

Simulation und Visualisierung von Netzwerken

Inhaltsverzeichnis

1. Netzwerke und ihre Bedeutung.....	1
Zur Idee.....	1
2. Die Anwendung.....	2
2.1. Heranziehen von Netzwerkdaten.....	2
Algorithmische Erstellung.....	2
Manuelle Erstellung.....	2
Laden.....	3
Virtuelle Netzwerkscanner.....	3
2.2. Darstellung von Netzwerken.....	3
Netzwerktopologie.....	3
Verteilungsansicht.....	4
Netzwerkzusammenfassung.....	5
3D-Projektion.....	5
2.3. Untersuchung von Netzwerken.....	5
Wege-Manager.....	6
Mittlere Weglänge.....	6
Rewiring.....	7
Clustering-Koeffizient.....	7
Robots.....	8
2.4. Laden und Speichern.....	8
Benutzerschnittstelle.....	8
Speicherformat.....	9
2.5 Zusammenhangsdiagramm der Klassen und Wirkungsweisen.....	9
3. Leben im Netzwerk: die Robots.....	10
(1) Robots als Signale und Gerüchte.....	10
(2) Robots als (Straßen-) Verkehr.....	11
4. Problemanalyse und Erfahrungen.....	12
5. Projekte in Arbeit und weitere Vorhaben.....	13
Neuronale Netzwerke.....	13
Robots und Ameisenalgorithmus.....	14
Erweiterung des Speichersystems und Skripting.....	14
6. Quellen.....	15

1. Netzwerke und ihre Bedeutung

Netzwerke begegnen uns nicht nur als Gegenstand wissenschaftlicher Betrachtung, sondern vor allem im Alltag, daher ist es naheliegend und sogar notwendig, sich mit ihnen auseinanderzusetzen. Wir treffen sie tagtäglich an, ohne uns dessen bewusst zu sein: Sie sind ein wichtiger Baustein in unserem organisiertem Leben. Wann immer wir uns durch unsere Welt bewegen, bewegen wir uns durch Netzwerke. So unterschiedlich der Straßenverkehr, das Vorzeigenetzwerk Internet oder der Baum am Straßenrand sind: Sie alle haben gemeinsam, dass wir sie als Netzwerk abstrakt mit Hilfe von Knoten und Kanten beschreiben können. Nicht zuletzt lassen sich verschiedenartige Problemstellung mit Hilfe der Graphentheorie lösen, die auch im Alltag Verwendung finden.

Mit *Advanced Network Simulation* (kurz: *ANS*) haben wir uns zur Aufgabe gemacht, eine Computeranwendung zu erstellen, mit der es möglich ist

- (a) reale Sachverhalte nachzubilden, die sich auf abstrakte Netzwerke reduzieren lassen
- (b) Eigenschaften und Ausschlag gebende Parameter sowie die Netzwerke selbst für den Anwender leicht interpretierbar darzustellen
- (c) umfangreiche Vergleichsanalysen durchzuführen
- (d) Netzwerke zu optimieren

Zur Idee

Das Projekt selbst ist im Zuge unserer Beschäftigung mit der Netzwerktheorie im PhysikClub Kassel entstanden. Da wir umfangreiche in Fachliteratur beschriebene Effekte nachvollziehen wollten, benötigten wir ein »Werkzeug« mit dem dies sich besonders einfach gestaltete. Eine umfangreiche Recherche beendeten wir mit der Feststellung, dass zwar eine Vielzahl mathematischer Abhandlungen vorhanden¹ ist, es jedoch keine Computeranwendung gibt, die Grundlagen und Anwendungsbezug der Netzwerktheorie in dem von uns geforderten Maß zum Gegenstand hat. So entschieden wir uns Anfang 2004 eine unseren Anforderungen entsprechende Anwendung zu programmieren, vielmehr, um selbst Forschung an Netzwerkeffekten zu betreiben, da wir auch erkannten, welches Potential in dem unserer Meinung nach trockenen Gebiet der Mathematik steckt, das nicht gerade dazu animiert, Problemstellungen selbstständig zu untersuchen – unabhängig davon, ob man es als Wissenschaftler oder als »Laie« tut.

Wir entwickelten Ideen zur Darstellung von Netzwerken. Komplett neue Aspekte wie die *Robots* entwickelten wir ebenso, wie wir bekannte Algorithmen einbauten, um sie unter einer Oberfläche zu vereinen, die einen Kompromiss zwischen Komplexität und einfacher Bedienbarkeit darstellt.

¹ Unseren Schwerpunkt bilden A. L. Barabásis Überlegungen zur Skalenfreiheit von Netzwerken und grundlegende graphentheoretische Aspekte u.a. Von S. N. Mendes und erweiterte Überlegungen von J. F. F. Dorogovtsev (s. Literaturverzeichnis)

2. Die Anwendung

2.1. Heranziehen von Netzwerkdaten

Mit Advanced Network Simulation stellen wir vier fundamentale Möglichkeiten bereit, Netzwerke und Netzwerkdaten heranzuziehen. Sie werden im Folgenden knapp beschrieben

Algorithmische Erstellung

Führt der Benutzer *ANS* aus, so ist die gebräuchteste Art, mit Netzwerken zu arbeiten, ein solches nach bestimmten Schemata algorithmisch zu erstellen. Mit einem Klick auf »Datei → Neues Netzwerk...« gelangt der Benutzer so zu einem Dialogfenster, das ihm ermöglicht, grundlegende Arten von Netzwerken zu erstellen. Bei diesem Dialog handelt es sich um eine grafische Schnittstelle zu in höchster Weise parametrisierbaren Algorithmen: Möchte man reale Sachverhalte nachstellen, so ist jede dieser Netzwerkstrukturen charakteristisch für eine Vielzahl unterschiedlicher Anwendungsbereiche.

Da *statische* – oder zufällige – *Netzwerke* in natura selten vorkommen, finden sie besonders Anwendung etwa in der Ausübung grundlegender Graphentheorie. Man erstellt sie, indem man eine gegebene Menge Knoten zufällig verbindet.

Wachsende Netzwerke benutzt man zur Simulation des Internets oder sozialer Strukturen. Das Wachstumsverhalten ist einstellbar. Winner-Takes-All-Netzwerke können nachgestellt werden infolge der Ausführung des Algorithmus mit dem sog. Preferential-Attachment (dabei erhalten stark verbundene Knoten tendenziell noch mehr Knoten).

Mit *Small-World-Netzwerken* lässt sich der sogenannte Small-World-Effekt beschreiben. Derartige Netzwerke werden in *ANS* durch das sogenannte *Rewiring* (s. Abschnitt 2.3) hochgeordneter Netzwerke erzeugt.

Perkolationsnetzwerke stellen die Möglichkeit dar, die Perkolation von Netzwerken zu untersuchen, also wie »durchlässig« ein Netzwerk ist. Dabei wird eine reguläre Struktur – man drückt dies in Dimensionen aus – mit einer bestimmten Anzahl von Kanten besetzt. Ein Knoten kann nur mit seinen nächsten Nachbarn verbunden werden. Anwendung findet dieser Netzwerktypus einerseits in der Geologie (s. Abschnitt »Clustering-Koeffizient«), andererseits aber in der Soziologie, Festkörperphysik und besonderen Szenarien – wie etwa Waldbränden – Anwendung.

Manuelle Erstellung

Eine vollkommen freie Netzwerkerstellung ist mit dem »Zeichnen« von Netzwerken möglich. Der Benutzer erstellt hierbei Knoten und verbindet diese durch Mausklick. Einfache Szenarien lassen sich dadurch schnell nachstellen und Algorithmen können anhand manuell erstellter Spezialfälle überprüft werden.

Laden

Es ist die Möglichkeit vorhanden, zuvor zusammengestellte Szenarien etc. aus gespeicherten Projektdateien zu laden. Somit ist der Datenaustausch von Benutzer sichergestellt, wodurch effektiver geforscht werden kann. Auch Vergleichsanalysen werden dadurch merklich vereinfacht.

Virtuelle Netzwerkscanner

Die Tatsache, dass es sich bei *ANS* um eine Computeranwendung handelt, legt die Überlegung nahe, schnell auf große, real existierende Netzwerkdaten zugreifen zu wollen. Zu diesem Zweck haben wir die »Virtuellen Netzwerkscanner« implementiert. Es handelt sich dabei um eine Schnittstelle, um auf eben derartige Computernetzwerke zuzugreifen.

Ein momentan implementierter Scanner liest einen gewählten Bereich des Dateisystems – also die Verzeichnisstruktur – ein und stellt diese als Netzwerk dar. Dies lässt sich etwa bei Dateisystemoptimierungen optimal einsetzen. Auch der Einsatz als »Dateimanager« wäre vorstellbar.

Unsere Anstrengungen konzentrieren sich momentan darauf, zweierlei weitere Scanner zu implementieren: Einerseits den Website-Scanner, der die Beziehungen zwischen Websites und deren Hyperlinks einliest. Andererseits arbeiten wir an der Schnittstelle zum Scannen des IRC-Netzwerks. Dies würde eine einfache und zugleich komplexe – also leistungsfähige – Möglichkeit darstellen, soziale Netzwerke zu untersuchen – denn nichts Anderes sind Internetchats. Clusterstrukturen ließen sich besonders hervorheben sowie das Kommunikationsverhalten und weitere Problemstellungen untersuchen.

2.2. Darstellung von Netzwerken

Unsere Computeranwendung ermöglicht das Betrachten von Netzwerken unter vier unterschiedlichen Gesichtspunkten.

Netzwerktopologie

Das Netzwerk an sich wird mit Knoten, roten Punkte, und Kanten, schwarzen Linien, auf dem Canvas visualisiert. Die Knoten werden zufällig verteilt, denn für Graphen ist die Topologie zunächst irrelevant.

Diese Ansichtsart ist interaktiv, sodass es möglich ist, Knoten und Kanten per Mausklick hinzuzufügen und zu entfernen sowie diese zu verschieben. Außerdem können Netzwerkelemente (Knoten und Kanten) markiert werden, wobei der Benutzer ausführlich über sie informiert wird (Grad, Identifikationsnummer, Clustering-Koeffizient usw.).

Mit der Zuweisung sog. Knoten- und Kantentags ist die Beschriftung der Elemente möglich. Ein Tag kann privat oder öffentlich sein. Öffentliche Tags sind benutzerdefinierte Tags (z.B. um ein Netzwerkelement zu benennen). Somit ist es möglich, auch den Grad oder den Clustering-

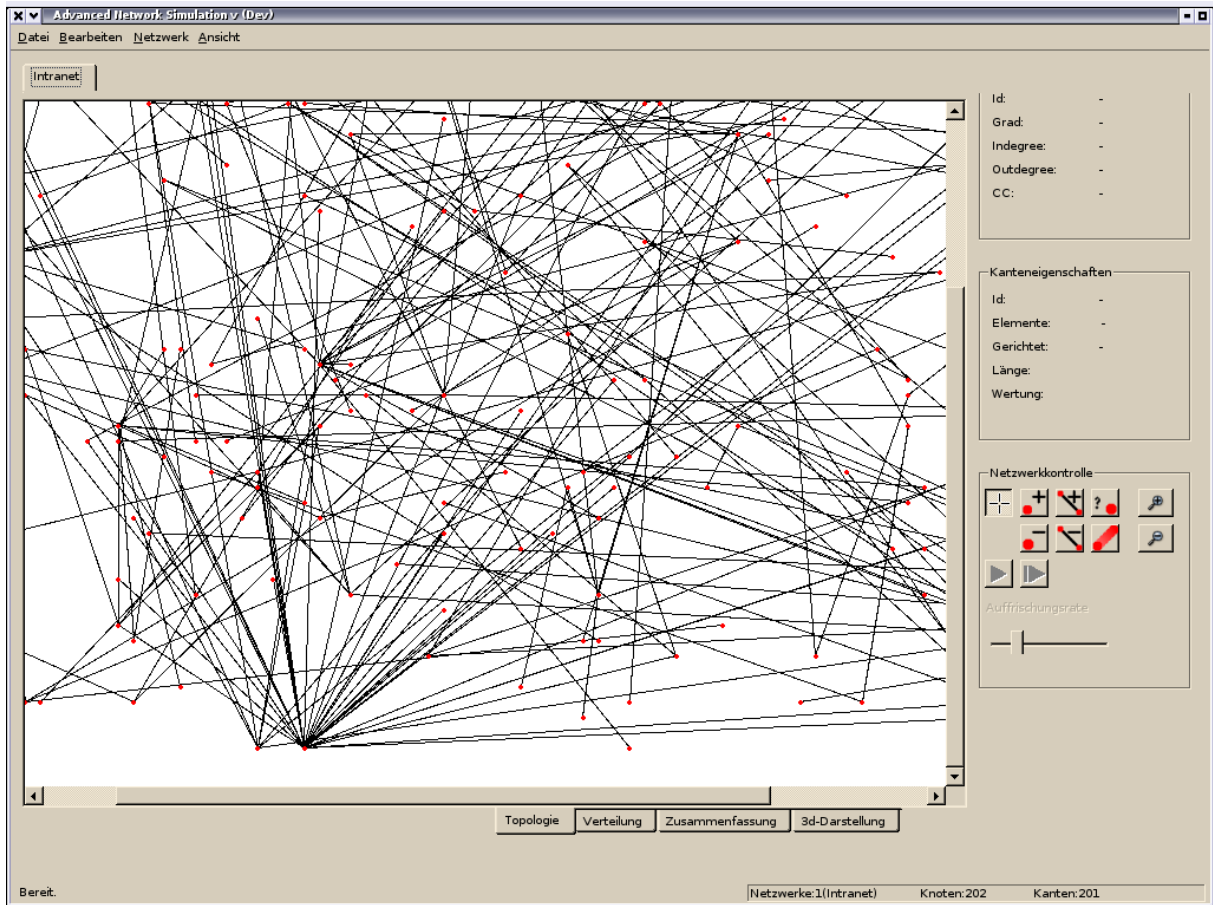


Abbildung 1: Topologie eines wachsenden Netzwerkes mit grafischer Benutzerschnittstelle

Koeffizienten neben dem entsprechenden Knoten einzublenden. Private Tags werden von Algorithmen und Klassen benutzt, um einen Knoten in bestimmter Weise zu markieren (zum Beispiel können Robots auf Knoten bzw. Kanten ablegen, ob sie diese bereits betreten haben).

Verteilungsansicht

Verschiedenste Verteilungskurven sind mächtige Werkzeuge der Graphentheorie. So lassen sich mit der Gradverteilungskurve (1. Achse: Grad, 2. Achse: Knotenanzahl) zufällige/statische Netzwerke von wachsenden und von skalenfreien sowie hochgeordneten Netzwerken unterscheiden. Die Bedeutung der Clustering-Koeffizienten-Verteilung ist Gegenstand unserer weiteren Untersuchungen.

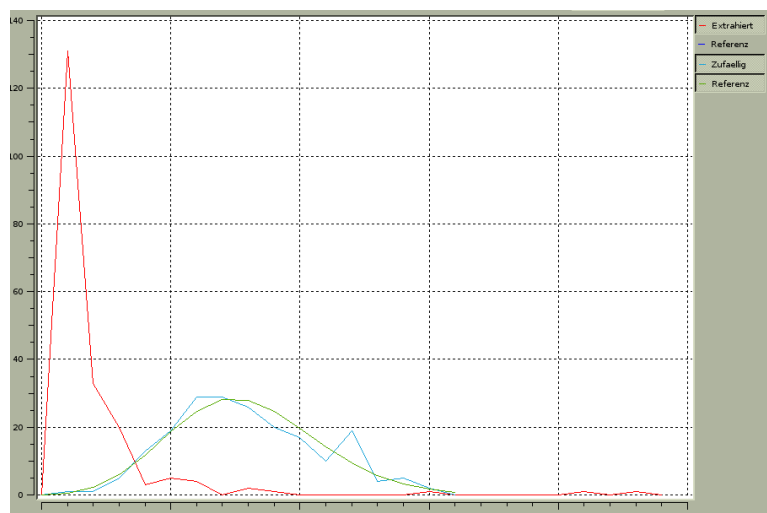


Abbildung 2: Gradverteilungsansicht - wachsendes Netzwerk (rot), extrahierte Gradverteilung eines statischen Netzwerkes (hellblau), berechnete Idealkurve des selben statischen Netzwerkes (grün)

Einerseits können Kurven aus Netzwerken beliebig extrahiert werden und andererseits ist das Erstellen von Referenzkurven möglich. Sie stellen Idealfälle wachsender bzw. statischer Netzwerke dar.

Netzwerkzusammenfassung

Der Modus *Netzwerkzusammenfassung* stellt wesentliche errechnete und extrahierte Eigenschaften aller geöffneten Netzwerke übersichtlich nebeneinander dar. Ein qualitativer Überblick kann somit schnell erlangt werden, ohne erst als Anwender die Topologie interpretieren zu müssen.

3D-Projektion

Zuletzt bleibt die *3D-Projektion*, bei dieser Ansicht kann wahlweise der Clustering-Koeffizient oder der Grad topologieabhängig auf das Netzwerk projiziert werden. Der Vorteil liegt darin, lokale Gebiete beurteilen zu können oder einen Gesamteindruck etwa bei größeren Netzwerken über den Grad zu erlangen. Auf der zweidimensionalen Ansicht ist oft schwer zu sagen, ob der Grad gewisser Knoten signifikant größer ist als der der übrigen Knoten. Auf der dreidimensionalen Ansicht lässt sich soz. ein Querschnitt durch Änderung des Betrachtungswinkels erzielen, oder bei einer Ansicht von schräg oben lassen sich gut Maxima oder Minima ausmachen.

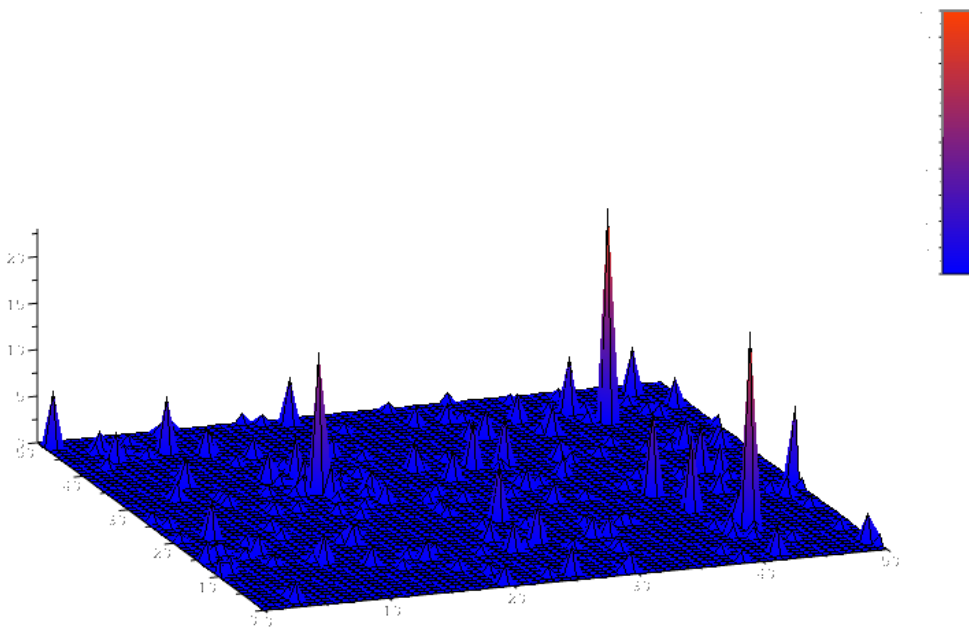


Abbildung 3: 3D-Projektion der Knotengrade (vorheriges Netzwerk aus Topologieansicht)

2.3. Untersuchung von Netzwerken

Wir stellen mit unserem Programm neben den bereits genannten Darstellungsmöglichkeiten verschiedenartige Werkzeuge zum Untersuchen von Netzwerken bereit. Angetrieben durch grundlegende Fragen wie etwa »Wodurch unterscheidet sich das Internet von der Gesellschaft und wie kann man das darstellen« oder »Welchen Einfluss haben verschiedenartige Eingriffe in das

Netzwerk auf dessen Störanfälligkeit« implementierten wir die Werkzeuge der kürzesten Wegsuche und der Kreissuche. Natürlich wurden gerade diese Fragen bereits ausführlich in wissenschaftlichen Publikationen behandelt. Doch waren es optimale Umstände, die bereits »professionell« durchgeführten Untersuchungen mit eigenen Mitteln nachzuvollziehen und u. U. zu erweitern.

Wege-Manager

Hinter diesem Begriff verbirgt sich eine grafische Schnittstelle, die dem Benutzer Zugriff auf drei wesentliche Algorithmen gestattet: Die angesprochene kürzeste Wegsuche und Kreissuche sowie eine Umsetzung des sogenannten Handlungsreisenden-Problems mithilfe des

»Ameisenalgorithmus«. Hierbei ist das Ziel, einen geschlossenen Weg zu finden, der alle Knoten des Netzwerkes einfach beinhaltet. Die Vorgehensweise, virtuelle Ameisen durch ein Netzwerk zu schicken und – ähnlich wie in der Natur – die Wege zu markieren, gibt dem Algorithmus seinen Namen. Die einzelnen Ameisen berücksichtigen dabei die Markierungen der vorhergehenden Ameisen und die Länge der Einzelstrecken des Rundweges und wählen dementsprechend ihren Weg aus. Ein virtueller Verwitterungseffekt beschleunigt das Ermitteln eines Resultats. Verglichen mit anderen Algorithmen ermittelt dieser Algorithmus einen annähernd optimalen bis optimalen Weg bei einem Minimum an Rechenoperationen.

Da der Ameisenalgorithmus prinzipiell sehr leistungsfähig ist, möchten wir interessante, möglicherweise alltägliche Problemstellungen untersuchen: Etwa das Verteilen von Sitzplätzen auf Veranstaltungen unter Berücksichtigung der Einzelpräferenzen oder das Verteilen von Schülern auf Kurse. Dazu arbeiten wir an einer Verallgemeinerung dieses Algorithmus.

Mittlere Weglänge

ANS extrahiert das arithmetische Mittel einer gegebenen Anzahl von Weglängen. Dafür wird der kürzeste Weglängen-Algorithmus mit einer benutzerdefinierten Anzahl von Knotenkombinationen ausgeführt. Je mehr Wege in Betracht gezogen werden, desto genauer wird der Wert, desto länger

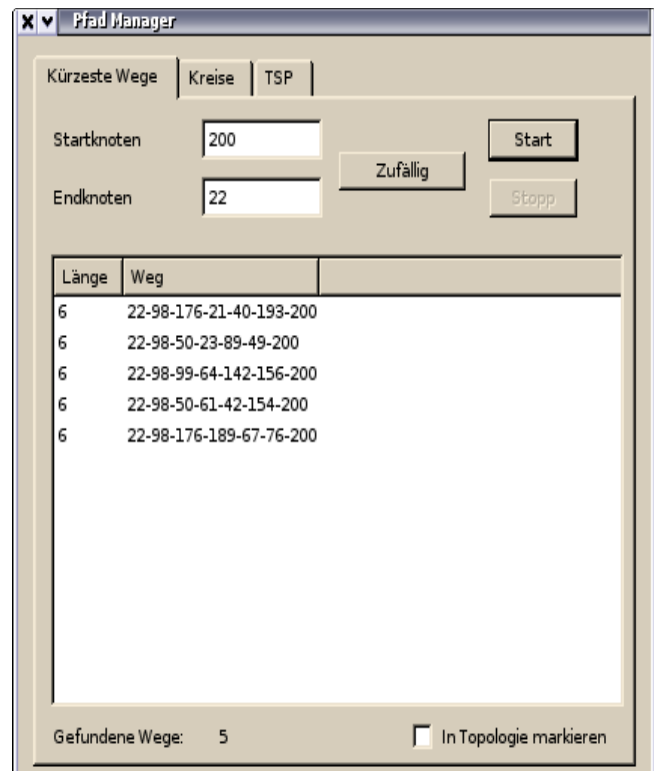


Abbildung 4: Der Wegemanager, zu sehen sind die kürzesten Wege zwischen den Knoten mit den Nummern 200 und 22

dauert aber auch die Operation.

Interessant sind Vergleichsanalysen parallel geöffneter Netzwerke, die sich in einzelnen Parametern geringfügig unterscheiden. Die mittleren Weglängen lassen sich als Diagramm anzeigen, den Einfluss einer Parameterveränderung auf die mittlere Weglänge somit leicht interpretieren. Diese Betrachtung ist wesentlich für eine Aussage über die Ausbreitungsgeschwindigkeit von Signalen über das Netzwerk.

Rewiring

Mit einem simplen Handgriff kann man hoch geordnete Netzwerke derart verändern, dass sie eine Nachbildung von sozialen Netzwerken darstellen: Es wird ein Kompromiss zwischen Ordnung und Zufälligkeit gefunden.

So haben wir das Wiederverknüpfen von Verbindungen implementiert (engl.: *rewiring*). Zunächst ermöglicht *ANS* dem Benutzer die Auswahl einer absoluten oder einer relativen Anzahl von Kanten. Schließlich muss noch bestimmt werden, ob ein Ende einer Kante mit einem anderen – zufällig ausgewählten – Knoten verbunden werden soll, oder ob Kanten einfach hinzugefügt werden sollen.

Clustering-Koeffizient

Die Perkolationstheorie der Graphentheorie macht man sich in Bereichen wie der Geologie oder Physik zu Nutze. So untersucht man in der Geologie das Maß an Porosität, das ein Gestein haben muss, um wasserdurchlässig zu sein, oder in der Physik etwa Effekte wie das Gefrieren von Flüssigkeiten.

Der Clustering-Koeffizient trifft Aussagen über den Nachbarschaftszusammenhang im direkten Umfeld eines Knotens, während im Gegensatz dazu die mittlere Weglänge u.a. ein Maß für Langstreckenverbindungen ist. Interessant ist nun die Untersuchung von »kritischen Perkolationsschwellen«, also einem kritischen Wert für den gesamten Netzwerk-Clustering-Koeffizienten, der den Übergang von Nicht-Durchlässigkeit zu Durchlässigkeit ausdrückt (der Punkt, an dem sich ein einziges großes *Cluster* im Netzwerk gebildet hat).

An der Untersuchung von Auswirkungen des Clustering-Koeffizienten auf die Störanfälligkeit von Netzwerken möchten wir uns selbst aktiv beteiligen, indem wir die Clustering-Koeffizienten-Verteilung mit der Ausbreitung von Signalen über unterschiedliche Netzwerke vergleichen. Führt man den Clustering-Koeffizienten-Algorithmus in statischen Netzwerken durch – nicht in Perkolationsnetzwerken – so ist insbesondere bei der Untersuchung von Perkolation zu beachten, dass es sich bei diesem Netzwerktyp um ein $(N-1)$ -dimensionales Perkolationsnetzwerk handelt, wobei N für die Anzahl der Knoten steht.

Die folgenden Grafiken stellen jedoch einen Aspekt dar, der mit Perkolationsnetzwerken möglich ist: In Kombination mit den Robots lassen sich Waldbrandsimulationen oder etwa

Epidemieprognosen durchrechnen. Der Clustering-Koeffizient eines jeden Knotens gibt Auskunft darüber, wie stark sich ein Signal ihm ausbreiten kann. Dementsprechend lassen sich etwa unterschiedliche Klassifikationen von »Brandherden« - oder besser: »Epizentren« - treffen.

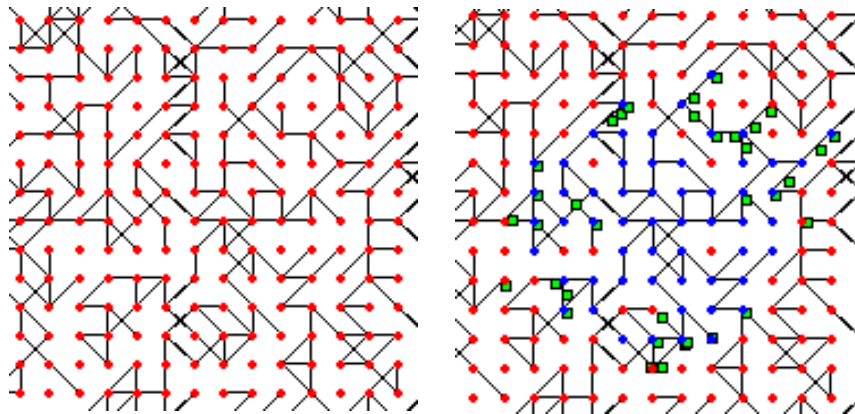


Abbildung 5: Ein Signal breitet sich konzentrisch von einem »Epizentrum« aus. Der Ausschnitt in unmodifizierter Form (links) ist hier demselben nach einigen Zeitschritten Signalausbreitung gegenübergestellt (rechts).

Robots

Die Robots werden gesondert beschrieben und sind hier der Vollständigkeit halber aufgeführt.

2.4. Laden und Speichern

Wir haben festgestellt, dass es absolut notwendig ist, Netzwerke zu speichern und zu einem anderen Zeitpunkt wieder zu laden um effektive Vergleichsanalysen von Netzwerken durchzuführen. Kam es zu einem Programmabsturz, waren bestimmte Netzwerke besonders beispielhaft oder führten wir empirische Untersuchungen auf einer Netzwerkstruktur durch, so verzögerte sich die Arbeit durch den ständigen manuellen Eingriff. Dies war insbesondere bei Vergleichsanalysen wachsender Netzwerke hinderlich, die wir durchgeführt haben.

Deswegen implementierten wir ein für unsere Ansprüche sehr leistungsfähiges Speicher- und Ladesystem.

Benutzerschnittstelle

Zunächst besteht die Möglichkeit zwischen dem Speichern aller geöffneten Netzwerke, Kurven und sonstiger Elemente oder dem Speichern mit mehreren Auswahlmöglichkeiten. Es können nun vorhandene Netzwerke und Kurven zu sogenannten *Bundles* zusammengefasst werden. Diese erhalten einen benutzerspezifischen Namen und sind dazu gedacht, Netzwerke in einem speziellen Kontext öffnen zu können (etwa dem Vergleich der erstellten wachsenden Netzwerke oder der Darstellung des Veränderungsprozesses an einem Small-World-Netzwerk). Schließlich können weitere Informationen zum Projekt angegeben werden: Projektname, Name des erstellenden Benutzers (optional), Beschreibung (optional), Datum (automatisch eingefügt), Speicherpfad.

Der Benutzerdialog zum Laden zeigt diese Informationen dann übersichtlich an. Zusätzlich dazu die Anzahl der gespeicherten Netzwerke, Kurven und die Dateigröße. Alle in dem Projekt

gespeicherten Netzwerke sind separat und kombiniert anwählbar, die Bundles sind hier gewisse Markierungsvoreinstellungen.

Speicherformat

Anfangs hatten wir Bedenken über die Größe zu speichernder Projekte, denn Netzwerke werden sehr schnell sehr speicherintensiv. Trotzdem ist uns ein vergleichsweise guter Kompromiss zwischen Arbeitsaufwand und Leistungsfähigkeit gelungen.

Ein Projekt besteht aus mehreren Einzeldateien, die zusammengefasst (libtar) und anschließend komprimiert (bzlib) werden. So ist in jedem Projekt ein Headerfile zu finden, welches allgemeine Informationen zum Projekt auf technischer und Anwenderseite enthält: Beschreibung, Datum, Benutzername etc. aber auch Informationen über das Speicherformat und die Versionsnummer. Zudem besteht eine Übersicht über alle Netzwerke, Bundles und Kurven des Projektes und die Namen der Dateien, durch welche sie genauer beschrieben werden.

Die Netzwerkbeschreibungsdateien enthalten allgemeine Informationen zu dem Netzwerk (Name, Beschreibung, Knotenzahl, Kantenzahl und den Namen der Datei zum Screenshot). Der Screenshot des Netzwerkes wird beim Laden angezeigt, sodass dem Benutzer die Erkennung des Netzwerkes erleichtert wird. Hinzu kommen die notwendigen Informationen zu jedem Knoten, jeder Kante und der dazugehörenden Tags.

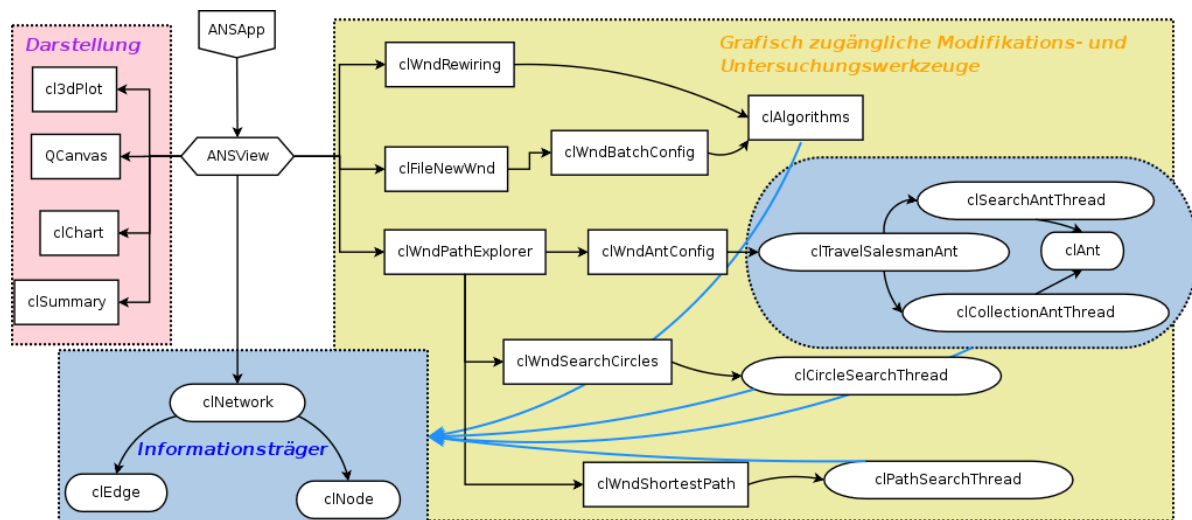
Schematisch sieht eine Projektdatei folgendermaßen aus:

```
header.xml
net0.xml,    ... ,    netN.xml
schnet0.xml, ... ,    schnetN.xml
curve0.xml,  ... ,    curveN.xml
bundle0.xml, ... ,    bundleN.xml
```

Die Einzeldateien sind XML-konform, sodass wir mit entsprechenden Zusatzbibliotheken auf effektive und komplexe Weise unser Speicherformat jederzeit ausbauen können.

2.5 Zusammenhangsdiagramm der Klassen und Wirkungsweisen

Das folgende Schaubild ist eine Darstellung einer Auswahl von Klassen, welche wesentliche Bestandteile unserer Anwendung bilden.



3. Leben im Netzwerk: die Robots

Die Untersuchung von Netzwerken, wie wir sie bisher betrachtet haben, ist sehr theoretisch und findet im Alltag keine deutliche Anwendung. Ebenso wenig lassen sich mithilfe derart gewonnener Ergebnisse konkrete Aussagen auf real existierende, mitunter komplexe Netzwerksysteme ziehen. Wesentlich ist nämlich ein Aspekt, der reale Netzwerke ausmacht: Sie existieren nicht nur, sondern »leben«, indem das Netzwerk selbst als Grundstruktur entsteht, die das Verhalten eines komplexen Systems ausmacht. Es bietet sich nun kaum an, diese Aspekte mithilfe von Algorithmen zu lösen, die – einmal in Gang gesetzt – nur mit einem hohen Grad an Verschachtelung eine vergleichbare Komplexität erreichen.

So machen wir uns die Objektorientierung zunutze und verwenden auf dem Netzwerk existierende Einheiten - wir nennen sie »Robots« - die je nach Anwendung unterschiedliche Aufgaben übernehmen. Es seien nun in Kürze einige genannt:

(1) Robots als Signale und Gerüchte

Hier geht man von einem Schaltkreis bzw. Stromnetzwerk aus (»Signale«) oder in dem anderen Fall von einem sozialen Netzwerk (»Gerüchte«). Indem die Robots nun Signale bzw. Gerüchte darstellen, kann etwa untersucht werden

- (a) wieviele Knoten entfernt werden müssen, um das Netzwerk instabil zu machen
- (b) welche Umspann- oder Kraftwerke rudimentär sind
- (c) wie man Schaltkreise auf Prozessoren optimiert
- (d) wie sich ein Gerücht verbreitet, wenn die Verbreitung nach bestimmten Regeln abläuft²

² Hayes, Brian: Gerüchte und Vehler (s. Quellenverzeichnis)

(2) Robots als (Straßen-) Verkehr

Anwendungen sind etwa Logistikbetriebe, welche mit möglichst geringen Kosten (Weglänge) die meisten Waren transportieren möchten, was de facto eine besonders komplexe Aufgabe darstellt, da hier nicht die Netzwerkstruktur an sich verändert wird sondern das Verhalten der »Lkw«. Diese Anwendung lässt sich auch mit dem Handlungsreisendenproblem (s. Abschnitt 2.3) kombinieren. Eine weitere Idee ist das internationale Flughafennetz und die Entwicklung eines Flugplanes, der eine optimale Auslastung der Flughäfen erreichen soll.

Unsere Auffassung von einer korrekten technischen Umsetzung der Robots hat sich seit einiger Zeit grundlegend geändert. Zu Beginn setzten wir nämlich auf die Betrachtung der Robots als unabhängig voneinander agierende Threads. Doch stießen wir hier auf technische Probleme, die wir nicht lösen konnten. So gab es hohe Leistungseinbußen bis zum kompletten Programmabsturz bei etwa 800 erstellten Threads. Für komplexe Systeme ist dies zu wenig, zudem hängt die Zahl stark von der Hardware ab. Auf einer 2,4 GHz CPU und 768 Mb RAM waren es

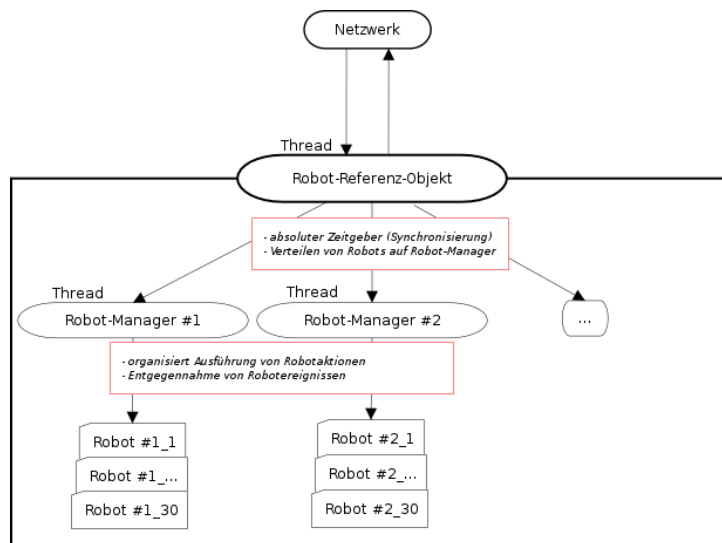


Abbildung 6: Funktionsweise der Robot-Interaktionen

durchschnittlich 1300 Threads, mit denen das Programm ohne Probleme lief, während es auf einer 1 GHz CPU mit 256 Mb RAM nur etwa 400 waren.

Das aktuelle Modell beruht auf einer noch ausgeprägteren Hierarchisierung: Ein Verwaltungsthread (clRobotManager) verwaltet eine bestimmte Anzahl an Robots (clRobot, Standard: 30), wobei auf die Verwaltungsthreads über eine Interfaceklasse (clRobotReferenceObject) von Netzwerkebene etc. zugegriffen wird. So übernimmt die Interfaceklasse die Verwaltung der Manager-Threads, wobei diese für die korrekte Ausführung und das Timing der einzelnen Robot-Objekte verantwortlich sind.

Die Synchronisierung wird dadurch bewerkstelligt, dass die nächste Iteration, in der Robots agieren können, erst freigegeben wird, wenn alle Robots ihre Aufgaben erfüllt haben.

Die höchste von uns gemessene Zahl von Robots, die nebeneinander agieren beträgt nun etwa 370000.

4. Problemanalyse und Erfahrungen

Dieser Abschnitt der Ausarbeitung ist eine kurze Reflexion über unsere Erfahrungen mit einem derart umfangreichen Projekt.

Zunächst ist anzuführen, dass wir uns bei Beginn der Arbeit an diesem Projekt nicht vorstellen konnten, welche Art von Problemen uns konkret erwartete und welches Ausmaß diese haben würden. Derart simple Entscheidungen, wie etwa die grafische Oberfläche oder gar das Betriebssystem, auf dem unsere Anwendung funktionsfähig sein sollte, schienen uns normal und waren nach einigem Hin und Her auch gelöst (so setzten anfangs anstatt auf die Qt-Bibliothek unter Linux auf den Borland-C++-Builder unter Windows). Ging es dann aber um elementare Fragen, wie den konkreten Aufbau einer Anwendung, die Netzwerke *simulieren* und *visualisieren* sollte, fertigten wir umfangreiche Skizzen an, die den roten Faden unserer Anwendung werden sollten und an den wir uns bis dato halten (von einigen Erweiterungen abgesehen). So musste etwa eine vom Programmierer leicht zu bedienende Netzwerkklass implementiert werden, die sich einen standardisierten Zugriff auf (fast) alle in ihr gespeicherten Daten erlaubt, um zum Beispiel unterschiedliche Darstellungsmethoden von Netzwerken nach einem einheitlichen Schema zu errichten. Mittlerweile umfängt diese Klasse mit Funktionsdefinitionen etc. etwa 3000 Zeilen Quellcode (abzüglich ca. 5 % Kommentare).

Ein anderer Problembereich sind die Algorithmen gewesen, die wir zum größten Teil mit Stift und Papier durchgingen, um lesbaren und funktionsfähigen Code zu erstellen. Dies ist ein Bereich, den wir zu respektieren begannen, denn die mathematischen Probleme schienen uns zu Beginn noch die unproblematischsten zu sein, wonach sich aber herausstellte, dass gerade diese Probleme einer besonderen Sorgfalt bedürfen und mitunter viel Entwicklungszeit in Anspruch nehmen; nicht zuletzt auf der endlos scheinenden Liste von Anwendungs- und Untersuchungsmöglichkeiten, die die Graphentheorie ermöglicht. So arbeiteten wir zum Beispiel über einen Zeitraum von sechs Wochen nur an den Algorithmen zur Kreissuche und zur Ermittlung der kürzesten Weglänge zwischen zwei Knoten.

Einen sehr zeitraubenden Aspekt stellten »kleinere« Erweiterungen der Programmfähigkeiten dar, wie etwa das Exportieren von Screenshots der Netzwerke, das Programmieren von Benutzerdialogen, die Multitabbing-Fähigkeit zur Darstellung der Netzwerktopologie etc. Programminterne Verarbeitungsmethoden zur Verwaltung von Netzwerken, grafischen Schnittstellenobjekten usw. haben sich als grundlegende Quellen für Speicherzugriffsfehler und plötzliche Programmabstürze herausgestellt.

Schließlich sind hier die »komplexen Probleme« anzuführen, die eine grundlegende Erweiterung des Programmes darstellen. Exemplarisch dafür stehen folgende Projekte:

- das Speichern (Konzeption einer Projektdatei, Erlernen des Umganges mit XML und den

entsprechenden Bibliotheken libxml++, libtar, bzlib, grafische Schnittstelle, Erstellen von Interfaceklassen und Integration in die Anwendung)

- Robots (Konzeption einer Aufgabenhierarchie einzelner Objekte, Synchronisation von Robots, Kommunikation untereinander, mit Anwendung und mit Netzwerk(-objekten), grafische Schnittstelle, Rücksicht auf Performance)

Sorgfalt war bei allen Schritten geboten, denn wir mussten stets darauf achten, dass sich Erweiterungen einfach einbauen ließen.

Letztendlich besteht die Hauptschwierigkeit – und damit unsere Grundproblemstellung – darin, eine Applikation zu entwickeln, welche einer Vielzahl grundlegender graphentheoretischer Mittel und selbst entworfener Werkzeuge derart bereitstellt, dass sich in universeller Weise Probleme lösen lassen. »Neurowissenschaftler sollen unsere Anwendung also gleichermaßen benutzen können wie der Logistikunternehmer oder Mathematikstudent.«

5. Projekte in Arbeit und weitere Vorhaben

Die wesentliche Arbeit an unserem Projekt stellte bisher das Programmieren einer Anwendung dar, die der Netzwerkuntersuchung mit dem Schwerpunkt Darstellung und Simulation ermöglicht. Komplexe Forschungen sind möglich, wie wir bereits erfahren haben, indem wir etwa Arbeitsgruppen des PhysikClubs in Kassel Netzwerke für Forschungszwecke simulierten.

Wir stellten weiterhin fest, dass die Komplexität und Universalität der Netzwerktheorie schier unbegrenzte Impulse bietet und wir deswegen auch eine Vielzahl noch nicht umgesetzter Ideen haben, die wir teilweise bereits implementieren.

Neuronale Netzwerke

Aktuellen Bezug besitzt das Thema »Neuronale Netze« - schon allein durch die intensive Forschung im Bereich der Biologie. Idee ist es, diesen Aspekt in zweierlei Hinsicht zu implementieren:

(1) Die Arbeitsweise und das Erscheinungsbild Neuronaler Netze wird (grafisch) dargestellt, wodurch etwa die Wirkung äußerer Einflüsse durch den Benutzer plastisch nachvollziehbar wird.

(2) Wir setzen Neuronale Netze als Backend ein, um zu simulierende Netzwerke besonders effektiv optimieren und an eigene Bedürfnisse anpassen zu können. Ziel ist es etwa, das Neuronale Netz auf eine gewollte Eigenschaft hin anzulernen. Erstellt man daraufhin Netzwerke oder gibt bestehende Netzwerke ein, so kann man sich dieses u. U. optimieren lassen – und zwar mit minimalen Veränderungen.

Z. Z. stellen wir die Verbindung zu unterschiedlichen Institutionen, die sich mit dieser Thematik beschäftigen, her. Vorläufiges Ergebnis ist die Erkenntnis, dass es eine Vielzahl äußerst spezialisierter Software gibt, die Neuronale Netze in keiner vergleichbaren Weise einsetzt.

Darauf aufbauend ließe sich durchaus in Kombination mit den Robots eine Art Simulation

künstlicher Intelligenz durchführen.

Geplant ist – zumindest für Stichpunkt 2 – Anfang April eine funktionsfähige Implementierung Neuronaler Netze als Backend zu integrieren.

Robots und Ameisenalgorithmus

Eine Verallgemeinerung des Ameisenalgorithmus auf andere Probleme als das des Handlungsreisenden ist auch in Arbeit sowie die Erweiterung von Robots, um etwa komplexere Sachverhalte zu simulieren (z. B. internationales Flughafennetz unter Berücksichtigung von Passagierzahlenzuwachs an Flughäfen, Kapazitäten von Flughäfen etc.). Mit der Synthese von Ameisenalgorithmus mit den Robots sollen Vorzüge und Fähigkeiten beider Modelle zusammengeführt werden.

Erweiterung des Speichersystems und Skripting

Momentan ist es möglich, Netzwerkprojekte für den weiteren Gebrauch mit *ANS* zu speichern. Größeren Profit hat der Anwender jedoch, wenn Netzwerke in einer universellen Sprache – durchaus von Menschen lesbar – abgespeichert werden können, was die Einbindung in wissenschaftliche Publikationen etc. erheblich erleichtern würde. Man denke etwa an die Ein- und Ausgabe von Adjazenzmatrizen.

Skripte sollen den Gebrauch von *ANS* leistungsfähiger machen. Szenarien wären schnell zu erstellen, automatisierbar und ihre Komplexität und Anpassbarkeit nähme zu.

6. Quellen

Barabási, A.L.: *Linked, The New Science of Networks*. Perseus Publishing, Cambridge 2002

Barabási, A.L. und Albert, R.: *Statistical mechanics of complex networks*. In: *Reviews Of Modern Physics*, Ausgabe 74, Januar 2002

Dorogovtsev, S.N. und Mendes, J.F.F.: *Evolution of Networks, From Biological Nets to the Internet and WWW*. Oxford University Press, New York 2003

Hayes, Brian: *Gerüchte und Vehler*, in: *Spektrum der Wissenschaft* 12/05, S. 116 – 121