen: Das Finden von verträglichen Einbettungen ist in Polynomialzeit möglich und die *Rigidity Theory* liefert – wenn auch eher in Form von Randbemerkungen – auch die Aussage, dass die Entscheidung, ob eine solche Einbettung eindeutig bestimmt ist, verhältnismäßig leicht ist. Außerdem ist der Grad der Vernetzung, das heißt die Anzahl der Verbindungen, die zur Rekonstruktion benötigt werden, signifikant niedriger. Diese Aspekte werden,

vor allem in der Gegenüberstellung zu entfernungsbasierter Rekonstruktion, in Kapitel 5 beleuchtet. In der Sensorlokalisierung wurden sie bisher verhältnismäßig wenig aufgegriffen. [EWM+03] geht zwar auf die Hintergründe und die Gemeinsamkeiten von Entfernungs- und Richtungsmessungen aus kombinatorischer Sicht ein, algorithmischer Nutzen ist aus den Unterschieden jedoch bisher kaum gezogen worden

Erst vor kurzem wurde mit [BGJ05] der Beweis geliefert, dass die Entscheidung, ob ein Graph unter Berücksichtigung einer Richtungszuweisung als Unit-Disk-Graph eingebettet werden kann, $\mathcal{NP}$-schwer ist. Vorgeschlagen wurde allerdings ein Lineares Programm als praktische, näherungsweise Lösung. Die Nebenbedingungen ergaben sich dabei zum einen aus der Richtungszuweisung, zum anderen daraus, dass unter Kenntnis der Richtungen aller Kanten in Unit-Disk-Graphen alle Kantenkreuzungen ermittelt werden können. Die Autoren geben ein Verfahren an, das die Anzahl der Variablen, die den Kantenlängen entsprechen, schnell verringert, indem die Kantenlängenverhältnisse in starren Subgraphen durch Triangulierung und Kantenüberlappung ermittelt werden. Dieses Vorgehen wird in Kapitel 6 noch näher beschrieben. Die Autoren geben an, dass ihr Verfahren bei geringen Fehlerne auch für fehlerhafte Eingaben dazu geeignet ist, eine Einbettung zu rekonstruieren, allerdings ohne die Bedingungen für die Kantenkreuzungen. Ihre Arbeit ist als Ausgangspunkt für die vorliegende Arbeit zu bezeichnen, wird sich allerdings vor allem in den folgenden, wesentlichen Punkten davon abheben:

In [BGJ05] wird nicht darauf eingegangen, dass das Finden verträglicher Einbettung ohne Berücksichtigung der Unit-Disk-Graph-Bedingungen tatsächlich nur dem Lösen des Linearen Programms entspricht und wann überhaupt zusätzliche Einschränkungen benötigt werden. Tatsächlich ist der Beitrag von Unit-Disk-Graph-Einschränkungen in vielen Fällen nur gering, abhängig von der Dichte des Netzes. Die vorliegende Arbeit wird deshalb das Problem auch für Quasi-Unit-Disk-Graphen betrachten.

Diese Erkenntnis ist dicht damit verknüpft, das in [BGJ05] zwar durch die Winkelmessungen eindeutig bestimmte Subgraphen identifiziert und genutzt wurden, um die Anzahl der Variablen zu verringern; dieses Verfahren ist aber nicht erschöpfend. Diese Arbeit wird viel Gewicht darauf legen, wie man auf Basis der Richtungsangaben *maximal viele* Variablen eliminieren kann. In diesem Zusammenhang werden hierfür auch schnelle Verfahren mit einer ausgedehnten Analyse vorgestellt, so dass sich zeigt, in welchen Fällen das Lineare Programm überhaupt noch zur Lösung beiträgt, beziehungsweise wie groß der Gewinn gegenüber dem vorigen Verfahren ist.

Obwohl [BGJ05] angibt, dass das beschriebene Verfahren auch bei fehlerhafter Eingabe angewandt werden könne, lassen sich Beispiele finden, in denen das dort vorgestellte Lineare Programm an beliebig kleinen Störungen scheitern kann, d. h. für die die formulierten Nebenbedingungen überhaupt nicht mehr erfüllbar sind. Für die fehlertolerante Rekonstruktion wird deshalb ein Verfahren vorgestellt, das ohne das Lösen eines Linearen Programms auskommt. Es basiert im wesentlichen auf denselben Schritten, die vorher zum Reduzieren der Variablen des Linearen Programms verwendet wurden, zusammen mit lokaler Optimierung und der Nutzung vorhandener Redundanz.

5. Kombinatorische Aspekte

Neben den skizzierten Aussagen zur $\mathcal{NP}$-Schwere bestimmter Probleme, die gewissermaßen Schranken für algorithmische Lösungen der Lokalisierungsprobleme aufweisen, liefert die Theorie zur Starrheit von Gerüsten, die *Rigidity Theory*[1], hilfreiche Aussagen darüber, welche Eigenschaften Graphen haben müssen, um eindeutige Lokalisierung bei bekannten Längen oder Richtungen der eingebetteten Kanten zu ermöglichen. Sie offenbart auch, wie fundamental der Unterschied beider Problemstellungen trotz enormer Analogien schon auf diesem Gebiet ist.

Im folgenden werde ich versuchen, eine kurze Zusammenfassung wesentlicher Ergebnisse und deren Motivation zu geben. Sie basiert in weiten Teilen auf der sehr zu empfehlenden Darstellung in [Whi96], wenngleich ich versuchen werde, Notationen und Aussagen nur soweit aufzugreifen, wie es hier dem Verständnis der Probleme nützt. An geeigneten Stellen werde ich mit eigenen Überlegungen ergänzen, inwiefern diese Ergebnisse für die Lokalisierung, insbesondere aus algorithmischer Sicht, genutzt werden können.

5.1. Vereinbarungen

Die *Rigidity Theory* hat ihre Wurzeln in der Betrachtung von Gerüsten als physikalischen Gebilden. Sie untersuchte die Frage, wann ein Gerüst, bestehend aus festen Holmen und flexiblen Gelenken, starr ist in dem Sinne, dass es keine kontinuierliche Verformung zulässt, ohne die Holme selbst zu verformen. Solche Gerüste werden *Gelenkgerüste* (*bar-and-joint frameworks*) genannt und lassen sich leicht motivieren: Gerüste dieser Art lassen sich sich in Brückenbau oder Fachwerkhäusern, ja selbst in schwedischen Selbstbaumöbeln finden, die Schrägverstrebungen benötigen, um nicht zusammenzuklappen. Mehr theoretischer Natur sind sogenannte *Teleskopgerüste* (*telescoping frameworks*). Sie bestehen aus Holmen variabler Länge, die durch starre Gelenke verbunden sind. Der Zusammenhang dieser beider Gerüsttypen mit Netzwerken, deren Kantenlängen oder -richtungen bekannt sind, ist schwer zu übersehen.

[1] In Ermangelung eines griffigen deutschen Begriffes werde ich bei diesem Namen bleiben

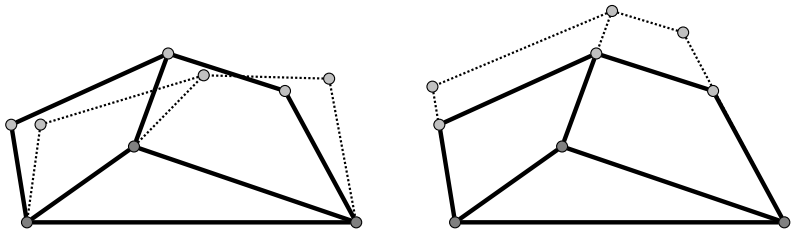


Abbildung 5.1.: Verformungen, die Gelenkgerüste (links) bzw. Teleskopgerüste (rechts) zulassen

5.2. Starrheit

5.2.1. Gelenkgerüste

Ich betrachte zunächst Eigenschaften von Gerüsten, die aus festen Stangen, aber flexiblen Verbindungen zusammengesetzt sind. Mit ihnen beschäftigt sich die *Rigidity Theory* klassischerweise. Die Frage, wann solche Gerüste eindeutig durch die Kenntnis der Kantenlängen festgelegt sind, korrespondiert direkt mit der Frage, wann ein entsprechendes Sensornetzwerk mit bekannten Kommunikationsentferungen eindeutig rekonstruierbar ist.

Die Eigenschaft eines Gerüstes, nicht kontinuierlich deformierbar zu sein, ohne die die Holme zu stauchen oder zu strecken, wird als *Starrheit* bezeichnet. Sie entspricht gewissermaßen einer lokalen Eindeutigkeit.

Definition 9 (Starrheit) *Ein Gerüst $G(\mathbf{p})$ heißt* flexibel*, wenn es eine stetige Abbildung $\mathbf{p}(t) : [0, 1) \to B(G, \mathbf{p})$ mit $\mathbf{p}(0) = \mathbf{p}$ gibt, so dass für ein t die Einbettung $\mathbf{p}(t)$ nicht kongruent zu $\mathbf{p}$ ist. Es heißt* starr*, wenn es nicht flexibel ist.*

Abbildung 5.2 zeigt, dass derselbe Graph mit unterschiedlichen Einbettungen ein starres oder aber ein flexibles Gerüst ergeben kann. Interessanter ist daher die *generische Starrheit*, die im Gegensatz zur Starrheit eine einem Graphen inhärente Eigenschaft ist. Generische Einbettungen bezeichnen dabei die Einbettungen ohne algebraische Abhängigkeiten der $\mathbf{p}(i)_k$. Diese Einbettungen stellen eine offene und dichte Menge im $\mathbb{R}^{d|V|}$ dar ([Whi96]), und wählt man zufällig nach einer stetigen Verteilung auf $\mathbb{R}^{d|V|}$ eine Einbettung, so ist diese mit Wahrscheinlichkeit 0 nicht generisch in diesem Sinne. Wenn im folgenden von „fast allen“ Einbettungen die Rede ist, sind damit die generischen Einbettungen gemeint.

Definition 10 (Generische Starrheit) *Ein Graph G heißt* generisch n-starr*, wenn $G(\mathbf{p})$ für fast alle Einbettungen $\mathbf{p} \in \mathbb{R}^n$ starr ist.*

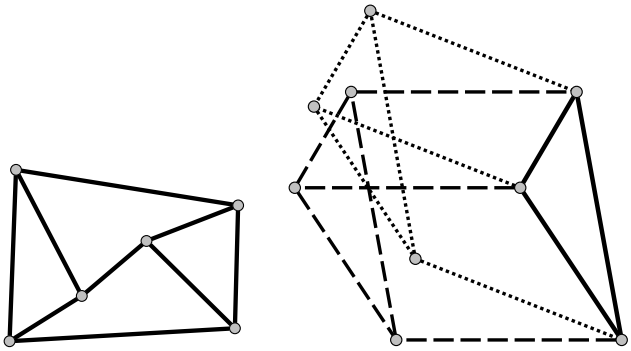


Abbildung 5.2.: starre und flexible Einbettung desselben Gerüstes

Ich werde zunächst beschreiben, wann ein Graph generisch 2-starr ist, was auf das Problem der Realisierung in der Ebene übertragen bedeutet, dass eine Einbettung durch eine realisierbare Längenzuweisung zumindest lokal eindeutig bestimmt ist. Die Beschränkung auf die Ebene ist hierbei notwendig, weil sich die Ergebnisse nicht bequem verallgemeinern lassen.

Infinitesimale Starrheit

Die gängigste Herangehensweise zur Analyse der Starrheit einer Einbettung ist die Berechnung von *infinitesimalen Bewegungen*, die ein Gerüst $G(\mathbf{p})$ zulässt, ohne dass sich dabei Kantenlängen verändern. Eine solche infinitesimale Bewegung wird beschrieben, indem den Knoten „Geschwindigkeiten“ durch einen Vektor $\mathbf{u} : V \to \mathbb{R}^2$ zugewiesen werden, wobei gelten muss, dass für alle $\{i, j\} \in E$

$$\begin{aligned} (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{u}_i - \mathbf{u}_j) &= 0 \\ \iff (\mathbf{p}_i - \mathbf{p}_j) \cdot \mathbf{u}_i + (\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{u}_j &= 0 \ , \end{aligned}$$

was nichts anderes bedeutet, als dass die relative Geschwindigkeit von i zu j, $\mathbf{u}_i - \mathbf{u}_j$ senkrecht zur die beiden Knoten verbindenden Strecke stehen muss. Diese Geschwindigkeiten $\mathbf{u} \in \mathbb{R}^{d|V|}$ lassen sich also als Kern der *Rigidity-Matrix* $R_G(\mathbf{p}) \in \mathbb{R}^{|E| \times d|V|}$ beschreiben, wobei die zur Kante (i, j) korrespondierende Zeile gebildet wird als

$$\begin{pmatrix} \mathbf{0} & \dots & \mathbf{p}_i - \mathbf{p}_j & \dots & \mathbf{p}_j - \mathbf{p}_i & \dots & \mathbf{0} \end{pmatrix}.$$

Abbildung 5.3 zeigt ein Beispiel für eine Einbettung des K_4.

Bemerkung 2 *Das Matroid, das die Zeilen der Matrix $R_G(\mathbf{p})$ durch lineare Unabhängigkeit bilden, lässt sich übertragen auf die Kanten des Graphen. Dieses Matroid wird als* (ebenes) Rigidity-Matroid $\mathcal{R}_2(G, \mathbf{p})$ *bezeichnet, sein Rang ist der Rang der zugehörigen Matrix. Ein ebenes Gerüst $G(\mathbf{p})$ heißt* unabhängig, *wenn* $\operatorname{rank}(\mathcal{R}_2(G, \mathbf{p})) = |E|$. *Für den Fall $G = K_n$ wird auch $\mathcal{R}_2(n, \mathbf{p})$ geschrieben.*

$$\begin{pmatrix} \mathbf{p}_1-\mathbf{p}_2 & \mathbf{p}_2-\mathbf{p}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{p}_1-\mathbf{p}_3 & \mathbf{0} & \mathbf{p}_3-\mathbf{p}_1 & \mathbf{0} \\ \mathbf{p}_1-\mathbf{p}_4 & \mathbf{0} & \mathbf{0} & \mathbf{p}_4-\mathbf{p}_1 \\ \mathbf{0} & \mathbf{p}_2-\mathbf{p}_3 & \mathbf{p}_3-\mathbf{p}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_2-\mathbf{p}_4 & \mathbf{0} & \mathbf{p}_4-\mathbf{p}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{p}_3-\mathbf{p}_4 & \mathbf{p}_4-\mathbf{p}_3 \end{pmatrix} = \begin{pmatrix} -2 & -1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 3 & -1 & -3 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 \end{pmatrix}$$

Abbildung 5.3.: $R_{K_4}(\mathbf{p})$ für $\mathbf{p}_1 = (0,0)$, $\mathbf{p}_2 = (2,1)$, $\mathbf{p}_3 = (-1,2)$, $\mathbf{p}_4 = (0,1)$

Sieht man von dem Fall, dass in einem Graphen alle Knoten auf denselben Punkt eingebettet sind, dann gibt es einen dreidimensionalen Raum von *trivialen* Geschwindigkeiten, die sich zusammensetzen aus Verschiebungen entlang der Achsen und der Drehung. Eine Basis aus diesen beiden Verschiebungen T_x und T_y und einer Drehung (im Uhrzeigersinn) T_r lässt sich explizit angeben und leicht verifizieren:

$$\begin{pmatrix} T_x \\ T_y \\ T_r \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & \dots \\ 0 & 1 & 0 & 1 & \dots \\ -y_1 & x_1 & -y_2 & x_2 & \dots \end{pmatrix} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_1 & \dots \\ \mathbf{e}_2 & \mathbf{e}_2 & \dots \\ \mathbf{p}_1^\perp & \mathbf{p}_2^\perp & \dots \end{pmatrix}$$

Es ist also $T_x(i) = (1,0)$, $T_y(i) = (0,1)$, $T_r(i) = (-y_i, x_i)$ für alle $i \in V$. Entsprechend ist $\operatorname{rank}(R_G(\mathbf{p})) \leq 2|V| - 3$ für alle $G = (V,E)$ und alle Einbettungen in die Ebene $\mathbf{p}$.

Definition 11 *Ein ebenes Gerüst $G(\mathbf{p})$ heißt* infinitesimal 2-starr*, wenn es nur triviale Geschwindigkeiten zulässt.*

Diese infinitesimale Starrheit hat darüber hinaus schon einen generischen Charakter:

Lemma 1 *Für jeden Graphen $G = (V,E)$ sind äquivalent:*

1. *Es gibt eine Einbettung in die Ebene $\mathbf{p}$, so dass $G(\mathbf{p})$ infinitesimal 2-starr ist.*
2. *G ist infinitesimal 2-starr für fast alle Einbettungen in die Ebene.*

Beweis. Hat $R_G(\mathbf{p})$ den maximalen Rang $2|V| - 3$, dann lässt sich durch entsprechende Wahl von je $2|V| - 3$ Zeilen L und Spalten S ein Minor $[R_G(\mathbf{p})]_{L,S}$ mit vollem Rang (und somit $\det[R_G(\mathbf{p})]_{L,S} \neq 0$) finden. Fasst man $\mathbf{p}$ als Variable im $\mathbb{R}^{d|V|}$ auf, ist $\det[R_G(\mathbf{p})]_{L,S}$ ein Polynom in den durch $\mathbf{p}$ zugewiesenen Koordinaten. Als solches ist es entweder überall 0 oder aber von 0 verschieden in fast allen Punkten. □

Zwischen infinitesimaler Starrheit und Starrheit gibt es einen direkten Zusammenhang.

Lemma 2 (Infinitesimale Starrheit und generische Starrheit) *Für einen Graphen G und eine Einbettung $\mathbf{p}$ gilt:*

1. *Ist das Gerüst $G(\mathbf{p})$ infinitesimal starr, ist es auch starr.*
2. *G ist generisch starr genau dann, wenn er für fast alle Einbettungen infinitesimal starr ist.*

Der Beweis ist in [Whi96] und den dort angegebenen Quellen nachzulesen, interessant ist zur Motivation allerdings, dass sich $R_G(\mathbf{p})$ auch herleiten lässt als Jakobi-Matrix der indzuzieren Längenzuweisung $\ell_{G,\mathbf{p}}$.

Was bleibt, ist die Frage, wie sich Graphen charakterisieren lassen, die sich infinitesimal starr einbetten lassen. Sie sind gemäß Lemma 1 für fast alle Einbettungen infinitesimal starr, also für fast alle Einbettungen starr nach Lemma 2, also letztlich generisch starr.

Lemma 3 (Counting Lemma) *Die Kantenmenge E eines Gerüstes $G(\mathbf{p})$ kann nur unabhängig sein, d. h. $E \in \mathcal{R}_2(G, \mathbf{p})$, wenn für alle Teilmengen $E' \subseteq E$ mit $E' \neq \emptyset$ gilt, dass $|E'| \leq 2|V[E']| - 3$.*

Diese Aussage ergibt sich unmittelbar daraus, dass der maximale Rang von $R_{G[E']}(\mathbf{p})$ $2|V[E']|) - 3$ ist, mehr Zeilen also nie linear unabhängig sein können. Die Umkehrung gilt auch, sie bildet einen der zentralen Sätze der Rigidity Theory, aber obwohl Beweis sehr interessant ist, nähme er hier unverhältnismäßig viel Platz ein, deshalb sei auch hier auf [Whi96] verwiesen.

Theorem 1 (Lamans Theorem) *Ein Graph $G = (V, E)$ ist genau dann generisch starr, wenn es eine Kantenmenge $E' \subseteq E$ gibt mit $|E'| = 2|V| - 3$, so dass für alle nichtleeren $E'' \subseteq E'$*

$$|E''| \leq 2|V[E'']| - 3$$

5.2.2. Teleskopgerüste

Teleskopgerüste sind wie oben erwähnt, Gerüste aus Holmen *variabler* Länge, die in fixierten Winkeln miteinander verbunden sind.

Die Starrheit von Teleskopgerüsten lässt sich zunächst weitgehend analog zum Fall der Gelenkgerüste definieren. Macht man sich klar, dass bei solchen Gerüsten durch die Richtung einer Kante die Richtung aller Kanten festgelegt wird, kann man sich bei der Suche nach Verformungen auf Einbettungen beschränken, bei denen alle Kanten ihre ursprüngliche Richtung beibehalten. Auf der anderen Seite sind Einbettungen hier nicht nur dann unwesentlich verschieden, wenn sie *kongruent* sind, sondern auch noch, wenn sie *ähnlich* sind, denn jedes Teleskopgerüst $G(\mathbf{p})$ lässt sich natürlich auch unter Beibehaltung der Richtungen einbetten durch $G(c\mathbf{p})$ für beliebiges $c \in \mathbb{R}^+$.

Definition 12 (Parallele Starrheit) *Ein Gerüst $G(\mathbf{p})$ heißt* parallel flexibel*, wenn es eine stetige Abbildung $\mathbf{p}(t) : [0,1) \to D(G,\mathbf{p})$ mit $\mathbf{p}(0) = \mathbf{p}$ gibt, so dass für ein t die Einbettung $\mathbf{p}(t)$ nicht ähnlich zu $\mathbf{p}$ ist. Es heißt* parallel starr*, wenn es nicht parallel flexibel ist.*

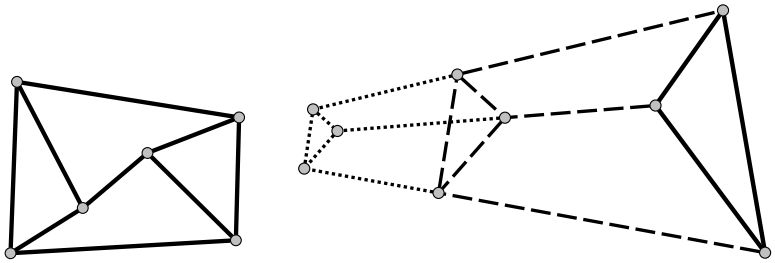


Abbildung 5.4.: parallel starre und flexible Einbettung desselben Teleskopgerüstes

Auch hier gibt es Graphen, für die sich sowohl Einbettungen finden lassen, die zu paralleler Starrheit des Gerüstes führen, als auch solche, für die das Gerüst parallel flexibel bleibt (Abbildung 5.4). Entsprechend interessiert auch hier *generische* parallele Starrheit als Eigenschaft eines Graphen:

Definition 13 (Generische Starrheit) *Ein Graph G heißt* generisch parallel n-starr*, wenn $G(\mathbf{p})$ für fast alle Einbettungen $\mathbf{p} \in \mathbb{R}^n$ parallel starr ist.*

Parallele Einbettungen

Wie zuvor werden wir auch bei der parallelen Starrheit zunächst eine Abwandlung des Problems betrachten, nämlich die Frage, welche parallelen Einbettungen es zu einem Gerüst überhaupt gibt. Wir werden uns auch hier auf den zweidimensionalen Fall einschränken.

Bemerkung 3 (Parallele Einbettung) *Eine Einbettung $\mathbf{q}$ ist eine* parallele Einbettung *eines ebenen Gerüstes $G(\mathbf{p})$ genau dann, wenn für alle $\{i,j\} \in E$*

$$\begin{aligned} (\mathbf{p}_i - \mathbf{p}_j)^\perp \cdot (\mathbf{q}_i - \mathbf{q}_j) &= 0 \\ \iff (\mathbf{p}_i - \mathbf{p}_j)^\perp \cdot \mathbf{q}_i + (\mathbf{p}_j - \mathbf{p}_i)^\perp \cdot \mathbf{q}_j &= 0\ . \end{aligned}$$

Die parallelen Einbettungen eines Gerüstes bilden also den Kern der $|E| \times d|V|$-Matrix $P_G(\mathbf{p})$, deren zur Kante $\{i,j\} \in E$ korrespondierende Zeile

$$\begin{pmatrix} \mathbf{0} & \dots & (\mathbf{p}_i - \mathbf{p}_j)^\perp & \dots & (\mathbf{p}_j - \mathbf{p}_i)^\perp & \dots & \mathbf{0} \end{pmatrix}$$

lautet, d. h. $P(G, \mathbf{p}) = \ker P_G(\mathbf{p})$.

Die Ähnlichkeit mit $R_G(\mathbf{p})$ ist frappierend, und es lässt sich wieder schnell ein dreidimensionaler Raum trivialer Lösungen identifizieren, der aufgespannt wird von den trivialen parallelen Einbettungen, den entlang der Achsen verschobenen Einbettungen T_{xs}, T_{ys} und den skalierten Einbettungen T_D:

$$\begin{pmatrix} T_{xs} \\ T_{ys} \\ T_D \end{pmatrix} = \begin{pmatrix} \mathbf{p}_1 + \mathbf{e}_1 & \mathbf{p}_2 + \mathbf{e}_1 & \dots \\ \mathbf{p}_1 + \mathbf{e}_2 & \mathbf{p}_2 + \mathbf{e}_2 & \dots \\ \frac{\mathbf{p}_1}{2} & \frac{\mathbf{p}_2}{2} & \dots \end{pmatrix} .$$

Dass diese drei Einbettungen tatsächlich im Kern von $P_G(\mathbf{p})$ liegen und für Gerüste mit zwei nicht gleich eingebetteten Knoten (also Knoten i und j mit $\mathbf{p}_i \neq \mathbf{p}_j$) linear unabhängig sind, lässt sich leicht überprüfen, wenn man sich klarmacht, dass $\langle T_{xs}, T_{ys}, T_D \rangle = \langle T_x, T_y, \mathbf{p} \rangle$, dass die drei genannten Einbettungen also denselben Raum aufspannen wie

$$\begin{pmatrix} T_x \\ T_y \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_1 & \dots \\ \mathbf{e}_2 & \mathbf{e}_2 & \dots \\ \mathbf{p}_1 & \mathbf{p}_2 & \dots \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & \dots \\ 0 & 1 & 0 & 1 & \dots \\ x_1 & y_1 & x_2 & y_2 & \dots \end{pmatrix} .$$

Man beachte, dass eine Einbettung $\mathbf{q}$ genau dann trivial parallel ist, wenn $G(\mathbf{p})$ und $G(\mathbf{q})$ ähnlich sind.

Bei all diesen Gemeinsamkeiten und der Ähnlichkeit der Definition verwundert nicht weiter, dass für Gerüste infinitesimale Starrheit und parallele Eindeutigkeit äquivalent sind:

Lemma 4 *Für jedes ebene Gerüst $G(\mathbf{p})$ ist* $\operatorname{rank}(R_G(\mathbf{p})) = \operatorname{rank}(P_G(\mathbf{p}))$

Beweis. Ist $R_G(\mathbf{p}) \cdot \mathbf{u} = \mathbf{0}$, dann ist für $\mathbf{q}$ mit $\mathbf{q}_i := \mathbf{u}_i^\perp$

$$\begin{aligned} & \forall \{i,j\} \in E : (\mathbf{p}_i - \mathbf{p}_j) \cdot \mathbf{u}_i + (\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{u}_j = 0 \\ \iff & \forall \{i,j\} \in E : (\mathbf{p}_i - \mathbf{p}_j)^\perp \cdot \mathbf{u}_i^\perp + (\mathbf{p}_j - \mathbf{p}_i)^\perp \cdot \mathbf{u}_j^\perp = 0 \\ \iff & \forall \{i,j\} \in E : (\mathbf{p}_i - \mathbf{p}_j)^\perp \cdot \mathbf{q}_i + (\mathbf{p}_j - \mathbf{p}_i)^\perp \cdot \mathbf{q}_j = 0 \ , \end{aligned}$$

also $\mathbf{q} \in \ker P_G(\mathbf{p})$ und vice versa. Da aber $\mathbf{u}, \mathbf{v}, \mathbf{w}, \dots \in \mathbb{R}^{|E| \times 2|V|}$ genau dann linear unabhängig sind, wenn $\mathbf{u}^\perp, \mathbf{v}^\perp, \mathbf{w}^\perp, \dots$ dies sind, lässt sich zu jeder Basis von $\ker R_G(\mathbf{p})$ eine gleichgroße Basis von $\ker P_G(\mathbf{p})$ finden und umgekehrt. □

Korollar 1 *Folgende Eigenschaften eines ebenen Gerüstes $G(\mathbf{p})$ sind äquivalent:*

1. *$G(\mathbf{p})$ ist infinitesimal starr.*
2. *$G(\mathbf{p})$ hat nur triviale parallele Einbettungen.*

Für den nächsten Schritt ist folgendes wesentlich:

Lemma 5 *$P(G,\mathbf{p})$ ist konvex und für $\mathbf{q} \in D(G,\mathbf{p})$ und $\mathbf{q}' \in P(G,\mathbf{p})$ gibt es ein $\epsilon \in (0,1]$, so dass für alle $c \in [0,\epsilon)$*

$$(1-c)\cdot\mathbf{q} + c\cdot\mathbf{q}' \in D(G,\mathbf{p}) \ .$$

Beweis. Zunächst ist $P(G,\mathbf{p})$ als Untervektorraum des $\mathbb{R}^{2|V|}$ sicher konvex. Dann verändern sich auf dem Weg von $\mathbf{q} \in D(G,\mathbf{p})$ nach $\mathbf{q}' \in P(G,\mathbf{p})$ die *gerichteten* Kantenlängen sicher stetig, d. h. für $\mathbf{q}_i(t) := (1-t)\cdot\mathbf{q}_i - t\cdot\mathbf{q}'_i$ für $t \in [0,1)$ ist jede der Funktionen

$$\vec{\ell}_{i,j}(t) := \frac{1}{|\mathbf{p}_j - \mathbf{p}_i|}(\mathbf{p}_j - \mathbf{p}_i)\cdot(\mathbf{q}_j(t) - \mathbf{q}_j(t))$$

für alle $(i,j) \in E$ stetig und *positiv* für $\mathbf{q}(t) \in D(G,\mathbf{p})$, also zumindest für $t = 0$ und also auch für $t < \epsilon$ für hinreichend kleines ϵ. □

Lemma 6 *Ein ebenes Gerüst ist parallel 2-starr genau dann, wenn es nur triviale parallele Einbettungen zulässt.*

Beweis. Ist ein Gerüst $G(\mathbf{p})$ nicht parallel 2-starr, dann gibt es eine nicht-ähnliche Einbettung $\mathbf{q} \in D(G,\mathbf{p})$. Es gibt also eine nichttriviale parallele Einbettung. Gibt es umgekehrt eine nichttriviale parallele Einbettung $\mathbf{q} \in P(G,\mathbf{p})$, dann sind auch alle $\mathbf{p} + c\mathbf{q}$ nichttriviale parallele Einbettungen und es gibt ein ε, so dass $\mathbf{p} + c\mathbf{q} \in D(G,\mathbf{p})$ für alle $c \leq \varepsilon$. □

Lemma 7 *Für einen Graphen G sind äquivalent:*

1. *$G = (V,E)$ enthält eine Kantenmenge $E' \subseteq E$ mit $|E'| = 2|V| - 3$ so, dass für alle nichtleeren $E'' \subseteq E'$ $|E''| \leq 2|V[E'']| - 3$*
2. *G ist generisch 2-starr.*
3. *G ist generisch parallel 2-starr.*

Beweis. Die Äquivalenz von 1. und 2. entspricht Lamans Theorem, die Äquivalenz von 2. und 3. ergibt sich aus Lemma 2, Korollar 1 und Lemma 6. □

5.3. Eindeutigkeit und verträgliche Einbettungen

Die Untersuchung der Starrheit von Gerüsten setzt die Kenntnis einer Einbettung voraus; es wird nach kontinuierlichen Verformungen gesucht. Bei der Lokalisierung von Netzwerken kann man davon ausgehen, dass es eine tatsächliche Einbettung gibt, die mit den erhobenen Daten, also einer Längen- oder Richtungszuweisung, verträglich ist, die Aufgaben bestehen aber darin, eine solche zu finden und Eindeutigkeit zuzusichern. Dafür ist die Starrheit der tatsächlichen Einbettung natürlich notwendige Voraussetzung, aber nicht unbedingt ausreichend, wie wir sehen werden.

5.3.1. Gelenkgerüste

Abseits von kontinuierlichen Verformungen lassen Gelenkgerüste auch diskontinuierliche Verformungen zu. Ein einfaches Beispiel hierfür sind Spiegelungen von Subgraphen an zwei Knoten (Abbildung 5.5). Insofern geht Frage nach der Eindeutigkeit einer Einbettung wirklich über die Frage nach der Starrheit hinaus.

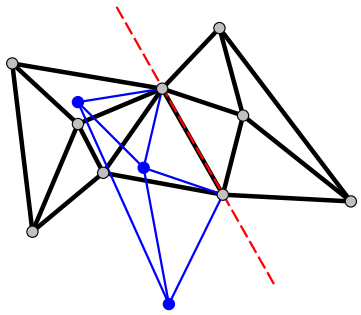


Abbildung 5.5.: Spiegelung an zwei Knoten; jeder nicht dreifach knotenzusammenhängende Graph lässt diese nicht kontinuierliche Verformung zu.

Hendrickson setzt sich in [Hen92] mit der Frage nach dieser Eindeutigkeit wieder unter der Einschränkung auseinander, dass die ursprüngliche Einbettung generisch ist. Er führt die Beobachtung an, dass Spiegelungen nur dann unmöglich sind, wenn der Graph dreifach knotenzusammenhängend ist, weil jeweils zwei Knoten, die den Graphen separieren, eine zulässige Spiegelachse vorgeben. Leider ist das immer noch nicht ausreichend; trotz dreifachen Knotenzusammenhangs sind für generisch starre Graphen, die nach Wegnahme einer Kante nicht mehr generisch starr sind, Verformungen wie in Abbildung 5.6 möglich. Graphen, die generisch starr sind auch nach Wegnahme einer beliebigen Kante noch erfüllen, werden dagegen als *redundant* generisch starr bezeichnet.

Das Ergebnis von [Hen92] sind *notwendige* Bedingungen dafür, dass (generisch) eingebettete Graphen durch die Kantenlängen eindeutig beschrieben werden, und auch

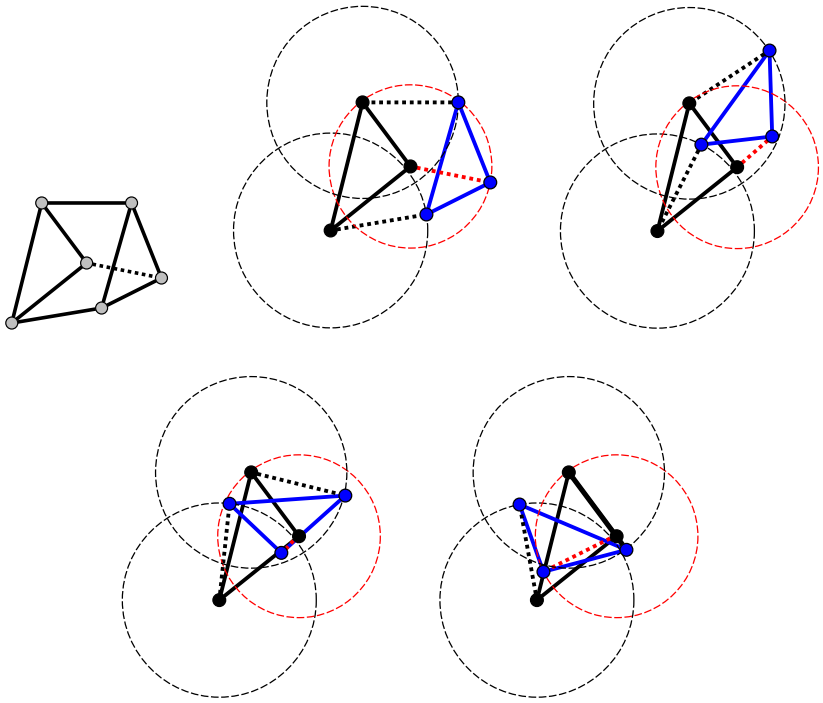


Abbildung 5.6.: Ist ein Graph (links oben) nach Wegnahme einer Kante nicht mehr generisch starr, gibt es eine kontinuierliche Verformung (Bewegung des blauen Subgraphen), bei der alle anderen Längen eingehalten werden. Die Kurve, die der Endpunkt der entfernten Kante beschreibt, schneidet eine zweite zulässige Position.

Algorithmen zur Identifizierung von Subgraphen, die diese Bedingungen erfüllen:

1. redundante generische Starrheit, impliziert generische Starrheit

2. dreifacher Knotenzusammenhang (im $\mathbb{R}^d$ $(d+1)$-fach)

Leider ist nicht bekannt, ob diese Bedingungen zumindest in der Ebene auch hinreichend sind; in höheren Dimensionen gibt es explizite Gegenbeispiele.

Trotzdem lassen sich leicht Graphen konstruieren, für die generische Einbettungen sicher eindeutig rekonstruierbar sind. Die wohl einfachsten sind Trilaterationsgraphen ([EGW+04]), die sich konstruieren lassen, indem man ausgehend von einem Dreieck (K_3) neue Knoten immer so hinzufügt, dass er mit mindestens drei bereits vorhandenen Knoten verbunden ist. Diese Klasse lässt sich noch erweitern um Vereinigung zweier solcher Graphen, die sich an mindestens drei Knoten überlappen, und so weiter. Ich werden diese Vorgehensweise ähnlich und etwas formaler für Teleskopgerüste vorführen.

Finden verträglicher Einbettungen

So wie es an Aussagen dazu mangelt, wann ein Graph generisch eindeutig durch seine Kantenlängen beschrieben ist, so mangelt es auch an entsprechenden Verfahren zur Rekonstruktion. Lässt man algebraische Abhängigkeiten der Knotenpositionen zu, also auch nicht generische Einbettungen, ist dieses Problem $\mathcal{NP}$-schwer, für den generischen Fall fehlen weitere Ergebnisse.

Leicht dagegen ist das Finden verträglicher Einbettungen wieder für generisch eingebettete Trilaterationsgraphen. Eine zulässige Auslegung des zugrundeliegenden Dreiecks lässt sich leicht finden, danach sind die Knoten sukzessive durch jeweils drei Längen eindeutig in ihrer Position festgelegt.

5.3.2. Teleskopgerüste

Sehr viel übersichtlicher gestaltet sich die Lage bei den Teleskopgerüsten, wie schon in Abschnitt 5.2.2 zur parallelen Starrheit anklang. Hier ist Starrheit nicht nur notwendig, sondern auch hinreichend.

Lemma 8 *Genau jeder generisch starre Graph $G = (V, E)$ ist für fast alle Einbettungen $\mathbf{p}$ durch $\alpha_{G,\mathbf{p}}$ eindeutig bis auf Streckung, Drehung und Verschiebung bestimmt.*

Beweis. Diese Aussage ist schon auf dem Weg zur parallelen Starrheit abgefallen, von der wir wissen, dass sie äquivalent ist zum Fehlen nichttrivialer paralleler Einbettungen (Lemma 6). Generische parallele Starrheit wiederum ist dasselbe wie generische Starrheit (Lemma 7). □

Ich werde diesen Zusammenhang im weiteren nicht ständig wiederholen; wann immer von *generisch starren* Graphen die Rede ist, liegt das Hauptaugenmerk darauf, dass diese Graphen für allgemeine Einbettungen bis auf unvermeidliche Freiheitsgrade eindeutig durch die Kantenrichtungen festgelegt sind.

Lemma 9 *Über folgende Schritte lassen sich leicht einfache Graphen konstruieren, generisch starr sind*[2]*:*

(i) K_2 *ist generisch starr.*

(ii) Ist $G = (V, E)$ generisch starr, dann auch $G' = (V', E')$ mit $V' = V \cup \{v'\}$ und $E' = E \cup \{\{u_1, v'\}\{u_2, v'\}\}$ für $u_1, u_2 \in V$ (2-verbundene Knotenaddition).

(iii) Ist $G = (V, E)$ generisch starr, dann auch jeder Graph $G' = (V', E')$ mit $V' = V$ und $E \subseteq E'$ (Kantenaddition).

[2] Der offensichtliche Fall, dass Graphen genau dann generisch starr sind, wenn ein isomorpher Graph generisch starr ist, ist nicht aufgezählt.

(iv) *Sind* $G = (V, E)$ *und* $G' = (V', E')$ *generisch starr und* $V \cap V' \geq 2$*, dann ist auch* $\hat{G} = \left(\hat{V}, \hat{E}\right)$ *mit* $\hat{V} = V \cup V'$ *und* $\hat{E} = E \cup E'$ *generisch starr (Knotenüberlappung).*

(v) *Sind* $G = (V, E)$ *und* $G' = (V', E')$ *generisch starr und* $E \cap E' \neq \emptyset$*, dann ist auch* $\hat{G} = \left(\hat{V}, \hat{E}\right)$ *mit* $\hat{V} = V \cup V'$ *und* $\hat{E} = E \cup E'$ *generisch starr (Kantenüberlappung).*

Graphen, die sich nur durch (i), (ii) und (iii) charakterisieren lassen, heißen 2-simpel*, kommen (iv) bzw. (v) hinzu, spricht man von* 2-knotenüberlappenden *bzw.* kantenüberlappenden *2-simplen Graphen.*

Beweis. Die Beweise ließen sich natürlich alle auch auf Basis von Lamans Theorem (Theorem 1) führen; einsichtiger sind sie aber wohl, wenn man jeweils über die Eindeutig einer parallelen Einbettung (bis auf Streckung und Verschiebung) argumentiert.

(i) Der K_2 ist bei festgelegter Kantenrichtung offensichtlich eindeutig auszulegen

(ii) Hat man für G eine Einbettung gewählt, ist für v' offenbar nur eine Position zulässig.

(iii) Hinzufügen von Kanten kann Eindeutigkeit natürlich nicht zerstören.

(iv) Wählt man je eine Einbettung für beide Subgraphen und fixiert die eine, dann gibt es offensichtlich nur genau eine Streckung und Verschiebung der zweiten, so dass die Schnittknoten an dieselbe Stelle eingebettet werden.

(v) Dies ist lediglich ein Sonderfall von (iv).

□

Bemerkung 4 *Es lassen sich nicht alle generisch starren Graphen durch die Schritte aus Lemma 9 charakterisieren. Abbildung 5.9 zeigt ein einfaches Gegenbeispiel.*

Finden verträglicher Einbettungen

Auch die Frage nach verträglichen Einbettungen, ob eindeutig oder nicht, gestaltet sich sehr viel übersichtlicher. Unter Kenntnis einer Richtungszuweisung α ist eine Längenzuweisung ℓ entweder mit α nicht zu vereinbaren ist, oder eine Einbettung ist durch α und ℓ bis auf Drehung und Verschiebung eindeutig beschrieben. Die Suche nach verträglichen Einbettungen für einen Graphen G mit einer Richtungszuweisung α lässt sich daher auch vereinfachen zu einer Suche nach verträglichen Längenzuweisungen,

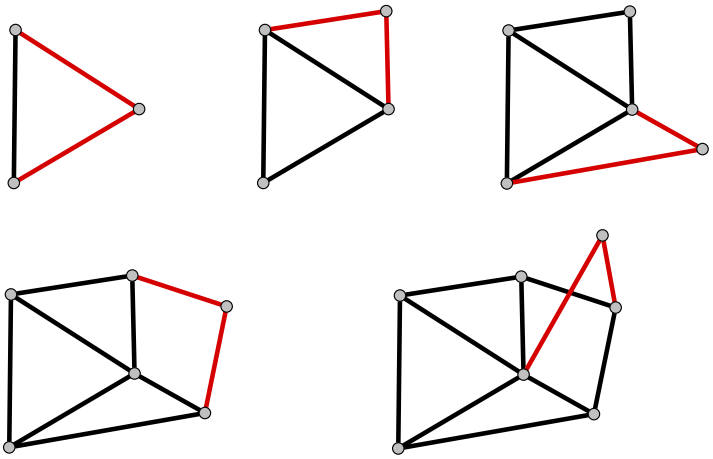


Abbildung 5.7.: Ein 2-simpler Graph

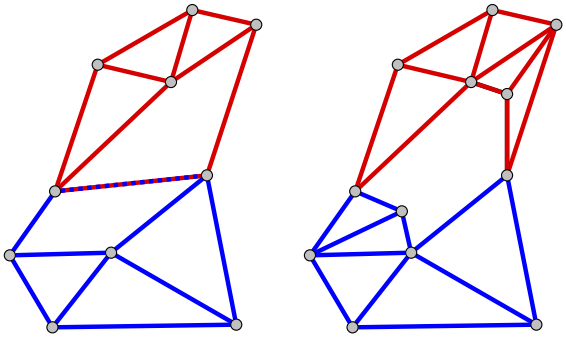


Abbildung 5.8.: Kantenüberlappung und Knotenüberlappung 2-simpler Graphen

wobei man zugleich auf die Freiheitsgrade der Drehung und Verschiebung verzichtet. Einzig die Streckung bleibt; ist eine Längenzuweisung ℓ für einen Graphen G mit α vereinbar, dann ist es auch $c\ell$ für alle $c \in \mathbb{R}^+$. Nun lässt sich eine solche Längenzuweisung wiederum sehr leicht charakterisieren, wenn man sich klarmacht, dass jede Zuweisung von *positiven* Kantenlängen an einen Spannbaum zulässig sein muss, und das erst Kreise solche Längenzuweisungen weiter einschränken.

Lemma 10 *Sei $G = (V, E)$ ein zusammenhängender Graph mit einer Richtungszuweisung α. Dann sei $S \subseteq E$ ein Spannbaum von G. Sei für eine Nichtspannbaumkante $(i, j) \in E \setminus S$ mit $C_{ij} = \{(i_1, j_1), \ldots, (i_k, j_k)\}$ der (eindeutige) Pfad von j nach i auf dem Spannbaum. Seien dann $C_{ij}^+ := \{(i, j) \in S \mid (i, j) \in C\}$ die* Vorwärtskanten *und analog $C_{ij}^- := \{(i, j) \in S \mid (j, i) \in C\}$ die* Rückwärtskanten *Eine Längenzuweisung ℓ*

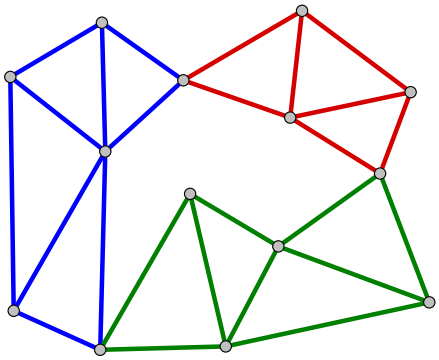


Abbildung 5.9.: Generisch starrer Graph, der sich nicht durch einen Schritt aus Lemma 9 aufbauen lässt.

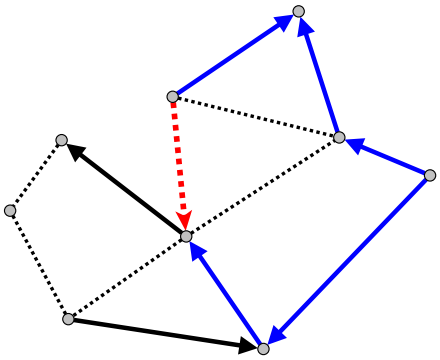


Abbildung 5.10.: Jede Nichtspannbaumkante (gestrichelt) schließt einen Kreis zusammen mit Spannbaumkanten. Jeder solche Kreis (z. B. aus der roten Nichtspannbaumkante und den blauen Spannbaumkanten) lässt sich nach Kanten aufteilen, die in dieselbe Richtung zeigen wie die Nichtspannbaumkante zeigen (C^+) und die anderen (C^-), wenn man von einer beliebigen Kantenrichtung ausgeht.

ist genau dann vereinbar mit α, wenn für alle Nichtspannbaumkanten $(i,j) \in E \setminus S$:

$$\sum_{(u,v)\in C^+\cup\{(i,j)\}} \ell(u,v)\cdot\alpha(u,v)_x - \sum_{(u,v)\in C^-} \ell(u,v)\cdot\alpha(u,v)_x$$
$$\sum_{(u,v)\in C^+\cup\{(i,j)\}} \ell(u,v)\cdot\alpha(u,v)_y - \sum_{(u,v)\in C^-} \ell(u,v)\cdot\alpha(u,v)_y$$

Beweis. Zum einen ist klar, dass die Bedingungen für jede verträglich Einbettung erfüllt sein müssen, denn sie beschreiben nach x- und y-Koordinate getrennt genau den Kreis aus Kanten, den die Nichtspannbaumkante auf dem Spannbaum schließt (siehe Abbildung 5.10. Zeichnet man zum anderen den Spannbaum mit den Längen

einer Lösung der Gleichungen, dann ist für die entstehende Einbettung $\mathbf{p}$

$$(\mathbf{p}_j - \mathbf{p}_i)_x = \sum_{(u,v)\in C^+} \ell(u,v)\cdot\alpha(u,v)_x - \sum_{(u,v)\in C^-} \ell(u,v)\cdot\alpha(u,v)_x = \ell(i,j)\cdot\alpha(i,j)_x$$

(analog für die y-Koordinate). Die Längenzuweisung ℓ entspricht also $\ell_{G,\mathbf{p}}$ und ist damit konsistent.

□

Korollar 2 *Identifiziert man eine Längenzuweisung mit einem Punkt des $(\mathbb{R}^+)^{|E|}$, dann entsprechen zulässige Lösungen den primal zulässigen Lösungen dieses Linearen Programmes*

$$\begin{array}{ll} \max & f(\ell) \\ u.d.N. & A(G,\alpha,S)\cdot\ell = \mathbf{0} \\ & \ell \geq \mathbf{0} \end{array} \tag{5.1}$$

für eine beliebige lineare Optimierungsfunktion f und $A(G,\alpha,S)$, die sich aus den in Lemma 10 beschriebenen Gleichungen ergebende Matrix.

Korollar 3 *Zu gegebenem Graphen G und einer Richtungszuweisung α lässt sich in Polynomialzeit entscheiden, ob es eine verträgliche Einbettung $\mathbf{p}$ gibt und gegebenenfalls eine solche angeben.*

Korollar 4 *Sei ein Graph $G = (V,E)$ generisch starr, S ein beliebiger Spannbaum und $\mathbf{p}$ eine allgemeine Einbettung in die Ebene. Dann ist der Kern von $A(G,\alpha_{G,\mathbf{p}},S)$ eindimensional und für $\mathbf{0} \neq l \in \ker A(G,\alpha_{G,\mathbf{p}},S)$ ist entweder $l \geq \mathbf{0}$ oder $-l \geq \mathbf{0}$. Entsprechend induzieren $\alpha_{G,\mathbf{p}}$ und l bzw. $-l$ die bis auf Drehung, Streckung und Verschiebung eindeutige Einbettung $\mathbf{p}$.*

5.4. Höhere Dimensionen

Ein weiterer Unterschied zwischen den vorgestellten Ergebnissen für bekannte Kantenlängen beziehungsweise Kantenrichtungen ist der, dass sich die Ergebnisse für bekannte Richtungen leicht verallgemeinern lassen. Es bleibt hier dabei, dass parallele Starrheit dasselbe ist wie parallele Eindeutigkeit, und diese lässt sich auch durch eine allgemeine Form von Lamans Bedingung charakterisieren:

Theorem 2 (Parallele Starrheit im $\mathbb{R}^d$) *Ein Graph ist generisch parallel d-starr genau dann, wenn es eine Kantenmenge $E' \subseteq E$ gibt mit $(d-1)|E'| = d|V| - (d+1)$ so, dass für alle nichtleeren $E'' \subseteq E'$*

$$(d-1)|E''| \leq d|V[E'']| - (d+1) \ .$$

Analoges Vorgehen scheitert für die Variante, in der Kantenlängen bekannt sind, leider völlig. Hier lässt sich noch nicht einmal die generische Starrheit leicht verallgemeinern. Es sei allerdings nicht verschwiegen, dass es eine Vielzahl von hinreichenden Charakterisierungen generisch 3-starrer Klassen von Graphen gibt. Siehe hierzu wieder [Whi96].

6. Richtungsbasierte Lokalisation in Quasi-Unit-Disk-Graphen

Eine mit einer lokalen Richtungszuweisung verträgliche Einbettung eines Graphen zu bestimmen ist nicht schwerer als das Lösen eines Linearen Programms. In Szenarien mit Zehntausenden von Knoten ist das immer noch außerhalb des Möglichen, insbesondere, wenn man vor Augen hat, dass bei jedem nicht verteilten Vorgehen die Stärke eines Sensornetzwerkes, viele Berechnungen parallel durchführen zu können, völlig brachliegt, während die Schwäche, die begrenzte Leistungsfähigkeit jedes einzelnen Knotens, voll zum Tragen kommt. Auf der anderen Seite sind in Kapitel 5 keine weiteren Annahmen an die Struktur der Netzwerke eingeflossen, wie wir sie in Kapitel 4 bereits kennengelernt haben. Im folgenden werde ich ausgehend von einigen Annahmen beschreiben, durch welche Schritte sich die Anzahl der Variablen schnell reduzieren lassen, und wie eine große Abdeckung einer Lokalisierung auch ohne das Lösen eines Linearen Programmes oder vergleichbar aufwendigen Verfahren erreichen kann. Sie bauen auf den Ideen von Bruck et al. [BGJ05] auf, gehen aber ab Abschnitt 6.2.2 über diese hinaus und werden außerdem eingehend auf ihre Laufzeit hin untersucht. Die vorgestellten Verfahren werden hier nur sequentiell entworfen und implementiert, allerdings werde ich einige Gedanken zur verteilten Berechnung im Ausblick in Kapitel 9 aufgreifen.

Das Problem, mit dem ich mich im weiteren beschäftige, ist eine Form des Realisierungsproblems für lokalen Richtungszuweisungen für Quasi-Unit-Disk-Graphen mit einem beliebigen Parameter $d \in (0, 1]$. Das Problem lässt sich also beschreiben als

Realität Ein Parameter d und ein zusammenhängender d-Quasi-Unit-Disk-Graph $G = (V, E)$ mit einer entsprechenden Einbettung $\mathbf{p}$ in die Ebene, einschließlich einer Ausrichtung der Sensoren

Abstraktion Der Parameter d, der Graph G und die lokale Richtungszuweisung $\omega_{G,\mathbf{p}}$

Rekonstruktion Eine Einbettung $\mathbf{q}$ hoher Qualität gemessen an der Realität $\mathbf{p}$ beziehungsweise der Abstraktion

Der letzte Punkt ist bewusst vage gehalten; es werden verschiedene Qualitätskriterien untersucht.

Zusätzlich werde ich von zwei weiteren Voraussetzungen ausgehen, die für realistische Szenarien sehr plausibel sind und die Analyse drastisch vereinfachen: Davon

ausgehend, dass die Knoten üblicherweise zumindest in entfernter Näherung zufällig, gleichmäßig und nicht unnötig dicht in einem gewissen Gebiet verteilt sind, kann man zum einen voraussetzen, dass sich die Knoten in allgemeiner Lage befinden, d. h. dass es keine algebraischen Abhängigkeiten zwischen den tatsächlichen Knotenposition gibt, und dass zum anderen von einem durch eine Konstante beschränkten maximalen Knotengrad von G ausgegangen werden kann, also $\Delta \in \mathcal{O}(1)$ und damit auch $m \in \mathcal{O}(n)$. Zur Motivation des letzten Punktes lässt sich ergänzen, dass sich natürlich auch durch gleichmäßiges Ausbringen von immer mehr Knoten in einem Gebiet fester Größe jeder Schranke für einen maximalen Knotengrad brechen lässt. Dem gegenüber steht allerdings, dass ab einer bestimmten Dichte die Qualität einer Rekonstruktion kaum mehr zunimmt; will man einen maximalen Knotengrad der Eingabe garantieren, kann man die Eingabe ohne signifikante Einbußen sukzessive um Knoten mit zu hohem Grad bereinigen. Darüber hinaus werden wir später (in Abschnitt 6.3.1) sehen, wie sich für Knoten ab einem bestimmten Knotengrad in einer Vorberechnung schnell inzidente Kanten entfernen ließen.

6.1. Datenstruktur

Zunächst werde ich eine Datenstruktur einführen, auf der die folgenden Algorithmen aufbauen. Sie vereinfacht hoffentlich die Verständlichkeit und Analyse und baut im wesentlichen auf einer listenbasierten Datenstruktur für Graphen auf, indem sie parasitär Informationen zu Knoten und Kanten speichert. Abbildung 6.1 zeigt den prinzipiellen Aufbau in UML-ähnlicher Weise. Nicht näher bezeichnet sind die Details der aus `Graph`, `Knoten` und `Kante` gebildeten Graphendatenstruktur (siehe dazu Abschnitt 3.3 zu Graphendatenstrukuren). Diese Klassen sind nur aufgeführt, um zu zeigen, welche Teile direkt mit ihnen korrespondieren.

Die Datenstruktur lässt sich zunächst nach Funktion der Klassen beschreiben und dann nach der der Methoden, die bei der Lokalisierung verwendet werden.

Klassen und Felder

`Netzwerk, Sensor, Verbindung` Diese Strukturen organisieren das Sensornetzwerk als Graph. Zusätzlich kennt jede Verbindung die beiden lokalen Winkelzuweisungen bezüglich der verbundenen Sensoren (`richtung`).

`Einbettung` Diese Struktur repräsentiert (partielle) Einbettungen der Sensoren eines Netzwerks Dabei werden die von einer Einbettung erfassten Knoten in einer Liste gespeichert (`sensoren`), und alle Einbettungen sind in einer Liste des Netzwerks gespeichert. Außerdem enthält jede Verbindung eine Liste der Einbettungen, bezüglich der *beide* betroffenen Sensoren bereits lokalisiert sind.

`Lokalisierung` Eine Lokalisierung bezeichnet die Einbettung eines einzelnen Knotens. Sie besteht aus reellen x- und y-Koordinaten, einem Winkel als Ausrichtung

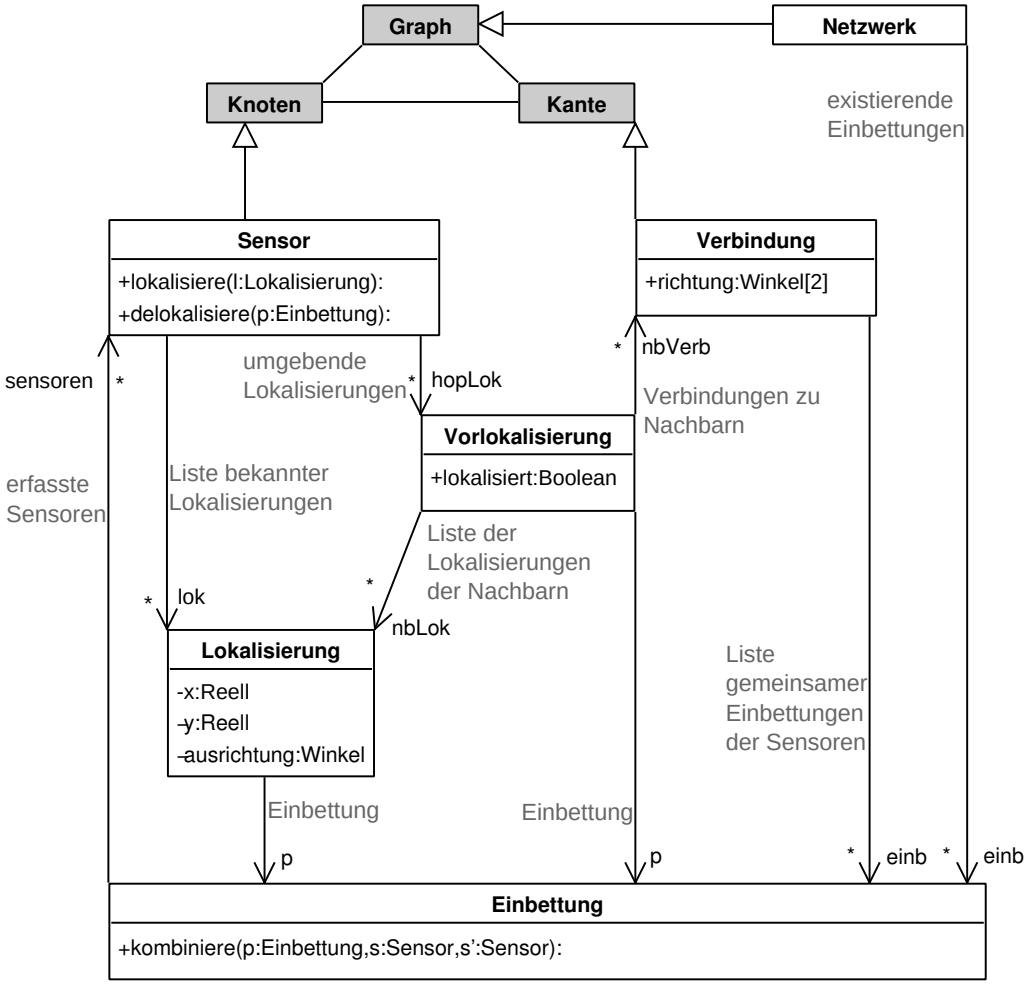


Abbildung 6.1.: Die verwendete Datenstruktur: Netzwerk, Sensoren und Verbindungen entsprechen der Datenstruktur für Graphen. Weiter gibt es (partielle) Einbettungen bestehend aus Lokalisierungen (aus Koordinaten und Ausrichtung) von Sensoren. Jeder Sensor kennt alle ihn betreffenden Lokalisierungen und die Lokalisierungen der benachbarten Sensoren, geordnet nach Einbettungen als Vorlokalisierungen. Die mit „⋆“ gekennzeichneten Felder sind Listen.

und der Einbettung, zu der sie gehört.

`Vorlokalisierung` Eine Vorlokalisierung enthält die Informationen, die einem Knoten aus den Lokalisierungen seiner Nachbarn zur Verfügung steht. Jedem Sensoren sind Vorlokalisierungen zu einer Einbettung bekannt, sobald mindestens ein Nachbar von dieser Einbettung erfasst wurde. Eine Vorlokalisierung besteht aus

der betroffenen Einbettung und zwei Listen, in denen für jeden erfassten Nachbarn zum einen dessen Lokalisation und zum anderen die Kante, über die man ihn erreicht, aufgeführt wird. Die beiden Listen sind parallel organisiert, d. h. gleichzeitiges Traversieren liefert immer zueinander gehörige Paare. Sie enthält außerdem die Information, ob der Sensor selbst bereits von der Einbettung erfasst wird. Solange eine Vorlokalisierung Informationen über eine Einbettung enthält, die den entsprechenden Knoten noch nicht erfasst, d. h. wenn `lokalisiert=false`, spreche ich auch von einer *offenen* Vorlokalisierung.

Methoden Während der Beschreibung der wenigen wichtigen Methoden, die für Algorithmen zur Verfügung stehen, lässt sich gleich feststellen, welche Laufzeitschranken sich zusichern lassen. Dabei werde die jeweils aktuelle Anzahl der Lokalisierungen bzw. Vorlokalisierungen eines Sensoren s als $k_L(s)$ bzw. als $k_V(s)$ bezeichnet.

`lokalisiere`(s, l) Ein Sensoren kann einer Einbettung hinzugefügt werden. Dabei wird die Lokalisierung der entsprechenden Liste des Sensoren in konstanter Zeit hinzugefügt, sowie der Sensor der Liste der lokalisierten Sensoren der Einbettung, ebenfalls in konstanter Zeit. Die Liste der Vorlokalisierungen wird nach einem entsprechenden Eintrag in $\mathcal{O}(k_V(s))$ Schritten durchgegangen und `lokalisiert` auf `true` gesetzt. Für alle verbundenen Sensoren werden die Vorlokalisierungen auf einen Eintrag zur betroffenen Einbettung durchsucht (gegebenenfalls wird einer eingefügt) und um die Lokalisierung l und die zugehörige Verbindung erweitert. Ist der Nachbarsensor s' bereits lokalisiert, wird die Einbettung der Liste `einb` der Verbindung hinzugefügt. Insgesamt sind das $\mathcal{O}(\deg s + \sum_{s' \in \mathrm{Nb}(s)} k_V(s'))$ Schritte. Wichtig ist: Da der Grad aller Sensoren durch eine Konstante Δ beschränkt ist, lässt sich `lokalisiere`(s, l) in konstanter Zeit ausführen, wenn ein Algorithmus garantiert, dass sich auch k_V für jeden Knoten durch eine Konstante beschränken lässt.

`delokalisiere`$(s, \mathbf{p})$ Die Lokalisierung eines Sensoren in einer Einbettung kann rückgängig gemacht werden, indem die zugehörige Lokalisation selbst wieder aus der Liste entfernt wird (in $\mathcal{O}(k_L(s))$ Schritten), und ansonsten analog die Schritte von `lokalisiere`(s, l) rückgängig gemacht werden. Entsprechend lässt sich diese Aufgabe in $\mathcal{O}(k_L(s) + \deg s + \sum_{s' \in \mathrm{Nb}(s)} k_V(s'))$ Schritten durchführen, also auch in konstanter Zeit, wenn der Algorithmus garantiert, dass sich für alle Sensoren die Anzahl der Lokalisierungen und der Vorlokalisierungen durch Konstanten beschränken lassen.

`kombiniere`$(\mathbf{p}, \mathbf{p}', s, s')$ Haben zwei Sensoren s und s' jeweils Lokalisierungen bezüglich zweier Einbettungene $\mathbf{p}$ und $\mathbf{p}'$, lassen sich die Koordinaten *eindeutig* zwischen den Einbettungen umrechnen, wenn man dafür nur Transformationen zulässt, die sich aus einer Skalierung, einer Drehung und einer Verschiebung zusammensetzen. Das wird unmittelbar einsichtig, wenn man sich überlegt, dass die Skalierung sich aus den Abständen in den Einbettungen eindeutig ergibt,

die Drehung ebenso eindeutig festgelegt ist, und sich danach die Verschiebung ergibt, wenn für jeden der beiden Sensoren gelten muss

$$c \cdot R(\alpha) \cdot \mathbf{p}'(s) + (x, y) = \mathbf{p}(s)$$

und $c \neq 0$ und α bekannt sind. In diesem Fall kann man auf die Einbettung $\mathbf{p}'$ verzichten und dafür für alle Sensoren, die bezüglich $\mathbf{p}'$ lokalisiert waren, die Lokalisierungen bezüglich $\mathbf{p}$ ausrechnen. Dies geschieht, indem über die Liste aller bezüglich $\mathbf{p}'$ lokalisierten Sensoren iteriert wird und dabei die Lokalisation bezüglich $\mathbf{p}'$ rückgängig gemacht wird (`delokalisiere`), und diese Sensoren bezüglich $\mathbf{p}$ lokalisiert (`lokalisiere`) werden. Es entsteht also bei konstanter Beschränkung von k_V und k_L maximal linearer Aufwand in der Anzahl der in $\mathbf{p}'$ lokalisierten Sensoren.

Die Initialisierung dieser Struktur benötigt offenbar nur lineare Laufzeit; es werden nur die Strukturen für Sensoren, Verbindungen und das Netzwerk angelegt, alle Listen sind initial leer, es gibt keine Einbettungen, Lokalisierungen oder Vorlokalisierungen.

6.2. Kantenüberlappende Triangulierung

Wie schon in Abschnitt 5.3.2 beschrieben, gibt es sehr schlichte Graphen, deren Einbettung bei bekannten Kantenrichtungen eindeutig zu rekonstruieren ist. Der erste Schritt nach der Initialisierung bestimmt zunächst die Einbettungen von maximalen *kantenüberlappenden 2-simplen* Subgraphen in $\mathcal{O}(n \log n)$ Schritten bei einem Speicherbedarf von $\mathcal{O}(n)$. Algorithmus 2 beschreibt die Vorgehensweise. Sie folgt der in [BGJ05] geäußerten Idee, zur Reduzierung der Variablenzahl, für die allerdings kein Algorithmus oder Laufzeitschranke angegeben wird. Hierauf liegt im folgenden der Schwerpunkt. Dabei bezeichne zu jedem Zeitpunkt $V_{\mathbf{p}}$ die von einer Einbettung $\mathbf{p}$ erfassten Sensoren.

Der Algorithmus besteht aus drei *elementaren Schritten*, jeweils den Rümpfen der Zeilen S (Saat), W (Wachstum) und K (Kombination) entsprechend. In kurzen Worten besteht die Vorgehensweise daraus, eine Kante (i, j) als Grundlage einer Einbettung zu wählen und die Endpunkte entsprechend zu lokalisieren (siehe Abbildung 6.2) als

$$\begin{array}{lcllcl} \mathbf{p}(i) & := & (0,0) & \mathbf{p}^{\uparrow}(i) & := & -\omega(i,j) \\ \mathbf{p}(j) & := & (1,0) & \mathbf{p}^{\uparrow}(j) & := & \pi - \omega(j,i) \ . \end{array}$$

Das entspricht in Algorithmus 2 den Zeile *S* (Saat) folgenden Zeilen. Solange nun ein Sensor s existiert, der zwei Nachbarn t und t' hat, die von einer Einbettung $\mathbf{p}$ lokalisiert sind, während er es nicht ist, wird einer dieser Sensoren lokalisiert. Da die allgemeine Lage der Sensoren vorausgesetzt wird, reicht dazu einfache Triangulierung (siehe Abbildung 6.3). Aus den bekannten, eingezeichneten Winkeln und Strecken

Algorithmus 2 : Kantenüberlappende Triangulierung

wiederhole
- **K** **wenn** ∃ Verbindung $\{i, j\}$ mit zwei Einbettungen $\mathbf{p}, \mathbf{p}'$ **dann**
  - **wenn** $|V_{\mathbf{p}}| \geq |V_{\mathbf{p}'}|$ **dann**
    - Kombiniere $\mathbf{p}, \mathbf{p}'$ zu $\mathbf{p}$ //$\mathbf{p}'$.`kombiniere`$(\mathbf{p}, i, j)$
  - **sonst**
    - Kombiniere $\mathbf{p}, \mathbf{p}'$ zu $\mathbf{p}'$ //$\mathbf{p}$.`kombiniere`$(\mathbf{p}', i, j)$
- **W** **sonst wenn** ∃ offene Vorlokalisierung v aus mind. 2 Nachbarn **dann**
  - lokalisiere v.`sensor` durch Triangulation, Abb. 6.3
- **S** **sonst wenn** ∃ Verbindung $\{i, j\}$ ohne Lokalisierung **dann**
  - Führe neue Einbettung $\mathbf{p}$ ein
  - $\mathbf{p}_i \leftarrow (0, 0)$, $\mathbf{p}_i^{\uparrow} \leftarrow -\omega(i, j)$ //i.`lokalisiere`$(\dots)$
  - $\mathbf{p}_j \leftarrow (1, 0)$, $\mathbf{p}_i^{\uparrow} \leftarrow \pi - \omega(j, i)$ //j.`lokalisiere`$(\dots)$

bis keiner der drei Fälle mehr erfüllt ist

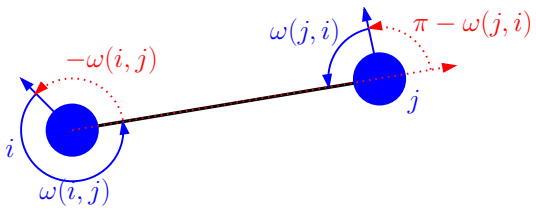


Abbildung 6.2.: Lokalisierung der ersten beiden Sensoren bei Einführung einer neuen Einbettung.

ergeben sich

$$
\begin{aligned}
|\mathbf{p}_s - \mathbf{p}_t| &= |\mathbf{p}_t - \mathbf{p}_{t'}| \cdot \frac{\sin\omega(t,s) - \sin\omega(t',s)}{-\arg(\mathbf{p}_t - \mathbf{p}_{t'}) + \mathbf{p}_{t'}^{\uparrow} + \omega(t',s)} \\
\arg(\mathbf{p}_s - \mathbf{p}_t) &= \mathbf{p}_t^{\uparrow} + \omega(t,s) \\
\mathbf{p}_s &= \mathbf{p}_t + |\mathbf{p}_s - \mathbf{p}_t| \cdot (\sin\arg(\mathbf{p}_s - \mathbf{p}_t), \cos\arg(\mathbf{p}_s - \mathbf{p}_t)) \\
\mathbf{p}_s^{\uparrow} &= \mathbf{p}_t^{\uparrow} + \omega(t,s) + \pi - \omega(s,t) \ .
\end{aligned}
$$

Das entspricht in Algorithmus 2 Zeile W (Wachstum) und den folgenden. Lässt sich so kein weiterer Knoten bezüglich $\mathbf{p}$ lokalisieren, wird die nächste Kante ohne Einbettung, d. h. ohne gemeinsame Einbettung der Endknoten, für eine neue Einbettung ausgewählt, und wieder sukzessive erweitert. Wird während dieses Verfahrens einer Verbindung eine zweite Einbettung zugeordnet, werden entsprechend Zeile Kff. in Algorithmus 2 vor weiteren Schritten die beiden Einbettungen kombiniert. Die Grundlage dafür bildet die Überlegung in Abschnitt 6.1 zur Implementierung von `kombiniere`. Die umgekehrte Reihenfolge dieser drei elementaren Schritte im Algorithmus entspricht der jeweiligen Priorität.

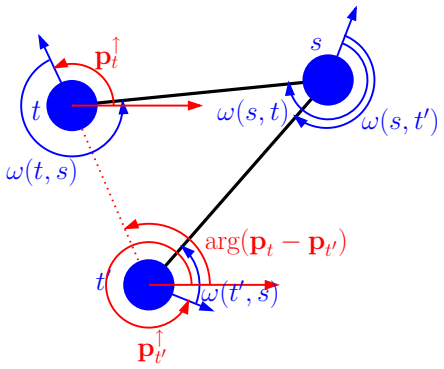


Abbildung 6.3.: Lokalisierung durch Triangulation bei der Anbindung eines Sensoren an eine bestehende Einbettung

6.2.1. Laufzeit und Speicherbedarf

Grundlegend für die Laufzeitschranken sind die folgenden Überlegungen von Lemma 11 und Lemma 12.

Lemma 11 *Folgende Invarianten gelten nach jedem Durchlauf einer der Blöcke S, W und K:*

1. *Die von einer Einbettung erfassten Sensoren induzieren einen Subgraphen $G[V_{\mathbf{p}}]$, der entweder dem K_2 entspricht oder aber zweifach knotenzusammenhängend ist.*
2. *Jeder Sensor hat zu jedem Zeitpunkt maximal $\Delta + 1$ Lokalisierungen.*
3. *Jede Verbindung ist zu jedem Zeitpunkt maximal von $\Delta + 1$ Einbettungen lokalisiert.*
4. *Jeder Sensor hat zu jedem Zeitpunkt maximal $\Delta^2 + \Delta$ Vorlokalisierungen.*

Beweis.

1. Eine Einbettung fängt immer bei zwei Knoten an, die verbunden sind (K_2). Wächst eine Einbettung durch Lokalisierung eines neuen Knotens, dann ist dieser Knoten durch mindestens zwei Kanten mit bereits lokalisierten Knoten verbunden, der Subgraph bleibt also zweifach knotenzusammenhängend. Werden zwei Einbettungen kombiniert und das Ergebnis wäre nicht zweifach knotenzusammenhängend, gäbe es einen Knoten, bei dessen Entfernung der abgedeckte Subgraph zerfiele. Da aber bei Entfernung eines Knotens keiner der beiden Teile zerfällt und die beiden auch immer noch mindestens einen Knoten gemeinsam haben, muss auch das Ergebnis der Kombination zweier Einbettungen zweifach knotenzusammenhängend sein.

2. Die Anzahl der Lokalisierungen eines Knotens erhöht sich jedesmal bei einer Lokalisation durch Einführen einer neuen Einbettung oder durch Anbinden an eine bestehende; auf jeden Fall nicht durch das Kombinieren von Einbettungen. Beim Kombinieren von Einbettungen verringert sich die Anzahl der Lokalisierungen eines Knotens, wenn der Knoten in beiden lokalisiert war. Eine Erhöhung der Anzahl der Lokalisierungen ist nur dann nicht direkt von einer Verringerung gefolgt, wenn sie nur Kanten betrifft, die noch gar keine Lokalisierung haben (sonst wäre der nächste Schritt eine Kombination der beiden Einbettungen). Das ist aber nur Δ mal möglich. Immer wenn also eine $(\Delta + 1)$-te Lokalisation hinzukommt, muss der nächste Schritt eine Kombination der beiden Einbettungen sein.

3. Jede Verbindung kann nur von Einbettungen erfasst sein, in denen beide Sensoren lokalisiert sind. Das können offenbar nicht mehr sein als $\Delta + 1$, wenn jeder der Sensoren nur so viele Lokalisierungen haben kann.

4. Jeder Sensor hat nur maximal Δ Nachbarn mit jeweils maximal $\Delta + 1$ Lokalisierungen, also Vorlokalisierungen bezüglich maximal $\Delta^2 + \Delta$ Einbettungen.

□

Lemma 12 *Der Aufwand für das Ermitteln von Verbindungen ohne Lokalisierung, Verbindungen mit mehr als einer Lokalisierung und Vorlokalisierungen mit mehr als einer Lokalisierung kann im $\mathcal{O}$-Kalkül vernachlässigt werden.*

Beweis. Eine Liste mit Verbindungen ohne Lokalisierungen lässt sich in $\mathcal{O}(n)$ initialisieren, dieser Aufwand kann der Initialisierung der Verbindungen zugeschlagen werden. Diese Liste und eine Liste der Verbindungen mit mehr als einer Lokalisierung (die leer initialisiert wird) lassen sich stets aktuell halten, indem beim Hinzufügen einer Einbettung zu einer Verbindung diese Verbindung aus den beiden Listen gelöscht bzw. eingefügt wird (jeweils in konstanter Zeit, siehe Abschnitt 3.3 zu verwendeten Listen). Die Liste mit den Vorlokalisierungen mit mehr als einer Lokalisierung wird leer initialisiert. Eine Vorlokalisierung wird ihr in konstanter Zeit hinzugefügt, wenn einer Vorlokalisierung eine zweite Lokalisierung angefügt wird und ebenfalls in konstanter Zeit entfernt, wenn entweder `lokalisiert` auf `true` gesetzt wird oder entfällt, weil die zugehörige Einbettung in einer anderen aufgegangen ist. In beiden Fällen lässt sich der Aufwand leicht diesen Operationen zuschlagen. □

Lemma 13 *Die Anzahl der elementaren Schritte S, W und K liegt jeweils in $\mathcal{O}(n)$.*

Beweis. Diese Aussage ist offensichtlich für Schritt S, da jede Kante maximal einmal betroffen sein kann. Entsprechend gilt es auch für Schritt K, denn wenn nur $\mathcal{O}(n)$ Einbettungen eingeführt werden, muss auch die Anzahl der Kombinationen in $\mathcal{O}(n)$

liegen, weil dabei immer eine Einbettung gelöscht wird. Für Schritt W lässt sich die Schranke einsehen, wenn man sich klarmacht, dass für jede Lokalisierung eines Sensors s in einer Einbettung zwei Nachbarn t, t' von s „verantwortlich“ sind, die bereits lokalisiert sind. Solche Paare von Nachbarn gibt es nur $\frac{\Delta^2-\Delta}{2}$ viele, und es lässt sich leicht einsehen, dass jedes Paar maximal $\frac{\Delta}{2}$ mal für die Lokalisation Pate stehen können, denn ist ein Paar t, t' von Nachbarsensoren k mal Grundlage einer Lokalisation eines Sensoren s, dann lässt sich zeigen, dass t mindestens den Grad $2k$ hat, indem induktiv jeder Lokalisation exklusiv zwei zu t inzidente Kanten zugewiesen werden:

IV Nach der ersten Lokalisierung enthält der Subgraph zur entsprechenden Einbettung mehr als nur eine Kante, ist also nach Lemma 11 zweifach knotenzusammenhängend, es lassen sich also zwei Kanten von t auswählen und dieser Lokalisation zuordnen. Die zwei Fälle dazu zeigt Abbildung 6.4.

IS Vor jeder weiteren Lokalisierung müssen t und t' bezüglich der entsprechenden Einbettung $\mathbf{p}$ lokalisiert sein. Der zugehörige Subgraph kann nicht *nur* die Kante $\{t, t'\}$ enthalten, denn dann wäre diese Einbettung in Schritt S entstanden, eine Verbindung $\{t, t'\}$ hätte aber schon vor der ersten Lokalisierung von s durch t und t' lokalisiert sein müssen. Der zugehörige Subgraph ist also zweifach knotenzusammenhängend und enthält damit auch mindestens zwei zu t inzidente Kanten. Hätten diese einen Schnitt mit bereits zugeordneten Kanten, dann käme es vor einer Lokalisierung von s zu einem Schritt K. Diesen Ablauf skizziert das dritte Bild von Abbildung 6.4.

□

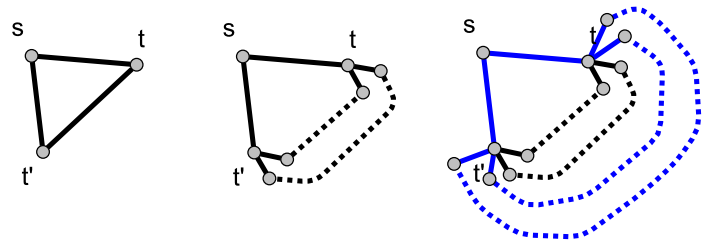


Abbildung 6.4.: Skizze zu Lemma 13: Für die erste Lokalisierung von s muss t mindestens den Grad zwei haben (links, mitte). Gegebenenfalls (links) muss man dafür die Kante zu s mitzählen. Für jede weitere Lokalisierung von s mmuss t mit t' über zwei *neue* Pfade verbunden sein, sonst wären die Einbettungen kombiniert worden

Theorem 3 *Die Laufzeit von Algorithmus 2 liegt in* $\mathcal{O}(n \log n)$*, der Platzbedarf in* $\mathcal{O}(n)$*.*

Beweis. *Laufzeit:* Nach Lemma 11 sind die Bedingungen erfüllt, damit eine Lokalisierung in konstanter Zeit durchgeführt werden kann. Damit lassen sich die jeweils $\mathcal{O}(n)$ Ausführungen der Schritte S und W auch in insgesamt $\mathcal{O}(n)$ Schritten durchführen. Den Zeitaufwand von $\mathcal{O}(n \log n)$ sieht man ein, wenn man sich überlegt, dass bei jeder Kombination zweier Einbettungen $\mathbf{p}$ und $\mathbf{p}'$ mit $|V_{\mathbf{p}}| \geq |V_{\mathbf{p}'}|$ jeder Sensor $s \in V$ einem der vier folgenden Fälle zuzuordnen ist, wobei jeweils konstanter Aufwand für alle Sensoren entsteht, die bezüglich $\mathbf{p}'$ lokalisiert sind:

$s \notin V_{\mathbf{p}} \cup V_{\mathbf{p}'}$: In diesem Fall ist s nicht betroffen, es entsteht kein Aufwand.

$s \in V_{\mathbf{p}} \setminus V_{\mathbf{p}'}$: Auch in diesem Fall ist für s nichts zu tun, es entsteht wieder kein Aufwand.

$s \in V_{\mathbf{p}'} \setminus V_{\mathbf{p}}$: In diesem Fall ist die Lokalisation bezüglich $\mathbf{p}'$ zu löschen und die Lokalisation bezüglich $\mathbf{p}$ einzufügen. Nach den Überlegungen aus Abschnitt 6.1 und Lemma 11 reicht dafür konstante Zeit.

$s \in V_{\mathbf{p}} \cap V_{\mathbf{p}'}$: In diesem Fall ist die Lokalisation bezüglich b schon vorhanden, aber die Lokalisation bezüglich $\mathbf{p}'$ ist zu entfernen. Das benötigt nicht mehr Aufwand als der vorangegangene Fall.

Interessant sind also nur letzten beiden Fälle. Wie oft kann ein Sensor einem dieser Fälle angehören? Jeder Sensor kann nur maximal konstant oft gemäß Schritt S lokalisiert werden und auch nur konstant oft gemäß Schritt W (siehe den Beweis zu Lemma 13). Die Anzahl der Lokalisation von s kann sich also nur konstant oft erhöhen. Im letzten Fall für $s \in V_{\mathbf{p}} \cap V_{\mathbf{p}'}$ nimmt die Anzahl der Lokalisationen des Sensoren ab, dieser Fall kann also auch nur konstant oft eintreten. Betrachtet man auf der anderen Seite eine Einbettung, in der s in Schritt S oder W lokalisiert wurde, lässt sich deren Anzahl durch eine Konstante abschätzen. Somit liegt die Anzahl der Kombinationen, bei der diese Einbettung bis zum Ende der Laufzeit als kleinerer Teil mit einer anderen Einbettung kombiniert wird, in $\mathcal{O}(\log n)$ (siehe dazu das folgende Lemma 14). Somit kann auch die Zahl der Kombinationen, für die für einen Knoten s der dritte Fall eintritt, in $\mathcal{O}(\log n)$. Insgesamt fällt für jeden Sensoren also nur ein Aufwand von $\mathcal{O}(\log n)$ Operationen in Schritt K an, also über alle Sensoren $\mathcal{O}(n \log n)$ *Platzbedarf:* Der Platzbedarf ergibt sich aus dem vorangegangenen Überlegungen, dass die Strukturen für Sensor und Verbindung nur $\mathcal{O}(n)$ oft angelegt werden müssen, insgesamt auch nur $\mathcal{O}(n)$ Einbettungen eingeführt werden und sich die Länge aller Listen jeweils durch Konstanten beschränken lassen. Da aber die Strukturen für Lokalisationen und Vorlokalisationen nur in solchen Listen vorkommen, kann es zu jedem Zeitpunkt nur linear viele Instanzen geben mit einem Gesamtplatzbedarf in $\mathcal{O}(n)$. □

Lemma 14 *Eine Einbettung kann nur in Einer Häufigkeit aus $\mathcal{O}(\log n)$ kleinerer Teil bei einer Kombination sein, wenn für jeden Sensor nur $c \in \mathbb{N}$ mal der Fall eintreten darf, dass er vor einer Kombination in beiden Teilen vorkommen darf.*

Beweis. Induktiv lässt sich zeigen, dass nach k Kombinationen, bei denen die betrachtete Einbettung jeweils die kleinere ist, mindestens $2^{\lfloor \frac{k}{c+1} \rfloor}$ Sensoren lokalisiert sind. Das gilt offenbar vor der ersten Kombination, denn eine Einbettung existiert nur mit mindestens 2 lokalisierten Sensoren. Hat aber eine Einbettung nach k solchen Kombinationen $2^{\lfloor \frac{k}{c+1} \rfloor}$ lokalisierte Sensoren, so muss für jeden Sensoren in mindestens einer der folgenden $c+1$ Kombinationen mit einem größeren Teil ein neuer Sensor hinzukommen, denn in mindestens einer dieser Kombinationen darf der Sensor nicht in beiden Teilen gleichzeitig liegen: Bei jeder Kombination mit n_- Knoten exklusiv im kleineren Teil und n_+ Knoten exklusiv im größeren bei $n_=$ gemeinsamen Knoten hat das Ergebnis $2n_- + n_= \geq 2n_-$ Knoten. Damit gilt aber, dass nach $k+c+1$ Schritten mindestens $2 \cdot 2^{\lfloor \frac{k}{c+1} \rfloor} = 2^{\lfloor \frac{k+c+1}{c+1} \rfloor}$ Sensoren in der betrachteten Einbettung (beziehungsweise ihren Nachfolgern) liegen müssen. Da aber jede Einbettung nur maximal n Sensoren abdecken kann, gilt

$$n \geq 2^{\lfloor \frac{k}{c+1} \rfloor} \Longrightarrow \mathrm{ld} n \geq \left\lfloor \frac{k}{c+1} \right\rfloor \Longrightarrow \mathrm{ld} n + 1 \geq \frac{k}{c+1} \Longrightarrow \underbrace{(c+1) \cdot (\mathrm{ld} n + 1)}_{\in \mathcal{O}(\log n)} \geq k$$

□

6.2.2. Nachbedingungen und Erweiterung

Das Ergebnis von Algorithmus 2 lässt sich vor allem durch die Subgraphen charakterisieren, die von den partiellen Einbettungen abgedeckt werden.

Lemma 15 *Nach Algorithmus 2 ist jede Verbindung in genau einer Einbettung lokalisiert. Die Subgraphen die in einer Einbettung lokalisiert sind, sind genau maximale kantenüberlappende 2-simple Subgraphen.*

Beweis. Dass jede Verbindung in genau einer Einbettung lokalisiert ist, ist klar, sonst hätte der Algorithmus nicht terminiert. Dass jeder lokalisierte Subgraph kantenüberlappend 2-simpel ist, ist ebenso klar, denn das ist eine Invariante über jeden der drei elementaren Schritte S, W und K. Andersherum gibt es für jeden maximalen kantenüberlappenden 2-simplen Subgraphen eine Folge von elementaren Schritten (jeweils in Korresponz mit einem der Punkte zum induktiven Aufbeu solcher Graphen aus Lemma 9), die dazu führen, dass dieser Subgraph von einer Einbettung abgedeckt wird. Betrachtet man diese Folge von Schritten *nach Algorithmus 2*, so sind entweder alle im Ergebnis bereits vorweggenommen, d. h. das Auslegen eines K_2 betrifft eine Kante, die schon irgendwo eingebettet ist, die Knotenaddition betrifft einen Knoten, der schon von derselben Einbettung erfasst wird, die Kantenüberlappung betrifft zwei Subgraphen, die schon gemeinsam eingebettet sind, oder aber es gäbe einen ersten

Schritt, der einen Subgraphen echt vergrößern würde[1]. Dieser Schritt wäre dann aber auch in Algorithmus 2 schon ausgeführt worden. □

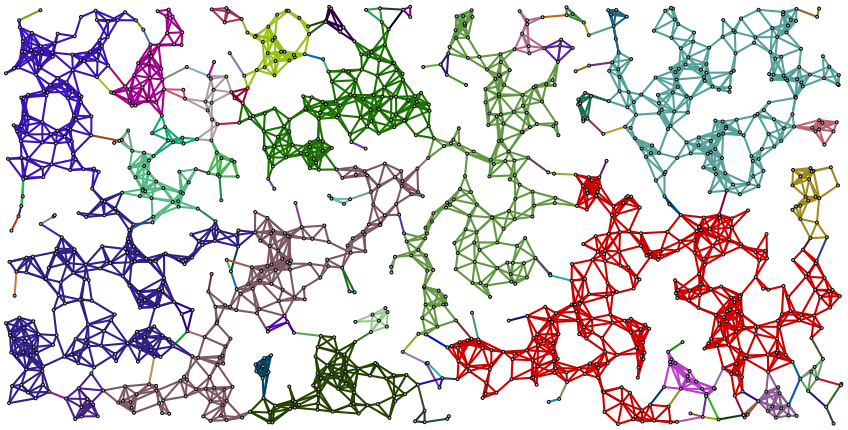


Abbildung 6.5.: Ergebnis der kantenüberlappenden Triangulierung am Beispiel eines Graphen mit 1400 Knoten bei einer Knotendichte von 3.5. Der größte lokalisierte Subgraph ist rot eingefärbt.

Dieses Verfahren hat in vielen Fällen schon eine enorme Abdeckung, d. h. der größte so lokalisierte Subgraph enthält schon für geringe Dichten einen Großteil der Sensoren. Abbildung 6.6 zeigt diesen Anteil in Abhängigkeit von der Sensordichte bei zufälligen Quasi-Unit-Disk-Graphen für $d = 0.5$. Da solche Graphen bei geringer Dichte nicht notwendig zusammenhängend sein müssen, ist die Größe eines maximalen zusammenhängenden Subgraphen mit angegeben. Auf ihn beschränkte sich der Algorithmus.

Dieses Verfahren lässt sich leicht dahingehend erweitern, dass es auch maximal 2-überlappende 2-simple Subgraphen lokalisiert. Dazu ist es notwendig, Kombinationsoperationen nicht nur dann durchzuführen, wenn es ein *adjazentes* Sensorpaar gibt, das in zwei Einbettungen lokalisiert wurde, sondern auch, wenn dies für ein *beliebiges* Sensorpaar gilt. Führt man diesen Algorithmus 3 nach Algorithmus 2 aus, bezeichne n_E die Anzahl der Einbettungen nach Algorithmus 2.

Lemma 16 *Die Erweiterung um Algorithmus 3 lässt sich so implementieren, dass sich Laufzeit und Platzbedarf um $\mathcal{O}(n_E^2)$ erhöhen.*

Beweis. Indiziert man die bestehenden Einbettungen eineindeutig als $\mathbf{p}_1, \ldots, \mathbf{p}_{n_E}$ (neue Einbettungen kommen nicht mehr hinzu), reicht es, ein zusätzliches $n_E \times n_E$-Array von Listen $L_{i,j}$ anzulegen und

[1] Man beachte, dass in Algorithmus 2 die Kantenaddition quasi *en passant* immer dann ausgeführt wird, wenn sie möglich ist

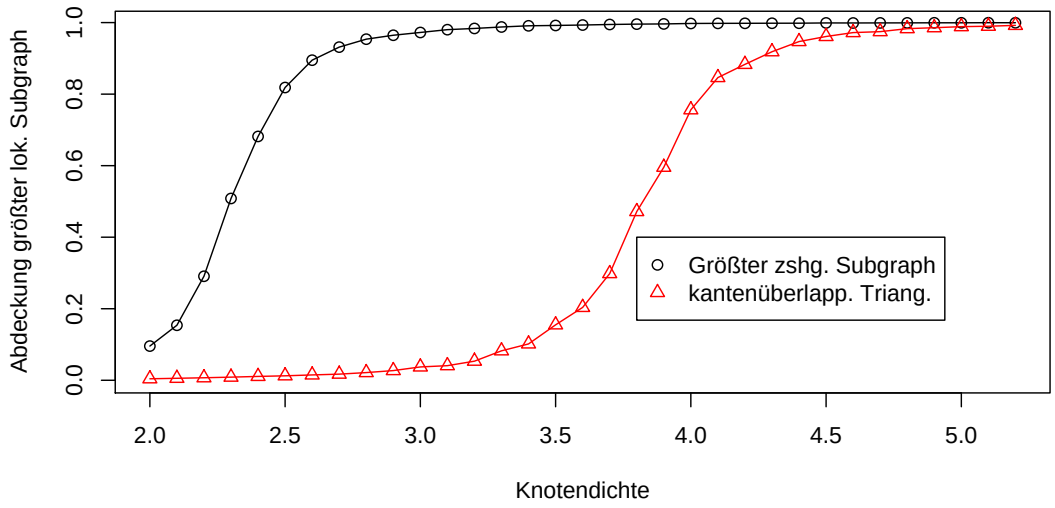


Abbildung 6.6.: Abdeckung des größten von Algorithmus 2 eingebetten Subgraph bei zunehmender Knotendichte im Vergleich zur Größe des maximalen zusammenhängenden Subgraphen bei 8000 Knoten.
[50 Messungen, Breite des 0.95-Vertrauensintervalls immer unter 0.1]

(i) initial alle Knoten zu durchlaufen und für je zwei Lokalisierungen $\mathbf{p}_i, \mathbf{p}_j$ eines Knotens s den Knoten s der Liste $L_{i,j}$ hinzuzufügen, sofern dort noch keine zwei Knoten vorhanden sind,

(ii) bei jeder Lokalisation eines Knotens s in einer Einbettung $\mathbf{p}_i$, der bereits bezüglich $\mathbf{p}_{j_1}, \ldots, \mathbf{p}_{j_r}$ lokalisiert ist, den Listen L_{i,j_k} den Sensor s hinzufügen,

(iii) beim Kombinieren einer Einbettung alle zugehörigen Listen zu leeren,

(iv) eine Schlange zu verwalten, in der alle Listen $L_{i,j}$ vermerkt sind, die schon zwei Knoten enthalten.

Damit liegt der Platzbedarf sicher in $\Theta(n_E^2)$, aber allein die Initialisierung benötigt auch eine Laufzeit in $\Theta(n_E^2)$. Darüber hinaus erhöht sich die Laufzeit aber nicht, denn bei Initialisierung und der Lokalisation eines Knotens sind immer nur maximal $\Delta + 1$ Lokalisierungen vorhanden, also sind von (i) maximal $\frac{\Delta^2+\Delta}{2}$ Listen $L_{i,j}$ betroffen, von (ii) maximal Δ viele. (iii) entspricht amortisiert dem Aufwand der Initialisierung und (iv) ist leicht durch parasitäres Speichern während der anderen Schritte zu erledigen. □

Lemma 15 gilt jetzt analog für Knotenüberlappung. Abbildung 6.8 zeigt, wie sich die Abdeckung durch diesen Schritt erhöht.

Algorithmus 3 : Knotenüberlappende Triangulierung

wiederhole
- I: **wenn** ∃ Sensorpaar $\{i, j\}$ mit zwei Einbettungenen $\mathbf{p}, \mathbf{p}'$ **dann**
  - **wenn** $|V_{\mathbf{p}}| \geq |V_{\mathbf{p}'}|$ **dann**
    - Kombiniere $\mathbf{p}, \mathbf{p}'$ zu $\mathbf{p}$ //$\mathbf{p}'$.`kombiniere`$(\mathbf{p}, i, j)$
  - **sonst**
    - Kombiniere $\mathbf{p}, \mathbf{p}'$ zu $\mathbf{p}'$ //$\mathbf{p}$.`kombiniere`$(\mathbf{p}', i, j)$
- W: **sonst wenn** ∃ offene Vorlokalisierung v aus mind. 2 Nachbarn **dann**
  - lokalisiere v.`sensor` durch Triangulation, Abb. 6.3

bis keiner der beiden Fälle mehr erfüllt ist

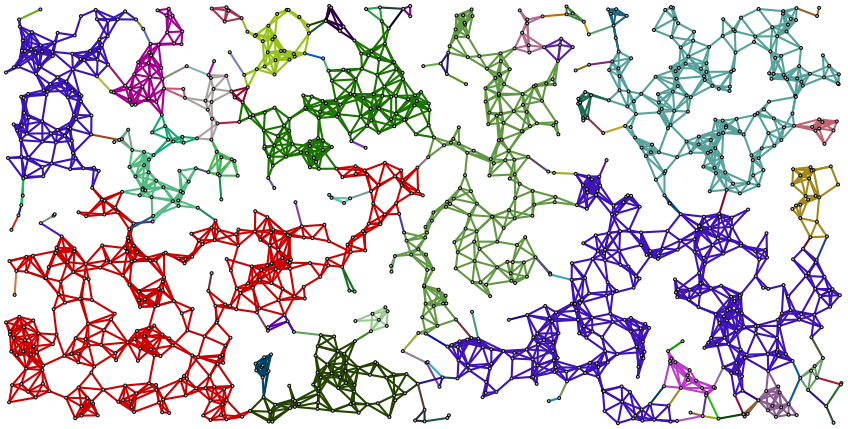


Abbildung 6.7.: Der Graph aus Beispiel 6.5 nach Algorithmus 3. Die Knotenüberlappung sorgt zumindest an einer Stelle für eine Kombination, was dazu führt, dass nun ein anderer Subgraph größter lokalisierter Subgraph ist (rot).

6.3. Maximale eindeutige Subgraphen

Die ersten beiden Stufen haben Sensoren so lokalisiert, dass maximale 2-knotenüberlappende 2-simple Graphen vollständig in einer Einbettung lokalisiert wurden. Damit ist aber noch nicht garantiert, dass tatsächlich auch maximale eindeutige Subgraphen lokalisiert wurden. Abbildung 5.9 zeigte bereits ein einfaches Gegenbeispiel.

Wir sind nach den bisherigen Schritten in der Situation, dass wir eine „Aufteilung“ des Graphen $G = (V, E)$ in Subgraphen $S_1 = (V_1, E_1), \ldots, S_{n_E} = \left(V_{n_E}, E_{n_E}\right)$ kennen, derart, dass die Kanten der Subgraphen die Kanten von $G = (V, E)$ partitionieren und dass eine Einbettung jedes der Subgraphen S_i durch die lokale Richtungszuweisung $\omega_{G,\mathbf{p}}$ eindeutig bis auf Drehung, Streckung und Verschiebung bestimmt ist (die S_i sind generisch starr und $\mathbf{p}$ ist eine allgemeine Einbettung). Für jeden S_i ist eine zulässige Einbettung $\mathbf{x}$ bekannt.

Ein Weg zur Einbettung größerer Subgraphen besteht nun darin, sukzessive Mengen

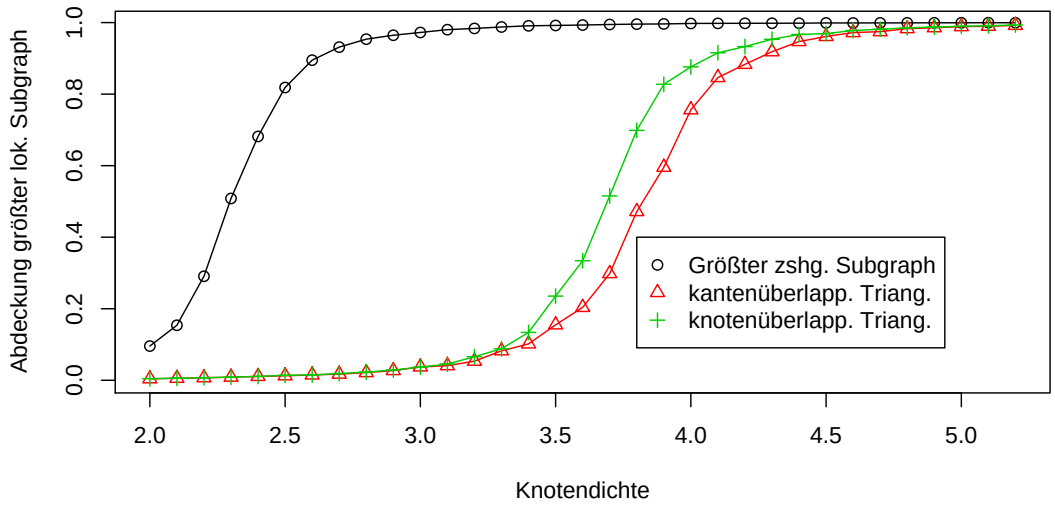


Abbildung 6.8.: Abdeckung von Algorithmus 2 und Algorithmus 3 bei zunehmender Knotendichte im Vergleich zur Größe des maximalen zusammenhängenden Subgraphen bei 8000 Knoten in einem quadratischen Gebiet
[50 Messungen, Breite des 0.95-Vertrauensintervalls immer unter 0.1]

von Subgraphen zu finden, die *zusammen* einen generisch starren Subgraphen bilden und für ihn gemäß Korollar 4 eine ebenso eindeutige Einbettung zu berechnen.

Definition 14 *Eine Menge von Subgraphen* $\mathcal{S} = \{S_1, \ldots, S_k\}$ *mit* $S_i = (V_i, E_i)$ *eines Graphen* $G = (V, E)$ *heiße* generisch starre Partitionierung *von* $G = (V, E)$*, wenn*

- *Die Subgraphen alle Knoten und alle Kanten abdecken; die Kanten dabei disjunkt, also*
$$V = \bigcup_{i=1}^{k} V_i \quad \textit{und} \quad E = \dot{\bigcup_{i=1}^{k}} E_i$$
und
- *alle Subgraphen* S_i *generisch starr sind.*

Es werden dann für jede Teilmenge $\mathcal{S}' \subseteq \mathcal{S} := \{S'_1 = (V'_1, E'_1), \ldots, S'_l = (V'_l, E'_l)\}$ *induzierte Knoten und Kantenmengen bezeichnet durch* $V(\mathcal{S}') := \bigcup_{i=1}^{l} V'_i$ *und* $E(\mathcal{S}') := \bigcup_{i=1}^{l} E'_i$*, der induzierte Subgraph durch* $G(\mathcal{S}') := (V(\mathcal{S}'), E(\mathcal{S}'))$ *und durch*

$$\mathrm{mv}_{\mathcal{S}'}(v) := |\{S' = (V', E') \in \mathcal{S}' \mid v \in V'\}|$$

die Vielfachheit *oder* Mehrfachverwendung *des Vorkommens eines Knotens in den ausgewählten Subgraphen.*

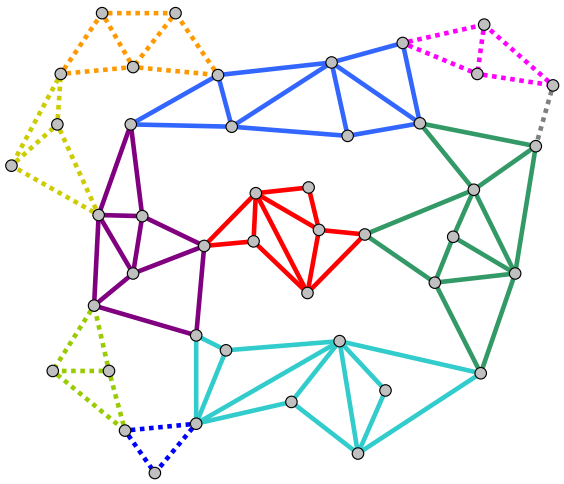


Abbildung 6.9.: Eine generisch starre Partitionierung eines Graphen mit eingefärbten Subgraphen. Die Menge der Subgraphen $\mathcal{S}'$ mit durchgezogenen Kanten ist minimal darin, dass $3\,(|\mathcal{S}'|-1) \leq 2\sum_{v\in V(\mathcal{S}')}(\mathrm{mv}_{\mathcal{S}'}(v)-1)$ und erfüllt damit Theorem 4. Der Übersichtlichkeit halber ist für jeden Subgraphen nur eine *unabhängige* Kantenmenge eingezeichnet.

Theorem 4 (Eindeutige Subgraphenmengen) *Sei $\mathcal{S}$ eine generisch starre Partitionierung von $G=(V,E)$. Ist $\mathcal{S}$* minimal *mit folgenden Eigenschaften:*

(a) $|\mathcal{S}| \geq 2$

(b) $3\,(|\mathcal{S}|-1) \leq 2\sum_{v\in V(\mathcal{S})}(\mathrm{mv}_{\mathcal{S}}(v)-1)$

dann ist $G(\mathcal{S})$ generisch starr.

Beweis. Jeder der Graphen $S_i=(V_i,E_i)\in\mathcal{S}$ ist generisch starr und hat $2|V_i|-3$ bezüglich jeder allgemeinen Einbettung $\mathbf{p}$ *unabhängige* Kanten. Wir betrachten für jeden dieser Graphen im folgenden *nur* noch eine solche Kantenmenge als E_i, willkürlich gewählt. Der Graph $G(\mathcal{S})$ hat nur

$$|V(\mathcal{S})| = \left|\bigcup_{(V_i,E_i)\in\mathcal{S}} V_i\right| = \sum_{(V_i,E_i)\in\mathcal{S}} |V_i| - \sum_{v\in V(\mathcal{S})}(\mathrm{mv}_{\mathcal{S}}(v)-1) \tag{6.1}$$

Knoten (die Knotenzahl summiert sich wegen Mehrfachvorkommen nicht einfach auf), aber

$$|E(\mathcal{S})| = \left|\dot{\bigcup}_{(V_i,E_i)\in\mathcal{S}} E_i\right| = \sum_{(V_i,E_i)\in\mathcal{S}} (2|V_i|-3) \tag{6.2}$$

Kanten. Aus 2. wird so

$$\begin{array}{rrcl} & 3\left(|\mathcal{S}|-1\right) & \leq & 2\sum_{v\in V(\mathcal{S})}\left(\mathrm{mv}_{\mathcal{S}}(v)-1\right) \\ \Leftrightarrow & 3|\mathcal{S}| & \leq & 2\sum_{v\in V(\mathcal{S})}\left(\mathrm{mv}_{\mathcal{S}}(v)-1\right)-3 \\ \Leftrightarrow & 2\sum_{(V_i,E_i)\in\mathcal{S}}|V_i|-3|\mathcal{S}| & \geq & 2\sum_{(V_i,E_i)\in\mathcal{S}}|V_i|-2\sum_{v\in V(\mathcal{S})}\left(\mathrm{mv}_{\mathcal{S}}(v)-1\right)-3 \\ \Leftrightarrow & \sum_{(V_i,E_i)\in\mathcal{S}}\left(2|V_i|-3\right) & \geq & 2\sum_{(V_i,E_i)\in\mathcal{S}}|V_i|-2\sum_{v\in V(\mathcal{S})}\left(\mathrm{mv}_{\mathcal{S}}(v)-1\right)-3 \\ \Leftrightarrow & |E(\mathcal{S})| & \geq & 2|V(\mathcal{S})|-3 \end{array} \tag{6.3}$$

Nun lässt sich Lamans Theorem (1) auch wie folgt lesen: *Ein Graph $G=(V,E)$ mit mindestens $2|V|-3$ Kanten ist entweder generisch starr, oder er enthält einen (nichtleeren) Subgraphen $G'=(V',E')$ mit $|E'|>2|V'|-3$ Kanten*[2]. Nehmen wir also an, $G(\mathcal{S})$ wäre nicht generisch starr. Dann gibt es einen *minimalen* Subgraphen $G'=(V',E')$ mit $|E'|>2|V'|-3$. Dieser Subgraph ist generisch starr, denn sonst gäbe es (wieder nach Lamans Theorem) einen kleineren Subgraphen im Widerspruch zur Minimalität von G'. Betrachtet man die Subgraphen $S_i\in\mathcal{S}$, die mit G' eine Kante teilen, die Menge von Subgraphen $\mathcal{S}'$, gegeben durch

$$\mathcal{S}':=\{S_i=(V_i,E_i)\in\mathcal{S}\mid E_i\cap E'\neq\emptyset\}\quad,$$

ist wegen der Kantenüberlappung generisch starrer Graphen klar, dass $G(\mathcal{S}')$ generisch starr ist. Nach Annahme ist aber $G(\mathcal{S})$ nicht generisch starr, also muss $\mathcal{S}'\subsetneq\mathcal{S}$ sein. Es gilt nun aber auch, dass $|E(\mathcal{S}')|\geq 2|V(\mathcal{S}')|-3$ und ganz analog zu Umformung 6.3, dass dann auch (b) erfüllt ist. Gleichzeitig muss aber auch (a) erfüllt sein, also $|\mathcal{S}'|\geq 2$, denn für $|\mathcal{S}'|=1$ wäre $G(\mathcal{S}')=S_i\in\mathcal{S}$ einer der generisch starren Graphen, die wir auf die notwendigen Kanten reduziert hatten, hier müsste eine Kantenmenge $E'=E_i$ unabhängig sein. Dass nun aber $\mathcal{S}'$ (a) und (b) erfüllt und gleichzeitig echt kleiner als $\mathcal{S}$, steht im Widerspruch zur Minimalität von $\mathcal{S}$. □

Dieses Theorem ergänzt gewissermaßen Lemma 9 um eine weitere Art, wie man aus generisch starren Graphen einen neuen generisch starren Graphen zusammensetzen kann, indem man sicherstellt, dass sie, ohne Kanten mehrfach zu nutzen, *insgesamt* genug Knoten mehrfach benutzen, ohne dass dies schon für eine Teilmenge der Fall ist. Genauer sind die Knotenüberlappung und damit auch die Kantenüberlappung nur Sonderfälle für $|\mathcal{S}|=2$. Der Beweis liefert eine weitere Aussage (fast) mit, die den vorangegangenen Charakterisierungen fehlte: Diese Charakterisierung ist erschöpfend.

Lemma 17 *Seien $\mathcal{S}'\neq\mathcal{S}$ generisch starre Partitionierung eines Graphen G so, dass $\mathcal{S}$ eine Vergröberung von $\mathcal{S}'$ darstellt, d. h. so, dass es für jedes $S_i'=(V_i',E_i')\in\mathcal{S}'$ ein $S_j=\left(V_j,E_j\right)\in\mathcal{S}$ gibt, so dass $V_i'\subseteq V_j$. Dann enthält $\mathcal{S}'$ eine Menge von Subgraphen $\mathcal{S}^s\subseteq\mathcal{S}'$ mit den in Theorem 4 geforderten Eigenschaften.*

Beweis. Da $\mathcal{S}$ eine echte Vergröberung von $\mathcal{S}'$ darstellt, muss es eine Menge $\mathcal{S}^{\mathrm{s}}=\{S_1^{\mathrm{s}}=(V_1^{\mathrm{s}},E_1^{\mathrm{s}}),\ldots,S_k^{\mathrm{s}}=(V_k^{\mathrm{s}},E_k^{\mathrm{s}})\}\subseteq\mathcal{S}'$ geben mit $|\mathcal{S}^{\mathrm{s}}|>1$ und ein $S_j=\left(V_j,E_j\right)\in\mathcal{S}$

[2] für mehr als $2|V|-3$ Kante kann auch beides eintreten

gibt, so dass $V_i^{\mathrm{s}} \subseteq V_j$ für alle $S_i^{\mathrm{s}} = (V_i^{\mathrm{s}}, E_i^{\mathrm{s}}) \in \mathcal{S}^{\mathrm{s}}$. Da G_j generisch starr sein sollte, erfüllt $\mathcal{S}^{\mathrm{s}}$ (entsprechend der Umformung 6.3) beide Bedingungen aus Theorem 4. Dann gibt es natürlich auch eine minimale solche Menge $\mathcal{S}^{\min} \subset \mathcal{S}'$. □

Das sukzessive Auffinden und Zusammenfassen von generisch starren Subgraphen in einer generisch starren Überdeckung nach Theorem 4 ist also erst dann nicht möglich, wenn eine generisch starre Überdeckung nicht mehr zu vergröbern ist, die enthaltenen generisch starren Subgraphen also maximal sind.

6.3.1. Graphen-Schnittknoten-Netzwerk

Eine generisch starre Partitionierung korrespondiert mit einem leicht zu bildenden bipartiten Graphen zwischen den Subgraphen und den Schnittknoten, wobei die Kanten dem Enthaltensein entsprechen:

Definition 15 *Sei $\mathcal{S}$ eine generisch starre Partitionierung eines Graphen $G = (V, E)$. Der bipartite Graph $B(\mathcal{S}) = (L, \mathcal{S}, A)$ mit*

$$\begin{aligned} L &= \{v \in V \mid \mathrm{mv}_{\mathcal{S}}(v) > 1\} \\ A &= \{(v, (V_i, E_i)) \mid v \in V_i\} \end{aligned}$$

heißt Mehrfachverwendungsgraph *von $\mathcal{S}$.*

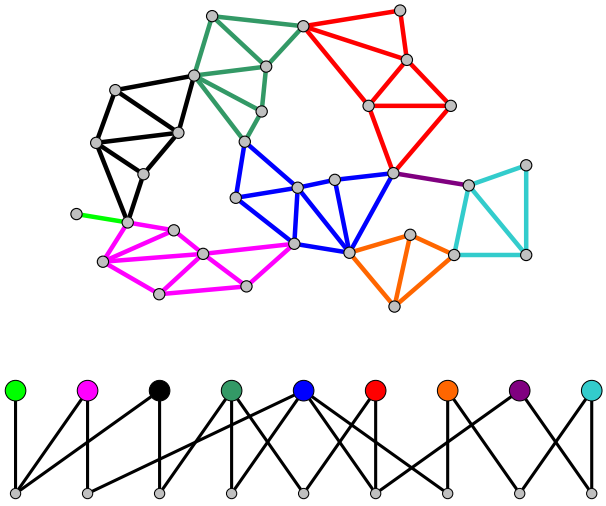


Abbildung 6.10.: Mehrfachverwendungsgraph einer generisch starren Partitionierung. Die Subgraphen und die zugehörigen Knoten im bipartiten Graph sind eingefärbt.

Bemerkung 5 *Ist $\mathcal{S}$ die generisch starre Partitionierung eines Graphen nach Algorithmus 2/3, dann sind $|L|, |\mathcal{S}|, |A| \in \mathcal{O}(n)$. Das folgt zum einen daraus, dass es nicht mehr Schnittknoten als Knoten (also $\mathcal{O}(n)$), nicht mehr (kantendisjunkte) Subgraphen als Kanten (also auch $\mathcal{O}(n)$) und nur konstant beschränkt viele Inklusionen jedes Schnittknotens in solchen Subgraphen geben kann, wenn der Grad jedes Knotens in $\mathcal{O}(1)$ liegt.*

Aber auch ohne die Gradbeschränkung, die für die Laufzeitabschätzung der Algorithmen 2/3 notwendig war, kann man sich überlegen, dass es *danach* unabhängig vom maximalen Knotengrad eine konstante Schranke für die Anzahl der Inklusionen jedes Knotens v in Subgraphen gibt, und somit die lineare Schranke für die Größe von A und $\mathcal{S}$:

Haben zwei Nachbarn von v einen Abstand geringer als d voneinander, dann haben sie sicher eine Verbindung und bilden damit ein Dreieck mit v. Sie liegen dann auf jeden Fall in demselben Subgraphen von $\mathcal{S}$. Auf die Art lässt sich *von vornherein* die Nachbarschaft von v als $\dot{\bigcup}_{i=0}^{h} N_i = \mathrm{Nb}(v)$ partitionieren (als transitiver Abschluss der Kantenrelation auf $G(\mathrm{Nb}(v))$, siehe Abbildung 6.11). Aus genannten Gründen ist für

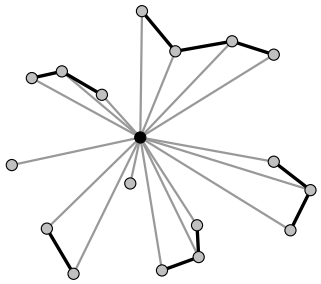


Abbildung 6.11.: Die Nachbarschaft jedes Knotens lässt sich nach Erreichbarkeit (innerhalb der Nachbarschaft) partitionieren

dieses v in $B(\mathcal{S})$ als Ergebnis von Algorithmus 2/3 die Anzahl der Kanten beschränkt durch h, denn Nachbarn, die Dreiecke mit v bilden, sind jeweils von derselben Einbettung erfasst. Nun nehmen wir an, es wäre möglich, einen Knoten und eine Nachbarschaft so einzubetten, dass $h > \left\lceil (2\sqrt{2}/d) \right\rceil^2 =: m(d)$. Wählt man aus jedem der N_i einen Knoten aus, hat man h Knoten, die *paarweise* unverbunden sind. Keiner der Knoten ist von v weiter entfernt als 1. Teilt man das achsenparallele Quadrat von $\mathbf{p}(v) - (1,1)$ bis $\mathbf{p}(v) + (1,1)$ aber in $m(d)$ Quadrate auf, muss es ein Quadrat geben, in dem mindestens 2 Knoten liegen. Es hat aber jedes Quadrat eine Kantenlänge kleiner oder gleich $2/\left(2\sqrt{2}/d\right) = d/\sqrt{2}$ und damit eine Diagonale kleiner oder gleich d. Die Knoten müssten somit verbunden sein im Widerspruch dazu, dass sie unterschiedlichen N_i entstammten. Es lässt sich festhalten, dass zu diesem Zeitpunkt kein Knoten mehr als $\left\lceil (2\sqrt{2}/d) \right\rceil^2$ Partitionen angehörte, selbst wenn der Knotengrad nicht a priori als konstant beschränkt angenommen werden könnte.

6.3.2. Bestimmung minimaler eindeutiger Subgraphenmengen

Theorem 4 und Lemma 17 liefern ein Verfahren, wie man ausgehend von einer generisch starren Partitionierung sukzessive vergröbert, bis es keine weitere Vergröberung gibt, und man damit auch maximale generisch starre Subgraphen identifiziert hat. Wie aber lassen sich Subgraphen einer generisch starren Partitionierung finden, die sich so zusammenfassen lassen?

Wir fassen dazu $B(\mathcal{S})$ als bipartites Flussnetzwerk (von L nach $\mathcal{S}$ auf) mit Knotenbewertungen $b : L \cup \mathcal{S} \to \mathbb{N}_0$ und Kantenkapazitäten $\kappa : A \to \mathbb{N}$. Ein Fluss $f : A \to \mathbb{N}_0$ heiße zulässig, wenn

$$\begin{array}{rrcl} \forall e \in A: & f(e) & \leq & \kappa(e) \\ \forall v \in L: & \sum_{e=(v,G)\in E} f(e) & \leq & b(v) \\ \forall G \in \mathcal{S}: & \sum_{e=(v,S)\in E} f(e) & \leq & b(S) \end{array}$$

und maximal, wenn $|f| := \sum_{e\in A} f(e)$ maximal unter allen zulässigen Flüssen ist.

Diese Formulierung eines Flussproblems lässt sich leicht auf ein einfaches Flussproblem mit einer Quelle und einer Senke zurückführen, indem man diese Knoten q und s, Kanten (s, v) mit Kapazität $b(v)$ und Kanten (S, t) mit Kapazität $b(S)$ zusätzlich einführt.

Hier liegen allerdings die Knotenbewertungen b nicht fest, sondern sie werden im Laufe des Verfahrens ändern. Kanten hingegen haben alle eine feste Kapazität von 2 ($\kappa \equiv 2$). Initial seien alle Bewertungen 0, und der Fluss $f \equiv 0$; f ist also zulässig. Zu sehen sind diese Ausgangswerte in Abbildung 6.12. Geändert werden dürfen Bewertungen nur erhöhend, dadurch bleiben zulässige Flüsse zulässig. Ein Fluss *sättigt* einen Subgraphen $S \in \mathcal{S}$, wenn $\sum_{e=(v,S)\in A} f(e) = b(S)$, er *lastet* einen Knoten $v \in L$ *aus*, wenn $\sum_{e=(v,S)\in A} f(e) = b(v)$.

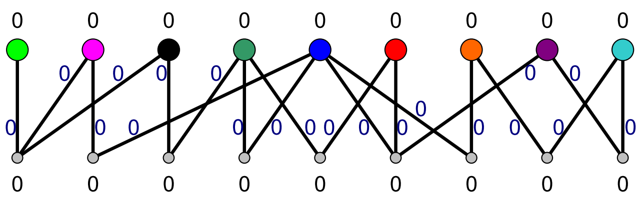


Abbildung 6.12.: Initiale Bewertungen und Fluss. Knoten und Subgraphen sind jeweils mit den Bewertungen, die Kanten mit dem Fluss markiert.

Grundlegender Algorithmus

Die grundlegende Vorgehensweise ist beschrieben in Algorithmus 4. Hier ist zu sehen, wie die Subgraphen aus $\mathcal{S}$ sukzessive einer Menge von „aktiven" Subgraphen

hinzugefügt werden (Zeilen 1/2). Der jeweils letzte hinzugefügte Subgraph S^{neu} hat $b(S^{\text{neu}}) = 0$ (wie initial), alle anderen aktiven Subgraphen S haben $b(S) = 3$ (Zeile 9). Für Knoten v, die in mindestens einem der Subgraphen aus $\mathcal{S}_{\text{aktiv}}$ enthalten sind, wird immer $b(v) = 2(\text{mv}_{\mathcal{S}_{\text{aktiv}}}(v) - 1)$ gesetzt (Zeile 4) und anschließend der Fluss maximiert. Diese ersten sechs Durchläufe sind in Abbildung 6.13 zu sehen.

Algorithmus 4 : Sukzessives Verschmelzen eindeutiger Subgraphenmengen

```
1  für alle S^neu ∈ 𝒮 tue
2      𝒮^aktiv ← 𝒮^aktiv ∪ {S^neu}
3      für alle v ∈ L : (v, S^neu) ∈ A tue
4          b(v) ← 2(mv_{𝒮^aktiv}(v) − 1)
5      maximiere f
6      wiederhole
7          (𝒮', L') ← starr(𝒮_aktiv, S^neu, f)
8          wenn (𝒮', L') ≠ (∅, ∅) dann  verschmelze Subgraphen aus 𝒮' zu S^neu
       bis (𝒮', L') = (∅, ∅)
9      b(S^neu) ← 3
```

Nach jeder Flussmaximierung wird überprüft, ob es eine Menge von Subgraphen gibt, die die Bedingungen aus Theorem 4 (minimal) erfüllen (Zeile 7). Das passiert in Algorithmus 5. Ist eine solche Menge gefunden worden, können die enthaltenen Subgraphen verschmolzen werden (Zeile 8). Auf die dafür notwendigen Schritte – das Berechnen einer gemeinsamen Einbettung und eine konsistente Anpassung von $B(\mathcal{S})$ – werde ich später noch eingehen. Dieser Schritt wird so lange wiederholt, bis in Zeile 7 keine Menge mehr gefunden wird, dann wird der nächste Subgraph aktiviert usw.

Algorithmus 5 : $\texttt{starr}(\mathcal{S}_{\text{aktiv}}, S^{\text{neu}}, f)$

```
1  für alle v ∈ L : (v, S^neu) ∈ A tue
2      für alle S ≠ S^neu ∈ 𝒮^aktiv : (v, S) ∈ A tue
3          𝒮' ← {S^neu, S'}
4          L' ← ∅
5          wiederhole
6              L' ← L' ∪ {v ∈ L | ∃S ∈ 𝒮' : f(v, S) > 0}
7              𝒮' ← 𝒮' ∪ {S ∈ 𝒮 | ∃v ∈ L' : f(v, S) < 2}
8              wenn 𝒮' einen ungesättigten Subgraphen enthält dann
9                  beginne mit nächstem Durchlauf von Schleife 2
10             gib zurück (𝒮', L')
           bis zum Abschluss
11 gib zurück (∅, ∅)
```

Algorithmus 5 überprüft, ob es in $\mathcal{S}^{\text{aktiv}}$ eine Menge gibt, die die in Theorem 4 ge-

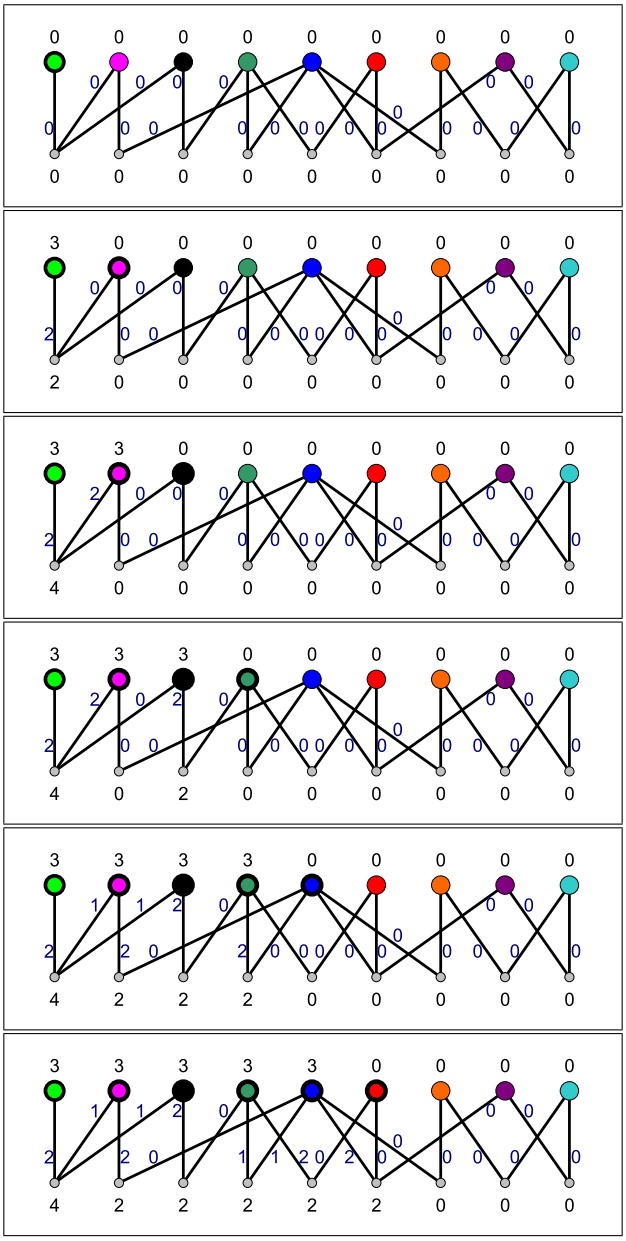


Abbildung 6.13.: Sukzessive Aktivierung der Subgraphen und Veränderung der Bewertungen. Aktivierte Subgraphen sind jeweils fett umrandet. Alle Flüsse sind maximal für den jeweiligen Schritt.

forderten Eigenschaften erfüllt. Dazu wird von dem letzten aktivierten Graphen S^{neu} und jeweils einem angrenzenden Subgraphen S (siehe Zeilen 1/2) der Abschluss über $(L' = \emptyset, \mathcal{S}' = \{S^{\text{neu}}, S\})$ so gebildet, dass L' am Ende alle Knoten enthält, die Einfluss nach $\mathcal{S}'$ haben und $\mathcal{S}'$ alle Subgraphen enthält, für die es eine Kante von einem Knoten $v \in L'$ gibt, die keinen maximalen Fluss führt. Anders ausgedrückt enthält der Abschluss immer genau die Subgraphen und Knoten, die auf einem erhöhenden

Pfad vom initialen $L' \cup \mathcal{S}'$ erreichbar sind. Abbildung 6.14 zeigt einen Fall, in dem

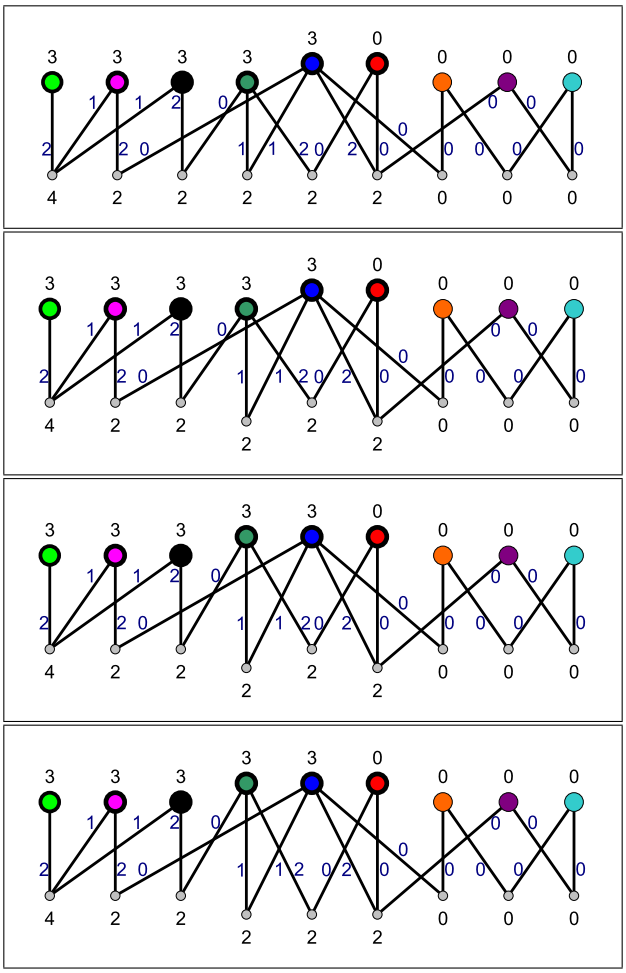


Abbildung 6.14.: Algorithmus 5 findet eine Menge von Subgraphen mit Schnittknoten als Abschluss, die zusammen generisch starr sind.

Algorithmus 5 ein Ergebnis zurückliefert. Die ausgewählten Subgraphen sind zusammen generisch starr (siehe dazu den ursprünglichen Graphen aus Abbildung 6.3.1). Im folgenden wird für $(\mathcal{S}', L')$ dieser Abschluss durch $(\overline{\mathcal{S}'}, \overline{L'})$ gekennzeichnet.

Korrektheit

Zunächst komme ich zur Korrektheit dieser Schritte *bis zur ersten Verschmelzung*, erst danach werde ich auf die Ausführung des Verschmelzungsschrittes aus Zeile 7 in Algorithmus 4 und die Korrektheit über diesen Schritt hinaus eingehen. Unter Korrektheit ist unter dieser Einschränkung zu verstehen, dass

1. eine Menge von Subgraphen zum Verschmelzen durch Algorithmus 5 ausgewählt

wird, sobald $\mathcal{S}^{\text{aktiv}}$ eine Menge von Subgraphen enthält, die zusammen einen generisch starren Subgraphen von G bilden und

2. eine ausgewählte Menge von Subgraphen immer einen generisch starren Subgraphen bildet.

Für beide Aussagen ist der Abschluss aus Algorithmus 5 besonders wichtig:

Lemma 18 *Werden bezüglich eines Flusses f Mengen $(\mathcal{S}', L')$ durch die Schritte der Zeilen 6/7 von Algorithmus 5 abgeschlossen, dann erfüllt für den Abschluss $(\overline{\mathcal{S}'}, \overline{L'})$ die Subgraphenmenge $\overline{\mathcal{S}'}$ das Kriterium (b) aus Theorem 4, wenn f alle Subgraphen aus $\overline{\mathcal{S}'}$ sättigt.*

Beweis. Durch die Art, wie die Mengen $\overline{L'}$ und $\overline{\mathcal{S}'}$ gebildet werden, gelten folgende zwei Eigenschaften:

(i) Subgraphen aus $\overline{\mathcal{S}'}$ bekommt nur Einfluss aus $\overline{L'}$

(ii) Knoten aus $\overline{L'}$ haben zu aktiven Subgraphen außerhalb von $\overline{\mathcal{S}'}$ immer den maximalen Fluss 2.

Zusätzlich gilt, dass Subgraphen aus $\overline{\mathcal{S}'}$ jeweils vollen Einfluss $b(S)$ haben. Der Einfluss nach $\overline{\mathcal{S}'}$ beträgt somit $3(|\overline{\mathcal{S}'}| - 1)$ (maximal der Subgraph S^{neu} hat $b(S^{\text{neu}}) = 0$, sonst gilt $b(S) = 3$). Für den Fluss nach $\overline{\mathcal{S}'}$ steht aber jedem Knoten $v \in \overline{L'}$ nur ein Fluss von $2\sum_{v\in\overline{L'}}(\text{mv}_{\overline{\mathcal{S}'}}(v) - 1)$ zur Verfügung, denn von den aktuellen $2\sum_{v\in\overline{L'}}(\text{mv}_{\mathcal{S}^{\text{aktiv}}}(v) - 1)$ muss für jeden Subgraphen $S \in \mathcal{S}^{\text{aktiv}} \setminus \overline{\mathcal{S}'}$ ja schon ein potentieller Fluss von 2 *nicht* nach $\overline{\mathcal{S}'}$ gehen. Da nun von Anfang an $|\overline{\mathcal{S}'}| \geq 2$ und $3(|\overline{\mathcal{S}'}| - 1) \leq 2\sum_{v\in\overline{L'}}(\text{mv}_{\overline{\mathcal{S}'}}(v) - 1)$, sind die beiden Kriterien erfüllt. □

Zunächst zum ersten Teil, dem Nachweis, dass Algorithmus 5 ein Ergebnis liefert, sobald $\mathcal{S}^{\text{aktiv}}$ eine generisch starre Vergröberung besitzt. Lässt sich $\mathcal{S}^{\text{aktiv}}$ vergröbern, dann gibt es nach Lemma 17 eine minimale Menge von Subgraphen $\mathcal{S}' \subseteq \mathcal{S}^{\text{aktiv}}$, die die Eigenschaften (a) und (b) aus Theorem 4 erfüllt. Davon ausgehend liefern die beiden folgenden Aussagen den gewünschten Nachweis.

Lemma 19 *Enthält $\mathcal{S}^{aktiv}$* durch Hinzufügen von S^{neu} *eine minimale Menge von Subgraphen $\mathcal{S}'$, die die Kriterien (a) und (b), erfüllen, dann sind nach der entsprechenden Flussmaximierung in Algorithmus 4 alle Subgraphen aus $\mathcal{S}'$ gesättigt.*

Beweis. Sei L' die Menge von Knoten, die zu mindestens zwei der Subgraphen aus $\mathcal{S}'$ gehören. Ignorieren wir zunächst, dass es auch aktivierte Subgraphen, die nicht zu $\mathcal{S}'$ und Knoten, die nicht zu L' gehören, gibt, dann ist die Bewertung der Knoten aus L' zusammen

$$\sum_{v\in L'} b(v) = \sum_{v\in L'} 2(\text{mv}_{\mathcal{S}'}(v) - 1) \geq 3(|\mathcal{S}'| - 1) = \sum_{S\in\mathcal{S}'} b(S) \ ,$$

also mindestens gleich der Bewertung der Subgraphen in $\mathcal{S}'$. Nehmen wir an, dass die Subgraphen aus $\mathcal{S}'$ nicht alle gesättigt sind, dann muss es einen Knoten $x \in L'$ geben, der noch keinen maximalen Ausfluss hat. Wird jetzt $(\mathcal{S}^x = \emptyset, L^x = \{x\})$ abgeschlossen wie in Algorithmus 5 zu $(\overline{\mathcal{S}^x}, \overline{L^x})$, also minimal gegen

1. $\mathcal{S}^x \leftarrow \mathcal{S}^x \cup \{S \in \mathcal{S}' \mid \exists v \in L^x : f(v,S) < 2\}$
2. $L^x \leftarrow L^x \cup \{v \in L' \mid \exists S \in \mathcal{S}^x : f(v,S) > 0\}$

dann ergibt sich folgendes Bild:

- $|\overline{\mathcal{S}^x}| \geq 2$, denn wie alle Knoten in L' ist die Knotenbewertung gerade ausreichend, um $\mathrm{mv}_{\mathcal{S}'}(x) - 1$ der ausgehenden $\mathrm{mv}_{\mathcal{S}'}(x)$ Kanten auszulasten. Da die Bewertung aber nicht ausgenutzt wurde, muss es schon zwei von x ausgehende Kanten (x, S) geben mit $f(x,S) < 2$.
- Alle Subgraphen in $\overline{\mathcal{S}^x}$ müssen gesättigt sein, denn durch die Art des Abschlusses ist sichergestellt, dass *alle* Subgraphen aus $\overline{\mathcal{S}^x}$ auf einem erhöhenden Pfad von x aus erreichbar sind. Der Fluss ist aber maximal nach Voraussetzung und x nicht ausgelastet.

Damit gilt für $(\overline{\mathcal{S}^x}, \overline{L^x})$ aber zusammen mit Lemma 18, dass $\overline{\mathcal{S}^x}$ die Kriterien (a) und (b) erfüllt. Das widerspricht aber entweder der Minimalität von $\mathcal{S}'$ (für $\overline{\mathcal{S}^x} \neq \mathcal{S}'$) oder der Annahme, dass ein Subgraph aus $\mathcal{S}'$ nicht gesättigt ist (für $\overline{\mathcal{S}^x} = \mathcal{S}'$).

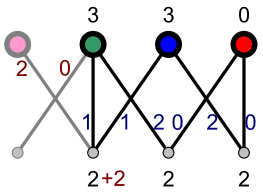


Abbildung 6.15.: Wird eine Menge von Subgraphen durch einen maximalen Fluss gesättigt von mehrfach genutzen Knoten gesättigt, ist das unabhängig von zusätzlichen Subgraphen oder Knoten.

Jetzt fehlt noch die Sicherheit, dass zusätzliche Knoten und Subgraphen daran nichts ändern können, dass bei einem maximalen Fluss alle Knoten von $\mathcal{S}'$ gesättigt sind: Zusätzliche Knoten könnten bestenfalls zusätzlichen Einfluss nach $\mathcal{S}'$ liefern, zusätzliche Subgraphen in $\mathcal{S}_{\text{aktiv}} \setminus \mathcal{S}'$ können aber von jedem der Knoten aus V' nur einen Fluss von 2 aufnehmen, während sie gleichzeitig die Knotenbewertung um zwei erhöhen. Abbildung 6.15 illustriert dies. □

Lemma 20 *Sobald eine Menge von Subgraphen aktiviert ist, die die Kriterien (a) und (b) erfüllen, dann liefert Algorithmus 5 ein Ergebnis.*

Beweis. Existiert eine solche Menge durch Hinzufügen von S^{neu}, dann existiert auch eine minimale solche $\mathcal{S}'$. Der letzte zu $\mathcal{S}^{\text{aktiv}}$ hinzugefügte Graph S^{neu} gehört ihr an, denn sonst hätte es schon vorher eine solche Menge von Subgraphen gegeben, außerdem sei S' ein weiterer Subgraph aus $\mathcal{S}'$, benachbart zu S^{neu}. Es bezeichne L' die Knoten mit $\text{mv}_{\mathcal{S}'}(v) \geq 2$. Nach Lemma 19 sind also nach der entsprechenden Flussmaximierung alle Subgraphen aus $\mathcal{S}'$ gesättigt. Betrachten wir den Durchlauf der in Zeile 5 beginnenden Schleife, in der der Abschluss $(\overline{\mathcal{S}''}, \overline{L''})$ über $(\mathcal{S}'' = \{S^{\text{neu}}, S'\}, L'' = \emptyset)$ gebildet wird. Angenommen, dieser wird nicht als Ergebnis zurückgeliefert. Dann enthält $\overline{\mathcal{S}''}$ einen nicht gesättigten Subgraphen $\hat{S} \notin \mathcal{S}'$. Die Art des Abschlusses garantiert, dass es von S' zu $\hat{S}$ einen erhöhenden Pfad gibt, denn S^{neu} ist für den Abschluss irrelevant wegen $b(S^{\text{neu}}) = 0$.

Es können die Knoten aus L' nur dann alle vollen Ausfluss haben, wenn alle Kanten von L' nach $\mathcal{S}^{\text{aktiv}} \setminus \mathcal{S}'$ vollen Fluss haben und gleichzeitig Subgraphen aus $\mathcal{S}'$ nur Fluss aus L' empfangen. Es muss auf dem Pfad von S' nach $\hat{S}$ aber entweder eine Kante $(v, S) \in (L \setminus L') \times \mathcal{S}'$ mit $f(v, S) > 0$ (hinzufügen von $v \notin L'$) oder eine Kante $(v, S) \in L' \times (\mathcal{S}^{\text{aktiv}} \setminus \mathcal{S}')$ mit $f(v, S) < 2$ (Hinzufügen von $S \notin \mathcal{S}'$) liegen. Also gibt es einen nicht ausgelasteten Knoten $x \in L'$ mit $\sum_{S \in \mathcal{S}^{\text{aktiv}}} f(x, S) < b(x)$.

Nun lässt sich wieder der Abschluss von $(\mathcal{S}^x = \emptyset, L_x = \{x\})$, $(\overline{\mathcal{S}^x}, \overline{L^x})$ bilden und analog zum Beweis von Lemma 19 gilt, dass $\overline{\mathcal{S}^x} \subset \mathcal{S}^{\text{aktiv}}$ (a) und (b) erfüllt. Dieser Abschluss muss $S^{\text{neu}} \in \overline{\mathcal{S}^x}$ enthalten, weil sonst vor dem Aktivieren von S^{neu} schon eine solche Menge existiert hätte, also auch eine minimale, die generisch starr gewesen wäre. Gleiches gälte, wenn $\overline{\mathcal{S}^x}$ nicht gleichzeitig auch noch einen an S^{neu} angrenzenden Subgraphen S'' enthielte. Damit muss aber zumindest der Durchlauf der in Zeile 5 beginnenden Schleife in Algorithmus 5, der den Abschluss über S^{neu} und S'' als Ergebnis eine Teilmenge von $\overline{\mathcal{S}^x}$ zurückgeliefert werden, denn der Abschluss kann nicht über $\overline{\mathcal{S}^x}$ hinausgehen und hier sind alle Subgraphen gesättigt. □

Damit haben wir die Garantie, dass, sobald eine Menge von Subgraphen, die zusammen einen generisch starren Subgraphen bildet, aktiviert ist, Algorithmus 5 ein Ergebnis liefert. Nun ist noch zu zeigen, dass sobald Algorithmus 5 ein Menge von Subgraphen als Ergebnis liefert, diese Subgraphen zusammen einen generisch starren Graphen bilden. Diese Aussage liefern die nächsten beiden Lemmata:

Lemma 21 *Findet Algorithmus 5 eine Menge von Subgraphen zum Verschmelzen, dann erfüllt diese die Kriterien (a) und (b) aus Theorem 4.*

Beweis. Diese Aussage folgt direkt aus Lemma 18: Algorithmus 5 bildet den Abschluss so, dass zum einen $\overline{\mathcal{S}'}$ mindestens zwei Subgraphen enthält, denn $\mathcal{S}'$ enthält schon zu Beginn zwei Subgraphen (Kriterium (a)) und zum anderen liefert es nur ein Ergebnis,

wenn alle Subgraphen aus $\overline{\mathcal{S}'}$ durch f gesättigt sind. Damit sind die Bedingungen aus Lemma 18 erfüllt, also auch Kriterium (b). □

Lemma 22 *Liefert Algorithmus 5 zum ersten Mal eine Menge von Subgraphen $\mathcal{S}^{res}$, dann ist $G(\mathcal{S}^{res})$ generisch starr.*

Beweis. Liefert Algorithmus 5 eine Menge von Subgraphen $\mathcal{S}^{\text{res}}$, dann erfüllt diese die Bedingungen (a) und (b) von Theorem 4. Diese Menge muss aber nicht notwendigerweise minimal sein (das zeigt Abbildung 6.16), ist aber trotzdem generisch starr:

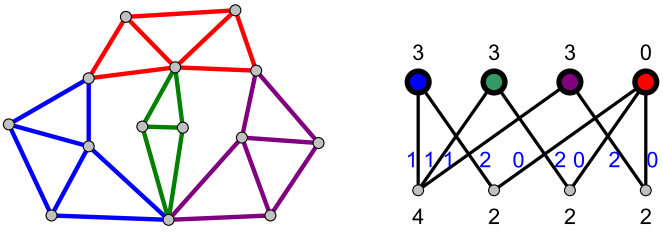


Abbildung 6.16.: In dieser Konstellation besteht die erste Menge von Subgraphen, die zurückgegeben wird, aus allen Subgraphen, obwohl jeweils drei minimal und generisch starr wären, wenn sie den roten Subgraphen einschließen.

Um das einzusehen, mache man sich zunächst klar, dass der letzte aktivierte Subgraph S^{neu} immer zu $\mathcal{S}^{\text{res}}$ gehört. Sonst hätte es schon früher eine Menge von Subgraphen gegeben, die zusammen (a) und (b) erfüllen und damit hätte Algorithmus 5 nach Lemma 20 auch schon früher ein Ergebnis geliefert. Außerdem ist entweder $\mathcal{S}^{\text{res}}$ selbst generisch starr, oder aber es gibt eine Teilmenge $\mathcal{S}^{\text{min}} \subseteq \mathcal{S}^{\text{res}}$ mit $|\mathcal{S}^{\text{min}}| \geq 2$, die generisch starr und minimal mit dieser Eigenschaft ist. Dabei muss wiederum $S^{\text{neu}} \in \mathcal{S}^{\text{min}}$ sein – mit derselben Begründung wie zuvor. Wir nennen die Knoten, die in in Subgraphen aus $\mathcal{S}^{\text{res}}$ vorkommen, L^{res}. Es reicht, zu zeigen, dass die Subgraphen aus $\mathcal{S}^{\text{res}}$ die Bedingungen (a) und (b) *immer noch* erfüllen, wenn man die Subgraphen aus $\mathcal{S}^{\text{min}}$ zu einem Subgraphen kontrahierte. Das kann man dann sukzessive wiederholen, bis $\mathcal{S}^{\text{res}}$ generisch starr ist; es kann schließlich nur endlich oft eine kleinere Teilmenge geben, die generisch starr und damit minimal ist, da sich die Größe von $\mathcal{S}^{\text{res}}$ mit jeder Kontraktion verkleinert. Bei der Kontraktion werden alle Subgraphen aus $\mathcal{S}^{\text{min}}$ durch einen Subgraphen $G[\mathcal{S}^{\text{min}}] =: S^{\text{min}}$ mit $b(S^{\text{min}}) = 0$ ersetzt, der damit die Stelle von S^{neu} einnimmt. Alle anderen Subgraphen aus $\mathcal{S}^{\text{aktiv}}$ werden einfach übernommen; $\hat{\mathcal{S}} := (\mathcal{S}^{\text{res}} \setminus \mathcal{S}^{\text{min}}) \cup \{S^{\text{min}}\}$ bezeichne die entstehende Subgraphenmenge. Die Knoten bleiben alle erhalten, alle Kanten zu Subgraphen aus $\mathcal{S}^{\text{aktiv}} \setminus \mathcal{S}^{\text{min}}$ ebenso, für jeden Knoten v, der zu mindestens einem Subgraphen aus $\mathcal{S}^{\text{min}}$ verbunden war, wird eine Kante (v, S^{min}) eingefügt. Abbildung 6.17 illustriert einen solchen Kontraktionsschritt.

Nun ist zu zeigen, dass $\mathcal{S}^{\text{res}}$ die Eigenschaften (a) und (b) nach der Kontraktion immer noch erfüllt. Dabei ist (a) trivial, denn da nicht alle Subgraphen kontrahiert werden,

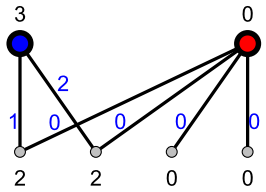


Abbildung 6.17.: Ergebnis nach der Kontraktion eines minimalen generisch starren Subgraphen aus einer Rückgabe; Bedingungen (a) und (b) bleiben erfüllt.

bleiben notwendig noch mindestens zwei übrig. Für (b) reicht es, zu zeigen, dass sich bei so einer Kontraktion ein konsistenter Fluss finden lässt, für den die Ergebnisse aus Lemma 18 weiter gelten. Dafür wählen wir den neuen Fluss $\hat{f}$ wie folgt:

- für alle $v \in L_{\text{aktiv}} \setminus L_{\min}, S \neq S^{\min}$ setze $\hat{f}(v,S) = f(v,S)$. Das schafft keine Probleme, da sich $b(v)$ für diese Knoten nicht ändert, weil sie ja keinem der kontrahierten Subgraphen angehören.
- für alle $v \in L_{\min}$ setze $\hat{f}(v,S^{\min}) = 0$. Auch hier kann es zu keiner Verletzung der Flusseigenschaften kommen.
- für alle $v \in L_{\min}, S \neq S^{\min}$ setze $\hat{f}(v,S) = f(v,S)$. Auch das schafft keine Probleme, denn der Einfluss in alle Subgraphen $S \neq S^{\min}$ bleibt insgesamt wie vor der Kontraktion (also 3), und für jeden Knoten $v \in V_i$ ist $b(v) = 2(\text{mv}_{\hat{\mathcal{S}}}(v)) - 1)$ bei $\text{mv}_{\hat{\mathcal{S}}}(v)) - 1$ ausgehenden Kanten und bereits einer Kante (der zu $S^{\min}$) mit Fluss 0, d. h. alle anderen Kanten können den alten Fluss (≤ 2) behalten.

Dieser Fluss sättigt alle Knoten, die vorher gesättigt waren, also auch alle aus $\mathcal{S}^{\text{res}}$, einschließlich des kontrahierten Knotens. □

Es fehlt nun die Zusicherung, dass nach einem Verschmelzungsschritt von Subgraphen aus $\mathcal{S}^{\text{aktiv}}$ die Voraussetzungen dieselben sind, wie wenn diese Subgraphen von vornherein verschmolzen gewesen wären. Genau genommen ist dieser im letzten Beweis schon umrissen worden, hier sei er der Vollständigkeit halber noch vollständig beschrieben.

Sind die Subgraphen $\mathcal{S}^{\text{aktiv}}$ aktiviert und wird eine Menge von Subgraphen $\mathcal{S}'$ mit zugehörigen Schnittknoten L' identifiziert, für die $G(\mathcal{S}')$ generisch starr ist und die den letzten aktivierten Subgraphen S^{neu} enthält, dann aktualisieren wir $B(\mathcal{S})$ durch folgende Schritte:

- Die Subgraphen aus $\mathcal{S}'$ außer S^{neu} werden entfernt.
- Alle Knoten aus $v \in L'$ werden mit S^{neu} verbunden, wenn sie es noch nicht sind. Hat ein Knoten danach *nur diese* Kante von S^{neu}, wird er gelöscht. Dabei wird die Bewertung jedes Knotens $v \in L'$ um $2(\text{mv}_{\mathcal{S}'}(v) - 1)$ verringert.

- Der Fluss f bleibt für alle alten Kanten unverändert, für neu eingefügte Kanten wird er auf 0 gesetzt.

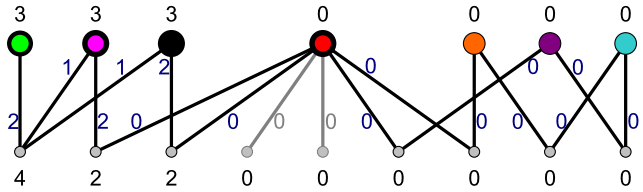


Abbildung 6.18.: Die Kontraktion am bekannten Beispiel. Zu beachten ist, dass der nächste Schritt *nicht* die Aktivierung des nächsten Subgraphen ist, sondern wieder die Suche aus Algorithmus 5, die dann die Subgraphen in Pink, Schwarz und das Ergebnis der Kontraktion auffindet.

Vergleichen wir nun das Ergebnis mit dem, was wir vorgefunden hätten, wenn wir von $\hat{\mathcal{S}} = (\mathcal{S} \setminus \mathcal{S}') \cup \{G(\mathcal{S}')\}$ ausgegangen wären und die Graphen aus $\mathcal{S}^{\text{aktiv}} \setminus \mathcal{S}'$ zuerst aktiviert worden wären – hier konnte ja eben keine interessante Menge von Subgraphen gefunden werden – und dann $G(\mathcal{S}')$:

- Es sind nach Konstruktion dieselben Subgraphen und Schnittknoten vorhanden.
- Die Bewertungen der Subgraphen stimmen überein, für $S \in \mathcal{S}^{\text{aktiv}} \setminus \mathcal{S}'$ ist $b(S) = 3$, für $S^{\text{neu}} = G(\mathcal{S}')$ ist $b(S^{\text{neu}}) = 0$.
- Die Bewertungen der Schnittknoten stimmen überein, denn nach Konstruktion ist die Bewertung für alle Schnittknoten, die in mindestens einem aktiven Subgraphen vorkommen, nach wie vor $2(\text{mv}_{\mathcal{S}^{\text{aktiv}}}(v) - 1)$: Bei der Kontraktion wurden nur Kanten zu aktiven Subgraphen entfernt, und entsprechend die Bewertung verringert.
- Der Fluss f bleibt zulässig. Hier gilt dieselbe Argumentation wie im Beweis zu Lemma 22: Interessant ist dabei nur Fluss von Schnittknoten, die Kanten zu aktiven Subgraphen (also Bewertung) verloren haben, also Knoten $v \in L'$. Diese Knoten haben aber sicher eine Kante zu S^{neu} mit $f(v, S^{\text{neu}}) = 0$. Entsprechend können alle anderen ausgehenden Kanten vollen Fluss haben, ohne dass zu viel Fluss v verlässt.

Dieses Vorgehen ist dasselbe wie in Abbildung 6.17 zuzüglich der Entfernung von Schnittknoten, die nur noch zum kontrahierten Subgraphen adjazent sind.

Berechnung einer gemeinsamen Einbettung

Neben der Identifikation von Subgraphen, die zusammengenommen einen generisch starren Graphen bilden, sollte immer auch eine mit den lokalen Richtungszuweisun-

gen verträgliche Einbettung jedes Subgraphen aus $\mathcal{S}$ bekannt sein. Das ist zu Beginn so, denn die Subgraphen, von denen man ausgeht, bestehen jeweils aus den Knoten, die bezüglich einer Einbettung lokalisiert wurden. Werden jetzt Subgraphen $\mathcal{S}' = \{S'_1 = (V'_1, E'_1), \ldots, S'_k = (V'_k, E'_k)\}$ gefunden, die zusammen auch noch generisch starr sind, dann lässt sich analog zu Korollar 4 eine verträgliche Einbettung berechnen:

1. Wähle aus jedem Subgraphen S, der n_S *der Schnittknoten* $\{v_1, \ldots, v_{n_S}\}$ enthält, $2n_S - 3$ unabhängige Kanten, die genau einen generisch starren Subgraphen über den betroffenen Schnittknoten aufspannen. Als solche Kantenmenge bietet sich $\{\{v_1, v_2\}\} \cup \{\{v_i, v_j\} \mid i \leq 2, j \geq 3\}$ an. Dabei ist es kein Problem, dass diese Kanten oft nicht existieren: Da die Lage aller Knoten relativ zueinander bekannt ist, lassen sich solche Kanten zu diesem Zweck „erfinden“, ihre Richtungen Abbildung 6.19 illustriert dies am Beispiel der nicht völlig trivialen Subgraphenmenge aus Abbildung 6.3.

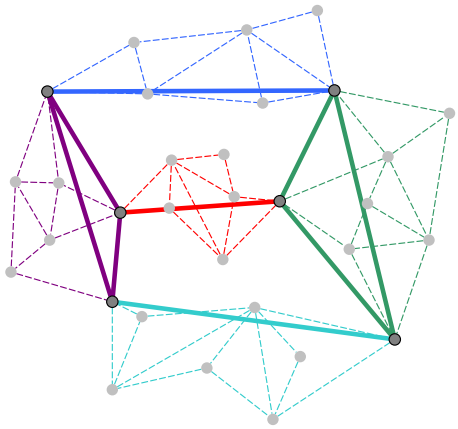


Abbildung 6.19.: Kantenauswahl zur Verschmelzung; gestrichelt die ursprünglichen Kanten, fett die zur Berechnung einer gemeinsamen Einbettung verwendeten Kanten

2. Bezüglich des sich ergebenden Graphen $\hat{G}$ lassen sich zulässige Längen jetzt als Kern der entsprechenden Matrix $A(\hat{G}, \alpha_{G,\mathbf{p}}, T)$ für einen beliebigen Spannbaum T berechnen und damit auch die Skalierungen der einzelnen Subgraphen und damit ausgehend vom größten Subgraphen entlang des Spannbaumes für jeden der anderen Subgraphen eine Umrechnung der Positionen der enthaltenen Knoten.

Laufzeitanalyse

Die Laufzeit dieses Vorgehens lässt sich zumindest grob in bezug auf $|\mathcal{S}|$, $|L|$ und $|A|$ beschränken. Die Initialisierung dieser Mengen lässt sich leicht in $\mathcal{O}(n)$ Schritten erledigen, indem man einmalig über alle Knoten iteriert und für diejenigen Knoten, die

bezüglich mehrerer Einbettungen ihre Positionen kennen, Schnittknoten anlegt und sie mit den betroffenen Einbettungen in $B(\mathcal{S})$ verbindet.

Zunächst zu den Schritten aus Algorithmus 4: Die Schleife, die in Zeile 1 beginnt, wird maximal $|\mathcal{S}|$ mal ausgeführt, die Schleife aus den Zeilen 3/4 kann *insgesamt* nur $|A|$ mal ausgeführt werden. Da die Bewertungen aller Schnittknoten aus L somit insgesamt weniger als $|A|$ mal um 2 erhöht wird, kann ein Maximalfluss immer nur einen Wert aus $\mathcal{O}(|A|)$ erreichen, die bis zu $|\mathcal{S}|$ Flusserhöhungen sind also durchführbar in $2|A| + |\mathcal{S}|$ (versuchten) Erhöhungsschritten mit einem jeweiligen Aufwand von $\mathcal{O}(|A|)$ Schritten für die Tiefen- oder Breitensuche nach einem erhöhenden Pfad. Insgesamt werden also in Algorithmus 4 *außerhalb der Schleife aus Zeile 6* nur $\mathcal{O}(|\mathcal{S}| \cdot |A| + |A|^2)$ Schritte ausgeführt.

Algorithmus 5 wird insgesamt maximal $2|\mathcal{S}|$ mal aufgerufen, weil jeder Aufruf entweder zum Abbruch der Schleife aus Zeile 6 führt oder aber zum Verschmelzen zweier Knoten. Selbst großzügig abgeschätzt wird dann in Algorithmus 5 nur maximal $2|\mathcal{S}|^2$ mal der Abschluss gebildet, jeweils, weil der Abschluss durch eine Tiefen- oder Breitensuche zu bilden ist, in $\mathcal{O}(|A|)$ Schritten. Insgesamt lässt sich die Laufzeit von Algorithmus 5 also durch $\mathcal{O}(|\mathcal{S}|^2|A|)$ abschätzen.

Das Verschmelzen einer Menge von Subgraphen $\mathcal{S}'$ benötigt nur $\mathcal{O}(|\mathcal{S}'|)$ Schnittknoten L', denn mehr als $3|\mathcal{S}'|$ Schnittknoten können nicht gleichzeitig Fluss nach $\mathcal{S}'$ haben. Damit ist auch die Anzahl der Kanten in $B(\mathcal{S})$ zwischen L' und $\mathcal{S}'$, $|A'| \in \mathcal{O}(|\mathcal{S}'|)$, denn jeder Schnittknoten kam schon zu Beginn nur in $\mathcal{O}(1)$ Subgraphen vor, was sich durch das Verschmelzen nicht erhöhen konnte. In $G(\mathcal{S}')$ werden nur $\mathcal{O}(|\mathcal{S}'|)$ Kanten $\hat{E}$ ausgewählt (siehe Abbildung 6.19), denn in jedem Subgraphen S, der n_S Schnittknoten enthält, werden $2n_S - 3$ Kanten ausgewählt, also im Schnitt für jedes Paar aus Schnittknoten und Vorkommnis in einem Subgraphen weniger als 2. Gleichzeitig gibt es für jedes solche Paar genau eine Kante in A', aber wie schon festgestellt, ist $|A'| \in \mathcal{O}(|\mathcal{S}'|)$.

Wie aufwendig ist es nun, eine verträgliche Einbettung für $\hat{G} = (L', \hat{E})$ aus jeweils $\mathcal{O}(|\mathcal{S}'|)$ Knoten (Schnittknoten) und Kanten zu berechnen? Für das Aufstellen der Matrix $A(\hat{G}, \alpha_{G,\mathbf{p}}, T)$ müssen ein Spannbaum konstruiert und vor allem Nichtspannbaumkanten identifiziert werden. Für jede Nichtspannbaumkante müssen der zugehörige Kreis über den Spannbaum traversiert und dabei die Einträge der Matrix aufgefüllt werden. Spannbaum- und Nichtspannbaumkannten auszuwählen, benötigt $\mathcal{O}(|\hat{E}|) \subseteq \mathcal{O}(|\mathcal{S}'|)$ Schritte, das Traversieren der Spannbaumkanten benötigt maximal $|L'| \in \mathcal{O}(|\mathcal{S}'|)$ Schritte, also insgesamt reichen $\mathcal{O}(|\mathcal{S}'|^2)$ Schritte, um die $(|\hat{E}| - |L'|) \times |L'|$-Matrix auszufüllen. Diese Matrix lässt sich sogar noch kleiner halten, indem man die *Spalten*, die zu Spannbaumkanten aus demselben Subgraphen S gehören, zusammenfasst; für die zugehörigen Kantenlängen ist das Verhältnis schon bekannt, die Matrix hat damit nur $|\mathcal{S}'|$ Spalten. Jetzt lässt sich auch die Anzahl der Schritte für die Berechnung des Kerns nach dem Gauß-Verfahren durch $\mathcal{O}(|\mathcal{S}'|^3)$ abschätzen. Es lässt sich nun für alle Schnittknoten schnell eine gemeinsamen Einbettung berechnen, wiederum durch Traversieren des Spannbaumes, und wenn man dabei von Anfang an vom größten

der beteiligten Subgraphen ausgeht, damit auch die Transformation der Einbettung, in der die Spannbaumkante lokalisiert ist, zur größten Einbettung. So kann man analog zur Laufzeitberechnung der kantenüberlappenden Triangulierung in Abschnitt 6.2.1 davon ausgehen, dass jeder Knoten v nur in einer Häufigkeit aus $\mathcal{O}(\log n)$ auf diese Art und Weise in einen neuen Subgraphen aufgeht. Dieser Aufwand von $\mathcal{O}(n \log n)$ Schritten ist gewissermaßen schon dadurch abgedeckt, dass wir diesen Aufwand für alle Verschmelzungen schon den Algorithmen 2/3 angerechnet haben.

Der Aufwand von $\mathcal{O}(|\mathcal{S}'|^3)$ pro Verschmelzung wiederum summiert sich zu maximal $\mathcal{O}(|\mathcal{S}|^3)$ auf. Das folgt daraus, dass in jedem Verschmelzungsschritt mit $k_i \geq 2$ Subgraphen ein Aufwand von $\mathcal{O}(k_i^3)$ anfällt, aber gleichzeitig gilt, dass $\sum_i k_i - 1 \leq |\mathcal{S}| - 1$, denn in jeder Verschmelzung „verschwinden“ $k_i - 1$ Subgraphen, maximal können aber $|\mathcal{S}| - 1$ Subgraphen verschwinden. Dass aber

$$\sum_i k_i^3 \leq \left(\sum_i (k_i - 1) + 1 \right)^3 \quad \text{für} \quad k_i \geq 2$$

ist im Anhang A.2 ausgeführt. Es entsteht also für Verschmelzungen insgesamt ein Aufwand aus $\mathcal{O}(|\mathcal{S}|^3)$.

Somit haben wir für die in Algorithmus 4 aufgeführten Schritte eine Laufzeit aus $\mathcal{O}(|\mathcal{S}| \cdot |A| + |A|^2)$, für alle Aufrufe von Algorithmus 5 eine (grob abgeschätzte) Laufzeit aus $\mathcal{O}(|\mathcal{S}|^2 \cdot |A|)$ und für alle Verschmelzungen insgesamt eine Laufzeit von $\mathcal{O}(|\mathcal{S}|^3)$.

In jedem Fall lässt sich diese Laufzeit abschätzen durch $\mathcal{O}(|L|^3)$. Das bedeutet im schlimmsten Fall $\mathcal{O}(n^3)$, allerdings spiegelt das nicht wider, dass in realistischen Fällen nach den Algorithmen 4 und 5 nur ein sehr kleiner Teil der Knoten Lokalisierungen in mehreren Einbettungen besitzen.

6.3.3. Abdeckung

Dieser weitere Schritt senkt die für eine weitgehende Abdeckung notwendige Dichte weiter ab, Abbildung 6.21 zeigt, wie sie sich Abdeckung (des jeweils größten lokalisierten Subgraphen) gegenüber den „leichteren“ Algorithmen unterscheidet.

6.3.4. Skalierung und Vervollständigung der Einbettung

In allen bisherigen Schritten wurden *ausschließlich* die Richtungsinformationen genutzt[3]. Das ist zum einen positiv zu bewerten, weil man so auf die Quasi-Unit-Disk-Graph-Beschränkungen verzichten kann. Zum anderen hat man so keinerlei Informationen über die Skalierung der Einbettung. Nimmt man hingegen die Information, dass die Eingabe in der Realität als Quasi-Unit-Disk-Graph eingebettet war, lässt sich die korrekte Skalierung c für eine rekonstruierte Einbettung $\mathbf{x}_{S_i}$ eines Subgraphen, d. h. die

[3] Mit Ausnahme der Laufzeitanalyse

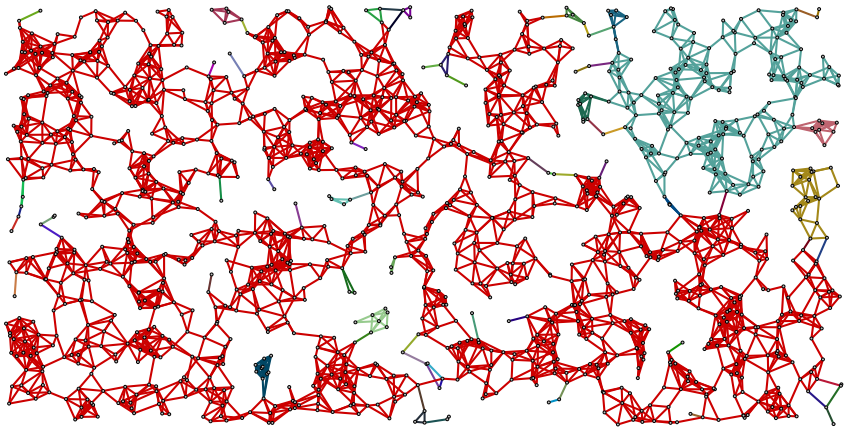


Abbildung 6.20.: Der Graph aus Beispiel 6.5 nach Algorithmus 4. Bei dieser Dichte erhöht dieser Schritt die Abdeckung üblicherweise deutlich.

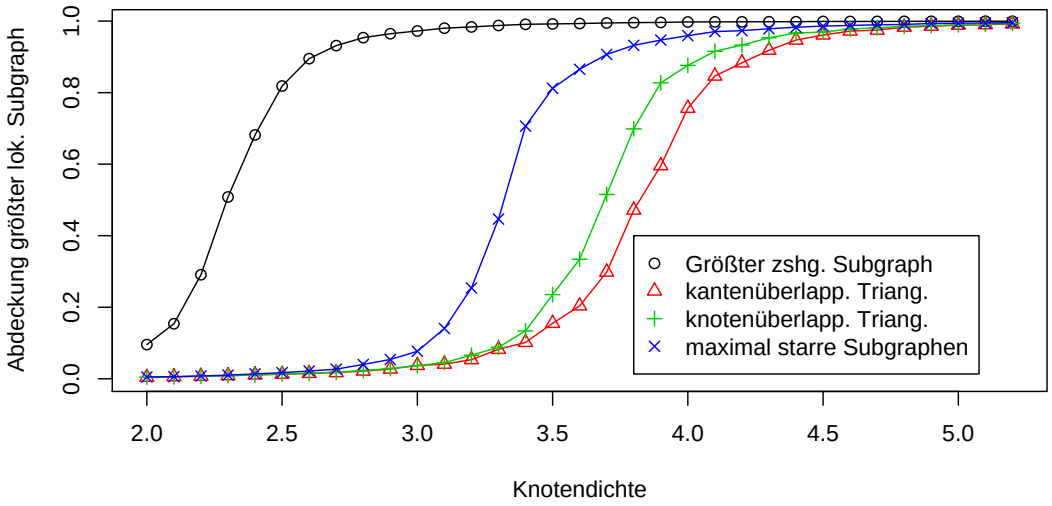


Abbildung 6.21.: Abdeckung aller vorgestellten Verfahren bei zunehmender Dichte im Vergleich zur Größe des maximalen zusammenhängenden Subgraphen bei 8000 Knoten
[50 Messungen, Breite des 0.95-Vertrauensintervalls immer unter 0.1]

Skalierung, für die $c\mathbf{x}$ sich nur noch um Drehung und Verschiebung von der tatasächlichen Einbettung $\mathbf{p}$ unterscheidet, bei hinreichend vielen Knoten sehr leicht eingrenzen, indem man

1. die Verbindung $\{i, j\} \in E$ ermittelt, für die $|\mathbf{x}_i - \mathbf{x}_j|$ maximal ist; dann muss

$c \leq c_{\max} := 1/|\mathbf{x}_i - \mathbf{x}_j|$ sein.

2. das Paar von Sensoren $\{i, j\} \notin E$ ermittelt, für das $|\mathbf{x}_i - \mathbf{x}_j|$ minimal ist; dann muss $c \geq c_{min} := d/|\mathbf{x}_i - \mathbf{x}_j|$ sein. Ist ein Subgraph S vollständig, ist $c_{\min}(S) := 0$.

Vor dem Hintergrund der bisherigen Laufzeitabschätzungen wirkt es störend, dass für den zweiten Punkt naiv $\Theta(n^2)$ Sensorpaare untersucht werden müssten. Hierfür gibt es zwei mögliche Abhilfen: Zum ersten kann man sich auf Sensorpaare $\{i, j\} \notin E$ beschränken, die einen gemeinsamen Nachbarn haben, d. h. für die es einen Sensoren k gibt mit $\{k, i\}, \{k, j\} \in E$. Zum anderen lässt sich das Gebiet, in dem die Sensoren durch $\mathbf{x}$ eingebettet sind, in ein Gitter aus Zellen der Kantenlänge $c_{\max}$ einteilen, um Sensorpaare zu finden, die dichter als $c_{\max}$ beieinander liegen, indem man nur benachbarte Gitterzellen untersucht. Die erste Variante hat den Nachteil, dass gegenüber der ursprünglichen Variante die Skalierung gegebenenfalls nicht so gut eingegrenzt wird (siehe 6.22). Dafür hat sie gegenüber der zweiten Variante den Vorteil, ausschließlich

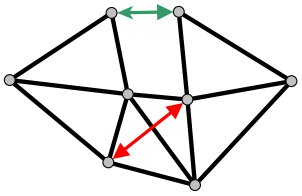


Abbildung 6.22.: Nicht immer haben dichteste nichtverbundene Knoten (grün) einen gemeinsamen Nachbarn.

lokale Information zu nutzen. Abbildung 6.23 zeigt die Konvergenz des Intervalls bei der Lokalisierung zufälliger 0.5-Quasi-Unit-Disk-Graphen, wenn man sich nur auf die Untersuchung von Sensoren mit einem gemeinsamen Nachbarn beschränkt.

Die bisher vorgestellten Algorithmen sichern die Eindeutigkeit berechneter Einbettungen zu, liefern dafür aber keine Einbettung für *alle* Sensoren, sondern immer nur für Teilmengen, für die es *nur aufgrund der Richtungsinformationen* möglich ist, eine Einbettung eindeutig zu bestimmen. Eine weitere Herausforderung ist es aber immer, *alle* Knoten zu lokalisieren, und dabei gegebenenfalls nur eine gute Näherung anzubieten.

Zu diesem Zeitpunkt kennen wir mit $\mathcal{S}$ eine generisch starre Partitionierung, die nicht mehr weiter zu vergröbern ist. Wir kennen für jeden Subgraphen eine Einbettung, die eindeutig bis auf die üblichen Freiheitsgrade ist, und haben die Skalierung jedes Subgraphen S bestimmt bis auf ein Interval $[c_{\min}(S), c_{\max}(S)]$. Das Verfahren aus Abschnitt 6.3.2 zum Verschmelzen von Subgraphenmengen bediente sich der Tatsache, dass die zu verschmelzenden Subgraphen zusammen generisch starr waren. Das war notwendig, um Lösungen als (eindimensionalen) Kern der entwickelten Matrix ablesen zu können. Ohne diese Eigenschaft bleibt aber immer noch der Rückschritt zum Ergebnis von Korollar 2: Eine verträgliche Lösung lässt sich immer noch finden als

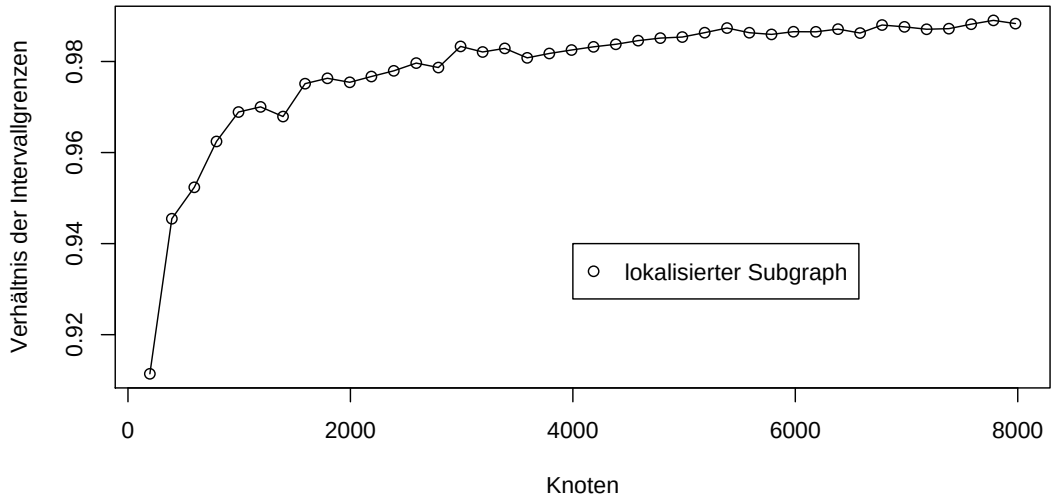


Abbildung 6.23.: Konvergenz von $c_{\min}/c_{max}$ bei zunehmender Zahl lokalisierter Knoten
[50 Messungen, Breite des Vertrauensintervalls kleiner 0.01]

Lösung des Linearen Programms 5.1 mit $|\mathcal{S}|$ Variablen, den Skalierungen jedes Subgraphen (siehe die Reduktion der Spalten in Abschnitt 6.3.2). Abbildung 6.24 zeigt so eine Rekonstruktion.

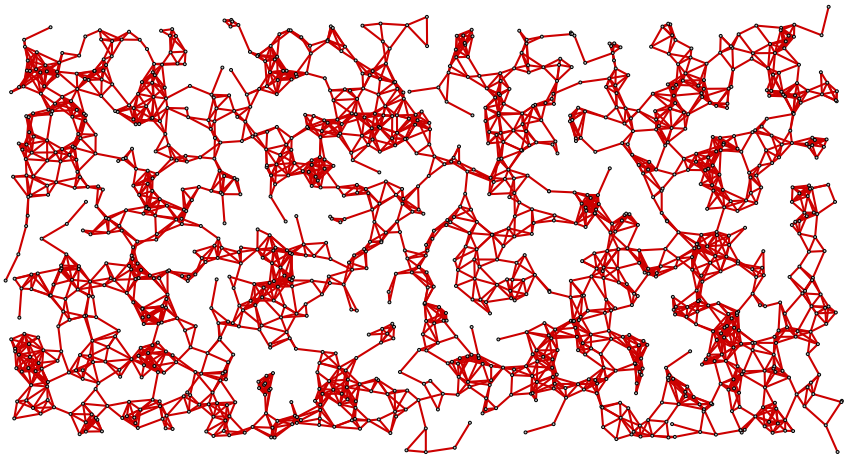


Abbildung 6.24.: Vollständige Rekonstruktion des Beispiels aus Abbildung 6.5

Zu beachten ist, dass so eine Rekonstruktion *garantiert*, dass alle Richtungen mit der Vorgabe übereinstimmen, aber *nicht garantiert*, dass die Rekonstruktion eine Ein-

bettung als d-Quasi-Unit-Disk-Graph darstellt. Das wäre vor dem Hintergrund, dass dieses Problem zumindest für $d = 1$ bewiesenermaßen $\mathcal{NP}$-schwer ist, auch sehr verwunderlich.

6.3.5. Testergebnisse

Ohne Verwendung des Linearen Programms entsprachen die ermittelten partiellen Einbettungen der tatsächlichen Einbettung bis auf Drehung, Verschiebung und Skalierung. Die interessanten Fragen sind daher nur die nach der Abdeckung und nach der Qualität der gewählten Skalierung – die anderen Freiheitsgrade sind ohne absolute Informationen nicht zu eliminieren. Die Abbildungen 6.21 und 6.23 zeigten das Verhalten für zufällig generierte Sensornetzwerke.

Bei der Verwendung des Linearen Programms sind aber tatsächlich Fehler zu erwarten. Sie lassen sich am besten ablesen am Fehler $Q_{\overline{\text{GTE}}}$, der Übereinstimmung mit der Eingabe bei optimaler Drehung, Verschiebung und Skalierung vergleicht (Abbildung 6.25), sowie der dafür verwendeten Skalierung (Abbildung 6.26). Die Ergebnisse zeigen, dass zum einen die optimierte Einbettung schon bei geringen Dichten sehr präzise die tatsächlichen Knotenpositionen annähert[4]

und dass zum anderen die dafür verwendete Skalierung einen immer kleiner werdenden Fehler aufweist. Gegebenenfalls ist bei den Tests die halbe *Breite* des 0.95-Vertrauensintervalls als l_{conf} angegeben, d. h. die Mittelwerte sind mit 95-prozentiger Sicherheit auf $\pm l_{\text{conf}}$ eingegrenzt.

Betrachtet man statt der Dichte eine variable Anzahl von Knoten, lässt sich ablesen, wie sich die Laufzeiten in verschiedenen Situationen empirisch verhalten. Abbildung 6.27 zeigt, wie sich die Laufzeit bei unterschiedlich weit gehender Implementierung jeweils bei einer Dichte verhält, bei der gerade eine hohe Abdeckung erreicht wird. So wird gewährleistet, dass die entsprechende Stufe tatsächlich einen Beitrag leistet. Es zeigt sich, daß der Gewinn, den das Auslegen 2-knotenüberlappender Subgraphen bringt, kaum durch Laufzeiteinbußen erkauft wurde, daß aber das Auffinden maximaler generisch starrer Subgraphen das Laufzeitverhalten bereits deutlich verschlechtert, allerdings bei weitem nicht so sehr wie das weitere Herabsenken der Dichte unter Nutzung des Linearen Programms.

Dieses Laufzeitverhalten gibt allerdings nicht wieder, welchen Gewinn das Auffinden maximaler starrer Subgraphen für das Lösen des Linearen Programms gebracht hat. Diesen kann man ablesen, wenn man die Anzahl der zur Optimierung verbleibenden Subgraphen – entsprechend den Variablen – betrachtet. Abbildung 6.28 zeigt, dass bei Knotendichten um 3.5 die Anzahl verbliebener Variablen bis auf 55% der nach der kantenüberlappenden Triangulerung noch vorhandenen Variablen gesenkt wurde.

[4] Hier wäre gegebenenfalls einzuwenden, dass sich durch das Steigern der Dichte bei gleicher Knotenzahl die Fläche des Gebietes verringert. Man beachte allerdings, dass sich bei einer Verdoppelung der Dichte die Kantenlänge des quadratischen Bereichs nur um den Faktor $1/\sqrt{2}$ verringert.

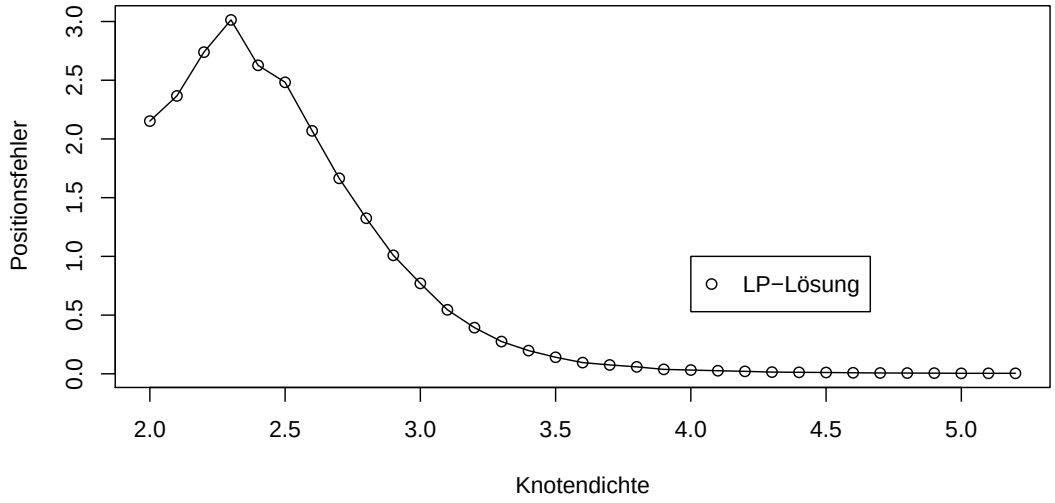


Abbildung 6.25.: Die mittlere Wert von $Q_{\overline{\text{GTE}}}$, des Mittelwerts der Abweichung der Knoten von ihrer tatsächlichen Position unter optimaler Drehung, Verschiebung und Skalierung bei 8000 Knoten und zunehmender Dichte
[50 Messungen, $l_{\text{conf}} < 0.05$]

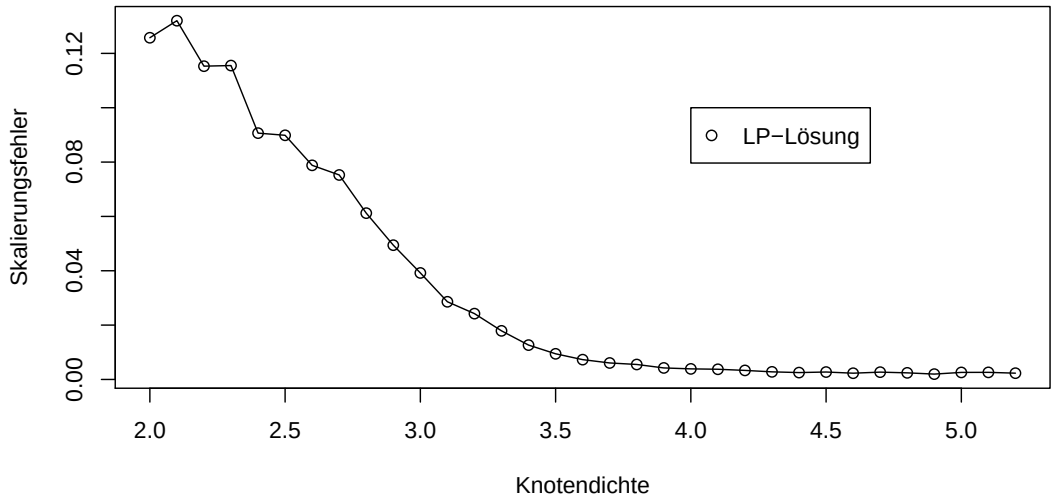


Abbildung 6.26.: Die durchschnittliche *absolute* Abweichung der für die Ergebnisse von Abbildung 6.25 verwendeten Skalierung
[50 Messungen, $l_{\text{conf}} \leq 0.01$]

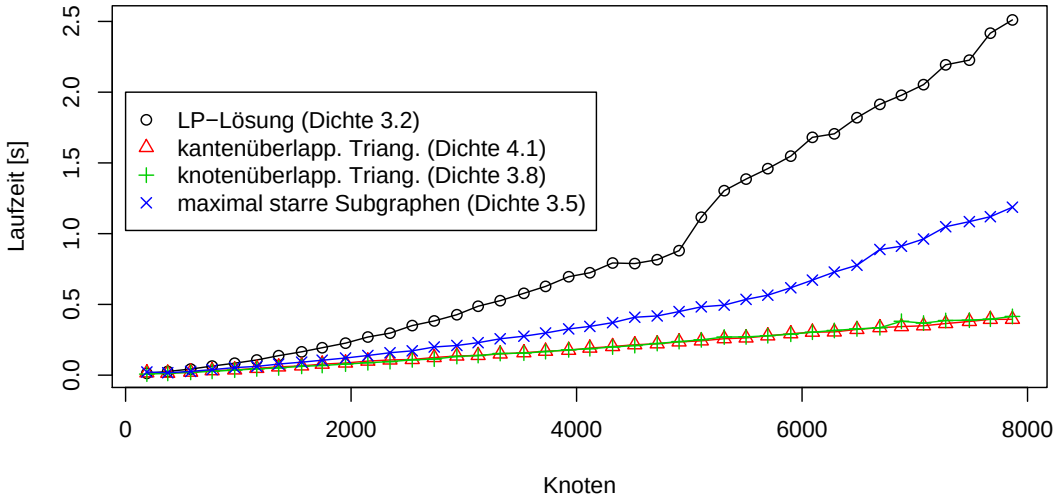


Abbildung 6.27.: Laufzeiten der vier Verfahren, jeweils bei Dichten, bei denen gerade eine hohe Abdeckung erreicht wird
[50 Messungen, $l_{\text{conf}} \leq 10\%$]

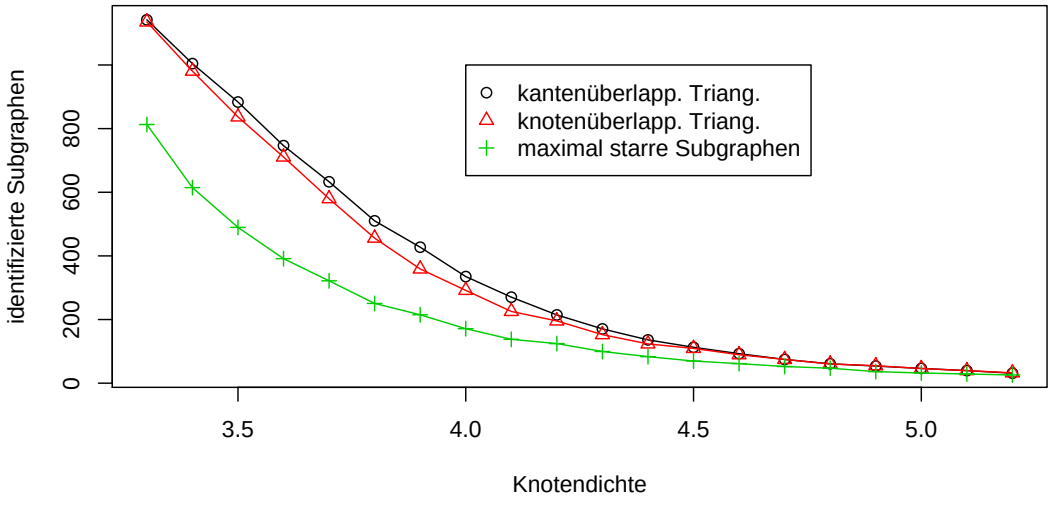


Abbildung 6.28.: Anzahl der verbleibenden Subgraphen, entsprechend den Variablen des LPs zum Finden verträglicher Lösungen bei zufälligen Graphen mit 8000 Knoten und unterschiedlichen Knotendichten
[50 Messungen, $l_{\text{conf}} \leq 10\%$]

7. Fehlertolerante Rekonstruktion

Die bisher vorgeschlagenen Algorithmen bauten wesentlich darauf auf, dass die Eingaben, d. h. die Richtungsmessungen *fehlerfrei* waren und die zusätzlichen Beschränkungen, dass z. B. die tatsächliche Einbettung die Eigenschaft des Kommunikationsgraphen, ein d-Quasi-Unit-Disk-Graph zu sein, belegt, strikt eingehalten wurden. Diese Voraussetzungen sind zwar für theoretische Betrachtungen fruchtbar gewesen, aus schon bei der Vorstellung verwendeter Sensorik in Kapitel 2 angeführten Gründen wäre es allerdings wünschenswert, Verfahren zur Hand zu haben, die auch für fehlerbehaftete Eingaben zu guten Ergebnissen kommen.

Solche Fehler lassen sich dadurch modellieren, dass die Eingabe, sprich alle $\omega(i,j)$, nach einem bestimmten Modell verändert werden:

$$\widetilde{\omega}(i,j) = f_{\text{err}}(\omega(i,j))$$

Dabei kann f_{err} entweder einen zufälligen Messfehler modellieren als $f(x) := x + r$ mit einem gleich- oder normalverteilten r, aber auch Diskretisierungsfehler durch $f(x) := \lfloor x/a \rfloor \cdot a$.

Zu beachten ist, dass eine so fehlerbehaftete Eingabe mit fast völliger Sicherheit nicht mehr konsistent ist. Das lässt sich leicht an einem Dreieck aus verbundenen und eingebetteten Knoten erkennen: Eine lokale Richtungszuweisung umfasst 6 Richtungen zur jeweiligen Ausrichtung der Knoten, aus jeweils zweien lässt sich ein Innenwinkel des Dreiecks berechnen. Nur für sehr spezielle Messfehler (bzw. Einbettungen, wenn man Diskretisierungsfehler betrachtet) wird die Winkelsumme wieder π betragen.

Ein damit einhergehendes Problem besteht darin, dass eine Rekonstruktion zwar an der Abweichung zur Realität gemessen wird (objektive Qualität), man letztlich aber kaum erwarten kann, dass eine Rekonstruktion mehr leistet, als eine Lösung zu finden, die konsistent ist und möglichst wenig *von der Eingabe* abweicht (subjektive Qualität). Bisher habe ich nur Qualitätsmaße für den ersten Fall vorgestellt. Der wesentliche Nachteil ist, dass bei fehlerbehafteter Eingabe anhand dieser nur schwer Aussagen darüber zu treffen sind, welchen Anteil an einer fehlerhaften Rekonstruktion bereits die fehlerhafte Eingabe hat. Da die Eingabe im wesentlichen nur aus einer lokalen Richtungszuweisung besteht, betrachte ich die drei lokalen Richtungszuweisungen ω (Realität), $\widetilde{\omega}$ (Eingabe) und $\hat{\omega}$ (Rekonstruktion) und bezeichne als *Abstand* zweier lokaler Richtungszuweisungen ω und ω':

$$d(\omega,\omega') := \sqrt{\frac{1}{2m} \cdot \sum_{\{i,j\}\in E} \left(\left(\omega(i,j) - \omega'(i,j)\right)^2 + \left(\omega(j,i) - \omega'(j,i)\right)^2 \right)}$$

Das entspricht für eine feste Kantenmenge E dem skalierten euklidischen Abstand im $\mathbb{R}^{2m}$. Alle paarweisen Abstände sind hier von gewissem Interesse:

$d(\omega, \widetilde{\omega})$**:** Der Abstand der fehlerbehafteten Eingabe gegenüber der durch die ursprüngliche Einbettung induzierten Richtungszuweisung; ein Maß für die Stärke des Fehlers.

$d(\omega, \hat{\omega})$**:** Der Abstand zwischen der Richtungszuweisung der ursprünglichen Einbettung gegenüber der der Rekonstruktion; ein Maß für die objektive Qualität der Rekonstruktion.

$d(\widetilde{\omega}, \hat{\omega})$**:** Der Abstand zwischen der Richtungszuweisung der Rekonstruktion und der fehlerbehafteten Eingabe; subjektives Maß für die Qualität der Rekonstruktion.

In diesem Zusammenhang sollte man ständig im Blick haben, dass eine Rekonstruktion, die im optimal ist in dem Sinne, dass sie minimal von der Eingabe abweicht, immer auch eine Korrektur *von der Realität* weg sein kann, d. h. das trotz $d(\widetilde{\omega}, \hat{\omega}) < d(\omega, \widetilde{\omega})$ – die Rekonstruktion liegt dichter an der Eingabe als die Realität – durch aus gelten kann, dass $d(\omega, \hat{\omega}) > d(\omega, \widetilde{\omega})$ – die Rekonstruktion weicht von der Realität noch weiter ab als die Eingabe; dem entgegenwirken kann nur die Redundanz vieler vorhandener Messungen.

7.1. Algorithmus

Trotz der Inkonsistenz der Eingabe lässt sich das iterative Vorgehen aus Kapitel 6 übertragen. Dieses Verfahren ließ sich grob in vier grundlegende Schritte unterteilen:

Saat Gibt es noch Kanten ganz ohne Lokalisation, kann eine neue Einbettung eingeführt werden, in dem genau die beiden Endknoten als $(0, 0)$ und $(1, 0)$ lokalisiert sind.

Wachstum Ein Knoten kann in einer Einbettung lokalisiert werden, wenn mindestens zwei Nachbarn bereits erfaßt sind.

Kombination zweier Einbettungen Überlappen zwei Einbettungen in ihren abgedeckten Knoten hinreichend, dann lassen sich die Koordinaten leicht umrechnen, die Einbettung mit weniger lokalisierten Knoten geht in der anderen auf.

Kombination vieler Einbettungen Die Anordnung der verbleibenden Einbettungen lässt sich iterativ ermitteln, indem man sukzessive nach *Mengen* von Einbettungen sucht, deren lokalisierte Subgraphen zusammen starre Subgraphen bilden. Es lassen sich dann die relativen Größen und damit eine gemeinsame Einbettung berechnen. In einem letzten Schritt wird nur noch nach verträglichen relativen Größen der verbleibenden Einbettungen gesucht.

Auf diesen Schritten wird auch das folgende Verfahren aufbauen; dabei wird es Redundanz und lokale Optimierung nutzen, um jeden Schritt zum einen *initial* möglichst gut zu wählen und zum zweiten Lösungen ständig zu verbessern. Im einzelnen bedeutet das für die ersten drei Schritte:

Saat Führt man neue Einbettungen jeweils für *Cliquen* ein, statt für einzelne Kanten, möglichst groß, aber eine feste Größe nicht überschreitend, bilden diese eine größere Basis für ein Wachstum; neue Knoten können von Anfang an über mehr als nur zwei Knoten trianguliert werden.

Wachstum Lokalisiert man Knoten in einer Einbettung durch *möglichst viele* Nachbarn, erhöht dies die Wahrscheinlichkeit, dadurch auch *gute* Lokalisierungen zu finden.

Kombination zweier Einbettungen Kombiniert man Einbettungen erst dann, wenn sie stärker als nur an zwei Knoten überlappen, lassen sich die für die Umrechnung notwendigen Parameter mitteln.

Alle diese Schritte haben einen Parameter, gewissermaßen den Grad der Redundanz, der sukzessive gesenkt wird bis hinab zum ursprünglichen Algorithmus. Algorithmus 6 skizziert dieses Verfahren in einer etwas allgemeinen Form.

Algorithmus 6 : Fehlertolerante Rekonstruktion

wiederhole
  wenn $\exists$ zwei Einbettungen $\mathbf{p}, \mathbf{p}'$ mit k gemeinsamen Sensoren **dann**
    Kombiniere die beiden Einbettungen
  sonst wenn $\exists$ Sensor $s \notin V_{\mathbf{p}}$ mit $|\mathrm{Nb}(s) \cap V_{\mathbf{p}}| \geq k$ **dann**
    lokalisiere s, sofern *befriedigende* Lokalisierung möglich //s. 7.1.2
  sonst wenn seit letzter Optimierung eine Einbettung gewachsen ist **dann**
    optimiere alle betroffenen Einbettungen
  sonst wenn $\exists$ Clique C mit $|C| = k$ ohne *gemeinsame* Einbettung **dann**
    Führe neue Einbettung ein, lokalisiere die Knoten aus C //s. 7.1.2
  sonst
    $k \leftarrow k - 1$
bis $k = 1$
solange $\exists$ lokalisierte Subgraphen $\mathcal{S}$, die zusammen generisch starr sind **tue**
  Verschmelze $\mathcal{S}$, optimiere das Ergebnis
Verschmelze restliche Subgraphen, optimiere das Ergebnis

Diese Beschreibung lässt noch sehr offen, wie die einzelnen Schritte auszuführen sind; hier lassen sich zum Teil auch verschiedene Varianten vorstellen, von denen ich mich jeweils für eine entschieden habe, die ich in den nächsten Abschnitten beschreibe.

7.1.1. Optimierung

Weil auch die anderen Schritte darauf aufbauen werden, hier zunächst zur Optimierung: Ausgehend von der Minimierungsfunktion $d(\widetilde{\omega}, \hat{\omega})^2$ lässt sich die Einbettung $\mathbf{x}_s$ eines Knotens auf zwei Arten unabhängig voneinander schrittweise verbessern:

Anpassung der Ausrichtung

Der Beitrag eines Knotens s zur Minimierungsfunktion ist nicht unabhängig von der Ausrichtung $\mathbf{x}_s^{\uparrow}$ des Knotens; die Summanden

$$\sum_{\{s,j\}\in E} \left(\omega(s,j) - \hat{\omega}(s,j)\right)^2$$

wird durch eine Drehung des Knotens s um einen Winkel γ verändert zu

$$\sum_{\{s,j\}\in E} \Big(\underbrace{\omega(s,j) - \hat{\omega}(s,j) + \gamma}_{\in[-\pi,\pi)}\Big)^2$$

Nun hat diese Funktion gegebenenfalls mehrere lokale Minima und Maxima, aber wenn man davon ausgeht, dass alle Abweichungen $\beta_j := \omega(s,j) - \hat{\omega}(s,j) \in [-\pi, \pi)$ verhältnismäßig klein sind, dann lässt sich bei der Optimierung von

$$\sum_{\{s,j\}\in E} \left(\beta_j + \gamma\right)^2 \tag{7.1}$$

nach γ auf die Rechnung modulo 2π verzichten und das Problem in $\mathbb{R}$ betrachten. Dort ist $\hat{\gamma} := -\frac{1}{\mathrm{Nb}(s)} \sum_{\{s,j\}\in E} \beta_j$, das arithmetische Mittel der Korrekturen, die einzige Minimalstelle, denn die Ableitung von (7.1)

$$\sum_{\{s,j\}\in E} 2 \cdot \left(\beta_j + \gamma\right)$$

ist an dieser Stelle

$$\sum_{\{s,j\}\in E} 2 \cdot \left(\beta_j + \hat{\gamma}\right) = |\mathrm{Nb}(s)| \cdot 2 \cdot \hat{\gamma} + \sum_{\{s,j\}\in E} \left(2 \cdot \beta_j\right) = \sum_{\{s,j\}\in E} \left(2 \cdot \beta_j\right) - \sum_{\{s,j\}\in E} \left(2 \cdot \beta_j\right) = 0$$

gleichzeitig ist es offenbar für alle $\gamma > \hat{\gamma}$ strikt größer 0, für alle $\gamma < \hat{\gamma}$ strikt kleiner 0. Auf diese Art lässt sich zu gegebener Einbettung leicht für jeden Knoten berechnen, welche Drehung optimal ist bei Beibehaltung aller Positionen. Abbildung 7.1 zeigt einen Knoten vor und nach dieser Optimierung.

Optimierung der Position

Die Berechnung einer optimalen Position *in einem Zug* ist nicht so leicht möglich. Aber analog lässt sich auch hier zunächst der durch die Position induzierte Fehler betrachten.

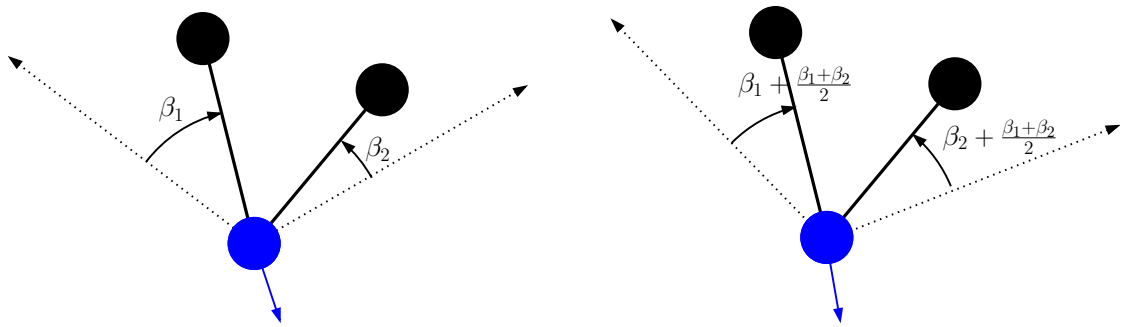


Abbildung 7.1.: Optimierung der Ausrichtung

Stellkräfte Betrachtent man dabei zuerst den Beitrag einer Abweichung $(\omega(s,j) - \hat{\omega}(s,j))^2$, also den Fehler zwischen der aktuellen Einbettung und der Messung für die Position eines Nachbarn durch einen Knoten s. Abbildung 7.2 zeigt diese Situation, es bezeichne $\omega_{\text{err}} := \hat{\omega}(s,j) - \omega(s,j)$. Für kleine Fehler ist in guter Näherung $\omega_{\text{err}} \approx \tan(\omega_{\text{err}}) = \frac{x_{\text{err}}}{|\mathbf{x}_s - \mathbf{x}_j|}$ und man kann wiederum in guter Näherung davon ausgehen, dass der Gradient minimal wird parallel zu $\mathbf{x}_j - \mathbf{x}'_j$ und dem Betrag nach proportional ist zu $\frac{\omega_{\text{err}}}{|\mathbf{x}_s - \mathbf{x}_j|}$, denn für $\mathbf{v} = \frac{\mathbf{x}_j - \mathbf{x}'_j}{|\mathbf{x}_j - \mathbf{x}'_j|}$ ist damit

$$\frac{\partial \omega_{\text{err}}^2}{\partial \mathbf{v}} \approx \partial \frac{x_{\text{err}}^2}{|\mathbf{x}_s - \mathbf{x}_j|^2} / \partial \mathbf{v} = \frac{2x_{\text{err}}}{|\mathbf{x}_s - \mathbf{x}_j|^2} \approx \frac{2\omega_{\text{err}}}{|\mathbf{x}_s - \mathbf{x}_j|} \quad .$$

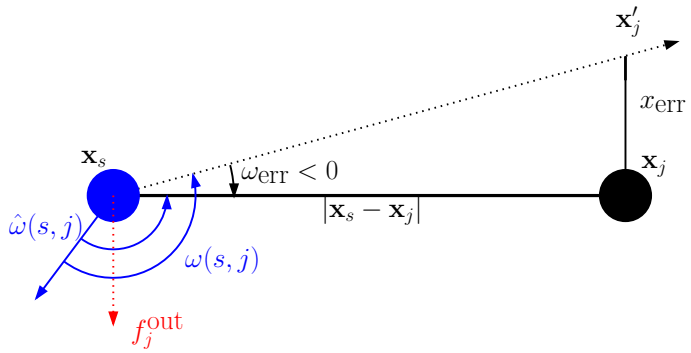


Abbildung 7.2.: Optimierung der Knotenposition durch eigene Messung der Richtung zu einem Nachbarn

Ganz analoge Überlegungen lassen sich auch für die Messung durch den Nachbarn anstellen; ein Verschieben eines Knotens s verändert alle $2|\text{Nb}(s)|$ Richtungsmessungen, mit denselben Ergebnissen für die Bezeichnungen aus Abbildung 7.3.

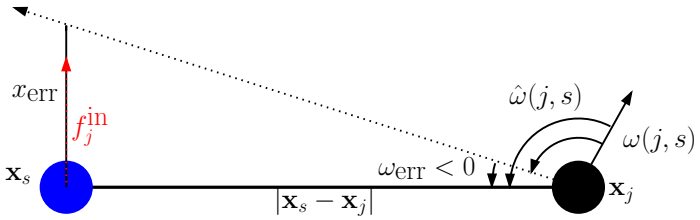


Abbildung 7.3.: Optimierung der Knotenposition durch Messung der Richtung durch einem Nachbarn

Damit ergibt sich zusammengenommen eine Optimierung in Richtung von

$$F_{\text{stell}}(s) = \sum_{\{s,j\}\in E} \frac{\overbrace{\hat{\omega}(s,j) - \omega(s,j)}^{\in[-\pi,\pi)} + \overbrace{\omega(j,s) - \hat{\omega}(j,s)}^{\in[-\pi,\pi)}}{|\mathbf{x}_s - \mathbf{x}_j|} \cdot R(\frac{\pi}{2}) \cdot (\mathbf{x}_j - \mathbf{x}_s)$$

Analog lässt sich auch wie in kräftebasierten Verfahren zur Längenrealisierung ein physikalisches Modell beschreiben, in dem die Kanten als feste Teleskopstangen durch Stellfedern in eine Richtung gezogen werden; auch hier würde die Kraft *senkrecht* zur Verbindung zum Nachbarknoten wirken und der Abweichung von der Ideallage proportional sein.

Weitere Kräfte Da in dieses Verfahren an keiner Stelle Informationen über die Skalierung einfließen, kann es schnell vorkommen, dass sich Fehler dahingehend aufsummieren, dass Teile des Graphen größer eingebettet werden als andere. Auf der anderen Seite steht mit der Einbettbarkeit als Quasi-Unit-Disk-Graph zumindest *eine* Skalierungsinformation zur Verfügung. Dabei macht das Einführen einer Komponente, die Kantenlängen oberhalb von 1 verhindern soll, wenig Schwierigkeiten, wenn man zum Beispiel eine Kraft

$$F_{\text{anz}}(s) = \sum_{\{s,j\}\in E} \begin{cases} \mathbf{0} & : |\mathbf{x}_j - \mathbf{x}_s| \leq 1 \\ \frac{|\mathbf{x}_j - \mathbf{x}_s| - 1}{|\mathbf{x}_j - \mathbf{x}_s|}(\mathbf{x}_j - \mathbf{x}_s) & : |\mathbf{x}_j - \mathbf{x}_s| > 1 \end{cases}$$

hinzunimmt. Schwerer einzubringen ist eine abstoßende Komponente, die das Nähern von unverbundenen Knotenpaaren auf unter d verhindert, aus denselben Gründen wie in Abschnitt 6.3.4 bei der Bestimmung des Intervalls gültiger Skalierungen eines starren Subgraphen: Zum einen sind massiv mehr Knotenpaare ($\Theta(n^2)$) betroffen, zum anderen ginge bei direkter Umsetzung jegliche Lokalität verloren. Deshalb bietet sich auch hier derselbe Weg an wie dort – es werden nur die Knotenpaare für abstoßende Kräfte betrachtet, die mit dem Knoten s einen gemeinsamen Nachbarn haben. Die

Kraft lässt sich dann ganz ähnlich wie bei der Anziehung definieren als

$$F_{\text{abst}}(s) = \sum_{j \in \text{Nb}^2(s):\{s,j\} \notin E} \begin{cases} \mathbf{0} & : |\mathbf{x}_j - \mathbf{x}_s| \geq d \\ \frac{d - |\mathbf{x}_j - \mathbf{x}_s|}{|\mathbf{x}_j - \mathbf{x}_s|}(\mathbf{x}_j - \mathbf{x}_s) & : |\mathbf{x}_j - \mathbf{x}_s| < d \end{cases}$$

Hier ließen sich eine Vielzahl von Varianten denken, vor dem Hintergrund, dass diese Kräft nur für eine korrekte Skalierung sorgen sollen, ist diese Wahl nicht allzu entscheidend; wichtiger ist, dass sie gemessen an F_{stell} schwächer einfließen und das Verhalten nicht dominieren.

Solche Optimierungen sind dabei unabhängig für unterschiedliche Einbettungen. Für die Optimierung der Einbettung eines Knotens in einer bestimmten Einbettung bezieht sich die Nachbarschaft jeweils nur auf Knoten, die auch von dieser Einbettung erfasst sind.

Ablauf Durch diese beiden Optimierungsschritte lässt sich eine Einbettung eines Knotens verbessern, indem beide Optimierungen solange abwechselnd ausgeführt werden, bis keinen nennenswerte Kräfte mehr wirken. Völlig analog lässt sich aber auch die Einbettung vieler Knoten gleichzeitig verbessern, indem wiederholt die beiden Optimierungen *für alle ausgewählten Knoten* ausgeführt werden.

Algorithmus 7 : Verbessere Einbettung von V'

```
wiederhole
    für alle v ∈ V' tue
        Optimiere Ausrichtung von v
        Verschiebe v um c_v (c_stell · F_stell(v) + c_anz · F_anz(v) + c_abst · F_abst(v))
bis keine nennenswerte Verbesserung eintritt
```

Dieses Verfahren aus Algorithmus 7 ist in vielfältigen Varianten aus kräftebasierten Layoutverfahren bekannt (siehe [BETT99]), und es sind viele Beschleunigungsverfahren untersucht worden. Diese Arbeit soll allerdings vor allem zeigen, dass mit solchen Verfahren auch unter der Einschränkung, dass nur lokale Informationen genutzt werden, Einbettungen trotz bestimmter Fehler gut rekonstruiert werden können. Für die Evaluation wurden die Parameter empirisch gewählt als $c_{\text{stell}} = 1, c_{\text{anz}} = 0.2, c_{\text{abst}} = 0.1$, die Schrittweite c_v wurde dynamisch angepasst, d. h. verkleinert, sobald bei einer Verschiebung eine Verschlechterung auftrat, und vergrößert, wenn auch eine zweite Verscheibung noch eine Verbesserung brachte.

7.1.2. Auslegung von Cliquen, Anbinden von Knoten

Die ersten ausgelegten Knoten sind immer die Knoten einer Clique. Sie bilden wegen ihrer vollständigen Messung eine sehr stabile Grundlage und erlauben Korrekturen

auch bei sehr stark fehlerbehafteter Eingabe. Aus Komplexitätsgründen wird hierbei die Suche auf Cliquen einer festen maximalen Größe –der Maximalgrad kann die Anzahl möglicher Cliquen und die Zeit für die Suche aus praktischer Sicht schon zu weit treiben, zumal 5 oder 6 Knoten schon völlig ausreichend sind – beschränkt. Auch ansonsten kann man sich hier weitgehend auf die lokale Optimierung verlassen, das Vorgehen aus Algorithmus 8 führt bereits zu sehr guten Ergebnissen. Es nutzt aus, dass sich bezüglich zweier fest gewählter Knoten ein dritter, verbundener Knoten leicht optimal einbetten lässt: Drei Knoten bilden genau dann ein Dreieck, wenn die Innenwinkel sich zu π aufsummieren. Dies tun sie – den Messungen nach – nicht; die Winkel, die sich aus den Messungen ergeben, können zusammen einen Überschuss haben oder aber eine zu geringe Summe haben. In beiden Fällen lässt sich der Fehler *gleichmäßig* auf die 4 nicht festgelegten Messungen verteilen, so dass die Optimierungsfunktion hier minimiert wird (siehe Abbildung 7.4).

Algorithmus 8 : Einbetten einer Clique $C = \{v_1, \ldots, v_{k_s}\}$

$\mathbf{x}_1 \leftarrow (0, 0)$
$\mathbf{x}_1^{\uparrow} \leftarrow -\omega(v_1, v_2)$
$\mathbf{x}_2 \leftarrow (1, 0)$
$\mathbf{x}_2^{\uparrow} \leftarrow \pi - \omega(v_2, v_1)$
für $i \in \{2, \ldots, k\}$ **tue**
  Lege v_i *optimal* bezüglich v_1, v_2 aus
Optimiere Einbettung von C

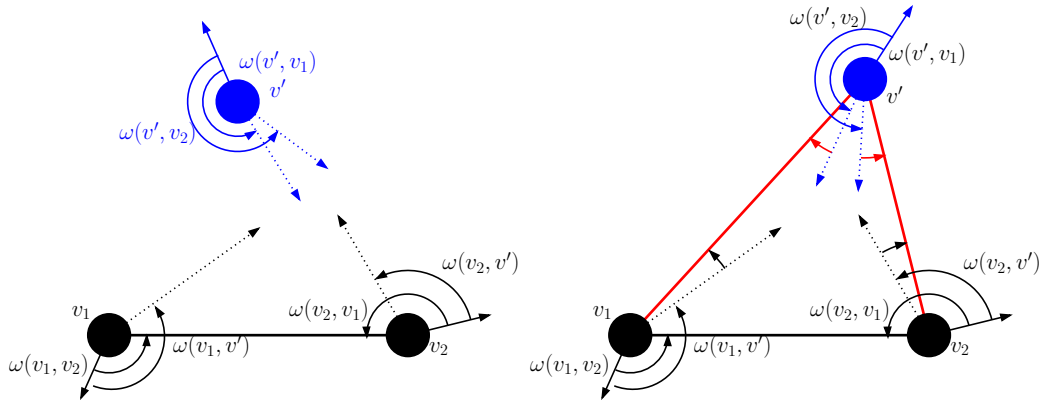


Abbildung 7.4.: Ein neuer Knoten wird so ausgelegt, dass die vier noch festgelegten Winkel gleichmäßig von der Vorgabe abweichen

Der Vollständigkeit halber sei erwähnt, dass dieses Vorgehen in wenigen Fällen nicht so einfach umzusetzen ist; in diesen Fällen muss man leicht von obigem Vorgehen abweichen.

1. Nicht immer ist die Drehrichtung eindeutig; in diesem Fall wird dem größten Winkel gefolgt.

2. Ist die Winkelsumme der Messungen zu groß, müssen die Innenwinkel *verkleinert* werden. Das kann dann zum Problem werden, wenn dabei einer der Winkel einen negativen Wert annehmen würde. In diesem Fall wird der betroffene Winkel nur bis zu diesem Punkt verkleinert, die anderen dafür entsprechend mehr.

Um allzu extreme initiale Einbettungen zu vermeiden, sollte in Punkt 2 die Untergrenze für einen Winkel auf einen kleinen positiven Wert (statt 0) gesetzt werden.

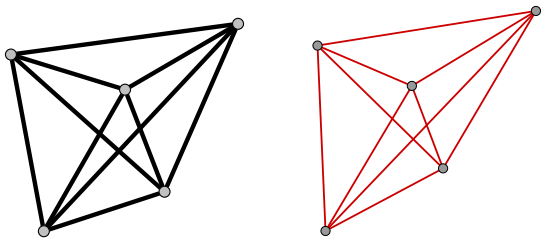


Abbildung 7.5.: Rekonstruierte Einbettung einer Clique bei gleichverteiltem Fehler im Intervall von $[-20°; +20°]$

Ein Beispiel für das Auslegen einer Clique der Größe 5 zeigt Abbildung 7.5.

Sehr ähnlich wie das Auslegen von Cliquen lässt sich auch das Anbinden neuer Knoten umsetzen. Hat ein Knoten k lokalisierte Nachbarn (bei mehr wird ausgewählt), können alle Paare darunter als Grundlage für eine Dreiecksbildung wie im letzten Abschnitt ausgewählt und das Ergebnis jeweils lokal unter Berücksichtigung *aller* Nachbarn optimiert werden. Die beste Lokalisierung wird übernommen, wenn keiner der einzelnen Winkel dabei zu weit von der Eingabe abweicht. Die besten Ergebnisse ließen sich allerdings erzielen, wenn zunächst für alle in Frage kommenden Knoten die jeweils beste Lokalisierung ermittelt wurde und diese dann immer für die Knoten, die jeweils den geringsten Fehler einführten, übernommen wurden.

Für das Herabsetzen der Redundanz der einzelnen Schritte sind verschiedene Schemata denkbar. Die einfachste, nämlich alle drei Parameter *parallel* herabzusetzen, führt zu guten Ergebnissen. Als Ausgangswerte ließen sich für größere Werte als $k = 6$ keine besseren Ergebnisse mehr erzielen. Wird beim Herabsetzten $k = 2$ erreicht, entspricht der Algorithmus *prinzipiell* dem Algorithmus für fehlerfreie Eingabe bis zur Knotenüberlappung, mit der Ausnahme, dass bisher auf das Anbinden von Knoten verzichtet wurden, die einen zu großen Fehler in die Einbettung hineintrugen. Auf diese Einschränkung muss in einem zusätzlichen Durchlauf verzichtet werden, um eine Lokalisierung aller Knoten in mindestens einer Einbettung zu garantieren.

7.1.3. Kombinieren der Einbettungen

Die Parameter für das Kombinieren zweier Einbettungen, Drehung, Skalierung und Verschiebung, waren im fehlerfreien Fall eindeutig definiert durch zwei Knoten, die

in beiden Systemen lokalisiert waren. Länge und Richtung der Differenzen ergaben Drehung und Skalierung, die Verschiebung ließ sich dann an einem der Knoten ablesen. Im fehlerbehafteten Fall kann die Ermittlung dieser Parameter auf Basis mehrerer Knotenpaare stabilere Ergebnisse liefern; die Drehung und Skalierung lässt sich aus den einzelnen Drehungen und Skalierungen aller Paare unter den k Knoten mitteln und anschließend lässt sich die Verschiebung ebenso aus den Verschiebungen aller Knoten mitteln.

In der letzten Phase werden, weitgehend wie im fehlerfreien Fall, Einbettungen gesucht, deren erfaßte Knoten und Kanten gemeinsam generisch starre Subgraphen bilden (siehe Algoritmus 4). Im Gegensatz zu Kapitel 6 lässt sich die Skalierung nun leider nicht als Kern einer Matrix ablesen, weil die Richtungen der Eingabe sogar sehr wahrscheinlich nicht konsistent sind; das ändert natürlich nichts daran, dass der Kern der Matrix $A(G, \alpha_{G,\mathbf{p}}, S)$ bis auf entartete Fälle eindimensional ist, es ist aber nicht garantiert, dass er einen Vektor mit ausschließlich positiven Einträgen enthält. Dafür sorgt die lokale Optimierung durchgehend für eine gleichmäßige Skalierung; für eine gute initiale Lösung ist es daher ausreichend, die Subgraphen ohne weitere Skalierung zusammenzusetzen. Das iterative Vorgehen hat, im Gegensatz zur Alternative, einfach alle nach der ersten Phase verbliebenen Subgraphen so miteinander zu verschmelzen, dabei trotzdem den Vorteil, dass immer wieder optimiert werden kann, sobald die Grundlage dafür, die Eindeutigkeit der Größenverhältnisse, geschaffen ist. Entsprechend wird dann auch im letzten Schritt beim Kombinieren der verbliebenen Einbettungen kein Lineares Programm gelöst, zumal gegebenenfalls auch noch die Nebenbedingungen aus längsten Kanten und kürzesten Nichtkanten widersprüchlich sein können, sondern auch hier werden die Einbettungen in der Skalierung miteinander kombiniert, die sich aus der lokalen Optimierung zuvor ergeben hat. Die Abbildungen 7.6 und 7.7 zeigen Beispiele für solche Rekonstruktionen.

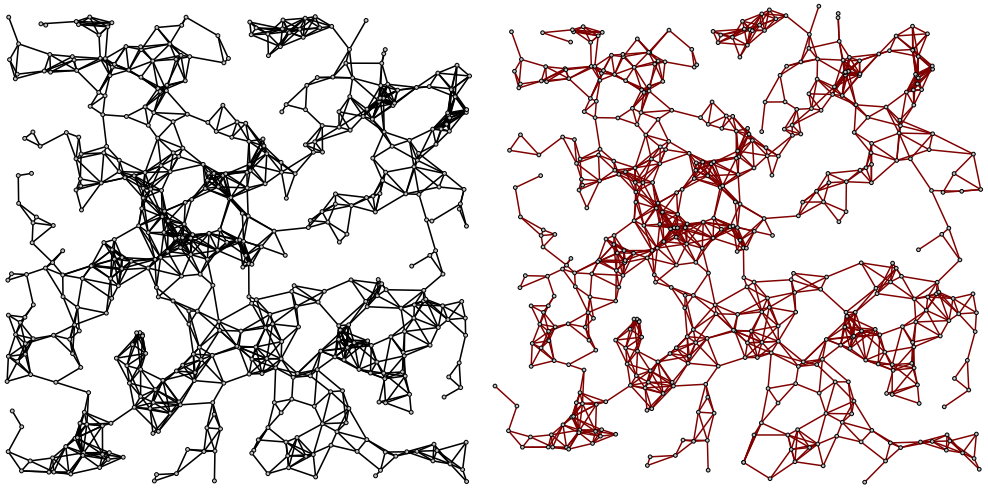


Abbildung 7.6.: Rekonstruktion eines mit einer Knotendichte von 3.5 verhältnismäßig dünnen Graphen bei gleichverteiltem Fehler aus $[-5°; 5°]$

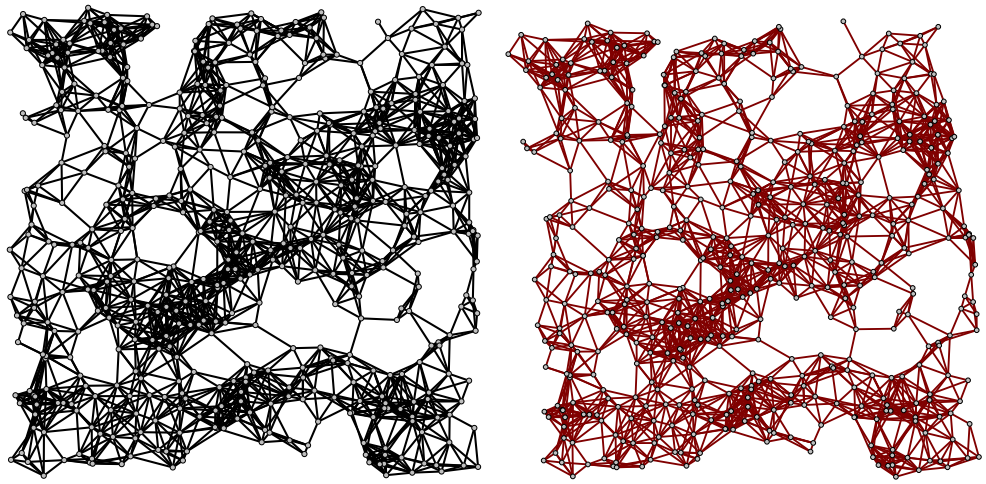


Abbildung 7.7.: Rekonstruktion eines mit einer Knotendichte von 5.0 deutlich dichteren Graphen bei gleichverteiltem Fehler aus $[-8°; 8°]$

In [BGJ05] wurde auch bei fehlerhafter Eingabe das Lineare Programm aus Bedingungen für die Kreise (entsprechend $A(G, \alpha_{G,\mathbf{p}}, S)$) und Kantenlängen, dort für $d = 1$ modelliert als

$$\begin{aligned} \forall e \in E &: 0 \leq \ell(e) \leq 1 \\ \forall e_1 = \{u, v_1\}, e_2 = \{u, v_2\} \in E, v_1 \neq v_2 \wedge \{v_1, v_2\} \notin E &: \ell(e_1) + \ell(e_2) > d \ , \end{aligned}$$

verwendet. Dass allerdings eine beliebig geringe Störung der Winkelmessung schon dazu führen kann, dass das entsprechende Lineare Programm keine zulässige Lösung mehr hat, zeigt Abbildung 7.8: Es lassen sich für jedes d *Strecken* konstruieren, die beliebig genau in ihrer Länge festzulegen sind und daraus Vielecke, die bereits bei minimaler Störung eines Winkels inkonsistent sind.

Diese Figur ist natürlich konstruiert, aber vor dem Hintergrund, dass die Skalierung von generisch starren Subgraphen durch die (Quasi-)Unit-Disk-Graph-Bedingungen schnell äußerst eng eingegegrenzt wird, verwundert es nicht, dass es tatsächlich und vor allem bei größeren Netzen zu solchen nicht erfüllbaren Nebenbedingungen kommt.

7.2. Testergebnisse

Das vorgestellte Verfahren wurde zur Ermittelung der Qualität der Rekonstruktion ausgiebig an zufällig generierten Quasi-Unit-Disk-Graphen getestet. Dafür wurden, sofern die Knotenzahl nicht variabel war, Graphen mit 500 Knoten generiert. Gegebenenfalls ist wieder die halbe *Breite* des 0.95-Vertrauensintervalls als l_{conf} angegeben, allerdings zum Teil eingeschränkt, weil die Ergebnisse in Bereichen, in denen die Rekonstruktion nicht durchgehend gelingt, die Streuung so enorm ist, dass statistische

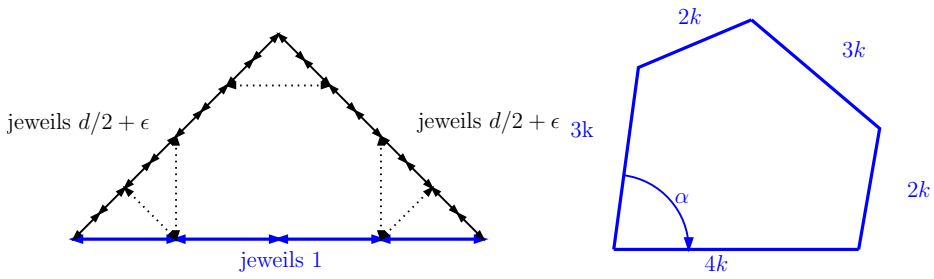


Abbildung 7.8.: Links: Ein Dreieck aus maximal langen Kanten an der Basis und minimal langen Kanten an den anderen Seiten. Die Anzahl der Abschnitte auf den Kanten können in diesem Rahmen recht frei gewählt werden, sie ergeben die Winkel; weitere notwendige Kanten können eingefügt werden (gestrichelt). Die Basis ist so durch die Winkel in der Länge beliebig genau festgelegt. Rechts: Stört man in einem Vieleck aus solchen Strecken an einer Stelle, wird das entstehende Lineare Programm inkonsistent.

Relevanz kaum zu erreichen ist. Diese Werte sind dann trotzdem als Anhaltspunkte in den Diagrammen enthalten. Betrachtet man in Tests zunächst die Abstände zu Beginn dieses Kapitels eingeführten Abstände zwischen den lokalen Richtungszuweisungen in der Realität (ω), der fehlerbehafteten Eingabe ($\widetilde{\omega}$) und der Rekonstruktion ($\hat{\omega}$), lässt sich folgendes Verhalten ablesen. So zeigt Abbildung 7.9 Ergebnisse für steigende Dichte bei einem normalverteiltem Fehler mit einer Standardabweichung von 5 Grad: Während der Abstand von der Eingabe zu tatsächlichen Einbettung natürlich durchgehend dieselbe ist, nimmt die Qualität der Richtungsrealisierung – egal, ob an der Realität oder an der Eingabe gemessen – mit zunehmender Dichte zunächst ab. Das lässt sich leicht dadurch erklären, dass für diese Knotendichten die Eingabe relativ wenig widersprüchlich ist, weil es nur verhältnismäßig kleine starre Subgraphen gibt. Die Eingabe kann entsprechend gut realisiert werden; der Abstand der Rekonstruktion ist also klein zur Eingabe, der Abstand zur Realität liegt zunächst im Bereich des Eingabefehlers. Dieser Effekt nimmt mit der Dichte ab, die Rekonstruktion ist bei Knotendichten um die 3.6 am schlechtesten. Nicht zufällig ist das der Bereich, ab dem nicht nur große Subgraphen eindeutig auszulegen sind, sondern zunehmend auch redundante Information enthalten. Mit weiter zunehmender Knotendichte verbessert sich die Qualität der Rekonstruktion dann; der Abstand zur Eingabe liegt im Bereich des Eingabefehlers, wogegen der Abstand zur Realität noch darunter fällt. Ähnliches Verhalten lässt sich auch für gleichverteilte Fehler und Diskretisierungsfehler beobachten (Abbildungen A.1 und A.2), wenn man davon absieht, daß Diskretisierungsfehler, weil sie systematisch in eine Richtung gehen, gegenüber der Eingabe leicht zu verbessern sind.

Interessant ist natürlich unter dem Strich vor allem, wie gut die Rekonstruktion der Topologie ist. Analog zu Abschnitt 6.3.5 wird dafür getrennt der Fehler $Q_{\overline{\text{GTE}}}$ und die dafür verwendete Skalierung betrachtet. Der um Drehung, Streckung und Skalierung bereinigte Fehler der Knotenpositionen nimmt – in Abbildung 7.10 für ausgewählte

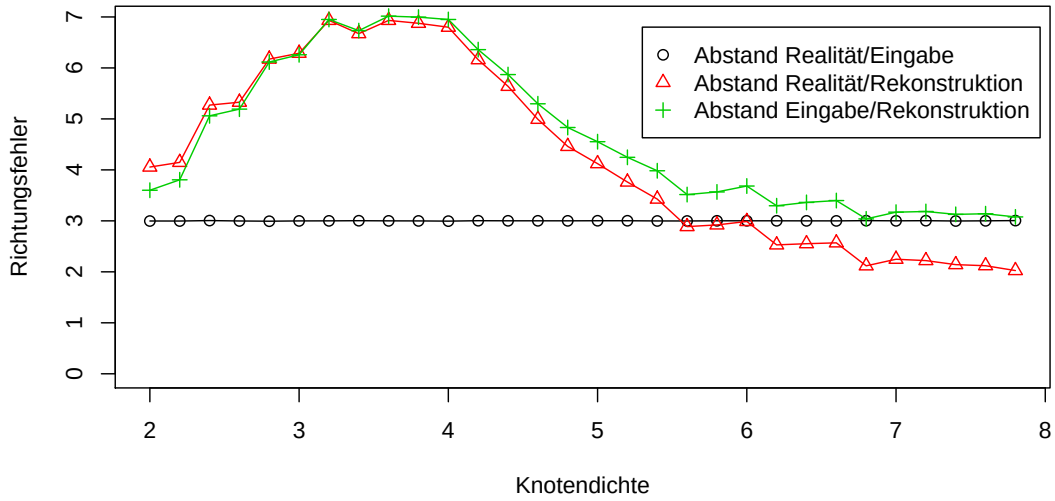


Abbildung 7.9.: Die drei Fehler $d(\omega, \widetilde{\omega})$, $d(\omega, \hat{\omega})$ und $d(\widetilde{\omega}, \hat{\omega})$ bei normalverteiltem Fehler bei einer Standardabweichung von 3° und zunehmender Knotendichte.
[200 Messungen, durchgehend $l_{\text{conf}} < 0.6°$, ab einer Dichte von 5 ist $l_{\text{conf}} < 0.4°$]

Fehler zu sehen – mit zunehmender Dichte ab. Für geringe Dichten ist nicht nur der durchschnittliche Fehler deutlich höher, sondern auch die Varianz. Das Risiko sehr schlechter Rekonstruktionen ist hier deutlich höher; hierauf sind auch die sehr ungenauen Werte für die Diskretisierungsfehler zurückzuführen. Festzuhalten ist aber, dass für die ausgewählten Fehler eine Dichte ab ungefähr 5 für eine gute Rekonstruktion ausreicht. Was die Skalierung der Rekonstruktion angeht, ist diese – wie in Abbildung 7.11 zu sehen – durchgehend zu klein (zur Berechnung von $Q_{\overline{\text{GTE}}}$ wurde das Ergebnis im Schnitt 10-15% vergrößert). Anzumerken ist allerdings, dass dieser Effekt der kräftebasierten Optimierung relativ vorhersagbar auftritt und sich so zum Teil kompensieren ließe.

Betrachtet man die Qualität der Rekonstruktion bei fester Dichte und zunehmendem Fehler, lässt sich ablesen, bis zu welcher Störung noch verwertbare Ergebnisse zu erwarten sind. Abbildung 7.12 zeigen diese Verhalten für normalverteilte Fehler bei drei ausgewählten Knotendichten. Die Abbildungen A.3 und A.4 zeigen ähnliches Verhalten für die anderen betrachteten Fehlertypen.

Abbildung 7.13 zeigt das empirische Laufzeitverhalten bei verschieden großem Fehler. Wegen der häufiger auszuführenden lokalen Optimierung ist die Laufzeit wesentlich von der Stärke des Fehlers abhängig.

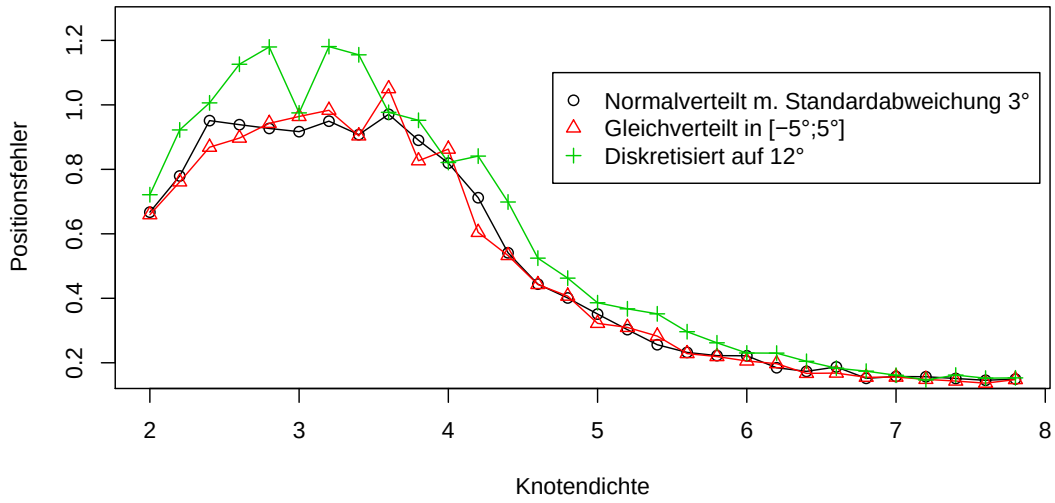


Abbildung 7.10.: Mittlere Knotenabstände $Q_{\overline{\mathrm{GTE}}}$ bei zunehmender Knotendichte für drei ausgewählte Fehler.
[200 Messungen, ab einer Dichte von 4.4 ist $l_{\mathrm{conf}} < 0.06$, mit Ausnahme der Dichten 3.4 bsi 3.8 ist sonst $l_{\mathrm{conf}} < 0.12$]

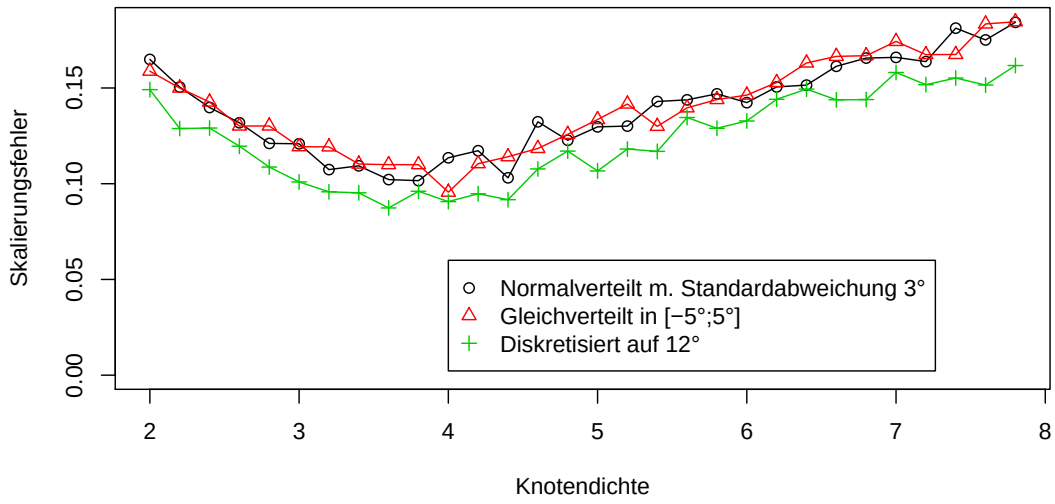


Abbildung 7.11.: Abweichung der für die Berechnung von $Q_{\overline{\mathrm{GTE}}}$ verwendeten Skalierung von 1 bei zunehmender Standardabweichung des normalverteilten Fehlers für ausgewählte Knotendichten
[200 Messungen, $l_{\mathrm{conf}} < 0.017$]

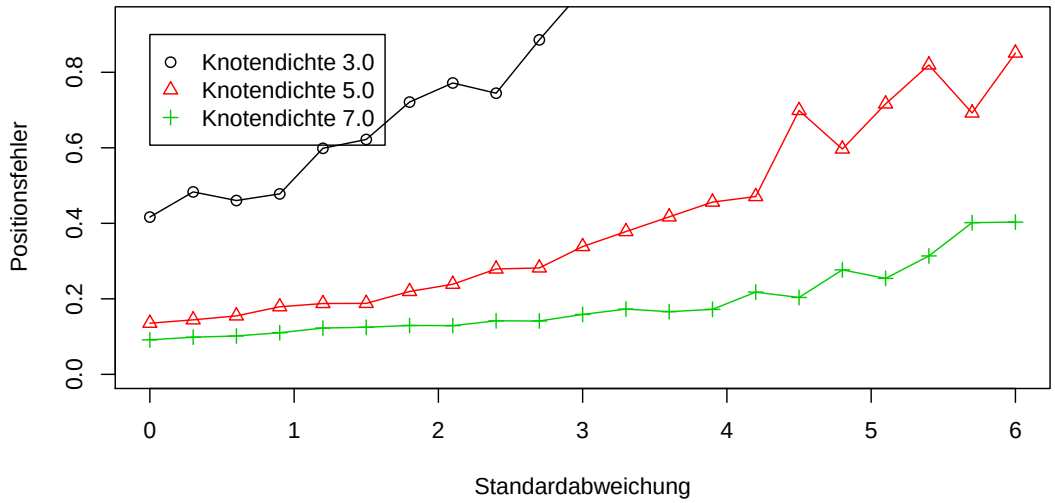


Abbildung 7.12.: Mittlere Knotenabstände $Q_{\overline{\text{GTE}}}$ bei zunehmender Standardabweichung des normalverteilten Fehlers für ausgewählte Knotendichten
[100 Messungen, für Knotendichte 7 ist $l_{\text{conf}} < 0.08$ (bis Standardabw. 4.5 $l_{\text{conf}} < 0.03$), für Knotendichte 5 $l_{\text{conf}} < 0.06$ bis zur Standardabweichung von 4.2)]

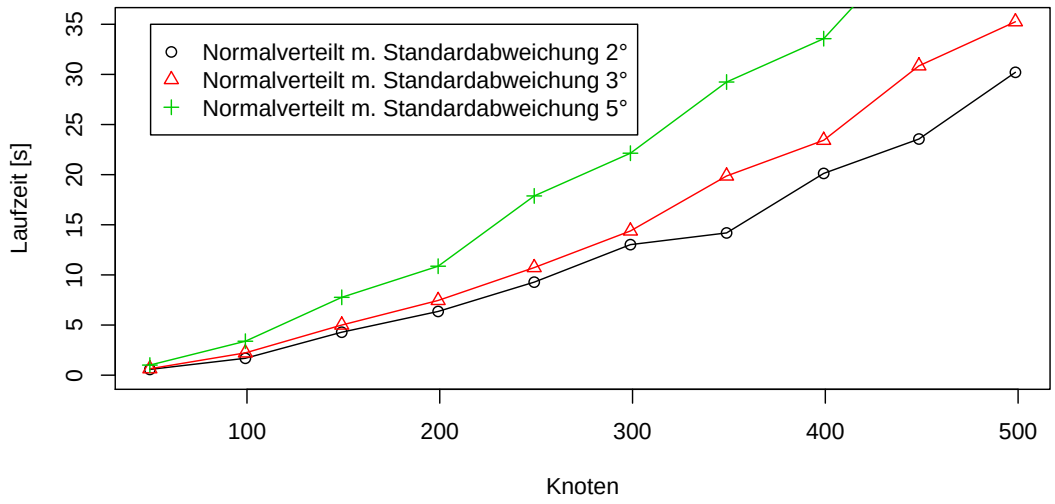


Abbildung 7.13.: Laufzeit bei zunehmender Knotenzahl bei normalverteilten Fehlern mit ausgewählten Standardabweichungen. Getestet wurden Graphen mit einer Knotendichte von 4.8.
[50 Messungen, $l_{\text{conf}} < 4s$]

8. Implementierung und Testumgebung

Die in den vergangenen Kapiteln vorgeschlagenen Algorithmen sind zur Evaluation und zur Überprüfung der Ergebnisse vollständig in Java in der aktuellen Version 1.5.0_04 implementiert.

Bei der Implementierung wurden folgende Bibliotheken von Drittanbietern verwendet:

yFiles Die Implementierung baut wesentlich auf den yFiles-Bibliotheken [Y] auf, einer kommerziellen Bibliothek für Graphen- und Layoutalgorithmik. Diese Bibliothek stellt die Datenstrukturen für Graphen und Listen wie in Kapitel 3 beschrieben, zur Verfügung. Darüber hinaus bietet sie eine Schnittstelle zur Integration von Algorithmen eine graphische Oberfläche, yEd; eine solche Integration ist für die beschriebenen Algorithmen umgesetzt worden, so dass die Ergebnisse leicht nachvollzogen werden können.

lp_solve, colt Als Teil von Algorithmus 4 werden Lösungen eines Linearen Programms bzw. eines homogenen linearen Gleichungssystems berechnet. Zur Lösung dieser Probleme wurden die frei verfügbaren Bibliothek lp_solve ([BEN05]), ein in C++ implementierter, quelloffener LP-Solver mit Java-Schnittstelle, bzw. die Colt-Bibliotheken [Col], die hochoptimierte Matrix-Arithmetik zur Verfügung stellt, inklusive schneller numerischer Verfahren zur Lösung linearer Gleichungssysteme, eingesetzt.

Die Ergebnisse der Tests basieren durchgehend auf mindestens 50 Messungen; die Zahl der Messungen sowie die Länge des Vertrauensintervalls ist jeweils angegeben. Alle Angaben und Statistiken zur Laufzeit wurden auf Rechnern mit Intel Xeon Prozessoren mit einer Taktrate von 2.8GHz und 512KB Cache (Messungen aus Kapitel 6) bzw. AMD Opteron 252 mit einer Taktrate von 2.6GHz und 1024KB Cache (Messungen aus Kapitel 7) gemessen und beziehen sich auf die *Systemzeit*. Obwohl darauf geachtet wurde, dass verwendete Klassen durch Vorläufe schon vor der Messung geladen sind und durch forciertes Aufrufen des Garbage-Collectors *vor* jedem Durchlauf Unterbrechungen durch die Java Virtual Machine verhindert werden, geben diese Messungen also kein ganz genaues Abbild. Da jedoch weder das Augenmerk dieser Arbeit auf Optimierung der Laufzeit liegt, noch die Implementierung besonders daraufhin untersucht wurde, reichen diese Messungen aus, um ein typisches Laufzeitverhalten prinzipiell zu beschreiben.

9. Zusammenfassung und Ausblick

9.1. Zusammenfassung

In der Lokalisation von Sensornetzwerken wurde bisher wenig auf lokale Richtungsinformation aufgebaut. Arbeiten, die sich mit dieser Variante beschäftigten, setzten sich wenig mit den fundamentalen Vorteilen auseindander. Diese Arbeit sollte vor allem folgende Ergebnisse vermitteln:

- Richtungsbasierte Lokalisierung stellt für Eindeutigkeit niedrigere Anfoderungen an die Struktur eines Netzes als entfernungsbasierte. Das Finden von verträglichen Einbettungen ist ohne weitere Einschränkungen in polynomieller Zeit möglich, im entfernungsbasierten Fall ist das ein $\mathcal{NP}$-schweres Problem.

- Für Sensornetzwerke plausible Annahmen erlauben den Entwurf von Algorithmen, die durch Richtungsinformationen eindeutig beschriebene Subgraphen identifiziert und korrekt einbettet. Diese Verfahren erreichen früh, d. h. schon bei geringen Knotendichten hohe Abdeckungen, bei denen für entfernungsbasierte Lokalisierung noch deutlich zu wenig Informationen vorlägen. Hier wurde das von Bruck et al. in [BGJ05] skizzierte Verfahren deutlich und bis zur Maximalität ausgedehnt und in der Laufzeit analysiert. So lässt sich oft auf die dort primär vorgeschlagene Vorgehensweise, das Lösen eines Linearen Programmes, verzichten.

- Die fehlende Skalierungsinformation ist durch Quasi-Unit-Disk-Graph-Bedingungen sehr präzise zu rekonstruieren; obwohl richtungsbasierte Lokalisierung für (Quasi-)Unit-Disk-Graphen wie die entfernungsbasierte Variante $\mathcal{NP}$-schwer ist, führt die gewissermaßen „orthogonale“ Information von Richtungen und (Quasi-)Unit-Disk-Graph-Annahmen trotzdem zu sehr guten Rekonstruktionen auch bei noch offenen Freiheitsgraden. Dies spiegelt direkt wider, dass verträgliche Realisierungen von Kantenrichtungen durch kontinuierliche Veränderungen ineinander zu überführen sind, während mit Kantenlängen verträgliche Einbettungen auch viel komplexer miteinander „verwandt“ sein können.

- Die beschriebenen Verfahren lassen sich, zusammen mit der Nutzung vorhandener Redundanz und lokaler Optimierung hervorragend zur Rekonstruktion bei fehlerbehafteter Richtungsinformation verwenden.

9.2. Ausblick

Lokalisationsverfahren sollten in letzter Konsequenz verteilt implementierbar sein; die vorgestellten sind sämtlich global entworfen und implementiert, allerdings durchgehend mit möglichst hoher Lokalität, so dass man zumindest Teile gut auf eine verteilte Umsetzung übertragen können sollte: Das Auffinden und Auslegen von kantenüberlappenden 2-simplen Subgraphen lässt sich prinzipiell leicht verteilen, indem alle (oder ausgewählte) Kanten zu Beginn die Grundlage einer neuen Einbettung bilden, dann wie gewohnt Einbettungen wachsen und bei Kantenüberlappung die Information zur Kombination entlang bereits eingebetteter Kanten propagiert werden. Entscheidungen, welche Einbettung bereits mehr Knoten lokalisiert hat, könnten dabei allerdings nicht getroffen werden; die Richtung der Kombination müsste aufgrund anderer (zwingend asymmetrischer) Informationen wie Knoten-IDs o. ä. festgelegt werden. Das Kombinieren knotenüberlappender generisch starrer Subgraphen würde bereits eine gewisse Organisation dieser Subgraphen voraussetzen. Ist allerdings eine Kommunikation zwischen Schnittknoten zu angrenzenden Subgraphen möglich (zum Beispiel entlang eines Spannbaumes), ließen sich doppelte Überlappungen durchaus mit vertretbarem Aufwand identifizieren. Bei der Kombination allgemeiner Subgraphenmengen kann natürlich beliebig weit reichende Kommunikation erfordern. Hier könnte allerdings eine Beobachtung aus der Praxis helfen: Die meisten Verschmelzungen nach Algorithmus 4 betreffen verhältnismäßig wenige Subgraphen, fast immer würde es reichen, wenn jeder Subgraph das Problem eingeschränkt auf eine sehr geringe Umgebung, z. B. direkt angrenzende Subgraphen und deren direkte Nachbarn, kennen würde. Komplexere Szenarien kommen zwar vor, sind aber eine deutliche Ausnahme. Alle in Kapitel 7 vorgestellten lokalen Optimierungsverfahren sind leicht auch verteilt umzusetzen.

Im Ausklang von Kapitel 5 wurde erwähnt, dass sich alle Ergebnisse zu Kantenrichtungen auf beliebige Dimensionen erweitern lassen. Das legt eine Beschäftigung vor allem mit dem dreidimensionalen Fall nahe, ohne dabei wie im entfernungsbasierten Fall einen viel komplexeren kombinatorischen Hintergrund (zudem mit vielen offenen Fragen) vorzufinden.

Was das Laufzeitverhalten angeht, zeigt sich die lokale Optimierung (trotz aller Beschränkungen) in der bisherigen Umsetzung als dominierend; verglichen mit bekannten Verfahren zur Längenrealisierung durch kräftebasierte Optimierung konvergierte das implementierte Verfahren jeweils verhältnismäßig langsam.

Über diese Punkte hinaus wird es mit steigendem tatsächlichen Einsatz von Sensornetzwerken immer mehr darum gehen, die sehr idealisierte Sichtweise vieler mathematisch geprägter Modellierungen den Gegebenheiten anzupassen, um *relevante* Forschung und Aussagen präsentieren zu können.

A. Anhang

A.1. Zusätzliche Tests

Die folgenden Seiten zeigen jeweils die in Kapitel 7 für normalverteilte Fehler vorgestellten Ergebnisse für gleichverteilte Fehler (in einem jeweils angegebenen Intervall) und für Diskretisierungsfehler (zu einer jeweils angegebenen Schrittweite

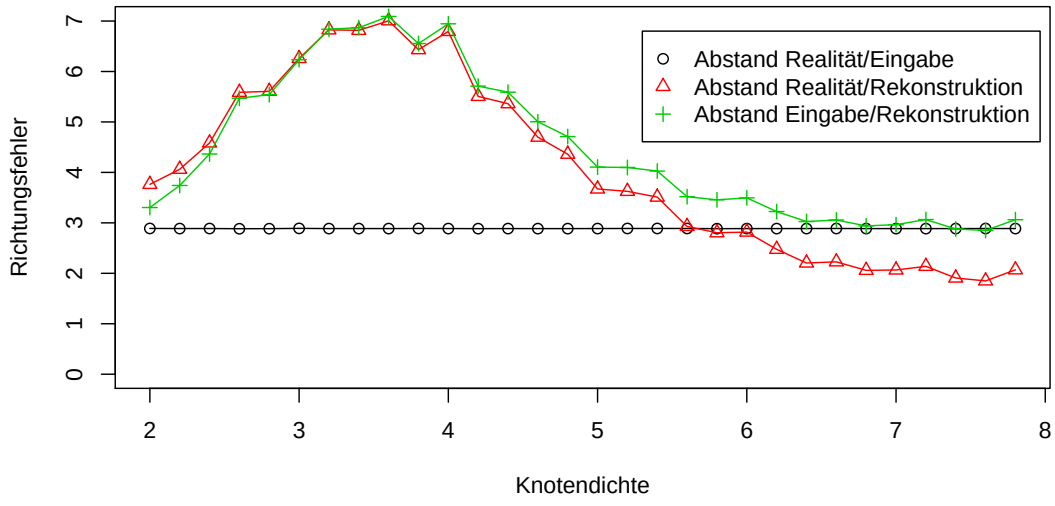


Abbildung A.1.: Die drei Fehler $d(\omega, \widetilde{\omega})$, $d(\omega, \hat{\omega})$ und $d(\widetilde{\omega}, \hat{\omega})$ bei gleichverteiltem Fehler in einem Intervall von $[-5°; 5°]$ und zunehmender Knotendichte.
[200 Messungen, durchgehend $l_{\text{conf}} < 0.6°$, ab einer Dichte von 4.4 ist $l_{\text{conf}} < 0.4°$]

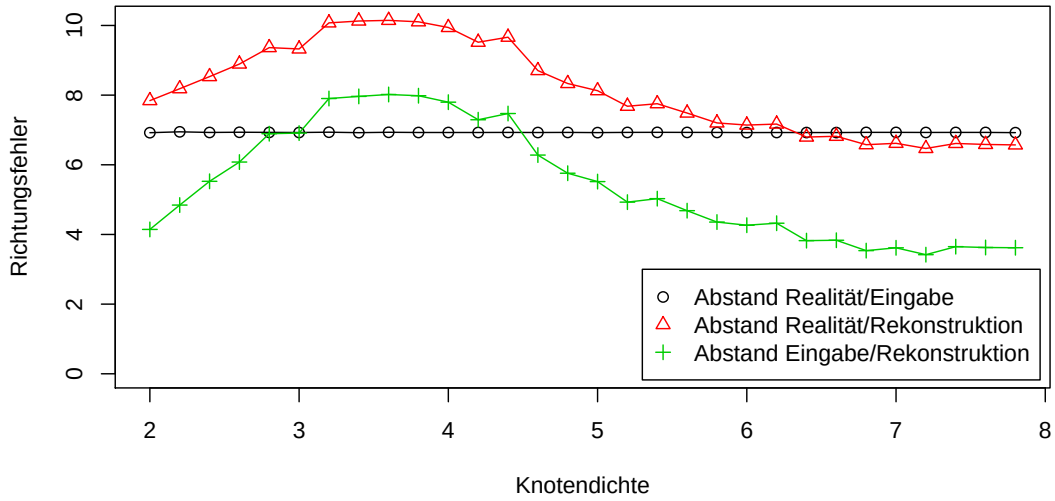


Abbildung A.2.: Die drei Fehler $d(\omega, \widetilde{\omega})$, $d(\omega, \hat{\omega})$ und $d(\widetilde{\omega}, \hat{\omega})$ bei Diskretisierung auf Vielfache von 12° und zunehmender Knotendichte.
[200 Messungen, durchgehend $l_{\text{conf}} < 0.6°$, ab einer Dichte von 5.2 ist $l_{\text{conf}} < 0.4°$]

A.2. Nebenrechnungen

Lemma 23 *Sind $k_1, \ldots, k_l \in \mathbb{N}$ ganze Zahlen mit $k_i \geq 2$, dann ist*

$$\sum_i^l k_i^3 \leq \left(\sum_i^l (k_i - 1) + 1 \right)^3$$

Beweis. Per Induktion über l lässt sich die Aussage zeigen:

IA, $l = 2$: Zu zeigen ist für $l = 2$, dass $a^3 + b^3 \leq (a + b - 1)^3$, durch Umformung für O.B.d.A. $a \geq b$:

$$\begin{aligned}
a^3 + b^3 &\leq (a+b-1)^3 \\
a^3 + b^3 &\leq (a+b)^3 - 3(a+b)^2 + 3(a+b) - 1 \\
a^3 + b^3 &\leq a^3 + 3a^2b + 3ab^2 + b^3 - 3(a+b)^2 + 3(a+b) - 1 \\
0 &\leq 3a^2b + 3ab^2 - 3(a+b)^2 + 3(a+b) - 1 \\
3a^2 + 6ab + 3b^2 + 1 &\leq 3a^2b + 3ab^2 + 3(a+b) \\
\underbrace{3a^2 + 3b^2}_{\leq 6a^2} + 6ab + 1 &\leq \underbrace{3a^2b}_{\geq 6a^2} + \underbrace{3ab^2}_{\geq 6ab} + \underbrace{3(a+b)}_{\geq 1} \\
6a^2 + 6ab + 1 &\leq 6a^2 + 6ab + 1
\end{aligned}$$

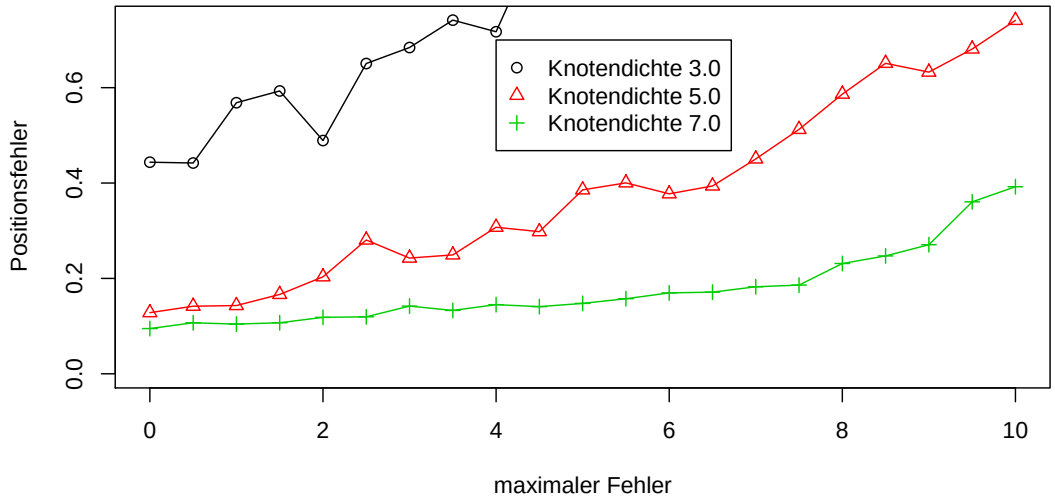


Abbildung A.3.: Mittlere Knotenabstände $Q_{\overline{\text{GTE}}}$ bei zunehmendem maximalen Fehler des gleichverteilten Fehlers für ausgewählte Knotendichten
[100 Messungen, für Knotendichte 7 $l_{\text{conf}} < 0.09$ (bis Standardabw. 9.0 $l_{\text{conf}} < 0.03$), für Knotendichte 5 $l_{\text{conf}} < 0.1$)]

IS, $l = n \rightsquigarrow l = n + 1$**:**

$$\begin{aligned}
\sum_{i}^{n+1} c \cdot k_i^3 &= \sum_{i}^{n} c \cdot k_i^3 + k_{n+1}^3 = \underbrace{\left(\sum_{i}^{l} (k_i - 1) + 1\right)}_{=:a>2}^3 + \underbrace{k_{n+1}}_{=:b}^3 \\
&= a^3 + b^3 \\
&\leq ((a-1) + (b-1) + 1)^3 \\
&= \left(\sum_{i=1}^{n} (k_i - 1) + 1 - 1 + (k_{n+1} - 1) + 1\right)^3 \\
&= \left(\sum_{i=1}^{n+1} (k_i - 1) + 1\right)^3
\end{aligned}$$

□

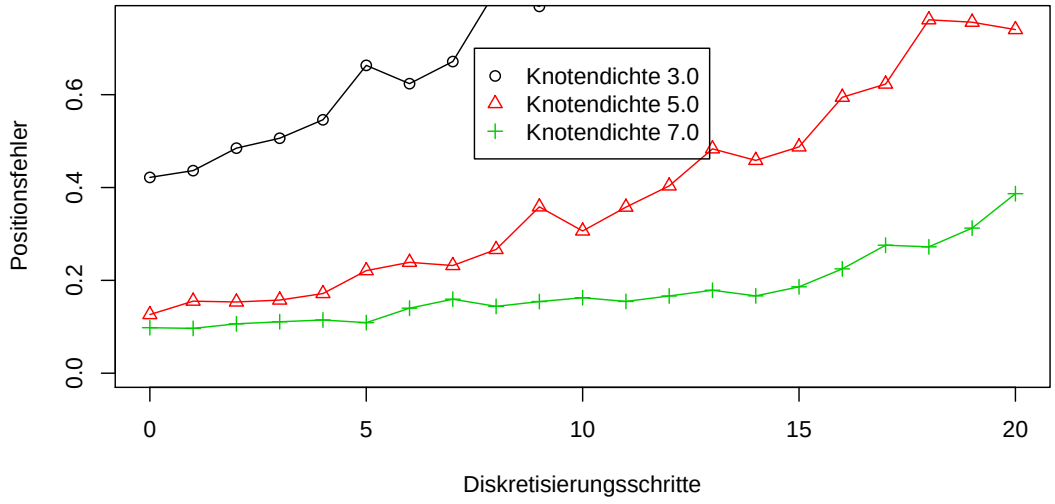


Abbildung A.4.: Mittlere Knotenabstände $Q_{\overline{\mathrm{GTE}}}$ bei zunehmender Schrittweite des Diskretisierungsfehlers für ausgewählte Knotendichten
[100 Messungen, für Knotendichte 7 $l_{\mathrm{conf}} < 0.06$, für Knotendichte 5 $l_{\mathrm{conf}} < 0.1$)]

Abbildungsverzeichnis

Literaturverzeichnis

[AGY04] Aspnes, J. ; Goldenberg, D. ; Yang, Y.: On the computational complexity of sensor network localization. In: *Proceedings of the First International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004

[BEN05] Berkelaar, M. ; Eikland, K. ; Notebaert, P. *lp_solve version 5.5.0.0*. http://lpsolve.sourceforge.net/5.5/. 2005

[BETT99] di Battista, G. ; Eades, P. ; Tamassia, R. ; Tollis, I. G.: *Graph Drawing*. Prentice Hall, 1999. – ISBN 0–13–301615–3

[BGJ05] Bruck, J. ; Gao, J. ; Jiang, A.: Localization and Routing in Sensor Networks by Local Angle Information. In: *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA : ACM Press, May 2005. – ISBN 1–59593–004–3, S. 181–192

[BK93] Breu, H. ; Kirkpatrick, D. G.: Unit Disk Graph Recognition is NP-Hard. Department of Computer Science, University of British Columbia, 1993. – Forschungsbericht

[BW04] Bischoff, R. ; Wattenhofer, R.: Analyzing Connectivity-Based Multi-Hop Ad-hoc Positioning. In: *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*. Washington, DC, USA : IEEE Computer Society, 2004. – ISBN 0–7695–2090–1, S. 165

[BY04] Biswas, P. ; Ye, Y.: Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. In: *IPSN'04: Proceedings of the third international symposium on Information processing in sensor networks*. New York, NY, USA : ACM Press, 2004. – ISBN 1–58113–846–6, S. 46–54

[CLR90] Cormen, T. H. ; Leiserson, C. E. ; Rivest, R. L.: *Introduction to Algorithms*. MIT Press, 1990. – ISBN 0–262–03141–8, 0–07–013143–0

[Col] *The Colt Project*. http://dsd.lbl.gov/~hoschek/colt/

[EGW+04] Eren, T. ; Goldenberg, D. K. ; Whiteley, W. ; Yang, Y. R. ; Morse, A. S. ; Anderson, B. D. O. ; Belhumeur, P. N.: Rigidity, Computation, and Randomization in Network Localization. In: *InfoCom '04:*. New York, NY, USA, March 2004, S. 2673 – 2684

[EWM+03] Eren, T. ; Whiteley, W. ; Morse, A. S. ; Belhumeur, P. N. ; Anderson, B. D. O.: Sensor and Network Topologies of Formations with Direction, Bearing and Angle Information between Agents. In: *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003

[GK05] Gotsman, C. ; Koren, Y.: Distributed Graph Layout for Sensor Networks. In: *Proceedings of the 12th International Symposium on Graph Drawing (GD'04)* Bd. 3383, 2005, S. 273–284

[Hen92] Hendrickson, B.: Conditions for Unique Graph Realizations. In: *SIAM J. Comput.* 21 (1992), Nr. 1, S. 65–84

[Jun99] Jungnickel, D.: *Graphs, Networks and Algorithms*. Berlin : Springer, 1999

[KMW04] Kuhn, F. ; Moscibroda, T. ; Wattenhofer, R.: Unit Disk Graph Approximation. In: *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*. New York, NY, USA : ACM Press, 2004. – ISBN 1–58113–921–7, S. 17–23

[KWZ03] Kuhn, F. ; Wattenhofer, R. ; Zollinger, A.: Ad-hoc Networks Beyond Unit Disk Graphs. In: *DIALM-POMC '03: Proceedings of the 2003 Joint Workshop on Foundations of Mobile Computing*. New York, NY, USA : ACM Press, 2003. – ISBN 1–58113–765–6, S. 69–78

[MLRT04] Moore, D. ; Leonard, J. ; Rus, D. ; Teller, S.: Robust Distributed Network Localization with Noisy Range Measurements. In: *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA : ACM Press, November 2004. – ISBN 1–58113–879–2, S. 50–61

[MOWW04] Moscibroda, T. ; O'Dell, R. ; Wattenhofer, M. ; Wattenhofer, R.: Virtual Coordinates for Ad hoc and Sensor Networks. In: *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*. New York, NY, USA : ACM Press, 2004. – ISBN 1–58113–921–7, S. 8–16

[NN03a] Niculescu, D. ; Nath, B.: DV Based Positioning in Ad Hoc Networks. In: *Telecommunication Systems, Volume 22, Issue 1 - 4, Pages 267 - 280*, 2003

[NN03b] Niculescu, D. ; Nath, B.: Position and orientation in ad hoc networks. In: *Elsevier Ad Hoc Networks*, 2003

[NN04] Niculescu, D. ; Nath, B.: Error Characteristics of Ad Hoc Positioning Systems (APS). In: *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA : ACM Press, 2004. – ISBN 1–58113–849–0, S. 20–30

[PBDT03] Priyantha, N. B. ; Balakrishnan, H. ; Demaine, E. ; Teller, S.: Anchor-Free Distributed Localization in Sensor Networks / MIT LCS. 2003 (TR-892). – Forschungsbericht

[PMBT01] Priyantha, N. B. ; Miu, A. K. L. ; Balakrishnan, H. ; Teller, S.: The Cricket Compass for Context-aware Mobile Applications. In: *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. New York, NY, USA : ACM Press, 2001. – ISBN 1–58113–422–3, S. 1–14

[Sax79] Saxe, J. B.: Embeddability of weighted graphs in k-space is strongly NP-hard. In: *Proc. 17th Allerton Conf. Commun. Control Comput.*, 1979, S. 480–489

[SGA+03] Savvides, A. ; Garber, W. ; Adlakha, S. ; Moses, R. ; Srivastava, M. B.: On the Error Characteristics of Multihop Node Localization in Ad-Hoc Sensor Networks. In: *The Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN 03)*, 2003

[SHS01] Savvides, A. ; Han, C.-C. ; Strivastava, M. B.: Dynamic Fine-grained Localization in Ad-Hoc networks of Sensors. In: *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. New York, NY, USA : ACM Press, 2001. – ISBN 1–58113–422–3, S. 166–179

[SRB01] Savarese, C. ; Rabaey, J. M. ; Beutel, J.: Locationing in distributed ad hoc wireless sensor networks. In: *Proc. 2001 Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 2001)* Bd. 4, IEEE, Piscataway, NJ, May 2001, S. 2037–2040

[TM03] Tubaishat, M. ; Madria, Sanjay: Sensor Networks: An Overview. In: *IEEE Potentials, 22(2)*, 2003, S. 20–23

[Whi96] Whiteley, W.: Matroids from Discrete Applied Geometry. In: *Matroid Theory, AMS Contemporary Mathematics*, 1996, S. 171–311

[Y] *The yFiles Java Library 2.4*. http://www.yworks.com/en/products_yfiles_about.htm