

Distributed Graph Layout for Sensor Networks

Ausarbeitung zum Seminar

Zeichnen von Graphen (WS04/05)

Prof. Dr. Ulrik Brandes

Basierend auf dem gleichnamigen Artikel von C. Gotsman und Y. Koren,
Proceedings of 12th International Symposium on Graph Drawing (GD'04).

von
Lars Volkhardt
Universität Konstanz

4. Mai 2005

Inhaltsverzeichnis

1	Einleitung	1
2	Das Problem	2
3	Das Verfahren von Gotsman und Koren	5
4	Experimentelle Ergebnisse	13

1 Einleitung

Sensor-Netzwerke kommen heutzutage in sehr vielen Bereichen zum Einsatz. Allen gemeinsam ist, dass mehrere Sensoren – auf bestimmte Art verteilt im Gelände – Daten über ihre Umwelt sammeln sollen. Angefangen bei der einfachen Bestimmung der Umgebungstemperatur, über Bestimmung der Luftfeuchtigkeit bis hin zu aufwendigen Messungen von Strahlung jeglicher Art, sind ihre Aufgaben so vielfältig wie ihre Einsatzgebiete. Vor allem das Militär hat Interesse an Sensor-Netzwerken, aber auch im Umweltbereich kommen sie häufig zum Einsatz. Ein vorstellbares Szenario ist beispielsweise der Einsatz eines Sensors-Netzwerks in der Landwirtschaft, zur Bestimmung des pH-Wertes an verschiedenen Stellen eines oder mehrerer Felder.

Um die erhaltenen Daten sinnvoll verwerten zu können, ist es wichtig, die geografische Lage jedes einzelnen Sensors genau bestimmen zu können. Bisher wurde für diesen Zweck oftmals ein GPS-Empfänger in jeden Sensor integriert, der die Position des Sensors mit Hilfe von Satelliten bestimmt. Dazu benötigt der GPS-Empfänger Kontakt zu mindestens drei Satelliten in der Erdumlaufbahn. Ein Einsatz des Sensor-Netzwerkes in geschlossenen Räumen würde damit praktisch schon ausgeschlossen.

Ein weiteres Problem sind die Kosten, welche mit dem Einsatz von GPS verbunden sind. Die Integration eines GPS-Empfängers in jeden einzelnen Sensor ist sehr teuer, außerdem steigt damit auch der Stromverbrauch unter Umständen um ein Vielfaches. Außerdem ist es vorstellbar, dass für die GPS-Empfänger schlicht kein Platz mehr vorhanden ist im Sensor-Netzwerk. Oder die Sensoren liegen so dicht beieinander, dass die Genauigkeit von GPS – welche im Bereich von 0.5 bis 1 m liegt – bei weitem nicht mehr ausreicht.

Besser wäre es also, wenn die Sensoren ihre globale Position im Gelände mittels Kommunikation untereinander bestimmen könnten. Eine Möglichkeit dafür besteht darin, die Position mit Hilfe geometrischer Messungen durchzuführen. Beispielsweise kann der Abstand zwischen zwei Sensoren aus dem

Zeitunterschied zwischen dem Absenden und dem Empfang eines Datenpakets errechnet werden. Eine andere Möglichkeit besteht in der Betrachtung der Signalstärke, welches ein versendetes Signal beim Empfänger noch aufweist. Eine größere Abschwächung der Signalstärke deutet dabei auf einen größeren Abstand zwischen den zwei Sensoren hin. Bei diesem Vorgehen muss allerdings beachtet werden, dass eine Kommunikationsmöglichkeit meistens nur zwischen je zwei benachbarten Sensoren besteht. Ein Datenaustausch zwischen zwei beliebigen Sensoren ist dadurch nur mit aufwändigen Routing-Algorithmen möglich. Es gilt also, die Kommunikation zwischen beliebigen Sensoren so gering wie möglich zu halten, und nur Daten zwischen benachbarten Sensoren auszutauschen.

Der vorliegende Artikel von Craig Gotsman und Yehuda Koren versucht dieses Problem auf effiziente Art zu lösen.

2 Das Problem

Der Artikel von Craig Gotsman und Yehuda Koren aus dem Jahr 2004 versucht folgendes Problem zu lösen:

Gegeben eine Menge von Sensoren verteilt in der Ebene, und ein Mechanismus mit dem jeder Sensor die Distanz zu wenigen benachbarten Sensoren messen kann, bestimme die Koordinaten jedes Sensors mittels lokaler Sensor-zu-Sensor Kommunikation.

Dabei werden die Koordinaten aller Sensoren als das *Layout* des Sensor-Netzwerks bezeichnet.

Das ganze Vorgehen ist also vergleichbar mit einem verteilten Layout-Algorithmus für Graphen, bei dem nur Daten zwischen benachbarten Knoten in die Berechnung einfließen. Dabei ist wichtig zu beachten, dass es meist keine eindeutige Lösung für das Problem gibt. Ein einfaches Beispiel dafür ist in Abbildung 1 zu sehen. Dort ist auf der linken Seite ein Graph mit drei

Knoten abgebildet, die auf einer Geraden liegen. Als Kriterium für ein gutes Layout wurde definiert, dass der Abstand zwischen benachbarten Knoten **1** betragen soll. Werden als Güte-Kriterium für ein Layout nur die Abstände zwischen benachbarten Knoten berücksichtigt, so lässt sich zum Beispiel die Position des dritten Knotens im Graph wie auf der rechten Seite der Abbildung gezeigt verändern. Auch der rechte Graph erfüllt also die geforderten Abstandsbedingungen zwischen benachbarten Knoten.

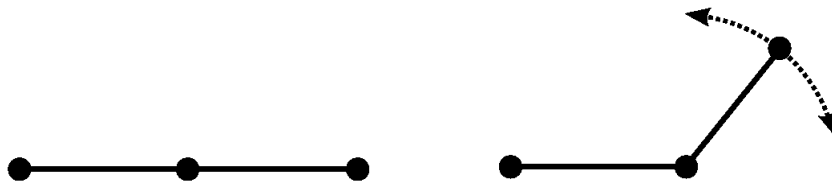


Abbildung 1: Einfacher Graph mit 3 Knoten

Es existiert also meistens mehr als eine Lösungen für ein gegebene Ausprägung des Problems. Oftmals kommt es dann zu Überlappungen eines Teilgraphs auf einen anderen. Dies ist in Abbildung 2 dargestellt. Hier begin-

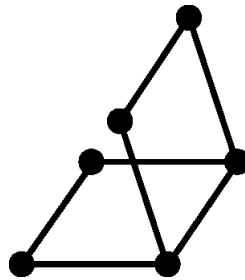


Abbildung 2: Überlappungen im Graph

nen sich zwei Teilgraphen zu überlappen, ohne dass die geforderten Bedingungen verletzt werden. Sämtliche Abstände zwischen allen benachbarten Knoten betragen weiterhin **1**. Selbst wenn die zwei oberen Knoten im Layout die zwei Knoten links im Layout vollständig überlagern, also jeder Knoten die selbe Position wie der jeweils andere hat, würden die geforderten Bedingungen erfüllt. Die größte Schwierigkeit ist eine Lösung für das Problem zu finden,

welche frei von Überlappungen ist.

Ein weiteres Problem in Sensor-Netzwerken ist das Auftreten von Störungen. Keine der geometrischen Messung zwischen den benachbarten Knoten kann störungsfrei erfolgen, insbesondere wenn die Abstände, und damit die gemessenen Zeiten, sehr klein werden. Die Störeinflüsse können gar so groß werden, dass es keine mögliche Lösung des Problems für die gemessenen Abstände zwischen den Sensoren gibt.

Gotsman und Koren definieren den Term „benachbart“ mit einer wahrscheinlichkeitstheoretischen Version des Kreis-Graphen Modells. Um einen Knoten v werden zwei Kreise mit Radien R_1 und R_2 gezogen, wobei $R_1 \leq R_2$. Jeder Knoten mit Abstand kleiner R_1 von v ist mit v benachbart, jeder Knoten mit Abstand größer R_2 ist nicht benachbart. Jeder Knoten zwischen R_1 und R_2 ist mit einer bestimmten Wahrscheinlichkeit p mit v benachbart (siehe Abbildung 3).

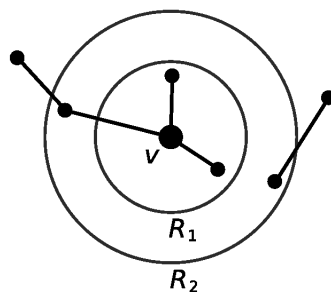


Abbildung 3: Probabilistischer Kreis-Graph

Das Verfahren von Gotsman und Koren besteht aus zwei Phasen. In der ersten Phase wird ein Layout für den gegebenen Graphen gesucht, welches das existierende Sensor-Netzwerk sehr gut approximiert. Dieses Ausgangslayout wird anschließend in der zweiten Phase mittels Stress-Minimierung optimiert.

3 Das Verfahren von Gotsman und Koren

Gegeben ist ein Graph $G(V = \{1, \dots, n\}, E)$, und für jede Kante $\langle i, j \rangle \in E$ seine euklidische Länge l_{ij} , die dem Abstand zweier Sensoren i, j im Sensor-Netzwerk entspricht. Ein 2D-Layout des Graphen wird repräsentiert durch zwei Vektoren x, y ($x, y \in \mathbb{R}^n$), in denen jeweils das i . Element der beiden Vektoren den Koordinaten des Knotens i entspricht. Der tatsächliche Abstand zwischen zwei Knoten i und j im Graph G wird durch d_{ij} repräsentiert: $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Im Idealfall ohne Störeinflüsse existiert ein Layout, welches die geforderten Abstände erfüllt ($d_{ij} = l_{ij}$). Ziel ist es, dieses Layout zu finden. Um die oben erwähnten Überlappungen auszuschließen, definieren Gotsman und Koren eine weitere Anforderung an das Layout des Graphen. Darin muss der Abstand zwischen zwei nicht benachbarten Knoten mindestens so groß sein wie R , wobei R der maximale Abstand zwischen zwei benachbarten Knoten i und j ist, also $R = \max_{\langle i, j \rangle \in E} l_{ij}$.

Das **Layout-Problem** wird nun wie folgt definiert:

Gegeben ein Graph $G(\{1, \dots, n\}, E)$ und für jede Kante $\langle i, j \rangle \in E$ seine (geforderte) Länge l_{ij} , finde ein optimales Layout (p_1, \dots, p_n) mit $p_i \in \mathbb{R}^2$ als Koordinaten des Knotens i , welches für alle $i \neq j$ erfüllt:

$$\begin{cases} \|p_i - p_j\| = l_{ij} & \text{falls } \langle i, j \rangle \in E \\ \|p_i - p_j\| > R & \text{falls } \langle i, j \rangle \notin E \end{cases}$$

Ein solches optimales Layout wäre vergleichbar mit einem Layout, welches von einem Kräfte-basierenden Layout-Algorithmus erzeugt würde. Dabei werden benachbarten Knoten zusammengezogen, während nicht-benachbarte Knoten sich abstoßen. Ein optimales Layout erhält man dann durch Minimierung der globalen Stress-Energie:

$$\sum_{i < j} \frac{(d_{ij} - l_{ij})^2}{l_{ij}^2}$$

Allerdings kommt diese Möglichkeit hier nicht in Frage, da die nötige Kommunikation im Sensors-Netzwerk so gering wie möglich bleiben soll. Auf den verwendeten Layout-Algorithmus bezogen bedeutet dies, dass keine Daten zwischen nicht-benachbarten Knoten in die Berechnung einfließen sollen.

3.1 Erste Phase: Gute Ausgangslayouts und Eigenwert-Projektion

Gotsman und Koren wandeln zur Vermeidung der globalen Berechnungen die globale Stress-Energie in eine *lokale* Stress-Energie und definieren diese durch:

$$\text{Stress}(x, y) = \sum_{\langle i, j \rangle \in E} (d_{ij(x, y)} - l_{ij})^2 \quad (1)$$

Da diese Form der Energie-Funktion nicht normiert – und die Funktion nicht konvex ist – besitzt sie eventuell viele lokale Minima. Diese können unter Umständen vom eigentlichen (globalen) Minimum weit entfernt sein. Damit die Minimierung der Stress-Funktion im globale Minimum konvergiert, starten Gotsman und Koren den Optimierungsprozess mit einem günstigen Ausgangslayout. Um dieses günstige Layout zu erhalten, versuchen sie nicht-benachbarte Knoten weit voneinander zu entfernen, und so ein Layout zu generieren, in welchem alle Knoten gut verteilt sind. *Gut* bedeutet in diesem Rahmen, dass keine Häufungen von Knoten entstehen, benachbarte Knoten nah beieinander und umgekehrt nicht-benachbarte Knoten weit entfernt voneinander sind. Eine Zielfunktion, die dieses berücksichtigt, definieren sie (zuerst für eine Dimension) durch:

$$E(x) = \frac{\sum_{\langle i, j \rangle \in E} w_{ij} \|x_i - x_j\|^2}{\sum_{i < j} \|x_i - x_j\|^2} \quad (2)$$

Deutlich erkennbar ist die Tatsache, dass weit entfernte Knoten i und j den Nenner der Funktion vergrößern, und dadurch – zumindest dann, wenn i und j nicht benachbart sind – die Energie $E(x)$ verkleinern. Dabei ist w_{ij} ein Maß für die Ähnlichkeit zweier benachbarter Knoten, welches abhängig sein sollte

von der Entfernung der beiden zueinander¹.

Um die Energiefunktion $E(x)$ zu berechnen, ohne globale Werte einbeziehen zu müssen, definieren Gotsman und Koren eine Matrix W mit den Einträgen w_{ij} wie oben. Außerdem eine Diagonalmatrix D , deren i -ter Diagonaleintrag die Summe der i -ten Zeile aus W ist. Das globale Minimum von $E(x)$ ist nun der Eigenvektor zum zweitgrößten Eigenwert λ_2 der Matrix $D^{-1}W$.²

Ließe sich der Eigenvektor dieser Matrix auf dezentrale Weise berechnen, bei der nur Werte zwischen benachbarten Knoten berücksichtigt werden, hätte man das gewünschte, optimale Ausgangslayout gefunden. Ein Beispiel wird diesen Sachverhalt verdeutlichen.

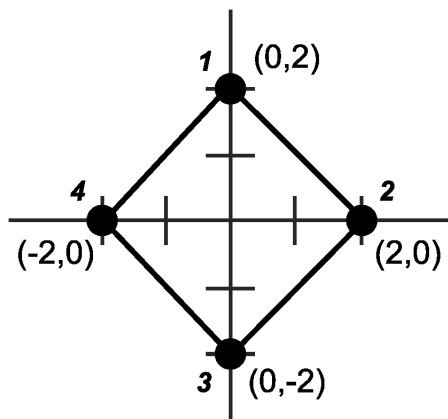


Abbildung 4: Beispiel

In Abbildung 4 ist ein einfacher Graph mit 4 Knoten gegeben. Der Abstand zwischen je zwei benachbarten Knoten i, j besträgt stets $d_{ij} = \sqrt{2^2 + 2^2} = \sqrt{8}$. Die Matrix W mit ihren Einträgen w_{ij} , definiert wie oben, und die dazugehörige Diagonalmatrix D sind in Abbildung 5 gegeben.

¹Gotsman und Koren setzen $w_{ij} = e^{-l_{ij}}$

²Dabei kann das Ergebnis unter Umständen eine Rotation und Skalierung des ursprünglichen Graphen darstellen, allerdings bleiben die Beziehungen der Knoten zueinander erhalten

$$W = \begin{pmatrix} 0 & e^{-\sqrt{8}} & 0 & e^{-\sqrt{8}} \\ e^{-\sqrt{8}} & 0 & e^{-\sqrt{8}} & 0 \\ 0 & e^{-\sqrt{8}} & 0 & e^{-\sqrt{8}} \\ e^{-\sqrt{8}} & 0 & e^{-\sqrt{8}} & 0 \end{pmatrix} \quad D = \begin{pmatrix} 2 \cdot e^{-\sqrt{8}} & 0 & 0 & 0 \\ 0 & 2 \cdot e^{-\sqrt{8}} & 0 & 0 \\ 0 & 0 & 2 \cdot e^{-\sqrt{8}} & 0 \\ 0 & 0 & 0 & 2 \cdot e^{-\sqrt{8}} \end{pmatrix}$$

Abbildung 5: Matrizen W und D

Die resultierende Matrix $D^{-1} \cdot W$ lautet:

$$D^{-1} \cdot W = \begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{pmatrix}$$

Die Eigenwerte von $D^{-1} \cdot W$ sind -1 , 0 , 1 und 0 . Die dazugehörigen Eigenvektoren lauten:

$$\begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}; \begin{pmatrix} 0 \\ -0.7 \\ 0 \\ 0.7 \end{pmatrix}; \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}; \begin{pmatrix} 0.7 \\ 0 \\ -0.7 \\ 0 \end{pmatrix}$$

Wie man erkennen kann, ist der Eigenvektor zum größten Eigenwert 1 der Matrix schlicht der Einheitsvektor. Der zweite Eigenvektor beinhaltet allerdings die x-Koordinaten (wenn auch skaliert) des gewünschten, optimalen Ausgangslayouts. Auch ist hier schon erkennbar, dass der dritte Eigenvektor (zum dritten Eigenwert der Matrix) die y-Koordinaten der Knoten enthält.

Um diese beiden Eigenvektoren der Matrix $D^{-1} \cdot M$ berechnen zu können, ohne globale Werte einzubeziehen, wenden Gotsman und Koren eine Potenziteration auf die Matrix $I + D^{-1} \cdot M$ an. Bei einer Potenziteration über eine Matrix A wird ein beliebiger³ Ausgangsvektor $q^{(0)}$ mit der Matrix A multipliziert. Anschließend wird der erhaltene Vektor (als $q^{(1)}$ bezeichnet) mit seiner Länge normiert, und abermals mit der Matrix multipliziert. Dieses

³mit gleicher Dimension wie die Breite der Matrix

Vorgehen wird iteriert, bis schließlich der Vektor gegen den Eigenvektor zum *größten* absoluten Eigenwert der Matrix A konvergiert. Man erhält also nach k Schritten eine Approximation $q^{(k)}$ des Eigenvektors zum größten Eigenwert der Matrix A . Allerdings sind die x- (bzw. y-) Koordinaten, wie oben im Beispiel gezeigt, im Eigenvektor zum zweitgrößten (bzw. drittgrößten) Eigenwert enthalten. Um mittels Potenziteration im zweiten Eigenvektor zu konvergieren, muss die Potenziteration mit einem Ausgangsvektor $v^{(0)}$ orthogonal zum größten Eigenvektor von A gestartet werden. Das gleiche macht man anschließend mit dem zweiten Vektor als Ausgangsvektor, um auf den Eigenvektor zum drittgrößten Eigenwert zu schließen.

Im konkreten Fall bei Gotsman und Koren ist die Potenziteration durch folgende Iteration gegeben:

$$x_i \leftarrow a \left(x_i + \frac{\sum_{\langle i,j \rangle \in E} w_{ij} x_j}{\sum_{\langle i,j \rangle \in E} w_{ij}} \right) \quad (3)$$

Hier wird die Position eines Knotens i iterativ angepasst an den Durchschnitt aller seiner Nachbarn j . Dabei bestimmt a das Wachstum der Koordinatenwerte. Um auf geeignete (orthogonale) Ausgangsvektoren – also geeignete Ausgangspositionen für die Knoten – zu kommen, konstruieren sie eine *lokale* Matrix A , so dass $A^T D x = 0$. *Lokal* bedeutet dabei, dass nur Einträge in der Matrix einen Wert $\neq 0$ haben, wenn die entsprechenden Knoten i, j im Graph verbunden sind (Abb. 6). Wird nun eine Potenziteration mit einem

$$A_{i,j} = \begin{cases} -x_j/D_{ii} & \text{falls } \langle i, j \rangle \in E \\ 0 & \text{falls } \langle i, j \rangle \notin E, i \neq j \\ -\sum_k A_{i,k} & \text{falls } i = j \end{cases} \quad i, j = 1, \dots, n$$

Abbildung 6: *Lokale* Matrix A

beliebigen Vektor $v^{(0)}$ auf die lokale Matrix A angewendet, (bei der nur lokale Operationen nötig sind), so erhält man den eigentlichen Ausgangsvektor,

also die Ausgangskordinaten für die Potenziteration in (3).

Damit durch die Iteration die Skalierung der x- und y-Koordinaten gleich bleibt, also $\|x\| = \|y\|$, werden die x- und y-Werte normiert mit der Spannweite der Koordinaten:

$$x_i \leftarrow \frac{x_i}{\max_i x_i - \min_i x_i}$$

So wird sichergestellt, dass sämtliche Koordinatenwerte im Bereich zwischen 0 und 1 liegen. Gotsman und Koren erwähnen an dieser Stelle, dass die Berechnung der Minima und Maxima bei jeder Iteration leicht lokalisierbar ist, also keine globalen Berechnungen dazu benötigt werden. Leider liefern sie keine Begründung für ihre Behauptung. Eine gewöhnliche Berechnung der Maxima würde sehr wohl globale Vergleiche zwischen den Knotenpositionen benötigen, und dies würde dem ganzen Verfahren – welches ja gerade auf lokale Berechnung des Layouts abzielt – zuwider laufen.

3.2 Zweite Phase: Optimierung der Stress-Energie

Ziel der zweiten Phase ist es, das in der ersten Phase gewonnene Ausgangs-layout zu optimieren, um ein Layout zu erhalten, welches das eigentlichen Sensor-Netzwerk so gut wie möglich approximiert. Durch Berechnung eines solchen Layouts hätten die Sensoren anschließend alle nötigen Informationen über ihre Position, welche für eine sinnvolle Auswertung der Daten notwendig ist.

Eine einfache, bisher gebräuchliche Art die Stress-Energie zu minimieren, ist das Gradientenabstiegsverfahren. Bei diesem iterativen Verfahren werden die Koordinaten eines Knotens i zum Zeitpunkt $t + 1$ derart angepasst, dass sich der Knoten zu einem benachbarten Knoten hinbewegt beziehungsweise von diesem wegbewegt, in Abhängigkeit von der tatsächlichen Länge d_{ij} zum Zeitpunkt t und der „gewünschten“ Länge l_{ij} :

$$x_i(t + 1) = x_i(t) + \delta \sum_{j:(i,j) \in E} \frac{(x_j(t) - x_i(t))}{d_{ij}(t)} \cdot (d_{ij}(t) - l_{ij})$$

Je größer die Differenz aus tatsächlichem und gewünschtem Abstand zwischen zwei Knoten i, j , desto stärker wird die Position des Knotens i angepasst. Eine Schwierigkeit dabei ist die Festlegung des Schrittfaktors δ , welcher die Geschwindigkeit des Konvergierens bestimmt. Bei einem zu großem Wert kann es zu suboptimalem Verhalten wie z.B. zum Springen von Knoten kommen. Ein zu kleiner Wert verlangsamt den Konvergenzprozess. Ein weiteres Problem ist die Skalenabhängigkeit des Gradientenabstiegsverfahrens. In die Berechnung der neuen Koordinaten eines Knotens fließen die (gewünschten) Ziellängen der Kanten direkt ein (l_{ij}), während die eigentliche Ziel-Stress-Funktion $E(x)$ [(2), siehe Seite 6] davon unabhängig ist.

Gotsman und Koren verwenden daher ein anderes Verfahren, das *Majorisati-
ons*-Verfahren. Beide oben genannte Probleme werden auf diese Weise gelöst. Dieses Verfahren verläuft ebenfalls iterativ, wobei das Layout zu einem Zeitpunkt t als Ausgangslayout für eine Reihe von Berechnungen genommen wird. Nach Durchführung der Berechnungen haben sich die Positionen der Knoten einem Layout $t + 1$ angenähert, bei dem man davon ausgehen kann, dass es eine geringere Stress-Energie aufweist im Vergleich mit dem Layout zum Zeitpunkt t .

Der theoretische Hintergrund dafür wird durch die Cauchy-Schwartz-Ungleichung geliefert. Diese besagt, dass das Quadrat des Skalarprodukts zweier Vektoren kleiner oder gleich dem Produkt des Betrages der Vektoren ist:

$$(x \cdot y)^2 \leq |x|^2 + |y|^2$$

Da wir unser (2D-)Layout (zu einem Zeitpunkt t) gegeben haben als zwei Vektoren ($x, y \in \mathbb{R}^n$) (für die x- bzw. y-Koordinaten), können wir die Stress-Energie (1) mit Hilfe der Cauchy-Schwartz-Ungleichung und einem anderen Layout (a, b) ($a, b \in \mathbb{R}^n$) für den gegebenen Graph begrenzen:

$$\text{Stress}(x, y) \leq x^T Lx + y^T Ly + x^T L^{(a,b)}a + y^T L^{(a,b)}b + c \quad (4)$$

Es gilt dann $\text{Stress}(x, y) \leq \text{Stress}(a, b)$. Hier ist L die einfache Laplace-Matrix des Graphen, während $L^{(a,b)}$ eine durch das Layout (a, b) gewichtete Laplace-

Matrix ist:

$$L_{i,j} = \begin{cases} -1 & \text{falls } \langle i, j \rangle \in E \\ 0 & \text{falls } \langle i, j \rangle \notin E \\ -\sum_{j \neq i} L_{i,j} & \text{falls } i = j \end{cases} \quad i, j = 1, \dots, n$$

$$L_{i,j}^{(a,b)} = \begin{cases} \frac{-l_{ij}}{\sqrt{(a_i - a_j)^2 + (b_i - b_j)^2}} & \text{falls } \langle i, j \rangle \in E \\ 0 & \text{falls } \langle i, j \rangle \notin E \\ -\sum_{j \neq i} L_{i,j}^{(a,b)} & \text{falls } i = j \end{cases} \quad i, j = 1, \dots, n$$

In der einfachen Laplace-Matrix macht eine Zeile i eine Aussage über die „Erreichbarkeit“ (den Grad) eines Knotens i . Demgegenüber haben in der gewichtete Laplace-Matrix die Kantenlängen des Ausgangslayouts (a, b) einen Einfluss auf die Einträge. Je stärker die Differenz aus tatsächlichen Abstand zweier Knoten i und j zur gewünschten Länge l_{ij} ist, desto stärker weicht der Eintrag in $L_{ij}^{(a,b)}$ von -1 ab.

Die rechte Seite der Gleichung in (4) kann durch folgendes Gleichungssystem minimiert werden:

$$\begin{aligned} Lx &= L^{(a,b)}a \\ Ly &= L^{(a,b)}b \end{aligned}$$

Wenden wir dieses Gleichungssystem auf das Layout (x, y) zu einem Zeitpunkt t an, erhalten wir ein Layout zum Zeitpunkt $t + 1$, in welchem die Stress-Energie verkleinert wurde:

$$\begin{aligned} L \cdot x(t+1) &= L^{(x(t), y(t))} \cdot x(t) \\ L \cdot y(t+1) &= L^{(x(t), y(t))} \cdot y(t) \end{aligned}$$

Dieses Gleichungssystem lösen Gotsman und Koren dezentral mittels Jacobi-Iteration[2]. Dabei werden in jedem Iterationsschritt $t \rightarrow t + 1$ vom Layout $(x, y)(t)$ ausgehend wiederholt die Knotenpositionen verändert, bis sich die

Abstände zwischen den Knoten (d_{ij}) annähern an die gewünschten Kantenlängen (l_{ij}):

$$x_i \leftarrow \frac{1}{\deg_i} \sum_{j:(i,j) \in E} \frac{(x_j + l_{ij}(x_i(t) - x_j(t)))}{d_{ij}(t)}$$

$$y_i \leftarrow \frac{1}{\deg_i} \sum_{j:(i,j) \in E} \frac{(y_j + l_{ij}(y_i(t) - y_j(t)))}{d_{ij}(t)}$$

Die Position eines Knotens i wird also gesetzt auf den Durchschnitt der Position aller Nachbarknoten multipliziert mit dem Verhältnis aus aktueller Kantenlänge und gewünschter Kantenlänge. Nachdem die Stress-Energie im Layout $(x, y)(t)$ auf diese Weise reduziert wurde, wird das Verfahren auf das erhaltene Layout $(x, y)(t + 1)$ erneut angewendet.

4 Experimentelle Ergebnisse

In ihrem Artikel geben die beiden Autoren im letzten Abschnitt einen Ausblick auf ihre Versuche mit dem Verteilten Layout-Verfahren für Sensor-Netzwerke. Leider bleiben die Einzelheiten und genauen Rahmenbedingungen dieser Versuche unklar, auch wird die Art und Weise der Implementation ihres Modells in keinsten Weise erwähnt. Die von ihnen im Artikel angeführ-

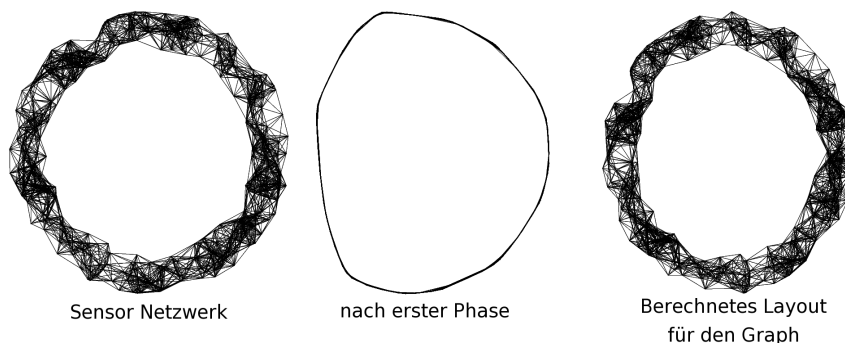


Abbildung 7: Layout-Verfahren angewendet auf ein Sensor-Netzwerk mit 350 Sensoren (links im Bild)

ten Resultate im Vergleich zu bisherigen Layout-Verfahren sind also leider nicht selbst nachvollziehbar. In Abbildung 7 ist ein Beispiel gegeben für ein Sensor-Netzwerk mit 350 Sensoren. Der linke Graph entspricht dem eigentlichen (realen) Sensor-Netzwerk. Das Ergebnis nach der ersten Phase ist in der Mitte zu sehen, und das abschließende Ergebnis nach der zweiten Phase sieht man rechts im Bild.

Literatur

- [1] Craig Gotsman and Yehuda Koren: *Distributed Graph Layout for Sensor Networks*. Proceedings of 12th International Symposium on Graph Drawing 2004 (GD'04).
- [2] G.H. Golub and C.F. Van Loan: *Matrix Computations*. Johns Hopkins University Press, 1996.
- [3] N.B. Priyantha, H. Balakrishnan, E. Demaine and S. Teller: *Anchor-Free Distributed Localization in Sensor Networks* Proc. 1st Inter. Conf. on Embedded Networked Sensor Systems (SenSys 2003), pp. 340–341.
- [4] Richard Bronson: *Matrix Methods: An Introduction*. Academic Press, 1970