Diplomarbeit

Virtual Coordinates Simulation

Guido Frerker

Betreuung: Regina O'Dell, Mirjam Wattenhofer

Prof. Dr. Roger Wattenhofer
Distributed Computing Group
Departement Informationstechnologie und Elektrotechnik
ETH Zürich

Sommersemester 2004

Zusammenfassung

Die Diplomarbeit behandelt einen Algorithmus zur Berechnung von virtuellen Koordinaten, welcher auf dem bestehenden Algorithmus aus dem Paper "Geographic Routing without Location Information" von C. Papadimitriou [1] aufbaut. Es wird dabei versucht, dass Kantenverhältnis im Graphen, der durch die virtuellen Koordinaten definiert ist,

$$\text{Rho} = \frac{\text{längste Distanz mit Kante}}{\text{kürzeste Distanz ohne Kante}}$$

stetig zu verbessern. Am Ende wird durch Simulationen gezeigt, dass die Einführung von Gewichten das Beste Resultat liefert.

Inhaltsverzeichnis

1 Einleitung

1.1 Virtuelle Koordinaten

In Sensornetzwerken haben die einzelnen Knoten nur beschränkte Energiereserven und es ist nicht möglich Koordinatenbestimmende Geräte wie GPS-Empfänger einzubauen. Auf der anderen Seite brauchen aber Applikationen die Sensorkoordinaten um Ereignisse korrekt einordnen zu können. So haben Forscher nach Möglichkeiten gesucht, um Koordinaten nur anhand weniger Ankerknoten zu bestimmen. Für gewisse Applikationen sind Ankerknoten sogar unnötig, und man kann Virtuellen Koordinaten von Sensoren berechnen, welche die Topologie des Netzwerkes genau reflektieren.

1.2 Aufgabenstellung

Zu Begin dieser Diplomarbeit stand das Ziel, Algorithmen für die Berechnung von so genannten Virtuellen Koordinaten (Virtual Coordinates, VC) besser zu verstehen und miteinander zu vergleichen. Dazu sollte eine Auswahl an bestehenden Algorithmen implementiert und eine umfassende Simulation durchgeführt werden.

Nach der Implementation des ersten Algorithmus anhand des Papers “Geographic Routing without Location Information“ von C. Papadimitriou [1], zeigte sich aber, dass das Ergebnis mit den Resultaten im Paper nicht überein stimmte. Als Folge davon wurde beschlossen, den Algorithmus mit eigenen Ideen zu verbessern und so von der ursprünglichen Idee einer genauen Implementation des Algorithmus abzuweichen.

1.3 Überblick

Das erste Kapitel gibt einen kurzen Überblick über die vorliegende Diplomarbeit. Im zweiten Kapitel wird erklärt, wie der Vorliegende Algorithmus aus dem Paper “Geographic Routing without Location Information“ implementiert wurde und die durchgeführten Simulationen werden präsentiert. Kapitel 3 führt dann mehrere Ideen zur Verbesserung des Algorithmus aus Kapitel 2 ein und zeigt das Resultat der endgültigen Implementation. In Kapitel 4 wird ein Fazit für diese Arbeit gezogen und es gibt ein persönlicher Rückblick.

2 Der Algorithmus nach Vorlage

Der Algorithmus aus dem Paper „Geographic Routing without Location Information“ wird in drei Phasen aufgeteilt. Am Anfang stellen wir Annahmen auf, die dann nach und nach wieder fallengelassen werden, sodass am Ende ein Algorithmus entsteht der Virtuelle Koordinaten berechnen kann. Die Annahmen der drei Phasen sind wie folgt bestimmt:

- Randknoten wissen, dass sie am Rand sind und kennen ihre Koordinaten.
- Randknoten wissen, dass das sie am Rand sind aber kennen ihre Koordinaten nicht.
- Knoten wissen weder dass sie am Rand sind, noch kennen sie ihre Koordinaten.

2.1 Randknoten kennen ihre Koordinaten

Randknoten werden in einem Rechteck angeordnet und kennen ihre exakten geographischen Koordinaten. Alle Nicht-Randknoten können jetzt ihre Koordinaten mit einem iterativen Ralaxations-Algorithmus bestimmen.

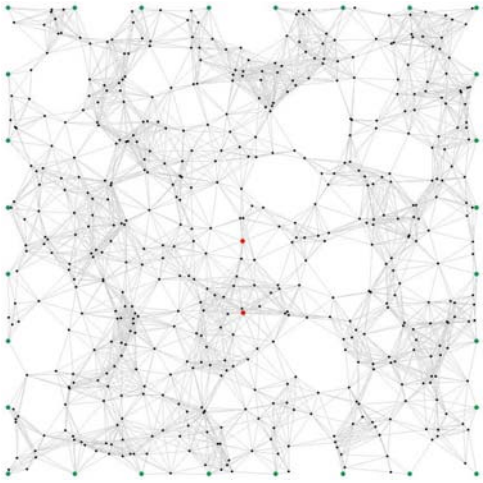

Abbildung 1: Beispiel eines Netzwerkes mit 512 Knoten und 28 Randknoten auf einem Feld von 10 auf 10 Einheiten. Der Sendebereich der Knoten ist eine Einheit. Dargestellt sind die realen Koordinaten.

2.1.1 Relaxations-Algorithmus

Der Relaxations-Algorithmus ist eine iterative Lösung um für alle Nicht-Randknoten die Koordinaten zu bestimmen. Dabei starten alle Knoten mit den gleichen initialen Koordinaten. Diese befinden sich in der Mitte des Feldes definiert durch die Randknoten. Ausgehend von den Randknoten sendet nun jeder Knoten seine Koordinaten an seine Nachbarn (Knoten die im Sendebereich liegen). Jeder Nicht-Randknoten berechnet dann in jeder Iteration seine neuen Koordinaten wie folgt:

$$x_i = \frac{\sum_{k \in neighbor\_set(i)} x_k}{size\_of(neighbor\_set(i))}$$

$$y_i = \frac{\sum_{k \in neighbor\_set(i)} y_k}{size\_of(neighbor\_set(i))}$$

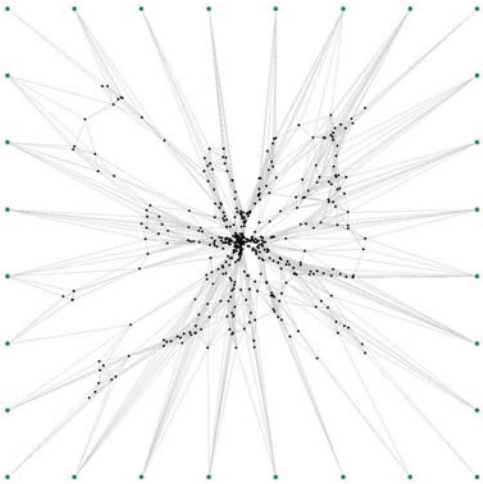

Abbildung 2

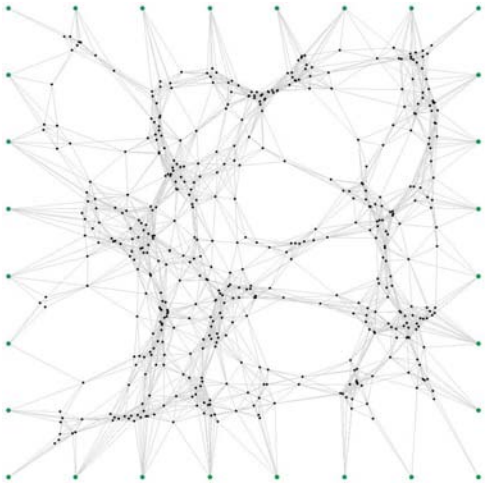

Abbildung 3

Die virtuellen Koordinaten von den Nicht-Randknoten aus Abbildung 1 nach 10 (Abbildung 2) bzw. 100 (Abbildung 3) Iterationen.

2.2 Randknoten sind bekannt

Die Bestimmung der Koordinaten der Randknoten erfolgt in drei Schritten.

Schritt 1: Alle Randknoten senden eine Hallo Nachricht ins ganze Netzwerk, sodass alle Randknoten ihre Distanz (in Hops) zu allen anderen Randknoten erfahren.

Schritt 2: Jeder Randknoten sendet nun diese Distanzen wiederum ins ganze Netzwerk. Am Ende kennen alle Randknoten alle Distanzen zwischen jedem paar von Randknoten.

Schritt 3: Jeder Randknoten benutzt nun einen Triangulations-Algorithmus [siehe Abschnitt 2.2.1], um die Koordinaten von allen Randknoten zu berechnen.

Weil die Distanzen über das ganze Netzwerk versendet werden, kennen auch die Nicht-Randknoten die Distanzen und können denselben Triangulations-Algorithmus ausführen um damit vernünftige Anfangskoordinaten zu berechnen, bevor sie mit dem Relaxations-Algorithmus beginnen.

Damit sichergestellt werden kann, dass alle Knoten für jeden die gleichen Koordinaten berechnen werden zwei Startknoten, so genannte „Bootstrapping beacons“, eingeführt. Diese Startknoten fluten das Netzwerk zuerst mit einer Hallo Nachricht. Die Randknoten beziehen dann diese Startknoten ebenfalls in ihre Triangulations-Berechnungen mit ein. Mit allen Koordinaten wird anschliessend das Gravitationszentrum (Center of Gravity, CG) berechnet. Das CG zusammen mit den zwei Startknoten bildet das neue Koordinatensystem. Dabei ist das CG der Ursprung, die Achse CG – erster Startknoten die x-Achse und der zweite Startknoten liegt auf der positiven y-Seite. Alle Koordinaten werden dann anhand dieser neuen Achsen normalisiert. Ein Randknoten muss also immer seine Distanzen zu den beiden Startknoten kennen.

Da die Startknoten irgendwo im Netzwerk sein können, müssen sie ihre Koordinaten ebenfalls mit dem Relaxations-Algorithmus anpassen. Nur die Randpunkte behalten ihre durch den Triangulations-Algorithmus berechneten Koordinaten.

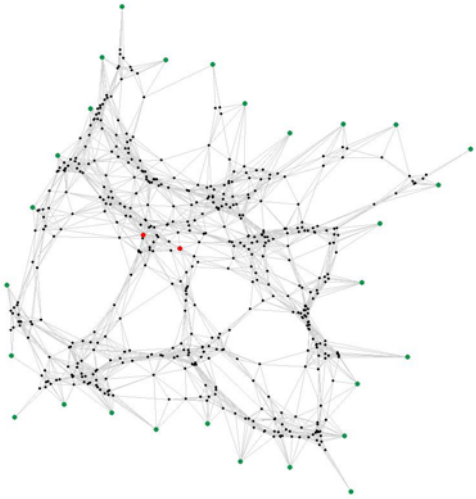

Abbildung 4

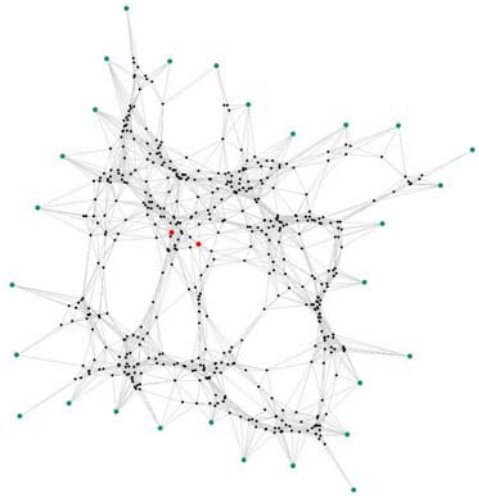

Abbildung 5

Die virtuellen Koordinaten von den Nicht-Randknoten aus Abbildung 1 nach 10 (Abbildung 4) bzw. 100 (Abbildung 5) Iterationen, wenn die Randknoten bekannt sind, sie aber ihre Koordinaten nicht kennen.

2.2.1 Triangulations-Algorithmus

Die Koordinaten werden so gewählt, sodass

$$E = \sum_{i,j \in perimeter\_set} (measured\_dist(i,j) - dist(i,j))^2$$

minimiert wird. Wobei *measured_dist(i,j)* die gemessene Hopdistanz zwischen Knoten i und j ist und *dist(i,j)* die euklidische Distanz zwischen den virtuellen Koordinaten der Knoten i und j ist.

Um das lokale Minimum von dieser Summe zu finden, müssen wir für jeden Knoten die partielle Ableitung lösen. Dies führt zu *2n* nichtlinearer abhängiger Gleichungen. Weil dieses System von Gleichungen ziemlich schwierig zu lösen ist, wird jeweils nur ein Knoten aufs Mal beachtet. Man sucht nun den Knoten mit der Höchsten Energie und verschiebt ihn an einen Ort mit lokalem Minimum. p_m ist nun dieser Knoten mit der Energiefunktion

$$\Delta_m = \sqrt{\left(\frac{\partial E}{\partial x_m}\right)^2 + \left(\frac{\partial E}{\partial y_m}\right)^2}$$

p_m wird nun schrittweise bewegt um Δ_m zu minimieren. Das Energieminimum liegt am Ort wo

$$\frac{\partial E}{\partial x_m} = \frac{\partial E}{\partial y_m} = 0$$

erfüllt ist. Dies wird erreicht durch iteratives Lösen der folgenden zwei linearen Gleichungen für δx und δy, und sie zu x_m bzw. y_m hinzufügen. t ist die laufende Iteration. Diese Schritte werden solange wiederholt, bis die neue Energiefunktion nicht mehr sinkt.

$$\frac{\partial^2 E}{\partial x_m^2}(x'_m, y'_m)\delta x + \frac{\partial^2 E}{\partial x_m \partial y_m}(x'_m, y'_m)\delta y = \frac{-\partial E}{\partial x_m}(x'_m, y'_m)$$
$$\frac{\partial^2 E}{\partial x_m \partial y_m}(x'_m, y'_m)\delta x + \frac{\partial^2 E}{\partial y_m^2}(x'_m, y'_m)\delta y = \frac{-\partial E}{\partial y_m}(x'_m, y'_m)$$

Der Algorithmus sieht in Pseudocode etwa so aus:

```
initialisiere Parameter
while (max Δ_i > ε)
    p_m ist der Knoten, der Δ_m = max Δ_i erfüllt;
    while (Δ_m > ε)
          berechne δx und δy;
          x_m ← x_m + δx
          y_m ← y_m + δy
    end while
end while
```

Die Nicht-Randknoten erhalten ihre Anfangs-Koordinaten, indem sie zuerst ebenfalls die Koordinaten der Randknoten berechnen. Anschliessend werden die Koordinaten der Randknoten fixiert und nur die Koordinaten der Nicht-Randknoten werden ins Energieminimum verschoben.

2.3 Weder Koordinaten noch Randknoten bekannt

In der endgültigen Version, wissen die Knoten nicht, ob sie am Rand sind oder nicht. Wir stellen dem Algorithmus von vorhin also eine weitere Phase voran, in der wir die Randknoten identifizieren. Dazu benötigen wir einen der Startknoten (bootstrap beacon node) von vorhin, der das ganze Netzwerk mit einer Hallo Nachricht flutet. Dadurch erfahren alle anderen Knoten ihre Hop-Distanz zu diesem Startknoten und können nun mit dem folgenden Kriterium entscheiden, ob sie am Rand sind oder nicht.

Wenn ein Knoten unter allen Nachbarn in Umkreis von zwei Hops am weitesten vom Startknoten entfernt ist, dann entscheidet er, dass er ein Randknoten ist.

Wenn zwei Knoten die gleiche Entfernung haben entscheidet z.B. die grössere ID welcher Knoten genommen wird.

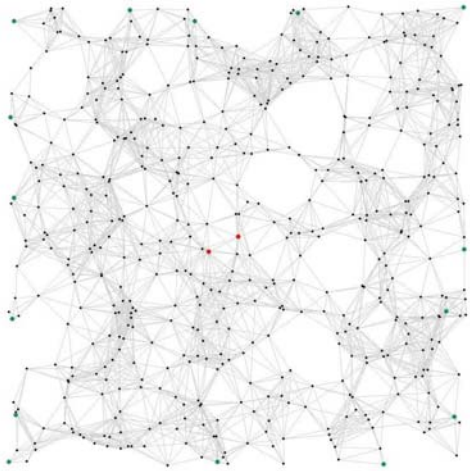

Abbildung 6: Ein Netzwerk mit 512 Knoten und 15 gefundenen Randknoten auf einem Feld von 10 auf 10 Einheiten. Der Sendebereich der Knoten ist eine Einheit. Dargestellt sind die realen Koordinaten.

Abbildung 7

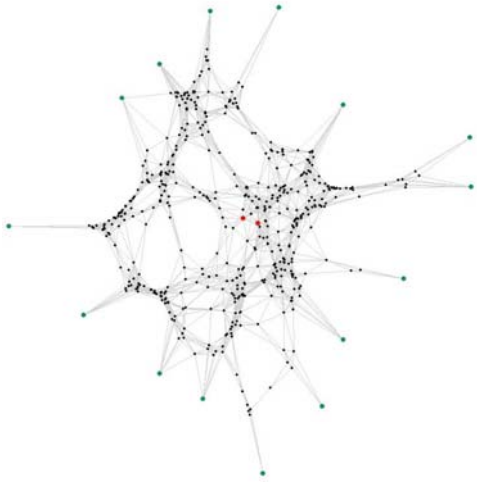

Abbildung 8

Die virtuellen Koordinaten von den Nicht-Randknoten aus Abbildung 6 nach 10 (Abbildung 7) bzw. 100 (Abbildung 8) Iterationen, wenn weder die Randknoten noch ihre Koordinaten bekannt sind.

2.4 Simulation

Für die Simulation wurde der vollständige Algorithmus für die Berechnung der virtuellen Koordinaten verwendet. Also der Algorithmus, wo weder die Koordinaten noch die Randpunkte bekannt sind.

2.4.1 Simulationsparameter

Die hier aufgelisteten Parameter zeigen für die Simulation verwendete Werte.

Anzahl Knoten: 200, 400 und 800. Die Anzahl Knoten die im Netwerk verteilt sind. Weil die Berechnung der Koordinaten in der Triangulations-Phase ziemlich lange dauert, waren mehr Knoten nicht möglich, um vernünftige Resultate zu erhalten.

Senderadius: 1 Einheit. Der Sendebereich eines Knoten liegt immer bei einer Einheit.

Knotendichte: Sie wird berechnet durch die Anzahl Knoten mal den Senderadius geteilt durch die Fläche des ganzen Netzwerkes.

$$\frac{\text{Anzahl Knoten} * \text{Senderadius}}{\text{Netzwerkfläche}}$$

Relaxations-Epsilon: 10^{-3}, 10^{-4} und 10^{-5}. Bei jeder Iteration des Relaxations-Algorithmus wird gemessen, welcher Knoten sich am Weitesten bewegt hat. Liegt dieser Wert unter dem Epsilon, beendet der Algorithmus die Berechnungen.

Triangulations-Epsilon: 0.1. Der Triangulations-Algorithmus wird solange wiederholt, bis alle Knoten weniger als Epsilon Energie haben

2.4.2 Messkriterien

Um gute Aussagen über einen Algorithmus zur Bestimmung virtueller Koordinaten zu machen, ohne dass man gleich einen Algorithmus für Geographisches Routing benötigt, braucht man Kriterien, die ein einfaches und messbares Ergebnis liefern.

Randknoten: Die Anzahl gefunden Randknoten in einem Graphen.

Rho: Das Kantenkriterium eines Graphen wird bestimmt durch den längste Abstand zwischen zwei Knoten, die ein Kante haben und dem kleinsten Abstand zwischen zwei Knoten, die keine Kante haben.

$$\text{Rho} = \frac{\text{längste Distanz mit Kante}}{\text{kürzeste Distanz ohne Kante}}$$

Gesendete Nachrichten: Die Anzahl für die Berechnung der virtuellen Koordinaten gesendeten Nachrichten.

Empfangende Nachrichten: Die Anzahl für die Berechnung der virtuellen Koordinaten empfangenen Nachrichten.

Relaxations-Iterationen: Die Anzahl im Relaxations-Algorithmus duchgeführten Iterationen.

Triangulations-Iterationen: Die Anzahl Iterationen, bis die definitiven Koordinaten für die Randknoten und die Startkoordinaten für die Nicht-Randknoten berechnet wurden.

2.4.3 Auswertung

Zuerst konzentrieren wir uns auf die Anzahl Randknoten, die vom Algorithmus gefunden werden. Sie werden ganz am Anfang der Algorithmus definiert und sind deshalb unabhängig von der Triangulations- und Relaxations-Parameter.

Anzahl Knoten	Dichte	Randknoten
800	25.13	40.63
800	14.87	38.73
800	11.17	49.21
400	25.65	29.33
400	15.51	29.29
400	10.39	34.04
200	25.13	20.94
200	17.45	18.71
200	9.82	23.02

Wie man sieht hängt die Anzahl der gefundenen Randknoten einerseits von der Anzahl Knoten ab, andererseits von der Dichte des Netzwerkes. Es fällt auf das ab einer bestimmten Dichte, die Anzahl Randknoten um einen konstanten Wert zu pendeln beginnen. Dass es bei kleiner Dichte mehr Randknoten gibt liegt daran, dass ein Graph dann am Rand viel zackiger ist als bei einem Graphen mit hoher Dichte.

Betrachten wir nun die anderen Parameter etwas genauer, allen voran interessiert uns das Rho (Kantenverhältnis).

gesendete Nachrichten	empfangene Nachrichten	Relaxations Epsilon	Relaxations Iterationen	Triangulations Iterationen	Rho Kantenverhältnis
96'971	1'472'894	0.00001	459	336	302.06
83'137	1'257'921	0.0001	393	285	1148.07
34'660	525'316	0.001	148	333	1861.64

Tabelle 1: Daten für ein Netzwerk mit 200 Knoten und einer Dichte von 15 und 20 Randknoten im Schnitt.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Epsilon	Relaxations Iterationen	Triangulations Iterationen	Rho Kantenverhältnis
314'393	4'373'092	0.00001	750	524	590.94
224'030	3'168'308	0.0001	526	478	765.22
125'143	1'736'831	0.001	278	493	438.19

Tabelle 2: Daten für ein Netzwerk mit 400 Knoten und einer Dichte von 15 und 30 Randknoten im Schnitt.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Epsilon	Relaxations Iterationen	Triangulations Iterationen	Rho Kantenverhältnis
931'517	12'969'782	0.00001	1118	675	767.34
789'898	10'934'608	0.0001	942	660	1285.47
404'618	5'609'677	0.001	464	611	1346.07

Tabelle 3: Daten für ein Netzwerk mit 800 Knoten und einer Dichte von 15 und 41 Randknoten im Schnitt.

Die Anzahl gesendeter Nachrichten bzw. die Anzahl empfangener Nachrichten ergibt sich logischerweise aus der Anzahl Knoten und der Grösse des Relaxations-Epsilon. Die Anzahl Iterationen für die Triangulations-Berechnungen ist vor allem auf die Anzahl der Randknoten zurückzuführen.

Das Kantenverhältnis Rho sollte nun eine Qualitative Aussage über den Algorithmus geben. Wie sich aber herausstellte hüpft der Wert unberechenbar zwischen willkürlichen Werten hin und her. Deshalb werden im Nächsten Kapitel Verbesserungen für den Algorithmus gesucht und eingeführt.

3 Erweiterungen und Verbesserungen

In diesem Kapitel wird der Algorithmus aus Kapitel zwei abgeändert und verbessert, um ein besseres Resultat für das Messkriterium Rho zu erhalten.

3.1 Eckpunkte

Anstatt die Randpunkte mit einem rechenaufwändigen Triangulations-Verfahren zu berechnen, werden erst einmal nur die Eckpunkte eines virtuellen Rechtecks berechnet. Dabei wird von einem zufälligen Startpunkt ausgegangen. Der erste Eckpunkt (Ecke a) ist der Knoten mit der grössten Distanz (Anzahl Hops) vom Startpunkt. Gibt es mehrere Punkte mit der grössten Distanz, so definiert sich der Knoten mit der höchsten Identifikations-Nummer als Ecke a. Die zweite Ecke (Ecke b) ist der Knoten, der von der Ecke a die grösste Entfernung hat, also diagonal gegenüber der Ecke a liegt. Für die Ecke c wird jener Knoten bestimmt, der den maximalen Wert des Produktes *Distanz(Ecke_a) * Distanz(Ecke_b)* hat. Die letzte Ecke (Ecke d) wird schlussendlich durch den grössten Wert des Produktes *Distanz(Ecke_a) * Distanz(Ecke_b) * Distanz(Ecke_c)* definiert. Die Koordinaten werden anhand der Hop-Distanzen, die die Eckknoten untereinander haben gesetzt. Knoten a liegt im Ursprung, Knoten c auf der y-Achse, Knoten d auf der x-Achse und Knoten b entsprechend der Distanzen zu c und d.

Alle anderen Knotenkoordinaten werden nun mit Hilfe zweier Geraden berechnet. Diese zwei Geraden eines Knoten werden dabei definiert als:
Alle Punkte, wo die Gleichung

$$\frac{\text{euklid. Distanz zu a}}{\text{euklid. Distanz zu b}} = \frac{\text{HopDistanz zu a}}{\text{HopDistanz zu b}}$$

erfüllt ist. Für die zweite Gerade wird c für a und d für b eingesetzt. Die initialen Koordinaten eines Knotens befinden sich im Schnittpunkt dieser beider Geraden.

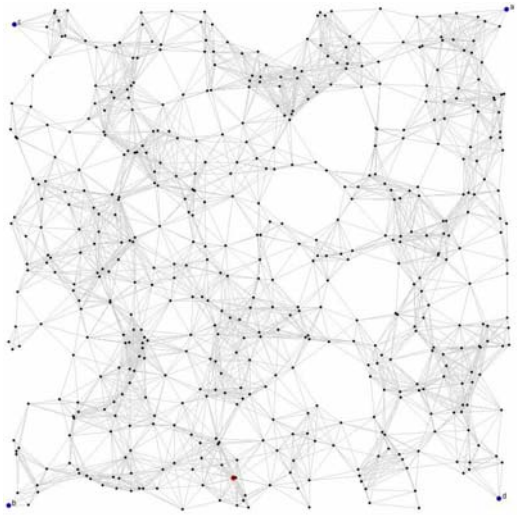

Abbildung 9: Ein Netzwerk mit 512 Knoten und den 4 gefundenen Eckknoten auf einem Feld von 10 auf 10 Einheiten. Der Sendebereich der Knoten beträgt eine Einheit.

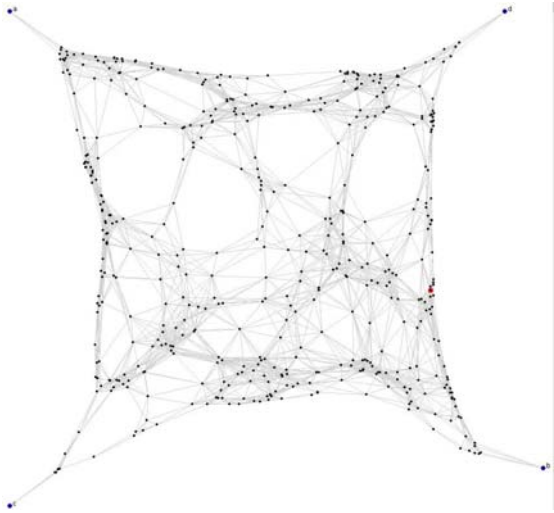

Abbildung 10: Das Netwerk aus Abbildung 9 nach einer Relaxations-Iteration.

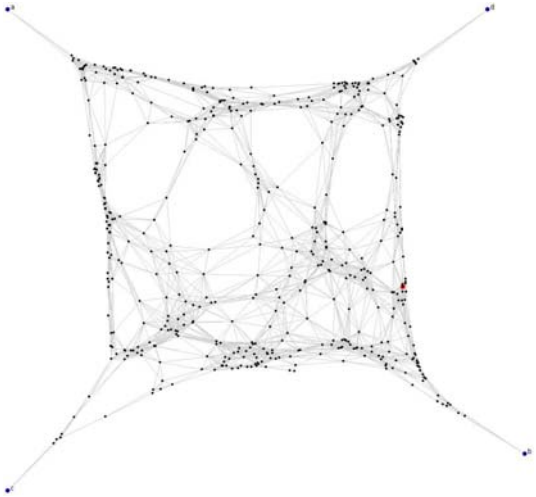

Abbildung 11: Das Netwerk aus Abbildung 9 nach 10 Relaxations-Iterationen.

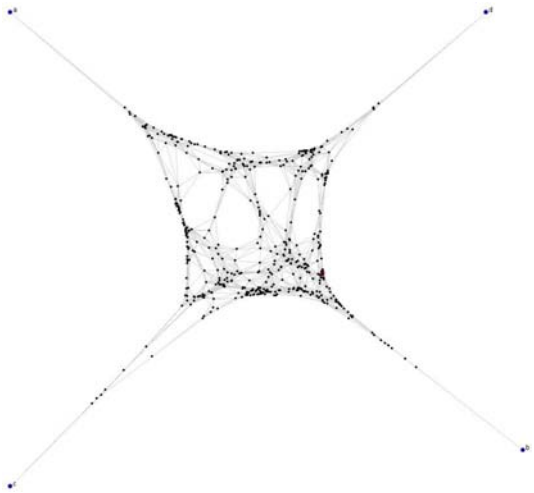

Abbildung 12: Das Netwerk aus Abbildung 9 nach 100 Relaxations-Iteration.

Man sieht in den Abbildungen 10 bis 12 deutlich, dass je mehr Iterationen durchgeführt werden, desto mehr zieht sich das Netzwerk zusammen und die Kanten zwischen einem Eckknoten und einem normalen Knoten werden in die Länge gezogen. Das wirkt sich auch auf das Kriterium Rho aus wie in den folgenden Simulationsergebnissen sichtbar wird.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
3242	48195	1	18.17
4987	73910	10	26.11
22808	336992	100	119.35

Tabelle 4: Daten für ein Netzwerk mit 200 Knoten und einer Dichte von 15. Die Daten sind die Durchschnittswerte von 50 Messungen.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
8629	121505	1	28.76
12141	170800	10	39.92
48024	673730	100	204.33

Tabelle 5: Daten für ein Netzwerk mit 400 Knoten und einer Dichte von 15.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
22441	312190	1	43.15
29726	413198	10	49.13
101808	1411416	100	165.36

Tabelle 6: Daten für ein Netzwerk mit 800 Knoten und einer Dichte von 15.

Um diesem Verhalten des Algorithmuses entgegenzuwirken werden zwei Lösungsvorschläge in den Abschnitten 3.2 und 3.3 eingeführt.

3.2 Gewichte

Der erste Lösungsvorschlag besteht darin, für die Eckpunkte grössere Gewichte einzuführen, d.h. sie werden mehrmals als Nachbarknoten gewertet. Wenn also ein Knoten ein Eckpunkt als Nachbar hat, summiert er alle normalen Nachbarknoten einmal und den Eckpunkt x-mal. Dabei ist x proportional zu den Anzahl Nachbarn, die ein Knoten hat. Damit wird erreicht, dass der Eckpunkt auch einen Einfluss hat, wenn der Knoten viele Nachbarn hat.

Neben dem Gewicht der Eckpunkte haben nun auch die Initialkoordinaten eines Knotens einen Einfluss. So bleiben die Knoten mehr oder weniger immer in der Nähe ihres Startpunktes. Die Initialkoordinaten werden ebenfalls x-mal gewichtet. Dies verhindert, dass sich das Netzwerk zusammenzieht und so einerseits die längste Distanz zwischen zwei Knoten mit einer Kante nicht grösser wird und andererseits die kürzeste Distanz zwischen zwei Knoten ohne Kanten nicht kleiner wird.

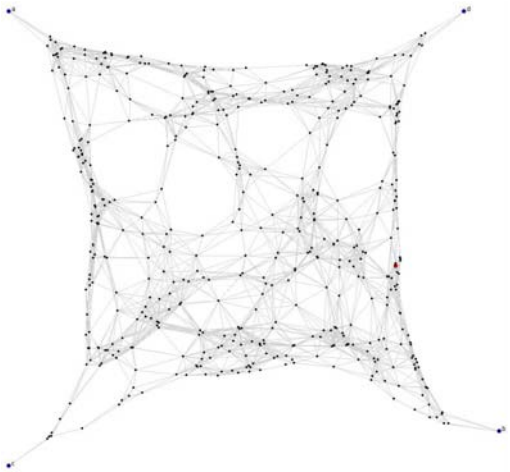

Abbildung 13: Das Netwerk aus Abbildung 9. Nach einer gewichteten Relaxations-Iteration.

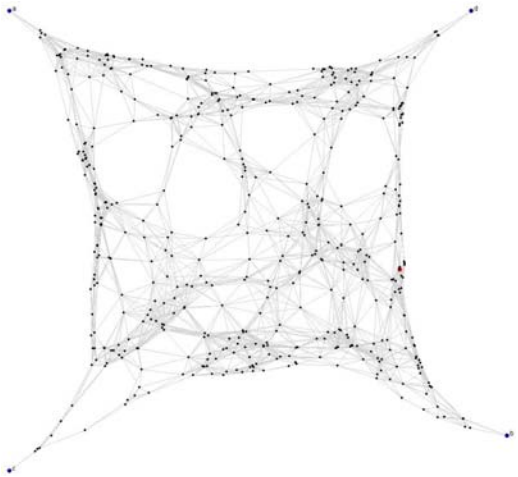

Abbildung 14: Das Netwerk aus Abbildung 9. Nach 7 gewichteten Relaxations-Iterationen. Der Algorithmus stoppte nach 7 Iterationen, weil das Relaxations-Epsilon von 0.01 unterschritten wurde.

Wie man sieht, ist durch das Gewichten das Kantenverhältnis Rho deutlich verbessert worden. Die Werte bleiben jetzt mehr oder weniger konstant.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
3277	48641	1	15.75
5054	74889	10	13.99
22832	337481	100	15.96

Tabelle 7: Daten für ein Netzwerk mit 200 Knoten und einer Dichte von 15. Die Daten sind die Durchschnittswerte von 50 Messungen.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
8446	119006	1	29.48
12178	171441	10	25.56
48008	673373	100	24.95

Tabelle 8: Daten für ein Netzwerk mit 400 Knoten und einer Dichte von 15.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
22805	317296	1	40.44
29745	413219	10	42.4
101786	1411768	100	41.75

Tabelle 9: Daten für ein Netzwerk mit 800 Knoten und einer Dichte von 15.

3.3 Randknoten bestimmen

Der zweite Lösungsvorschlag besteht darin wieder mehr Randknoten einzuführen. Dadurch sollen die Knoten an den Rand gezogen werden. Die Randknoten werden dabei durch den kürzesten Pfad zwischen zwei Eckpunkten gefunden. Gibt es mehrere kürzeste Pfade, wird jener ausgewählt, der am Weitesten von den gegenüberliegenden Eckknoten entfernt ist. Zum Beispiel für die kürzesten Pfade zwischen a und c wird jener ausgewählt, der von b die grösste Entfernung hat.

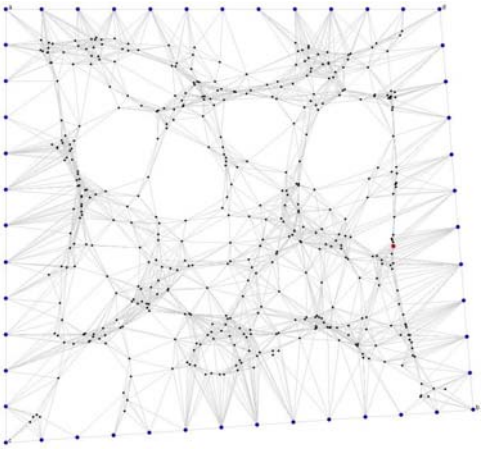

Abbildung 15: Das Netwerk aus Abbildung 9 mit 48 Randknoten. Nach einer normalen Relaxations-Iteration.

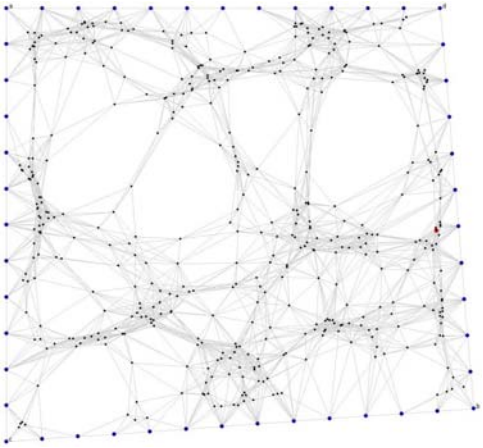

Abbildung 16: Das Netwerk aus Abbildung 9 mit 48 Randknoten. Nach 35 normalen Relaxations-Iterationen. Der Algorithmus stoppte nach 35 Iterationen, weil das Relaxations-Epsilon von 0.01 unterschritten wurde.

Die Einführung von Randknoten verhinderte ebenfalls, das sich die Knoten gegen die Mitte des Netzwerkes zusammenklumpen. Das Kantenverhältnis Rho zeigt aber weniger gute Werte auf, als das bei der Gewichtung vom vorherigen Abschnitt der Fall war.

Rand-knoten	gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
29	3343	49391	1	27.77
29	5105	75487	10	29.31
29	22979	339521	100	23.52

Tabelle 10: Daten für ein Netzwerk mit 200 Knoten und einer Dichte von 15. Die Daten sind die Durchschnittswerte von 50 Messungen.

Rand-knoten	gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
45	8872	124948	1	46.38
45	12503	175870	10	32.58
45	48289	677207	100	44.96

Tabelle 11: Daten für ein Netzwerk mit 400 Knoten und einer Dichte von 15.

Rand-knoten	gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
64	22948	319324	1	109.49
64	30162	418981	10	103.81
64	101779	1411195	100	109.92

Tabelle 12: Daten für ein Netzwerk mit 800 Knoten und einer Dichte von 15.

Weil die Grösse des Kantenverhältnisses vor allem von der kleinsten Distanz zwischen zwei Knoten, die keine Kante haben, abhängt wird im nächsten Kapitel ein weiteres Gewicht für die Relaxations-Iterationen eingeführt, das Negative Gewicht.

3.4 Negative Gewichte

Negative Gewichte werden eingeführt, weil Knoten, die zu einem anderen Knoten eine euklidische Entfernung kleiner als der Senderadius (bei uns eine Einheit) haben, ohne mit einer Kante verbunden zu sein, sich gegenseitig abstossen sollen. Haben zwei Knoten p und q den euklidischen Abstand x so befindet sich der virtuelle Nachbarknoten p' von q auf der von p gegenüberliegenden Seite mit dem Abstand $d = (Senderadius\text{-}x)^2$ von q.

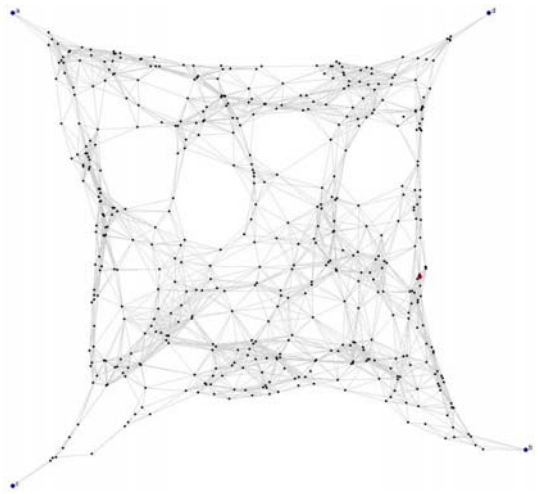

Abbildung 17: Das Netwerk aus Abbildung 9. Nach einer Relaxations-Iteration mit negativen Gewichten.

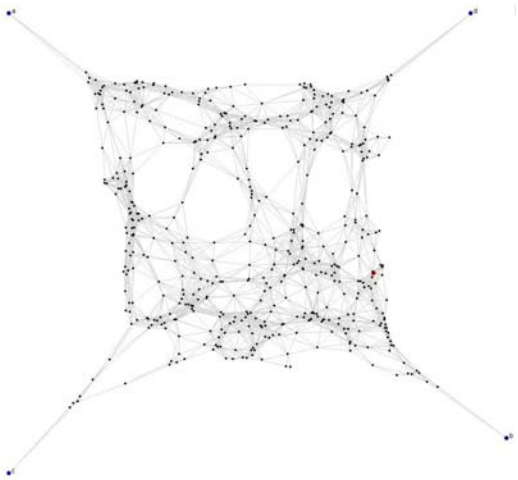

Abbildung 18: Das Netwerk aus Abbildung 9. Nach 75 Relaxations-Iterationen mit negativen Gewichten. Der Algorithmus stoppte nach 75 Iterationen, weil das Relaxations-Epsilon von 0.01 unterschritten wurde.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
3002	44499	1	82.02
4831	71503	10	5.7
22629	334351	100	7.5

Tabelle 13: Daten für ein Netzwerk mit 200 Knoten und einer Dichte von 15. Die Daten sind die Durchschnittswerte von 50 Messungen.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
8322	117114	1	22.78
11779	165443	10	7.55
47640	668091	100	9.58

Tabelle 14: Daten für ein Netzwerk mit 400 Knoten und einer Dichte von 15.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
21575	299611	1	35.11
28965	402384	10	7.65
100619	1395232	100	9.87

Tabelle 15: Daten für ein Netzwerk mit 800 Knoten und einer Dichte von 15.

Die negativen Gewichte verbessern das Kantenverhältnis Rho enorm. Aber erst nach mehreren Iterationen. Erstaunlicherweise haben die Anzahl Knoten auf Rho jetzt fast keinen Einfluss mehr. Der Unterschied zwischen 200 Knoten und 800 Knoten ist nur noch minimal. Der Anstieg von Rho zwischen 10 und 100 Iterationen ist wieder auf die Verklumpung der Knoten in Richtung Zentrum des Netzwerks zurückzuführen, d.h. die Kanten zwischen Ecken und nicht Ecken werden immer mehr gedehnt, während die kürzeste Distanz zwischen zwei Knoten im Graphen ohne Kante, sich in der Waage hält.

3.5 Gewichte und Negative Gewichte kombiniert

Es soll nun versucht werden, die Vorteile der zwei Verbesserungen der gewichteten Ecken und der Negativen Gewichte zu kombinieren.

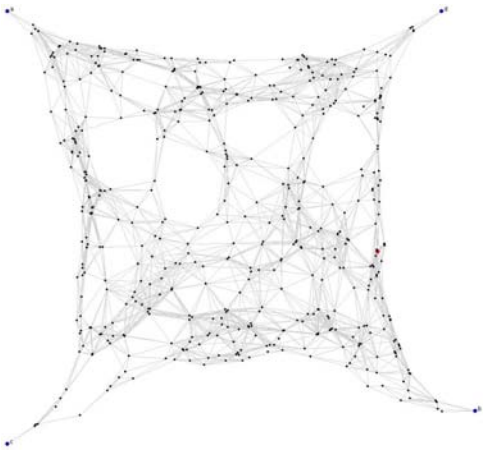

Abbildung 19: Das Netwerk aus Abbildung 9. Nach einer gewichteten Relaxations-Iteration mit zusätzlichen negativen Gewichten.

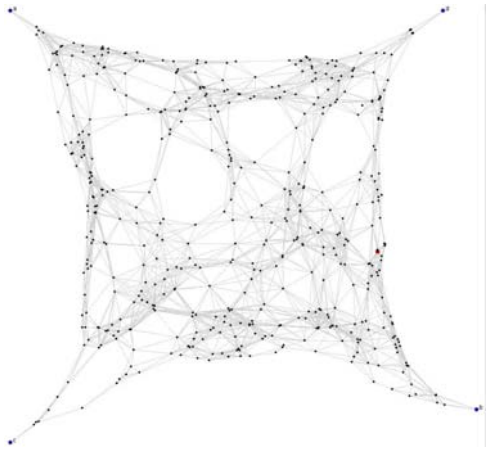

Abbildung 20: Das Netwerk aus Abbildung 9. Nach zehn gewichteten Relaxations-Iterationen mit zusätzlichen negativen Gewichten.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
3036	45399	1	8.86
4809	71749	10	5.89
22578	336111	100	5.45

Tabelle 16: Daten für ein Netzwerk mit 200 Knoten und einer Dichte von 15. Die Daten sind die Durchschnittswerte von 50 Messungen.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
7884	111459	1	22.07
11448	161689	10	8.41
47329	665981	100	6.72

Tabelle 17: Daten für ein Netzwerk mit 400 Knoten und einer Dichte von 15.

gesendete Nachrichten	empfangene Nachrichten	Relaxations Iterationen	Rho Kantenverhältnis
21598	301136	1	25.72
29167	405876	10	8.36
100870	1399803	100	7.57

Tabelle 18: Daten für ein Netzwerk mit 800 Knoten und einer Dichte von 15.

Vergleicht man nun die Tabellen 16 bis 18 mit den Tabellen 13 bis 15, sieht man schnell, dass der Wert von Rho wiederum verbessert worden ist. Zwar liegen die Werte nach zehn Iterationen noch über den Werten vom Algorithmus mit nur negativen Gewichten, doch befinden sie sich nach hundert Iterationen überall darunter.

3.6 Ausblick

Weil die Zeit leider nicht mehr gereicht hatte, um noch mehr Verbesserungen zu implementieren, werden hier nun weitere Ideen noch theoretisch aufgeführt.

3.6.1 Randknoten

Weil die Fläche des Netzwerks nicht unbedingt die Form eines Rechtecks haben muss oder der Rand ziemlich zackig und ausgefranst sein kann, ist es nicht immer sinnvoll den kürzesten Pfad für die Randknoten zu nehmen. Als Alternative wird hier ein Algorithmus aufgezeigt, der die Knoten schrittweise mit folgendem Kriterium auswählt. Zum Beispiel für die Randknoten von a nach c:

Der nächste Knoten ist immer entweder
    ein Hop näher zu c
oder
    ein Hop mehr von a entfernt
oder
    beides
und
    ist unter allen Kandidaten immer am weitesten von b und d entfernt, d.h. der Wert des Produktes *Hopdistanz(Knoten b) * Hopdistanz(Knoten d)* ist bei diesem Knoten am grössten.

Auch sollte ausprobiert werden, ob eventuell auf dem Pfad nur jeder zweite Knoten als Randknoten definiert werden kann. Der Grund, die Anzahl Randknoten zu verkleinern, liegt darin, dass die Knoten zu stark nach aussen gezogen werden und dadurch lange Kanten noch länger werden. Knoten, die sich in einem Knotenhaufen befinden, ziehen sich gegenseitig an, dadurch hat eine einzelne Kante zu einem Entfernten Knoten nur wenig Gewicht.

3.6.2 Gewichte

Also folge von den langen Kanten aus Abschnitt 3.6.1 kommt man auf die Idee, Kanten die länger als ein Senderadius betragen stärker zu Gewichten als Kanten, die weniger als den Senderadius betragen. Also ähnlich wie die negativen Gewichte, nur umgekehrt.

4 Fazit

In dieser Arbeit wird ein Algorithmus für die Berechnung von Virtuellen Koordinaten für Sensornetzwerke untersucht. Es wird versucht diesen Algorithmus der Vorlage entsprechend zu implementieren und mit Simulationen die Ergebnisse zu überprüfen. Weil dies aber nicht ganz gelingt, wird der Algorithmus durch diverse Anpassungen und Änderungen ergänzt, um damit das Kantenverhältnis Rho zu verbessern. Das Resultat am Ende zeigt einen guten, verbesserten Algorithmus, der ein kleines Rho liefert, das aber durchaus noch verbessert werden kann.

4.1 Persönlicher Rückblick

Am Anfang der Diplomarbeit war geplant, dass nach ca. zwei Monaten mehrere Algorithmen für die Berechnung von Virtuellen Koordinaten von Sensorknoten implementiert sind. Aber schon mit dem ersten Algorithmus zeigten sich Probleme, die nicht so schnell zu lösen waren. Viele Details, die wichtig für die Implementierung sind, werden im Paper nur ungenau beschrieben. Vorallem die Triangulations-Berechnungen sind im Paper nur dürftig aufgeführt und ich musste zuerst einen Algorithmus dafür finden, den ich implementieren konnte. Dazu diente mir die JLink Schnittstelle von Mathematica, mit der ich in Java den Mathematica-Kernel aufrufen konnte und so die rechenaufwändigen Differentialgleichungen für die Triangulations-Berechnungen zu lösen.

Literaturverzeichnis

[1] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In ACM MobiCom Conference, September 2003.