

Erklärung zum modified Yao-Steps angewandt auf den Delaunay Graphen

Tim Budweg

Zusammenfassung—Diese Arbeit erläutert die Spannerbildung mithilfe der Delaunay-Triangulation mit einem modified Yao-Step aus [2]. Der *stretch-factor* liegt bei 3.54 in Bezug zum Euklidischen Graphen und beschränkt den Ausgangsgrad der Knoten auf 14.

Index Terms—Spanner, ad-hoc Netze, Delaunay Graph, Modified Yao-Step

I. EINLEITUNG

Es gibt viele Anwendungsbereiche für drahtlose Sensornetze. Zum Beispiel können Waldbrände oder Tsunamis durch ein Sensornetz-Frühwarnsystem schneller erkannt werden. Sowohl in diesem Beispiel als auch in anderen Anwendungsgebieten ist es gewünscht, dass alle gesendeten Nachrichten ankommen. Garantierte Nachrichtenauslieferung wird von mehreren Algorithmen bereits erreicht (z.B.: Face-Routing; siehe [1]). Jedoch müssen dafür bestimmte Voraussetzungen an das Sensornetz erfüllt sein. Die wichtigste Voraussetzung ist die Planarität des Netzes, das heißt, dass keine Kantenschnitte existieren dürfen. Diese Planarität zu erreichen ist ein wichtiges Ziel von *Topologiekontrollen*.

Ein weiterer Punkt ist, dass diese Netze willkürlich groß werden können. Deshalb führt ein *zentralisierter* Algorithmus zu einer sehr langen Laufzeit mit hohem Energieverbrauch, weil Nachrichten in Abhängigkeit von der globalen Netzgröße verschickt werden müssen.

Damit die Nachrichten nicht über beliebig lange Pfade geroutet werden müssen, wird eine Beschränkung der Pfadverlängerung gewünscht. Pfadverlängerungen entstehen, wenn (spezielle) kantenlöschende Algorithmen auf Graphen angewendet werden. Der hier behandelte Algorithmus ist streng lokal, was die Anzahl der gesendeten Nachrichten minimiert. Außerdem produziert dieser einen *t-Spanner* des euklidischen Graphen. Diese Thematik wird im Folgenden anhand des Artikels von [2] erläutert. Dort wurde ein streng lokaler Algorithmus aufbauend auf den *Delaunay Graphen* angewandt. Das Ergebnis ist ein Graph, welcher im Bezug zum euklidischen Graphen einen stretch-factor von $t = 3.54$ und zusätzlich eine Beschränkung des Ausgangsgrad eines Knoten von $k = 14$ hat.

A. Graphen

1) *Euklidischer Graph*: Ein euklidischer Graph ist ein spezieller Graph. Alle Knoten sind mit allen anderen Knoten verbunden (Clique) und die Kantengewichte entsprechen der euklidischen Distanz beider Eckpunkte. Ein Beispiel finden Sie in Abbildung 1.

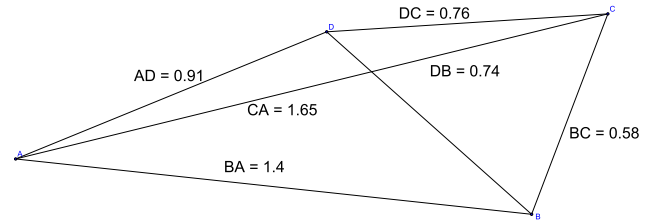


Abbildung 1. Ein euklidischer Graph

2) *Unit Disk Graph*: Der Unit Disk Graph ist ein euklidischer Graph ohne alle Kanten, die länger als ein konstantes $c \in \mathbb{R}$ sind. In Abbildung 2 sehen Sie den euklidischen Graph aus Abbildung 1 als Unit Disk Graph mit $c = 1$.

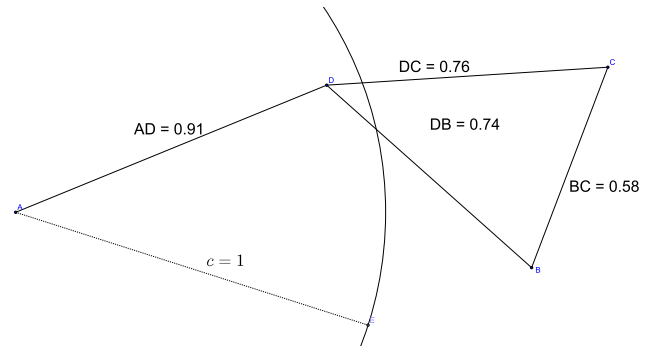


Abbildung 2. Der Unit-Disk-Graph zu Abbildung 1 mit $c = 1$

B. Spanner

Gegeben ist ein Graph G , welcher ein Subgraph vom euklidischen Graphen E ist. G enthält alle Knoten von E , aber nicht alle Kanten. Die Umwege, die durch das Löschen von Kanten entstehen, dürfen nur um einen konstanten Faktor ansteigen.

1) *Euklidischer Spanner*: G ist genau dann ein euklidischer Spanner von H , wenn die kürzesten Pfade zwischen allen Knoten maximal um einen konstanten Faktor t vergrößert werden. Die Notation $c_G(A, B)$ bedeutet: der kürzeste Pfad zwischen A und B im Graphen G . Formal bedeutet das:

$$c_H(A, B) \leq p \cdot c_G(A, B) \quad (1)$$

(Formel entnommen aus [2])

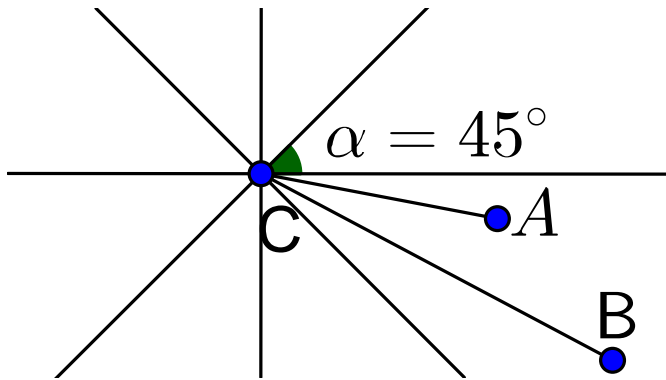


Abbildung 3. Punkt A mit acht 45° großen Kegeln. Kürzeste Kante AC wird ausgewählt

C. Yao Step

Gegeben ist ein Graph G . Für alle Knoten $A \in G$ wird folgender Algorithmus ausgeführt:

- Erzeuge k gleich große Kegel um A . $k \in \mathbb{N} > 6$
- Bestimme die kürzeste Kante in jedem Kegel ausgehend von C .
- Lösche alle Kanten, die nicht von beiden Endpunkten ausgewählt wurden.

D. Delaunay Triangulation

Die Delaunay Triangulation erzeugt aus einem beliebigen zusammenhängenden Graphen einen geometrischen (= euklidischen) Spanner mit dem Streckungsfaktor $c_{del} \approx 2.42$ und einem beliebig hohen Ausgangsgrad eines Knotens. Dazu werden alle Dreiecke betrachtet. Wenn der Kreis durch alle Eckpunkte des Dreiecks keine weiteren Punkte des Graphen enthält, sind diese drei Kanten auch im Delaunay Graphen. Um Fallunterscheidungen zu vermeiden wird angenommen, dass keine vier Punkte auf einem Kreis liegen.

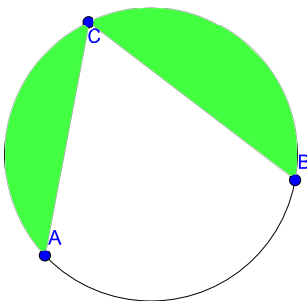


Abbildung 4. Die grünen Bereiche dürfen keine Punkte enthalten. Wenn doch, werden diese Punkte ausgewählt und die „alten“ Knoten liegen innerhalb des Kreises

E. Synchronität

Bei einem synchronen Algorithmus ist die Arbeitsweise der Knoten in Runden und diese in Phasen eingeteilt. Eine Runde sieht beispielsweise wie folgt aus:

- Zuerst erhalten alle Knoten ihre Nachrichten, falls es welche gibt.

- Danach verarbeiten sie die Nachrichten und stellen ggf. weitere Berechnungen an.
- Am Ende der Runde werden Nachrichten an andere Knoten verschickt.

Hier ist zu beachten, dass alle gesendeten Nachrichten zu Beginn der Folgerunde angekommen sind.

F. Lokale Algorithmen

Ein Algorithmus ist genau dann lokal, wenn jeder Knoten ausschließlich mit seiner k -Hop Nachbarschaft kommuniziert. ($k \in \mathbb{N}$) Formal reicht dies aus, um einen Algorithmus lokal zu nennen. Das Problem, dass der Algorithmus eine lange Laufzeit hat (siehe Einleitung), ist dadurch jedoch nicht gelöst. Anhand des *Maximal Independent Set* (Fortan: MIS) Algorithmus lässt sich ein Beispiel konstruieren, welches dieses Problem veranschaulicht. Betrachten Sie Abbildung 5.

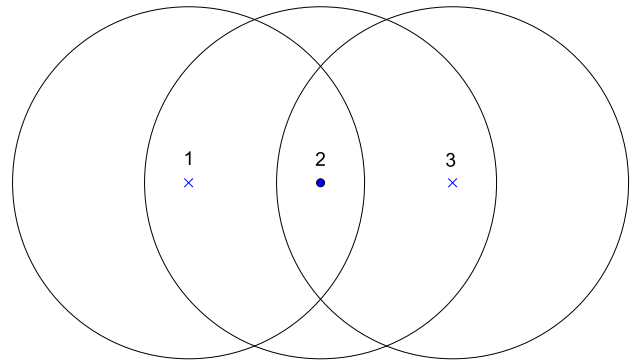


Abbildung 5. \times sind Knoten im Maximal Independent Set, \bullet sind keine Knoten des MIS, Die Kreise um die Knoten sind die jeweiligen Sendebereiche.

- **Runde 1:** Knoten 1 wird MIS-Knoten, da er von allen seinen Nachbarn, welche noch keine Rolle haben, derjenige mit der kleinsten ID ist. Knoten 2 und 3 warten, da es Knoten gibt, welche noch keine Rolle und eine kleinere ID haben.
- **Runde 2:** Knoten 2 entscheidet, dass er kein MIS-Knoten ist, da er in seiner Nachbarschaft Knoten 1 kennt, der schon MIS-Knoten ist. Knoten 3 wartet.
- **Runde 3:** Knoten 3 wird MIS-Knoten, da Knoten 2 kein MIS-Knoten ist und somit 3 die kleinste ID hat.

Diese Schritte lassen sich durch den gesamten Graphen sukzessiv durchführen. Der Algorithmus hat demnach eine Laufzeit in der Größenordnung von

$$\theta(\text{dia}(G))$$

wobei $\text{dia}(G)$ für den Durchmesser (längster kürzester Pfad) des Graphen steht.

Die Definition des lokalen Algorithmus trifft hier zu, weil jeder Knoten nur mit seinen unmittelbaren Nachbarn, der 1-Hop-Nachbarschaft, kommuniziert.

G. Strenge Lokalität

Wie im Abschnitt I-F beschrieben, sind lokale Algorithmen noch nicht ausreichend, um die benötigte Nachrichtenanzahl zu minimieren. Strenge Lokalität ist gegeben, wenn ein *synchroner* Algorithmus unabhängig von der Netzgröße in konstanter Zeit terminiert (siehe [3]).

II. GEOMETRISCHE SPANNER

Es gilt folgendes Lemma zu beweisen:

Theorem II.1. Für jedes $k \geq 14$ existiert ein Subgraph G' vom Delaunay-Graph G , sodass G' einen maximalen Ausgangsgrad von k und einen stretch-factor von maximal $1 + 2\pi(k \cos \frac{\pi}{k})^{-1}$ hat.

Gegeben sind zwei Kanten CA und CB im Delaunay-Graph G vom Euklidischen Graph E . Ohne Beschränkung der Allgemeinheit sei CA die kürzeste Kante. Die Autoren von [2] beweisen dieses Theorem mithilfe des *outward* und des *inward* Path. Der outward Path kann gebildet werden, wenn kein Knoten von G innerhalb von $\triangle ABC$ liegt. Der inward Path verbindet die Knoten A und B, wenn Knoten innerhalb von $\triangle ABC$ liegen.

A. Der outward Path

Die Autoren von [2] benutzen eine rekursive Definition um den Pfad von A nach B zu definieren. Es gilt zu beweisen, dass dieser Pfad in jedem Fall existiert, weil anderenfalls der Graph nicht zwingend verbunden ist. Dieser Fall darf nicht eintreten.

- **Basisfall:** Die Rekursion endet, wenn $AB \in G$. (vgl. Abbildung 3: AB müssen verbunden sein, weil B der einzige Knoten in einem Kegel von A ist.)
- **Rekursionsfall:** $AB \notin G$

Aufgrund der Delaunay-Eigenschaft, dass eine Kante nur dann nicht in G enthalten ist, wenn im Kreis $\odot ABC$ ein weiterer Punkt liegt, muss deshalb ein weiterer Punkt in $\odot ABC$ liegen. Um den Pfad von A nach B zu erhalten, wird ein Pfad von A zum intermediate Point T (siehe Abbildung 6) und von diesem zu B erstellt und konkateniert. Dieser Rekursionsfall kann öfter eintreten, da es bezüglich A und T oder B und T einen weiteren intermediate Point T' geben kann und dazwischen rekursiv fortführend bis der Basisfall eintritt. Der Rekursionsfall bewahrt immer folgende Eigenschaften:

- (O_1) und (O_2) sind vollständig innerhalb von (O) (vgl. Abbildung 7)
- α und β sind beide kleiner als der Winkel $\angle AOB$ (vgl. Abbildung 8)
- Alle drei Winkel sind kleiner als $\frac{4\pi}{k}$

B. Der inward Path

Die Menge S besteht aus allen Punkten innerhalb von $\triangle ABC$, Punkt A und Punkt B. In Abbildung 9 ist ein Beispiel eines Inward Path.

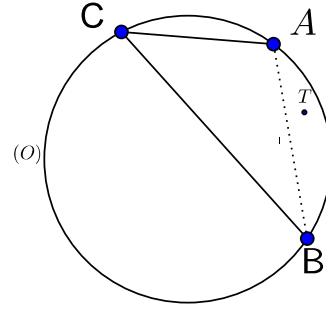


Abbildung 6. T wird intermediate Point genannt.

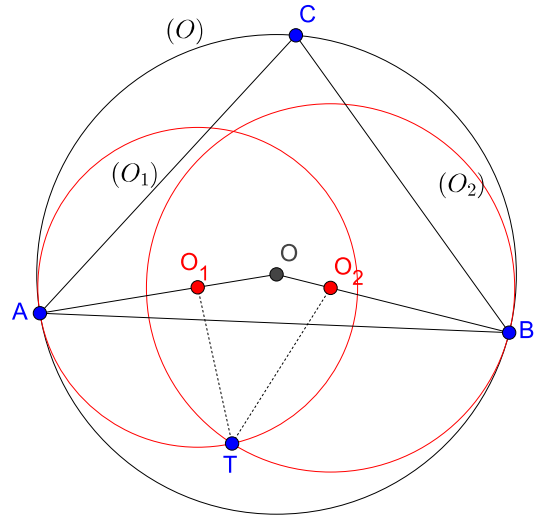


Abbildung 7. (O_1) und (O_2) sind vollständig innerhalb von (O) , (O_1) ist der Kreis durch A und T mit Mittelpunkt auf dem Segment AO, (O_2) verläuft durch B und T und der Mittelpunkt liegt auf dem Segment BO

LITERATUR

- [1] H. Frey and I. Stojmenovic, "On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '06. New York, NY, USA: ACM, 2006, pp. 390–401. [Online]. Available: <http://doi.acm.org/10.1145/1161089.1161133>
- [2] I. A. Kanj and L. Perkovic, "On geometric spanners of euclidean and unit disk graphs," *CoRR*, vol. abs/0802.2864, 2008. [Online]. Available: <http://arxiv.org/abs/0802.2864>
- [3] J. Suomela, "Survey of local algorithms," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 24:1–24:40, Mar. 2013. [Online]. Available: <http://doi.acm.org/10.1145/2431211.2431223>

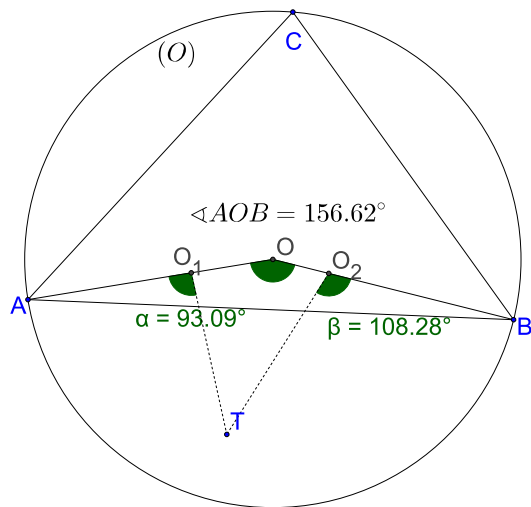


Abbildung 8. α und β sind beide kleiner als AOB , gleiche Konstruktion wie in Abbildung 7, A und B liegen im selben Kegel ausgehend von C .

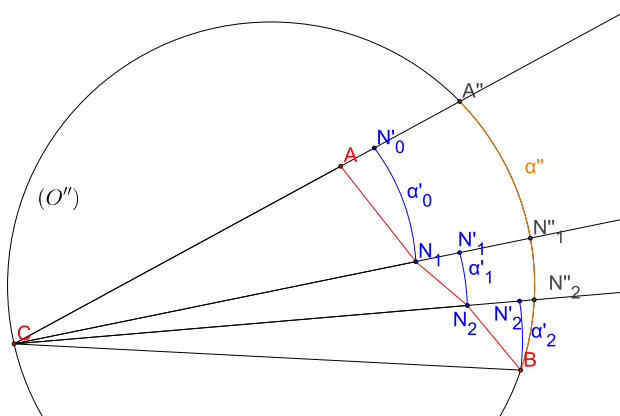


Abbildung 9. AN_1N_2B ist der inward Path von A nach B