Untersuchung von Algorithmen zur Generierung uniform verteilter Zufallsgraphen anhand einer Grad-Sequenz

Wolfgang Schlauch

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

BACHELORARBEIT

Untersuchung von Algorithmen zur Generierung uniform verteilter Zufallsgraphen anhand einer Grad-Sequenz

Wolfgang Schlauch

BETREUER:
Prof. Dr. Prof. h.c. Andreas Dengel
Dipl. Inf. Darko Obradovic

19. Oktober 2010

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Historie . . . 1
- 1.2 Motivation . . . 2

2 Einführung in die Graphentheorie **5**
- 2.1 Terminologie . . . 5
- 2.2 Spezielle Graphen . . . 8
  - 2.2.1 Zufallsgraph . . . 8
  - 2.2.2 Skalenfreies Netzwerk . . . 8
  - 2.2.3 Konfigurationsmodell . . . 11

3 Monte-Carlo-Markov-Ketten-Algorithmus **13**
- 3.1 Prinzip . . . 13
- 3.2 Havel-Hakimi-Algorithmus . . . 14
- 3.3 Markov-Ketten-Monte-Carlo-Algorithmus . . . 16
- 3.4 Vorteile gegenüber dem Konfigurationsmodell . . . 18

4 Statistische Evaluation **19**
- 4.1 Toy-Netzwerk . . . 19
- 4.2 Motiverkennung auf realen Netzwerken . . . 21
- 4.3 Powerlaw Netzwerke . . . 22

5 Berechnung der Uniformität **25**
- 5.1 Markov-Kette . . . 25
- 5.2 Engültiger Evaluationsansatz . . . 26
- 5.3 Die „Konstante“ q . . . 28

6 Fazit **31**
- 6.1 Zusammenfassung . . . 31
- 6.2 Ausblick . . . 32

Abbildungsverzeichnis

2.1 Gerichteter Graph mit Schleife 5
2.2 Ungerichteter und Gerichteter Graph 6
2.3 Multigraph 7
2.4 Gewichteter Graph 7
2.5 Erdös-Rényi-Graph 9
2.6 Skalenfreies Netzwerk 10
2.7 Konfigurationsmodell-Graph 11

3.1 Kantentausch, Beispiel 1 16
3.2 Kantentausch, Beispiel 2 16
3.3 Verbotener Kantentausch zur Schleife 17

4.1 Toy-Netzwerk 20
4.2 Verteilung der prozentualen Häufigkeit des Auftauchens der verschiedenen Konfigurationen des Toy-Netzwerkes 21
4.3 Feed-Forward-Motiv 22
4.4 Auswertung MCMC-Algorithmus 24

5.1 Knotenverteilung 26
5.2 Konfigurationsgraph Toy-Netzwerk 27
5.3 Evaluation PageRank 30

Tabellenverzeichnis

4.1 Auswertung Havel-Hakimi-Algorithmus 22

5.1 Auswertung PageRank-Algorithmus 29

Algorithmenverzeichnis

1 Konfigurations-Graph 12
2 Generierung von Zufallsgraphen 13
3 Algorithmus Havel-Hakimi 15
4 Algorithmus Monte-Carlo-Markov-Kette 17
5 Algorithmus Powerlaw-Generierung 23

Zusammenfassung

Die Untersuchung von Graphen ist generell ein interessantes Problem, beispielsweise für die Optimierung von Schaltkreisen, aber auch für andere Bereiche, wie die Biologie, relevant. Die Verkettungen von DNA-Sequenzen und ähnlichem in verschiedenen Lebewesen zu untersuchen, um damit Aussagen über sie zu treffen sind von großer Bedeutung für den Menschen. Graphenanalyse wird aber auch in Sozialen Netzwerken immer interessanter, da man durch die Untersuchung von Netzwerken Werbung besser einsetzen kann wenn man weiß, wer ein Meinungsmacher ist in einer Gruppe von Menschen. Ebenfalls ist die Erforschung des Aufbaus von Gruppen und Ordnungen innerhalb einer Gemeinschaft von Bedeutung für die Sozialwissenschaften.

Wenn bestimmte Muster in einem Netzwerk auftauchen, wird gerne getestet ob das Muster zufällig ist. Dies kann gemacht werden indem man dieses Muster in echten Netzwerken mit einer ähnlichen Struktur sucht, aber auch in dem man Zufallsnetzwerke erzeugt, die eine solche Struktur besitzen. Ein geeignete Algorithmenklasse zur Generierung dieser Zufallsnetzwerke sind Monte-Carlo-Markov-Ketten-Algorithmen, die in der Praxis gute Resulate liefern.

Leider ist die Laufzeit der MCMC-Algorithmen mit einem hohen multiplikativen Faktor versehen, den wir in dieser Arbeit genauer untersuchen wollen. Zu diesem Zweck wird ein Algorithmus zur Generierung von Zufallsnetzen entwickelt und durch empirische Tests dessen Korrektheit belegt. Auf dieser Basis wird dann gezeigt, dass der Faktor für spezielle Netzwerke bisher zu hoch gewählt war und er stark mit den Netzen variiert. Für konkrete Fälle wird eine Reduktion des Faktors auf einen Wert von 2 gezeigt, während er vorher bei 100 stand.

Die Ergebnisse sind bisher nur für spezielle Netzwerke gültig, der Algorithmus kann aber für allgemeine Fälle verwendet werden und kann somit als Grundlage für weitere Forschung dienen.

Abstract

The study of graphs is in general an interesting problem, for example it enables to optimize circuits. But also other areas benefit, for example the biological study of concurring DNS-sequences. Another important application area of graph analysis are social networks. Applications are, for example, to find opinion-leaders for improving advertisements on networks or investigating the construction of groups for social analysis research purposes.

If a certain pattern occurs in a network frequently, researchers tend to ask whether this is random. For this they can either take other real world networks and investigate them or take random networks with a similar structure. A class of algorithms to generate these similar random networks are the Monte-Carlo-Markov-Chain-algorithms that provide good results in practice.

Unfortunately the runtime of MCMC-algorithms suffers from a high multiplicative factor, which we will examine in this work. For this purpose we implemented an algorithm to generate random networks and supported by empirical tests that its correct. In addition we showed that the empirically found factors were chosen too high. For specific cases it is possible to reduce the factor from 100 to 2. So far, these results were for specific cases only, but they can serve as a starting point for future research in this direction.

Kapitel 1

Einleitung

1.1 Historie

Seit dem späten 18. Jahrhundert wurden Vorläufer der sozialen Netzwerke identifiziert. Eine der Hauptpersonen dabei war Ferdinand Tönnies, der als einer der ersten die Unterscheidung zwischen einer Gemeinschaft, die durch persönliche, gemeinsame Werte definiert ist, und einer Gesellschaft, die durch formelle Verbindungen definiert ist, festlegte.

Nach einem Tief in der Untersuchung in den ersten Jahrzehnten des 19. Jahrhunderts wurde die Idee wieder aufgenommen. Die Untersuchung von Netzwerken, insbesondere von sozialen Netzwerken, wird seit den 1920er Jahren intensiviert; wurde sie damals noch zu Zwecken der „rassenpolitischen" Analyse genutzt, bis zu den 40er Jahren [23], entwickelte sich ab d'ann der sozio-kulturelle Aspekt dieses Themengebietes stärker. Zwar wurde in den 30er Jahren auch auf diesem Gebiet geforscht, aber nur in kleinen Gruppen wie Klassensälen.

Im Jahre 1954 wurde der Term „Soziales Netzwerk" (engl. *social network*) von J. A. Barnes als ein feststehender Ausdruck verwendet um sich wiederholende Strukturen (engl. *pattern*) in verschiedenen Gruppierungen zu beschreiben, zum Beispiel in Stämmen, Familien oder auch in sozialen Kategorien [28].

Seitdem ging es mit der Untersuchung von sozialen Netzwerken langsam, aber stetig, weiter. Einerseits untersuchte Harrison White mit seiner Studiengruppe an der Harvard University, Department of Social Relations, dieses Themengebiet auf einer sozialen Ebene, während Charles Tilly sich mit sozialen Bewegungen und politischen Netzwerken beschäftigte [28]. Besondere Erwähnung verdient insbesondere Stanley Milgram, der mit der These der *kleinen-Welt-Phänomene* die Untersuchung der sozialen Netzwerke vorantrieb und durch die Idee, dass in den USA jeder Mensch mit jedem anderen über durchschnittlich sechs Verbindungen bekannt sei, ein großes Interesse erweckte, auch wenn dieses Experiment stark durch die Menschen beeinflusst wurde. Auch war die Wiederholung 1970 ein Indiz dafür, dass diese These sich nicht nur in begrenz-

ten Teilen der Bevölkerung anwenden läßt, sondern auch über unterschiedliche Bevölkerungsgruppen hinweg [22, 27]. Durch Duncan J. Watts und Steven H. Strogatz, Ende des zwanzigsten Jahrhunderts [21], wurde das kleine-Welt-Phänomen weiter untersucht. Sie brachten interessante, neue Ergebnisse hervor im Bezug auf bestimmte Effekte wie den Abstand zweier Knoten voneinander. Sie nahmen in der realen Welt vorkommende Netze, zum Beispiel den Graphen der Zusammenarbeit von Schauspielern, und untersuchten sie, schlugen aber auch die genauere Untersuchung von anderen Netzwerken vor, die bisher nur als nicht verbundene Graphen betrachtet wurden. Als Beispiel mag hier die Idee eines Netzwerkes von Sexualpartnern dienen; an diesen könnte man dann das Ausbreitungsverhalten von Sexualkrankheiten untersuchen.

1.2 Motivation

Seit damals hat sich viel getan, die soziale Netzwerkanalyse ist mittlerweile eine eigene Forschungslandschaft geworden; Forscher haben die Wahl komplette Netzwerke zu analysieren, nur spezielle Verbindungen, oder auch personsbezogene Netzwerke (engl. *egocentric networks*) [15]. Das Problem hierbei ist die Datanaquirierung und auf dieser basiert auch meist die Klasse des zu untersuchenden Netzwerkes. Liegen nur von einzelnen Instanzen (Personen, Firmen, etc.) spezielle Daten vor, von anderen ist nur die Verbindung zu ihnen bekannt aber nichts genaueres, so wird meist eine personenbezogene Analyse aufgezogen. Das bedeutet, es wird untersucht wie genau sich diese Instanz in einem Umfeld integriert hat. Dazu entgegengesetzt ist die Idee der kompletten Netzwerke, bei denen alle Daten jedes Individuum vorhanden sind, wie beispielsweise bei einem firmen-internen Netzwerk.

Der gravierende Unterschied zu dem alten Ansatz der sozial-wissenschaftlichen Studien liegt in der Distanzierung von den individuellen Faktoren der Individuen. SNA (engl. *Social Network Analysis*) achtet nicht mehr nur auf die Attribute und Eigenschaften der Individuen, sondern gibt den Beziehungen zwischen ihnen eine größere Beachtung.

Auf dieser Basis kann es nur von Interesse sein die Beziehungsstrukturen zwischen den einzelnen Knoten oder Individuen zu untersuchen. Es wäre aber redundant verschiedene reale Netzwerke zu untersuchen, da Personen die in einem Netzwerk verbunden sind, meist auch in einem anderen Netzwerk verbunden sind. Von daher werden auf den echten Netzwerken basierende Zufallsgraphen genutzt. Unter Beobachtung der sich ergebenden Konfigurationen wird dann festgestellt, ob ein beliebiges Muster, welches im Ursprungsgraphen beobachtet wurde, in den anderen Zufallsgraphen ungefähr genau so häufig auftritt oder ob das Muster im Ursprungsnetz besonders oft, beziehungsweise besonders selten auftritt.

Diese Untersuchungen wurden hauptsächlich von Uri Alon und seinen Mitarbeitern präsentiert [2, 13]. Diese Gruppe untersuchte das Vorkommen von so genannten Motiven (engl. *motifs*), insbesondere beim *Escherichia coli* Bakterium. Ein Motiv entspricht dabei einem besonderen Muster mehrerer

Knoten. Dabei wurden Genregulations-Netzwerke als Basis der Analysen genommen. Als ein besonderes Faktum wies Alon darauf hin, dass Muster, die bei diesem Bakterium vorkommen, sowohl bei anderen Bakterien als auch bei höher entwickelten Organismen auftreten.

Diese Untersuchungen wurden bisher überwiegend auf Bakterien und ähnlichen Organismen durchgeführt um diese Motive zu erkennen und die Häufigkeit oder Regelmäßigkeit festzustellen, aber durch die Einführung von Gewichten der Kanten oder Knoten könnte man, analog zu Proteinstrukturen und deren Verformungsenergie [3], die wahrscheinlichen Erkrankungen oder Rekombinationsmöglichkeiten von Genen bestimmen.

Ebenfalls ist diese Idee gut für die Untersuchung von Sozialen Netzwerken zu nutzen um besondere Strukturen zu erkennen. Während früher die Idee war nach besonderen Strukturen in den Netzwerken zu suchen, deren generelle Struktur bekannt war, aber nicht deren Größe - die bekannten Algorithmen hierfür benötigen entweder exponentielle Zeit oder sind als *greedy Algorithmen* einzustufen - kann mit dieser Idee ein Motiv im vorhandenen Sozialen Netzwerk markiert werden und durch eine Menge an Kantentauschen herausgefunden werden, ob dieser Subgraph zufällig ist oder nicht. Auf diese Weise kann der exponentielle Aufwand durch einen polynomiellen Aufwand verbessert werden.

Durch die Dynamik der Meinungsbildung und der Verbreitung von Meinungen und Mitteilungen über soziale Netzwerke, wie zum Beispiel twitter[1], facebook[2], wird diese Analyse auch für Werbefirmen und Politik interessanter. Bisher gab es leider nur wenige Studien auf diesem Bereich [29]; Farhad Manjoo stellte beispielsweise in einer Untersuchung fest, dass afroamerikanische Mitbürger auf twitter.com sogenannte *blacktags* kreieren würden - das bedeutet, dass sich Themen unter der negriden Bevölkerung, nicht nur lokal beschränkt sondern auf der ganzen Welt, sehr schnell verbreiten und in die Liste der *trending topics* gelangen, wo sie dann erst von der kaukasischen Bevölkerung übernommen werden [11]. Dieser Trend scheint eine Besonderheit des Netzwerkes twitter zu sein. Er könnte für eine größere Verbindung unter den Negriden zu sprechen, als negrid-kaukasische Bekanntschaften aber auch dafür dass sie mehr dazu neigen sich mit neuer Technologie Gehör zu verschaffen zu wollen (vgl [19]).

Die vorliegende Arbeit flieder sich wie folgt: Es wird eine kleine Einführung in die Graphentheorie folgen, im Anschluss werden die in der Entwicklungsphase programmierten Algorithmen im Wesentlichen erläutert, wobei auf Beweise dass sie funktionieren verwiesen wird. Im dritten Abschnitt werden die Testfälle untersucht und danach wird erklärt, wieso der verwendete Algorithmus die gewünschten Merkmale aufweist. Zum Schluss wird noch ein Ausblick auf weitere Forschungsmöglichkeiten gegeben und ein abschließendes Urteil gezogen.

[1] http://www.twitter.com
[2] http://www.facebook.com

Kapitel 2

Einführung in die Graphentheorie

Es folgt eine Erklärung einiger Schlagworte aus der Graphentheorie, mit denen man im Verlauf dieser Arbeit gearbeitet wird.

2.1 Terminologie

Knoten Als Knoten wird eine Einheit in einem Graphen bezeichnet.

Kante Eine Kante ist eine Verbindung zwischen zwei Knoten. Man spricht von einer gerichteten Kante $e = (u, v)$, wenn sie nur in eine Richtung verläuft. In diesem Fall wird sie durch einen Pfeil dargestellt.

Schleife Eine Schleife entspricht einer Kante, deren Startknoten und Endknoten identisch ist (siehe Abbildung 2.1).

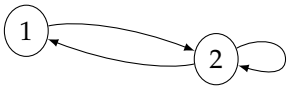
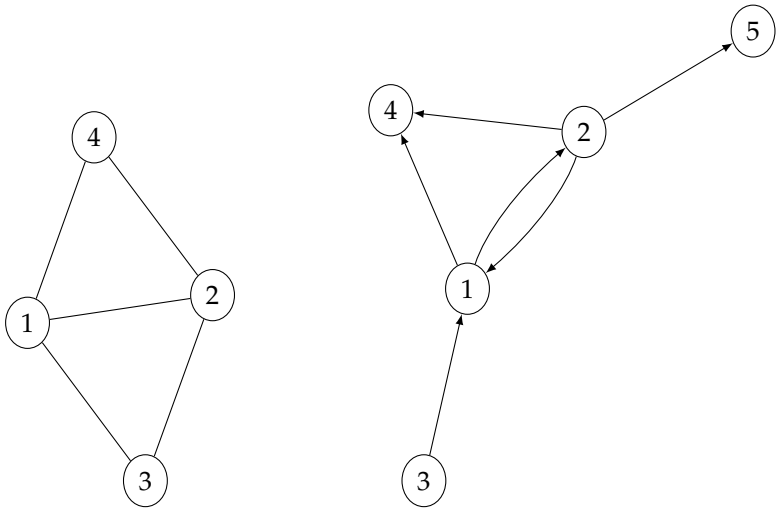


Abbildung 2.1: Gerichteter Graph mit Schleife

Graph

Ungerichteter Graph Ein ungerichteter Graph $G = (V, E)$ (siehe Abbildung 2.2(a)) besteht aus einer Menge von Punkten, genannt Knoten, V (engl. *vertices*) und einer Menge von Linien, genannt Kanten, E (engl. *edges*), die die Punkte miteinander verbinden; also gilt, dass $E \subseteq \{\{u, v\} \mid u, v \in V\}$. Erlaubt der Graph keine Schleifen so gilt $E \subseteq \{\{u, v\} \mid u\, v \in V \wedge u \neq v\}$.



(a) Ungerichteter Graph (b) Gerichter Graph

Abbildung 2.2:

Gerichteter Graph Ein gerichteter Graph $G = (V, E)$ besteht aus einer Menge von Knoten V und einer Menge von gerichteten Kanten, E, die die Knoten miteinander verbinden; also gilt, dass $E \subseteq \{(u, v) \mid u, v \in V\}$. Erlaubt der Graph keine Schleifen so gilt $E \subseteq \{(u, v) \mid u, v \in V \wedge u \neq v\}$. Ein Beispielsgraph ist in Abbildung 2.2(b) zu sehen.

Einfacher Graph Ein Graph $G = (V, E)$ wird als einfacher Graph bezeichnet, wenn er keine parallelen Kanten und keine Schleifen enthält. Im ungerichteten Graph bedeutet dies, dass zwei Knoten maximal durch eine Kante direkt verbunden sind und es keine Kante zu sich selbst gibt. Als ein Beispiel hierfür kann Abbildung 2.2(a) herangezogen werden. In einem gerichteten Graphen bedeutet dies, dass es zwischen zwei Knoten maximal ein Pfeil in jede Richtung geben darf und keiner der Knoten ein Pfeil auf sich selbst besitzt.

Multigraph Ein Multigraph $G = (V, E)$ unterscheidet sich von einem einfachen Graphen durch die Erlaubnis sowohl von Schleifen als auch von mehr als einer direkten Kante zwischen zwei Knoten. Als einfaches Beispiel siehe Abbildung 2.3

Gewichteter Graph Ein Graph $G = (V, E)$ ist ein gewichteter Graph, wenn die Elemente e der Kantenmenge E eine Wertigkeit zugeschrieben bekommen. Ein Beispielsgraph befindet sich in Abbildung 2.4.

Knotengrad Der Grad eines Knoten beschreibt, wie viele Kanten zu dem Kno-

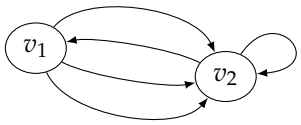


Abbildung 2.3: Ein Multigraph, erkenntlich an mehreren Kanten zwischen v_1 und v_2

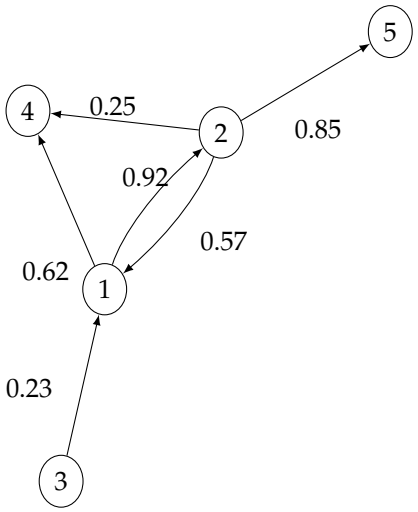


Abbildung 2.4: Gewichteter Graph

ten inzident sind, also wie viele Kanten mit dem Knoten verbunden sind. Bei ungerichteten Graphen wird der Knotengrad des Knoten v_i meist mit d_i bezeichnet, bei gerichteten Graphen benötigt man als ausreichende Beschreibung den Eingangsgrad und den Ausgangsgrad des Knoten v_i, dargestellt durch $\left(d_i^+, d_i^-\right)$. In Abbildung 2.4 hat Knoten 3 den Ausgangsgrad 1, da nur die Kante $(3, 1)$ von ihm ausgeht und den Eingangsgrad 0, da keine Kante hineingeht.

Sequenz Eine Sequenz $S = (d_1, d_2, \ldots, d_n)$ ist eine Menge von natürlichen Zahlen, welche die Knotengrade beschreibt. Der i-te Knoten hat genau den Grad d_i, für den Fall dass der Graph ungerichtet ist. Sollte ein gerichteter Graph mit einer Sequenz erschaffen werden, so gibt es zwei Möglichkeiten dies zu ermöglichen: als erstes kann man anstatt der Zahl d_i immer ein 2-Tupel als Eintrag angeben, so dass Eingangsgrad und Ausgangsgrad eines Knoten zusammenstehen $\left(\left(d_i^+, d_i^-\right)\right)$. Alternativ kann man es auch als zwei Sequenzen S_1 und S_2 mit $|S_1| = |S_2|$ angeben, so dass der i-te Eintrag in S_1 zusammen mit dem i-ten Eintrag in S_2 einen Knoten definieren. Der Graph in Abbildung 2.4 hat demnach die Sequenz

$$S = ((2,\,1)\,,(2,\,1)\,,(1,\,0)\,,(0,\,2)\,,(0,\,1))$$

Konfiguration Eine Konfiguration eines Graphen $G = (V,\,E)$ ist genau eine Darstellung eines Graphen mit genau $|V|$ Knoten und genau $|E|$ Kanten. Jede Konfiguration eines Graphen G steht für einen anderen Graphen mit genau der selben Kanten- und Knotenzahl wie G.

2.2 Spezielle Graphen

2.2.1 Zufallsgraph

Ein Zufallsgraph ist ein Graph, dessen Kanten zufällig erzeugt werden. Bekanntere Modelle hierfür sind zum Beispiel der *Erdös-Renyi-Graph* $G\,(n,\,p)$ (Beispiel siehe Abbildung 2.5) bei welchem die n Knoten mit Wahrscheinlichkeit $0 \leq p \leq 1$, p konstant, miteinander verbunden werden. Allgemeiner kann man den Zufallsgraphen wie eine normale Zufallsvariable betrachteten und die Erstellung des Zufallsgraphen als ein großes Urnenexperiment, bei dem die Kugeln den einzelnen Graphen entsprechen. Diese Sicht ist natürlich nur eine Veranschaulichung der Erstellung eines Zufallsgraphen mit n Knoten. Dennoch ist bei dieser Beschreibung ein sehr wichtiger Punkt implizit enthalten, nämlich die Uniformität einen Graphen zu ziehen. Jede Kante wird mit der gleichbleibenden Wahrscheinlichkeit p, $0 \leq p \leq 1$, erzeugt, was bedeutet dass die eigentliche Erstellung des Graphen für jeden Knoten $p \cdot (n-1)$ - da keine Selbkanten erlaubt sind - unabhängige Zufallsexperimente darstellt, also insgesamt $n \cdot [p \cdot (n-1)]$ von einandner unabhängige Zufallsereignisse in einem gerichteten Graphen. Zufallsgraphen sind am Ende aber nicht geeignet um die Strukturen sozialer Netzwerke darzustellen, da sie eher einen exponentiellen Verteilungsgrad besitzen [16] und reale Netzwerke, wie soziale Netzwerke oder das Internet eher zu polynomiellen Zusammenhängen neigen. Auch andere Eigenschaften von realen Netzwerken treten in diesen Zufallsgraphen nicht auf, zum Beispiel lokale Verbindungshäufigkeiten (Clustering-Koeffizient) [10].

Aus diesem Grund werden wir uns von dieser Art Zufallsgraphen zu erzeugen abwenden und uns einer anderen Art der Erzeugung von Netzwerken zuwenden. Diese Netzwerke haben andere Attribute, sie werden *skalenfreie Netzwerke* genannt.

2.2.2 Skalenfreies Netzwerk

Während bei Zufallsgraphen keine Knoten mit sehr hohem Verbindungsgrad im Vergleich zu den anderen Knoten existieren, ist es in echten Netzwerken häufig so, dass es Knoten, beziehungsweise Häufungspunkte gibt, die mit vielen anderen Knoten verbunden sind. Als Beispiel hierfür sei das WWW genannt, in dem Google[1] einer der Hauptknoten ist, da die Suchmaschine auf

[1] http://www.google.com

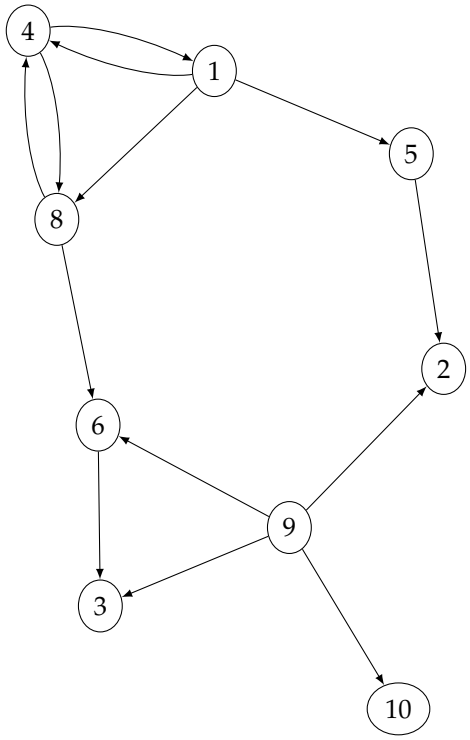


Abbildung 2.5: Mit dem Erdös-Rényi-Algorithmus erstelltes Netz.

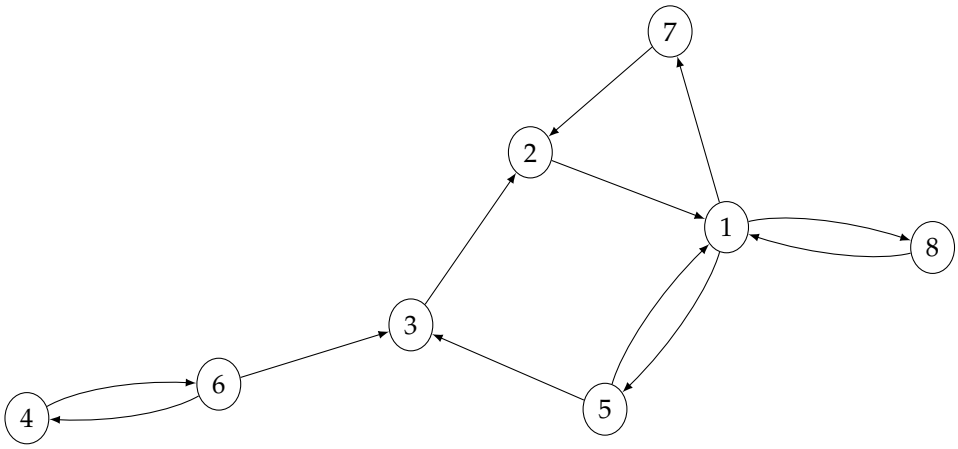


Abbildung 2.6: Beispiel eines skalenfreien Netzwerkes, generiert aus der Sequenz $S = ((3,3),(2,1),(2,1),(1,1),(1,2),(1,2),(1,1),(1,1))$

sehr viele andere Seiten verweist und auch von sehr vielen anderen Seiten verlinkt ist, andere hingegen aber nur sehr wenige Verlinkungen aufweisen. Andere Beispiele sind soziale Netze wie facebook, twitter und ähnliche. Auf twitter gibt es als besonders prägnantes Beispiel mindestens einen Häufungspunkt, der nur eine ausgehende Kante hat (*following 1 person*), aber über eine Millionen eingehende Kanten besitzt (*followers 1 722 295 persons*) [17].

Diese Art von Graphen basiert nicht auf einer gegebenen Wahrscheinlichkeit sondern auf einem Potenzgesetz $P \propto k^{-\gamma}$, γ ist eine einheitslose Zahl. Mittels diesem können Knotengrade bestimmt werden; es existieren Algorithmen um diese Netzwerke zu erzeugen, als Beispiel sei genannt der Algorithmus von *Barabási und Albert* [9]; diese agieren häufig nach dem Prinzip des *preferential attachment*, bei dem sich neu hinzukommende Knoten während des Generierungsprozesses lieber mit bereits stark verknüpften Knoten verbinden.

In dieser Arbeit wird nicht wie bei dem Algorithmus von *Barabási und Albert* mit einem kleinen Netzwerk anfangen und daraus dann durch Hinzufügen neuer Knoten und Kanten das Netzwerk vergrößert bis ein Schwellwert erreicht ist. Stattdessen wird vorgeben, wie viele Knoten vorhanden sein sollen und dem Potenzgesetz folgend die Knotengrade dazu generiert um daraus eine Knotengradsequenz zu konstruieren. Jene wird dann an Algorithmen weitergereicht, welche aus dieser Graphen generiert. Diese Unterart der Nutzung des Potenzgesetzes ist als Modellierungsart für skalenfreie Netzwerke bekannt, hat aber den Nachteil dass das Wachstum der Knotenmenge entfällt. Die entstehenden Graphen genügen dennoch dem Potenzgesetz, da die Knotengerade danach generiert sind, und erfüllen daher die an sie gestellten Ansprüche. Für Abbildung 2.6 wurde aus einer Sequenz S mit dem von Barabáasi und Albert vorgeschlagenem Algorithmus ein Netzwerk generiert. Es ist nur

ein kleines Beispiel, dennoch erkennt man, dass Knoten 1 sehr viele Verbindungen aufweißt, während andere, wie Knoten 4, wenige Verbindungen aufweisen.

Es existiert noch ein weiteres häufig genutztes Modell zur Erzeugung von Zufallsgraphen, das Konfigurationsmodell.

2.2.3 Konfigurationsmodell

Dieses Modell generiert aus einer Sequenz $S = (d_1, \ldots, d_n)$ einen Graphen $G = (V, E)$. Dafür wird für jeden der n Einträge in S d_i-mal die Nummer des Knoten, i, erzeugt und einer Menge W hinzugefügt, quasi als offene Andockstellen für Kanten. Danach werden zufällig zwei beliebige Andockstellen (engl. *stubs*) herausgenommen und diese miteinander verbunden. Das Problem hierbei ist, dass auf diese Art Multikanten und Schleifen entstehen können. Schleifen entstehen nur selten bei genügend großen Graphen [14], dennoch können Multikanten auch bei großen Graphen entstehen.

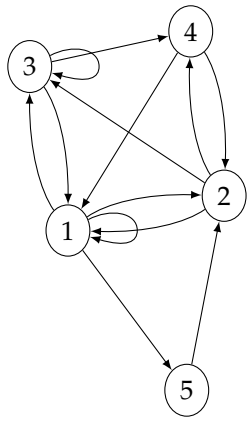


Abbildung 2.7: Graph, erstellt nach dem Konfigurationsmodell aus der Sequenz $S = ((4,4), (3,3), (3,3), (2,2), (1,1))$

Dieses Modell hat im Gegensatz zu dem Modell der Zufallsgraphen von Erdös-Rényi den Vorteil exakt die gewünschte Gradsequenz zu generieren [5], was für viele Untersuchungen von Vorteil sein kann.

Dennoch hat dieses Modell starke Nachteile, wenn man einfache Graphen untersuchen will. Dadurch, dass die Verbindungen uniform erzeugt werden, kann es passieren, dass sich Knoten mehrfach verbinden oder dass Knoten sich mit sich selbst verbinden. So ist in Abbildung 2.7 eine Sequenz angegeben, für die bekannt ist, dass kein einfacher Graph existiert, der die Sequenz realisiert. Für diesen Fall wurde das Konfigurationsmodell erweitert zu einem *erased configuration model*. Bei diesem wird das Netzwerk entsprechend Algorithmus 1 generiert. Danach werden aber nur Kanten beibehalten, wenn zwei verschiedene Knoten durch diese verbunden sind. Sollten mehere Kanten zwei Kno-

Algorithmus 1 Konfigurations-Graph

Require: Sequence $S = (d_1, d_2, \ldots, d_n)$
  $W = \emptyset$
  for $i = 1$ to n **do**
    $insert\left(W, \{d_i\}^{d_i}\right)$
  end for
  $V = (d_1, d_2, \ldots, d_3)$
  while unconnected element in W **do**
    $e = connectRandomElements\,(W)$
    $addToEdges\,(e)$
  end while
  $return\, G = (V, E)$

ten mit einander verbinden, so werden alle überzähligen Kanten gelöscht [18]. Dieser Vorgang ändert aber automatisch die Sequenz, die generiert wurde [1] und bisweilen werden so auch nicht als einfacher Graph realisierbare Sequenzen realisiert und als ein solcher präsentiert. Nicht nur daraus resultiert, dass die Graphengenerierung aller Graphen nicht uniform ist [5], aber dass das *erased configuration model* nicht mehr uniforme Ergebnisse liefern kann, da durch die Löschung von Kanten häufiger gleiche Netzwerke hervorgebracht werden, wird nur noch offensichtlicher.

Zu beachten ist weiterhin, dass bei diesem Modell ebenfalls nicht verbundene Graphen entstehen können. Daher achten andere Forscher (vgl. [20, 7, 2, 13]) darauf, dass ihre Netzwerke immer verbunden bleiben und führen dazu einen *connectivity-check* durch. Wenn das generierte Netzwerk verbunden ist, so vergrößern sie die Zeitspanne, bis sie das nächste mal überprüfen ob das Netz verbunden ist. Sie versuchen in ihren Algorithmen eine geeignete Schrittweite zu finden, wann sie noch einmal prüfen sollten, ob ihr Netz verbunden ist.

Kapitel 3

Monte-Carlo-Markov-Ketten-Algorithmus

In diesem Abschnitt wird der Algorithmus erklärt und eine Laufzeitanalyse geben; diese wird den worst-case betrachten um eine gute Abschätzung dafür zu erhalten was man erreichen kann.

3.1 Prinzip

Die algorithmische Idee (siehe Algorithmus 2), auf der das Verfahren der Generierung von Zufallsgraphen basiert ist folgende: man erhält eine Knotengradsequenz als Eingabe, generiert hieraus einen Graphen und durch das Verändern von Kanten, also ihrem Ziel oder ihrer Quelle, erhält man daraus einen anderen Graphen.

Algorithmus 2 Generierung von Zufallsgraphen

Require: $S = (s_1, s_2, \ldots, s_n)$
  $G = generateGraph\,(S)$
  while condition **do**
    $G = changeGraph\,(G)$
  end while

Diese Idee funktioniert sowohl auf ungerichteten Graphen als auch auf gerichteten, im Kontext dieser Arbeit wird der Fokus aber weiterhin auf gerichteten Graphen liegen.

Bei der Generierung des Graphen ist es nunmehr wichtig darauf zu achten, dass nicht hierbei schon das Kriterium der Einfachkeit verletzt wird. Daher wird der Havel-Hakimi-Algorithmus verwendet, welcher unter bestimmten Bedingungen (siehe [6]) garantiert einen einfachen Graphen zu generieren.

3.2 Havel-Hakimi-Algorithmus

Beschreibung

Die Eingabe des Havel-Hakimi-Algorithmus' besteht im Kontext dieser Arbeit aus einer Anzahl Knoten n, und zwei geordneten Mengen aus Integer-Werten, der Eingangsgrad-Menge und der Ausgangsgrad-Menge; geordnet sind sie so, dass der i-te Eintrag in der Eingrad-Menge mit dem dazugehörigen i-ten Eintrag in der Ausgrad-Menge die Konfiguration eines Knoten ausreichend spezifizieren.

Der Algorithmus (siehe Algorithms 3) basiert darauf, dass er einen beliebigen Eintrag i aus der Eingangsgrad-Menge auswählt und die Grade der größten Einträge in der Aus-gangsgrad-Menge um jeweils eins reduziert und dies so oft durchführt wie es die natürliche Zahl i vorgibt. Die dabei entstehenden Paarungen ergeben die Kanten des Graphen. Diese werden gespeichert für die spätere Verwendung. Danach muss die Menge der Ausgangsgrade sortiert werden, ansonsten wird beim nächsten Durchlauf mit einem neuen Eingangsgrad nicht der Knoten mit dem größten verbleibenden Ausgangsgrad gewählt. Alternativ könnte man dies auch über eine Maximumsfunktion machen, was aber keine Ersparnisse einbringen würde. Dieser Vorgang wird so oft wiederholt, bis alle Eingangsgrade vollständig genutzt wurden.

Dabei ist anzumerken, dass es einige Vorraussetzungen für diesen Algorithmus gibt; so muss die Summe der Eingangsgrade gleich der Summe des Ausgangsgrade sein, damit am Ende keine losen Kanten existieren. Auch darf keiner der Knoten einen negativen Grad haben.

Laufzeitanalyse

Die Erstellung der Knotenobjekte benötigt eine Laufzeit von $\mathcal{O}(n)$, denn jeder der n Knoten wird einzeln initialisiert.

Der Vorgang des Auswählens und Verbindens mit anderen Knoten wird für jeden Knoten ausgeführt; bei einer Anzahl von n Knoten bedeutet dies dann den Vorfaktor $\mathcal{O}(n)$. In der Schleife über die Knoten wird bei jedem neuen Betreten der Schleife die Menge der Eingangsrade sowie die Menge der Ausgangsgrade neu sortiert. Hier kommt es nun auf den Sortiertalgorithmus an; aber da für Standardsortieralgorithmen eine bekannte Schranke von $\mathcal{O}(n \cdot \log(n))$ existiert und in dieser Arbeit auch ein solcher Sortieralgorithmus verwendet wurde, wird dieser in die Rechnung einfließen. Auch wenn eigentlich immer nur ein Teil der Grade sortiert wird, da immer weniger freie Stellen zum Einhängen von Kanten vorhanden sind, wird dennoch als worstcase Abschätzung mit diesem Wert gerechnet.

Das Aufspannen von Kanten benötigt zwei Kontrollen.

1. dass nicht bereits eine Kante zwischen den Knoten existiert zur Vermeidung von Multikanten

Algorithmus 3 Havel-Hakimi

Require: $inDegreeSequence = (i_1, i_2, \ldots, i_n)$
Require: $outDegreeSequence = (j_1, j_2, \ldots, j_n)$
  $Nodes = \emptyset$
  $k = 1$
  while $inDegreeSequence \neq \emptyset$ **do**
    $inDegree = inDegreeSequence[k]$
    $outDegree = outDegreeSequence[k]$
    $Nodes.append\,(create\_node(inDegree, outDegree))$
    $inc(k)$
  end while
  for all Nodes **do**
    reverse_sort $\{inDegree\}$ Nodes
    sort $\{outDegree\}$ Nodes
    $k = 0$
    while current_node has free source slots **do**
      if $current\_node \neq Nodes[k]$ && $current\_node\ has\ free\ source\ slots$ **then**
        $create\_edge(current\_node, Nodes[k])$
        $inc(k)$
      end if
    end while
  end for

2. dass die zwei Knotenobjekte nicht den selben Knoten beschreiben zur Vermeidung von Selbstzyklen

Die Kontrollen selbst erfolgen jeweils in $\mathcal{O}(1)$, das Erstellen der Kante liegt ebenfalls in dieser Klasse. Die Häufigkeit dieses Vorganges kann der Einfachkeit halber auch mit $\mathcal{O}(n)$ abgeschätzt werden, auch wenn maximal $n-1$ die obere Schranke darstellen sollte. Ansonsten würde es mindestens eine Selbst-Kante geben oder eine doppelte Verbindung zwischen zwei Knoten.

Also kann die maximale Laufzeit dieses Algorithmus wie folgt abgeschätzt werden:

$$\begin{aligned} T_{HH}(n) &= \mathcal{O}(n) + \mathcal{O}(n) \cdot [\mathcal{O}(n \cdot \log(n)) + \mathcal{O}(n) \cdot (3 \cdot \mathcal{O}(1))] \\ &= \mathcal{O}(n) + \mathcal{O}(n) \cdot [\mathcal{O}(n \cdot \log(n)) + \mathcal{O}(n)] \\ &= \mathcal{O}(n) + \mathcal{O}(n) \cdot \mathcal{O}(n \cdot \log(n)) \\ &= \mathcal{O}(n) + \mathcal{O}\left(n^2 \cdot \log(n)\right) \\ &= \mathcal{O}\left(n^2 \cdot \log(n)\right) \end{aligned}$$

Bemerkung: Von Peter Erdös, István Miklós und Zoltán Toroczkai [6] wird eine andere Laufzeitsschranke für diesen Algorithmus angegeben, was abhängig ist von der Implementierung; so gibt zum Beispiel Sven Kosub [10] eine

Laufzeit von $\mathcal{O}((n+m) \cdot \log(n))$ an, da der Algorithmus dort über Queues implementiert wurde. Da aber, wenn man wieder mit der worst-case-Analyse herangeht, die Anzahl m der Kanten, die dort mit $m = \frac{1}{2} \cdot \sum_{i=1}^{n} d_i$ angegeben ist, mit $\frac{n \cdot (n-1)}{2}$ für einen ungerichteten Graphen nach oben abgeschätzt werden kann, wird auch dieser Algorithmus nicht mit einer besseren worst-case-Laufzeit auskommen als oben angegeben.

3.3 Markov-Ketten-Monte-Carlo-Algorithmus

Beschreibung

Für die Realisierung des Markov-Ketten-Monte-Carlo-Algorithmus wird als Eingabe ein bereits vorhandenes Netzwerk genutzt. Von diesem wird die Anzahl der Kanten m, multipliziert mit einer Konstanten q, als Anzahl der Iterationen abgeschätzt und dann für jede Iteration ein Kantentausch ausgeführt, bei dem verschiedene Dinge beachtet werden müssen. Wie auch beim Havel-Hakimi-Algorithmus sollten hier bei keine Selbst-Kanten entstehen, keine Multikanten und es sollten auch keine unnötigen Tauschoperationen durchgeführt werden. Eine unnötige Tauschoperation wäre hierbei durch Abbildung 3.1 gegeben.

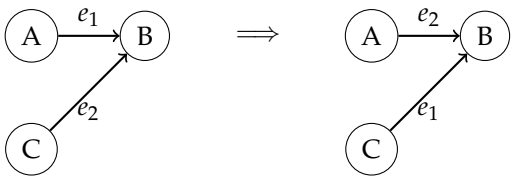


Abbildung 3.1: Kantentausch-Beispiel; an dieser Stelle unnötig da der resultierende Graph dem Ursprungsgraph entspricht

Dies wäre ein unötiger Tausch zweier Kanten, da sich daraus kein neuer Graph ergibt, wenn man die Kanten als identisch und nur die miteinander verbundenen Knoten als Referenzpunkte betrachtet. Ein sinnvoller Kantentausch wäre dem entsprechend in Abbildung 3.2 sichtbar.

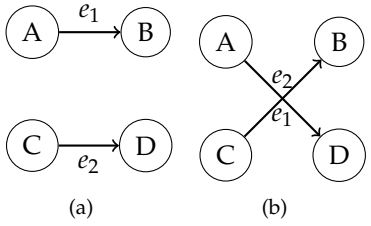


Abbildung 3.2: korrekter Kantentausch

Wenn ein Kantentausch nicht sinnvoll ist, da sich keine Änderung ergibt, wird er nicht ausgeführt; dennoch zählt der Versuch diesen Kantentausch auszuführen als ein Schritt im Algorithmus um die Veränderung des Graphen nicht zu erzwingen. Denn in einem einfachen Graphen gibt es eine maximal erlaubte Kantenzahl, $\frac{n \cdot (n-1)}{2}$, die Nicht-Beachtung von Selbstkanten und Multikanten, beziehungsweise unnötigen Tauschs würden diese Zahl aber verändern und die Ergebnisse somit stark verfälschen.

Weiterhin sind Kantentausche verboten, bei denen Schleifen entstehen würden. Auch in diesem Fall (Abbildung 3.3) wird der Tausch verworfen und ein weiterer Schritt gezählt.

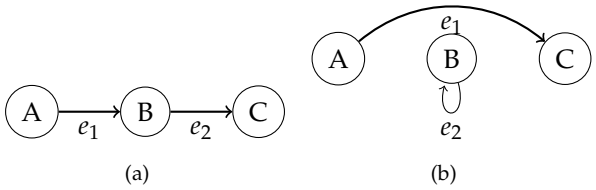


Abbildung 3.3: Verbotener Kantentausch zur Schleife

Diese Kantentausche werden nun insgesamt $q \cdot m$ mal mit zufälliggewählten Kanten durchgeführt und das jeweils resultierende Netzwerk wird zurückgegeben[1]. Sollte ein Kantentausch ungültig sein, so wird der Schritt dennoch gezählt als eine erfolgreiche Netzwerkgenerierung und der Tausch nicht durchgeführt. Da ein Großteil der Kantentausche bei dünnen Netzwerken eh zu ungültigen Netzwerken führen würde, also Netzwerken mit Schleifen oder Multikanten, ist dieses Vorgehen auch legitim.

Algorithmus 4 Monte-Carlo-Markov-Chain

Require: $G = (V, E)$
Require: q
  $m = |E|$
  $numberIterations = q \cdot m$
  while numberIterations > 0 **do**
    swapEdges
    $numberIterations = numberIterations - 1$
  end while

[1]Zu Beginn war noch ein Verbundenheits-Test enthalten; dieser wurde aber herausgenommen, was damit begründet ist, dass es auf die Uniformität des Algorithmus und damit auch die Erreichbarkeit aller generierbaren Netze ankommt.

Laufzeitanalyse

Da ein Netzwerk die Eingabe ist, wird die Erzeugung des Netzes nicht mit in die Analyse hineingerechnet.

Die Anzahl Iterationen des Algorithmus wird, wie oben beschrieben, über eine Konstante[2] q und die Kantenzahl m festgelegt, das bedeutet einen Vorfaktor von $\mathcal{O}(q \cdot m)$. Und das ist auch der Hauptaufwand, der die gesamte Laufzeit des Algorithmus bestimmt, denn der Kantentausch bedarf formal gesehen nur eines konstanten Aufwandes, den der Tests und des Kantentausch.

Es läuft also für den Markov-Ketten-Monte-Carlo-Algorithmus also auf eine Laufzeit von $\mathcal{O}(c \cdot q \cdot m) = \mathcal{O}(q \cdot m) = \mathcal{O}(m)$ hinaus.

3.4 Vorteile gegenüber dem Konfigurationsmodell

Beim Einsatz des Havel-Hakimi Algorithmus kann es nicht passieren, dass Multikanten oder Schleifen entstehen, da Havel-Hakimi abbricht, wenn andere Knoten zu wenige freie Andockstellen für die Kanten haben. Ebenso können beim Einsatz des MCMC-Algorithmus keine Multikanten oder Schleifen entstehen aus oben beschriebenen Gründen. Das spricht dafür, dass aus einer Sequenz eher mit Havel-Hakimi ein Netzwerk generiert werden sollte welches dann mittels MCMC-Algorithmus verändert wird, als dass aus der Sequenz verschiedene Netzwerke generiert werden, da es wie gesagt nicht garantiert ist, dass die Sequenz bei der Generierung mit dem Konfigurationsmodell erhalten bleibt. Ein weiterer Vorteil ist, dass eine Markov-Kette, die bestimmte Eigenschaften erfüllt, eine uniforme Verteilung darstellt. Nach Viger und Latapy [20] ist die Markov-Kette für diesen Algorithmus irreduzibel, symmetrisch und aperiodisch, was bedeutet, dass sie genau eine stationäre Verteilung besitzt. Nach [25] ist dies eine Bedingung dafür, dass man von jedem Punkt in der Markov-Kette zu jedem anderen Punkt mit einer festen Wahrscheinlichkeit gelangt. Nach [20] konvergiert die Markov-Kette daher zu einer uniformen Verteilung.

Ob der Algorithmus wirklich uniform sampelt wird in Kapitel 5 geklärt.

[2] wie konstant dieses Konstante ist, siehe 5.3

Kapitel 4

Statistische Evaluation

Zu Beginn wurde ein Toy-Netzwerk für die Evaluation des Havel-Hakimi Algorithmus und des Markov-Ketten-Monte-Carlo-Algorithmus genommen, da auf diese Weise ein direkter Verlgeich zwischen dieser Implementierung und einem Paper von Uri Alon [12] gezogen werden konnte. Danach wurde der Test auf verschiedene Netzwerke aus diesem Paper angewendet um Vergleichswerte zu erhalten und um zu sehen, ob sich ein Unterschied ergibt. Dabei muss erwähnt werden, dass für diese Evaluationen nicht die Netzwerke der Gruppe von U. Alon genommen werden konnten, da die verwendeten Testnetzwerke mehr Motive enthielten; da der Unterschied aber nicht sehr groß ist wird der Wert der angegeben ist als Näherungswert genommen [12, 13].

4.1 Toy-Netzwerk

Das Basis-Netzwerk besteht aus 12 Knoten; einer hat 10 Ausgangsgrade aber keine Eingangsgrade, einer hat 10 Eingangsgrade aber keine Ausgangsgrade, der Rest hat als Eingangs- und als Ausgangsgrad je 1.

Das linke Netz in Abbildung 4.1 ist das Spezial-Netzwerk, das rechte ist eine der Konfigurationen die durch Vertauschung von Kanten entstehen kann. Insgesamt können auf diese Weise nur 91 verschiedene Netzwerke entstehen, wie sich leicht nachzählen lässt.

In einem Test mit 100000 Simulationen ergab sich, dass die Netzwerke alle ungefähr gleich verteilt auftauchen mit einer geringen Schwankung von maximal 0,06 Prozent, was innerhalb der Messtoleranz liegt. Also produziert der Markov-Ketten-Monte-Carlo-Algorithmus, angewandt auf ein mit einem Havel-Hakimi-Algorithmus generiertem Netzwerk, eine uniforme Verteilung der verschiedenen Netzwerke in einer effizienten Laufzeit von durschnittlich 10 Sekunden pro 1000 Iterationen.

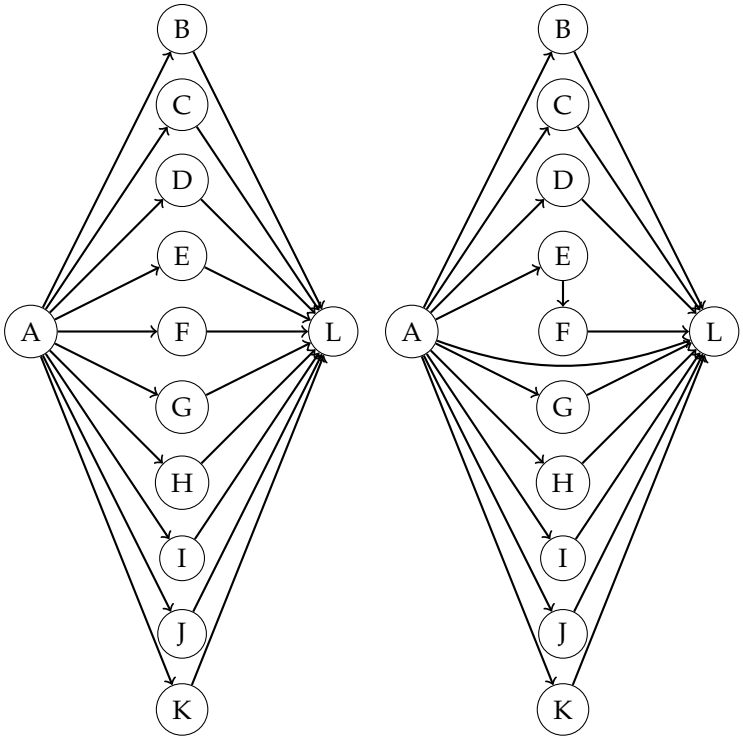


Abbildung 4.1: (a) ist die Darstellung des Startnetzwerkes (Spezialkonfiguration); (b) ist eine der neunzig anderen Konfigurationen

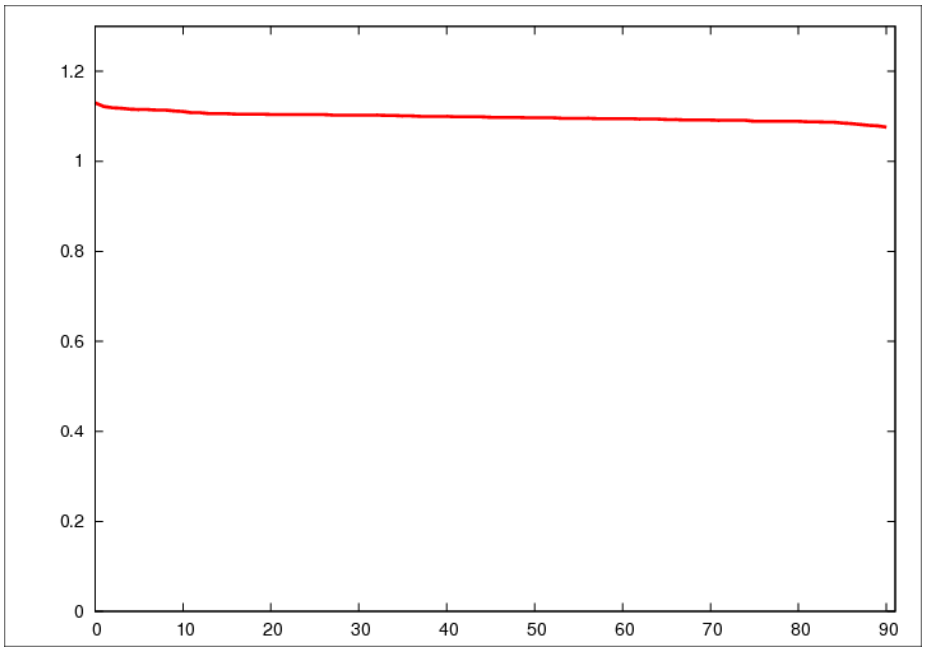


Abbildung 4.2: Verteilung der prozentualen Häufigkeit des Auftauchens der verschiedenen Konfigurationen des Toy-Netzwerkes

4.2 Motiverkennung auf realen Netzwerken

Als Faktor, nachdem man die Netzwerkgenerierung unterscheidet, wurden die $feed - forward\ loop$s gewählt. Als ein solcher wird eine Konfiguartion bestehend aus drei Knoten bezeichnet, bei denen Knoten A eine Kante zu B hat, B eine Kante zu C, A aber auch eine direkte Kante zu C hat (siehe Abbildung 4.3).

Die Motiverkennung wurde erst für reale Netzwerke angewandt, da sich im Toy-Netzwerk eine Messung solcher Konfigurationen nicht lohnen würde.

Als nächstes mussten einige Netzwerke ausgewählt werden, deren Motivzahlen ungefähr die Werte der Gruppe um Alon decken [12, 13], um einen Vergleichswert zu haben; es war nicht möglich exakt diese Netzwerke zu finden, aber für das Gen der *Saccharomyces cerevisiae*, auch Bierhefe genannt, und für das *Escherichia coli* Bakterium konnten Netzwerke gefunden werden, die sowohl in Knoten als auch in Kanten Anzahl mit den Netzwerken aus [12] übereinstimmten nur nur je 3 beziehungsweise 2 *feed-forward loops* mehr hatten. Es wäre noch wünschenswert gewesen das Neuronen-Netzwerk des Fadenwurmes *Caenorhabditis elegans* in einer ähnlichen Konfiguration zu finden, aber leider war es nicht möglich eine entsprechende Darstellung zu finden.

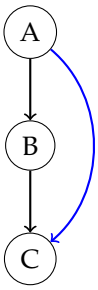


Abbildung 4.3: Die Kante von A nach C entspricht in dieser Konfiguration der sogenannten feed-forward Kante des Graphen;

	E. COLI			YEAST		
iterations	mean	std. dev.	Z-value	mean	std. dev.	Z-value
1000	7.79	3.14	10.89 (10.26)	14.35	4.13	14.2 (11.54 | | 13.47)
10000	7.65	3.08	11.15 (10.5)	14.54	4.14	14.2 (11.46 | | 13.39)
100000	7.66	3.11	11.04 (10.4)	14.58	4.19	13.94 (11.31 | | 13.22)

Tabelle 4.1: Mittelwert und Standardabweichung der Vorkommen des *feed-forward loop* basierend auf den genannten Netzwerken. Der Z-Wert gibt an wie groß der durchschnittliche Unterschied zwischen den generierten Netzwerken und den jeweiligen echten ist; die Wert in Klammern beim *Yeast*-Netzwerk beziehen sich auf [12], Rückrechnung, und [13], wo ein konkreter Wert von 70 Vorkommen angeben ist, während es bei diesem Netzwerk 73 Vorkommen waren.

Diese Ergebnisse, siehe Tabelle 4.1, sind ungefähr deckungsgleich mit denen der Gruppe des Weizman Institute of Science. Dies bedeutet, der entwickelte MCMC-Algorithmus scheint dem der Gruppe um Alon zumindest zu gleichen oder das selbe zu machen.

4.3 Powerlaw Netzwerke

Die bisherigen Daten scheinen zwar ein uniformes Sampling zu modellieren, aber sie basierten immer auf realen, nicht unbedingt zusammenhängenden Netzwerken. Aus diesem Grund sollte noch eine weitere Klasse Netzwerke untersucht werden, die realitätsnäher ist, aber dennoch zusammenhängend ist.

Dazu wurde mittels Potensgesetz (*power law*) eine Ein- und Ausgradfolge generiert, die nach [4] reale Graphen besser modeliert als zufällige Graphen dies können. Dies bedeutet, dass eine Verteilung $P \propto k^{-\gamma}$ angenommen wur-

de und solange neue Werte generiert wurden, die dieser Funktion folgen, bis ein Sättigungswert erreicht ist. Aus dieser Verteilung wurden dann Knoten-Eingrad-Folgen generiert. Auf Basis dieser Knoten-Eingrad-Folgen konnten nunmehr mit dem Havel-Hakimi-Algorithmus, der als Ausgrad-Folge entweder die gleiche, die reverse oder die gemischte Folge als Eingabe erhielt, ein Graph generiert werden, auf welchen dann der MCMC-Algorithmus angewandt werden konnte.

Zwar wäre es auch bei diesen möglich gewesen von dem ersten Netzwerk die Anzahl der *feed-forward* Motive zu zählen und danach zu messen wie häufig es in den anderen daraus generierten Graphen vorkommt, aber hier wurde ein anderer Weg eingeschlagen.

Neuer Evaluationsansatz

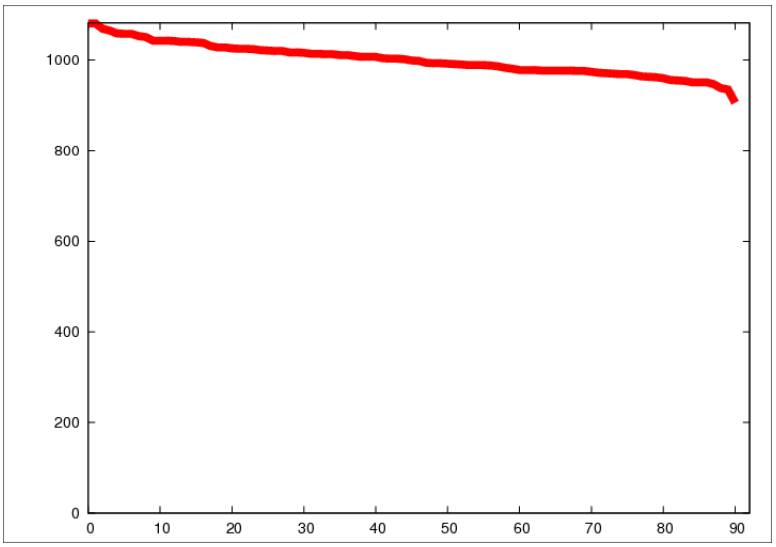
In eine einfache Datenbank wurde der Graph über einen Hashkey eingetragen, sowie auch die Konfiguration des Graphen, ob er zusammenhängend ist und wie oft er gefunden wurde. Letzteres festzustellen war möglich dank der Hashkeys die eine sehr geringe Kollisonsgefahr haben.

Algorithmus 5 Powerlaw-Generierung

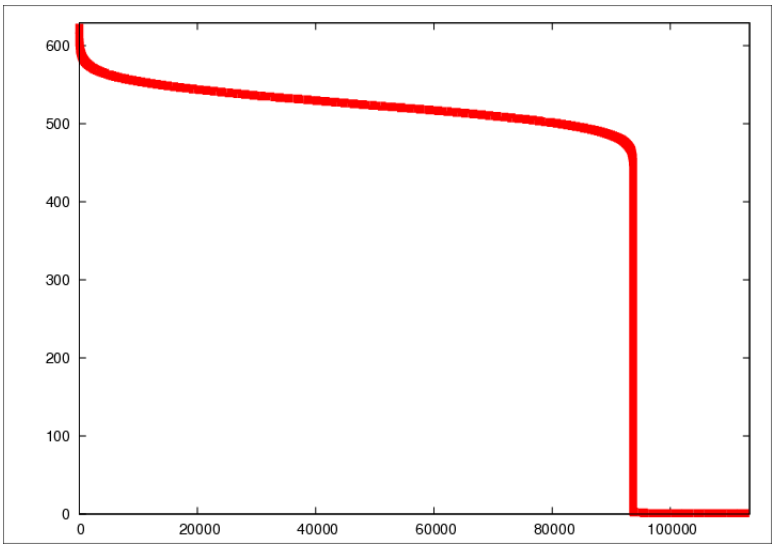
Require: startvalue N
Require: exponent k
  $result = 1$
  $i = 0$
  $resultList = \emptyset$
  while result > 0 **do**
    $result = int\,(N * * (-k * i))$
    $i = i + 1$
    if result > 0 **then**
      $resultList.append(result)$
    end if
  end while

Der Test wurde zuerst wieder mit dem Toy-Netzwerk (siehe Abbildung 4.1) durchgeführt, welches bei den vorigen Tests als uniform in einem sehr engen Rahmen bewertet wurde. Daher überraschte es keineswegs, dass es auch bei diesem Test als relativ uniform mit einer geringen Abweichung gemessen wurde.

Umso gravierender wurden die Ergebnisse, als der selbe Test auf einem weiteren Netzwerk der *powerlaw*-Klasse ausgeführt wurde. Während der Großteil der Ergebnisse weitesgehend auf Uniformität hindeutete, fiel ein anderer Teil sehr stark, so dass Zweifel an der Uniformität des Verfahrens auftraten (siehe Abbildung 4.4).



(a) Toy-Netzwerk



(b) Powerlaw-Netzwerk

Abbildung 4.4: (a) zeigt die Auswertung des MCMC-Algorithmus auf dem Toy-Netzwerk recht uniform um 1100 Wiederholungen pro Konfiguration. (b) zeigt die Auswertung des MCMC-Algorithmus auf einem *powerlaw*-Netzwerk mit Sequenz $S = ((3,3),(2,2),(2,1),(1,2),(1,1),(1,1),(1,1),(1,1))$

Kapitel 5

Berechnung der Uniformität

5.1 Markov-Kette

Der Anteil der Markov-Kette an diesem Algorithmus bezieht sich auf die Konfigurationen der Graphen mit fester Anzahl Kanten und Knoten. Wenn man jeden Graphen für sich als einen Knoten im Netzwerk der Konfigurationen betrachtet, so sind die Kantentausche innerhalb eines Graphen die Kanten im Netzwerk der Konfigurationen.

Für das Toy-Netzwerk ist der *diameter*, also die maximale Distanz zwischen zwei Knoten im Konfigurations-Netzwerk, genau 2, da man von jeder Konfiguration mit maximal einem Schritt zur Spezialkonfiguration gelangen kann und von dort aus mit ebenfalls maximal einem weiteren Schritt jede beliebige andere Konfiguration erreichen kann. Alle Knoten, bis auf die Spezialkonfiguration, haben einen Grad von 17, mit Schleife 18, nur die Spezialkonfiguration hat einen wesentlich höheren Grad von 90, mit Schleife 91, was aber auch verständlich ist, wenn man sie als Basiskonfiguration betrachtet von der aus man jede andere sehr schnell erreichen kann.

Für den Konfiguations-Graphen des *powerlaw*-Netzwerkes ist durchschnittlich ein gerundeter Grad von 32 festzustellen gewesen, von 29 bis 39 war die Spanne der Knotengrade. Es ist also ein wesentlich natürlicheres Basis-Netzwerk benutzt worden als beim Konfigurations-Graphen des Toy-Netzwerkes, welches sich wie beschrieben mit einer besonderen Konfiguration hervorhebt.

Die Bestimmung des Diameters des Konfigurations-Netzwerkes des *powerlaw*-Netzwerkes mittels Standard-Algorithmen wie dem *Floyd-Warshall*-Algorithmus [24] zur Lösung des *all-sources-shortest-paths*-Problems war leider nicht möglich aufgrund des hohen Speicheraufwandes, der mit $\mathcal{O}\left(n^2\right)$ quadratisch mit der Anzahl der Knoten im Netzwerk steigt möchte man die Pfade direkt mitspeichern steigt der Aufwand sogar in die Klasse $\mathcal{O}\left(n^3\right)$.

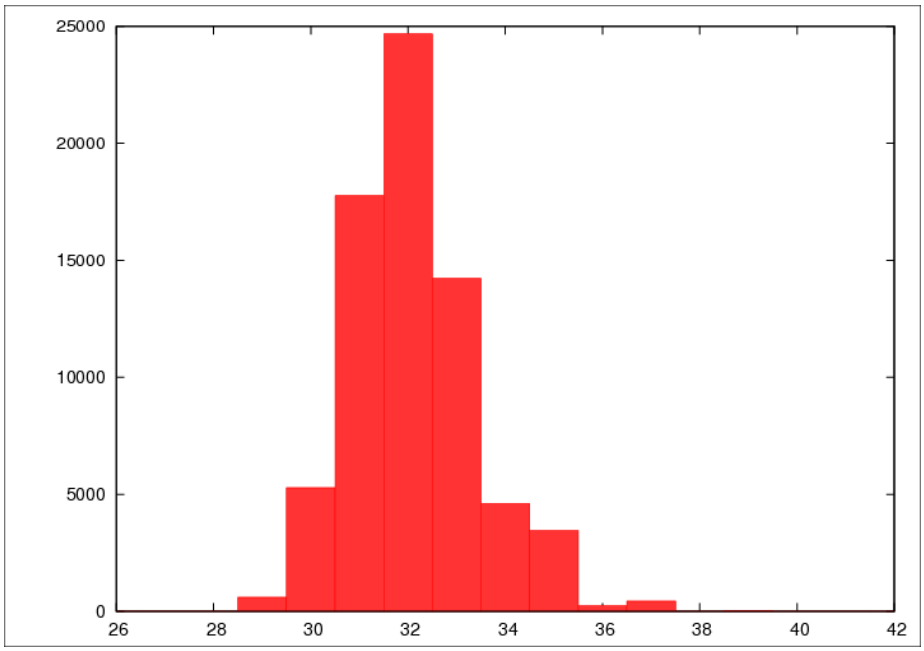


Abbildung 5.1: Verteilung der Knotengrade beim *powerlaw*-Konfigurations-Graphen

5.2 Engültiger Evaluationsansatz

Aufgrund der Tatsache, dass die Generierung nicht uniform aus dem Raum aller Graphen zu ziehen schien, wurde die Entscheidung getroffen die Markov-Kette, im weiteren Konfigurations-Graphen genannt, dieser Graphen zu generieren um festzustellen, ob der Algorithmus wirklich uniform sampelt beziehungsweise wie groß q gewählt werden muss, damit es uniform wird und jede Konfiguration gezogen wird.

Auf diese Netzwerke wurde ein modifizierter PageRank-Algorithmus angewendet [26]. Die Modifikation bestand darin, dass alle Knoten bis auf einer einen Initialwert von 0 hatten; als Ausnahmeknoten wurde der Knoten mit dem höchsten Eingangsgrad gewählt, dessen Gewicht für den PageRank-Algorithmus mit der Anzahl der Knoten im Konfigurations-Netzwerk initalisiert wurde. Die Idee dahinter ist, wenn es gleichwahrscheinlich ist zu jedem Knoten zu gelangen, wird der PageRank-Algorithmus nach einer endlichen Zeit bei allen Knoten einen approximativ zu 1 gleichen Wert aufweisen.

Für das Toy-Netzwerk ergaben sich zwei Ergebnisse; betrachtet man den Konfigurations-Graphen wie die anderen Netze auch ohne Schleifen, so ergab sich keine uniforme Verteilung. Dies ist offensichtlich darin begründet, dass man von jeder Konfiguration aus zur Spezial-Konfiguration wechseln kann

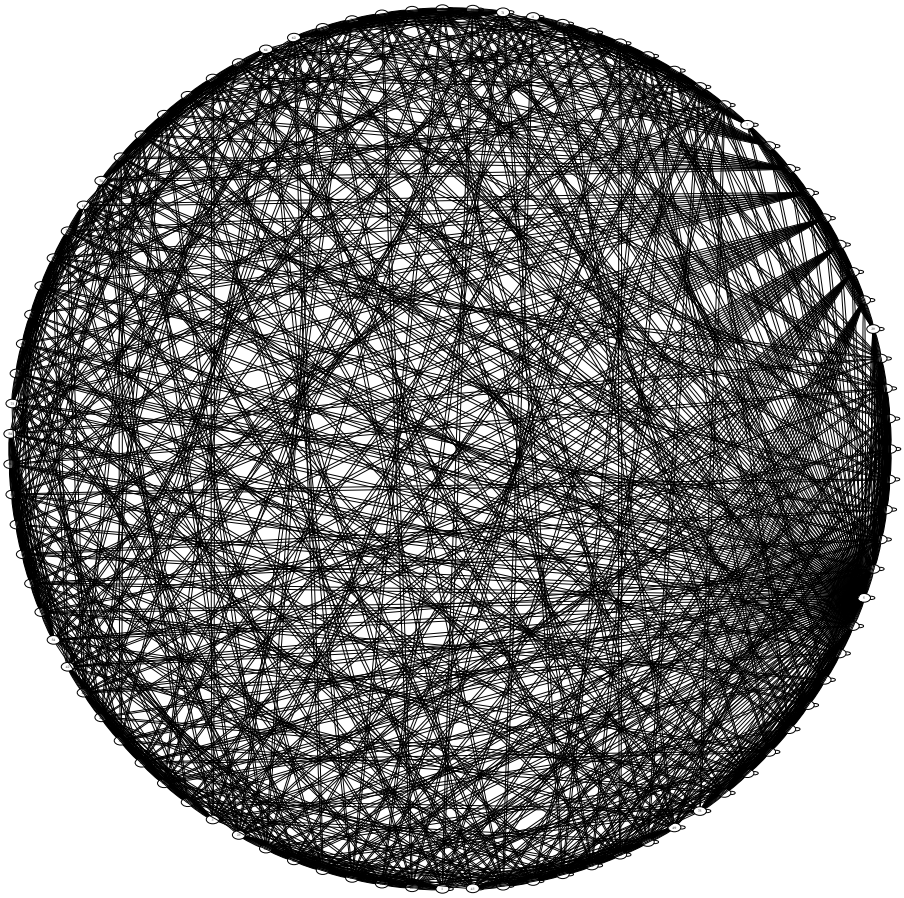


Abbildung 5.2: Graph der Konfigurationen des Toy-Netzwerkes

mit einem einzigen Kantentausch, während die neunzig anderen unter sich mehr Tausche benötigen, um einander zu erreichen. Betrachtet man die Kanten als gewichtet mit Gewicht 1, so ergibt sich direkt eine Vergrößerung des PageRank-Wertes bei der Spezial-Konfiguration; bemerkenswert ist dennoch, dass die anderen Knoten gleichverteilte PageRank-Werte haben. Das andere Ergebnis wird erzielt, wenn man mit Schleifen für unzulässige Tausche arbeitet. Gewichtet werden sie durch die Anzahl aller möglichen Kantentausche abzüglich der durchführbaren (in diesem Netzwerk also 163, bei der Spezialkonfiguration nur 90). Hier ergab es sich, dass eine perfekte uniforme Verteilung eintrat.

Bei dem *powerlaw*-Netzwerk wurde dieser Test ebenfalls durchgeführt. Auch dort verlief er gut in einem engen Rahmen (siehe Abbildung 5.3). Wenn der PageRank Algorithmus die Wahrscheinlichkeit angibt, dass die Netze gefunden werden, so wurde bei den durchgeführten Tests eine Verteilung sehr nahe der uniformen Verteilung gefunden (siehe Tabelle 5.1). Die dafür genutzten Netzwerke entsprachen *powerlaw*-Netzwerken, deren Eingangsgradsequenz $S = (3, 2, 2, 1, 1, 1, 1, 1)$ entsprach und deren Ausgangsgradsequenz entweder negativ korreliert, positiv korreliert oder zufällig verteilt war. Der modifizierte PageRank-Algorithmus auf der Markovkette des jeweiligen Netzwerkes konvergierte bei den durchgeführten Tests gegen 1.

5.3 Die „Konstante“ q

Es bleibt noch zu beachten, dass bei den späteren Evaluationen in dieser Arbeit die Konstante q, welche für den MCMC-Algorithmus 4 genutzt wurde, weggelassen wurde. Dies geschah aus zwei Gründen.

- Es war nur noch wichtig alle Konfigurationen eines Graphen zu finden
- Es war unbekannt welches q geeignet erschien

Die Gruppe um Alon schrieb in [12] dass ein q von 100 mehr als genügen würde um den MCMC-Algorithmus auf dem Toy-Netzwerk im Speziellen und viele andere Netzwerke im Allgemeinen uniform zu halten.

Aus den Untersuchungen aus dem vorigen Abschnitt konnten wir diesen hohen Wert für q bestätigen. Er ist für die getesteten Fälle ausreichend, aber in den meisten Fällen viel zu hoch und ist bei einer Laufzeit von $\mathcal{O}(q \cdot m)$, wenn man q als Variable betrachtet, nicht unbedingt genügend. Immerhin wurde gezeigt, dass sich eine quadratische Grenze etablieren könnte (vgl. [8]) Die Untersuchungen mit dem modifiziertem PageRank-Algorithmus ergaben, dass Werte von $q = \lceil \frac{21}{12} \rceil = \lceil 1,75 \rceil = 2$ für das Toy-Netzwerk schon mehr als genügen, für das *powerlaw*-Netzwerk genügten andere Werte, je nach Verteilung der Ausgrade lagen diese zwischen 4 und 8.

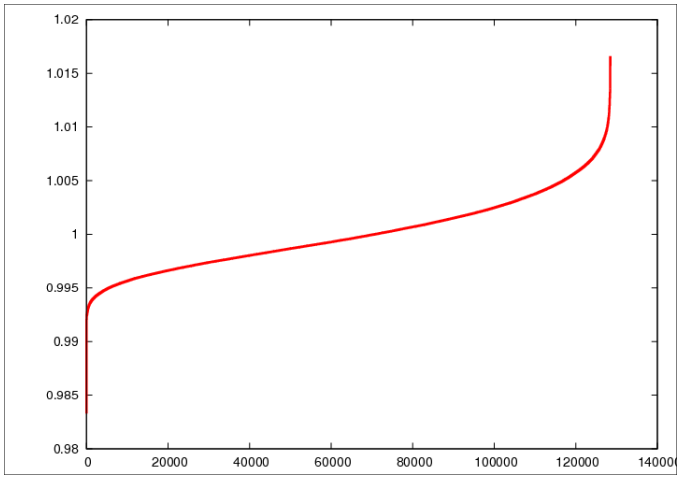
Der Vorteil an der Findung dieser Werte mittels der Konstruktion dieses Graphen aller Graphen liegt darin, dass spätere Simulationen darauf mit einem genauer bestimmten q schneller durchgeführt werden können und nicht

Netzwerk	Kantenzahl	Iterationen	q	$\pm \delta$
negativ korreliert	12	61	~ 5	0.009
positiv korreliert	12	89	~ 7	0.02
zufällig	12	78	~ 7	0.15
Toy-Netzwerk	12	21	~ 2	0

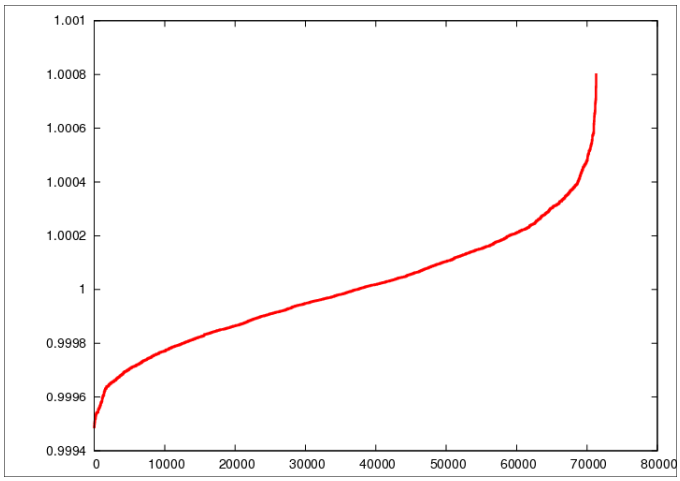
Tabelle 5.1: Iterationszahlen bis der PageRank-Algorithmus terminiert mit durchschnittlichem δ von der uniformen Verteilung

mehr auf eine Auswertung von anderen Werten, wie der Anzahl der Zusammenhangskomponenten oder dem Clusteringkoeffizienten, gewartet werden muss. Auch ist von Vorteil dass bekannt ist, wie viele Graphen es bei einer Anzahl von n Knoten und m Kanten geben kann und man nicht in die Verlegenheit gerät Graphen auszulassen, weil man sie eventuell übersehen hat oder sie durch den Algorithmus nie oder nur sehr selten erreicht werden.

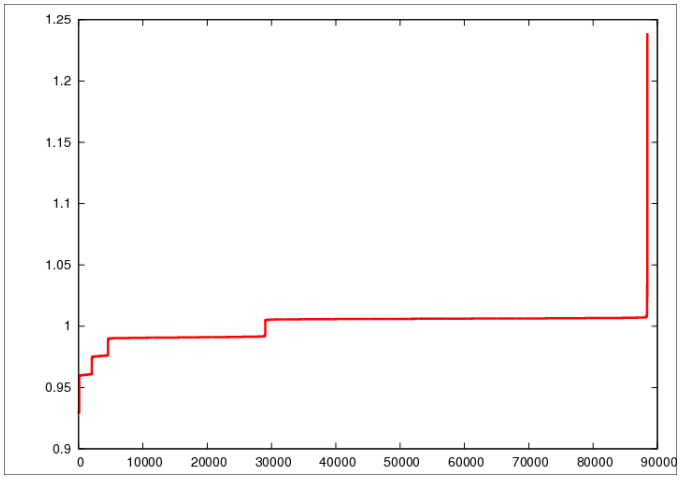
Der Nachteil ist, dass das Abspeichern in einer Datenbank oder sonstige Verfahren des Speichern dieser Werte sehr langwierig sein kann und das Speichern selbst wieder als ein Aufwand angesehen kann. Dieser ist aber vernachlässigbar, wenn man wirklich nur die Netzwerkanalysen betreiben will.



(a) *powerlaw*-Netz, reverse Ausgradsequenz



(b) *powerlaw*-Netz, Ausgradsequenz gleich Eingradsequenz



(c) *powerlaw*-Netz, vermischte Ausgradsequenz

Abbildung 5.3: Evaluierung des modifizierten PageRank-Algorithmus anhand verschiedener Graphen von Graphen von *powerlaw*-Netzwerke. An der x-Achse sind die einzelnen Graphen nummeriert abgetragen, an der y-Achse der Wert des PageRank-Algorithmus der einzelnen Netzwerke. Ersichtlich ist die starke Annäherung an den Wert einer uniformen Verteilung.

Kapitel 6

Fazit

6.1 Zusammenfassung

In dieser Arbeit habe ich nun im zweiten Kapitel den Havel-Hakimi Algorithmus zur Generierung von Graphen aus einer gegebenen Gradsequenz vorgestellt, danach bin ich zu einem Markov-Ketten-Monte-Carlo-Algorithmus weitergegangen, der den zufälligen Kantentausch realisiert. Beide Algorithmen wurden in Bezug auf ihre Laufzeit analysiert in Bezugnahme auf meine Implementation der beiden Algorithmen. Im dritten Kapitel habe ich in einem ersten Experiment nach Motiven gesucht und die Ergebnisse der selbst programmierten Algorithmen mit denen der Gruppe um Uri Alon verglichen. Danach erklärte ich kurz die Prinzipien der *powerlaw*-Netzwerk und beschrieb dann die Ergebnisse der Algorithmen auf dieser Familie von Netzwerken, wobei ich mich immer noch auf ein kleines Testnetzwerk beschränkte. Zum Abschluss untersuchte ich noch wie uniform die Algorithmen wirklich sind und begründete die Uniformität schließlich mit einem modifizierten PageRank-Algorithmus auf der Markov-Kette eines Netzes.

Es ist dabei festzustellen gewesen, dass die Algorithmen relativ schnell die Netzwerke generieren und auch abspeichern, aber je größer das Netzwerk wird, desto langsamer werden die Algorithmen. Während in der Entwicklungsphase der Algorithmen noch Probleme mit dem Zwischenspeicher bestanden, wurde der Fehler auf dem Weg behoben und mittlerweile sollte es möglich sein die Algorithmen auch auf einem Standardcomputer laufen zu lassen mit vernünftigen Laufzeiten. Verbessert werden könnte noch die Auslastung des Prozessors, die bei manchen Speichervorgängen oder bei der Anwendung des PageRank-Algorithmus noch sehr hoch war. Auch wenn in der Zeit der Multiprozessoren die maximale Auslastung eines Kernes noch nicht sonderlich tragisch sein sollte ist es dem Nutzer immer lieber wenn man es nicht merkt, wenn ein Programm im Hintergrund rechnet.

6.2 Ausblick

Als weitere Forschungsarbeit an diesem Gebiet könnte ich mir vorstellen, dass die Untersuchung der Variablen q ein größeres Gebiet darstellen könnte, beziehungsweise die Skalierung der Variablen anhand der Anzahl der Kanten des Netzes, was bei mir immer recht übersichtlich blieb. Es existiert zwar eine theoretische Obere Schranke die in $\mathcal{O}\left(m^2\right)$ liegt, aber diese würde die Laufzeit von $\mathcal{O}\left(m\right)$ nach $\mathcal{O}\left(m^3\right)$ schieben, von daher wäre eine weitere Evaluierung mit der neuen Idee der Vorherbestimmung dieses Wertes bestimmt von Interesse anstelle des Findens während des Algorithmus.

Weiterhin ist die Untersuchung von anderen Motiven, zum Beispiel von Cliquen, Paarungen oder anderen Beziehungen in sozialen Netzwerken von Interesse. Auch eine Applikation im Arbeitsbereich kann sinnvoll sein, wenn man damit das soziale Netzwerk einer Firma analysiert und bemerkt, dass Chefs und Angestelle nicht gut harmonieren, aber eine andere Kombination das Arbeitspotential stark vergrößern würde.

Literaturverzeichnis

[1] Daron Acemoglu and Asu Ozdaglar. 6.207/14.15: Networks Lecture 5: Generalized Random Graphs and Small-World Model. Technical report, MIT, 2009.

[2] Uri Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6):450–461, June 2007.

[3] Uri Alon. A biophysicist learns the art of hugging. *Nature*, page 701, Juni 2008.

[4] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks, October 1999.

[5] Fan Chung and Linyuan Lu. *Complex Graphs and Networks*. American Mathematical Society, August 2006.

[6] Péter L. Erdős, István Miklós, and Zoltán Toroczkai. A simple Havel-Hakimi type algorithm to realize graphical degree sequences of directed graphs. *Electronic Journal of Combinatorics*, 17(1):R66, 2010.

[7] Christos Gkantsidis, Milena Mihail, and Ellen Zegura. The Markov Chain Simulation Method for Generating Connected Power Law Random Graphs. In *In Proc. 5th Workshop on Algorithm Engineering and Experiments (ALENEX). SIAM*, 2003.

[8] M. Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, December 1989.

[9] Ansgar Jonietz. Skalenfreie Netze. `http://www.jonietz.de/personen/ansgar/Skalenfreie_Netze.pdf` – last visited 22th September 2010, 2005.

[10] Sven Kosub. Internet-Algorithmik. `http://www14.informatik.tu-muenchen.de/lehre/2006WS/ia/Internet-Algorithmik.pdf` – visited last 1th Otober 2010, 2010.

[11] Farhad Manjoo. How Black People Use Twitter -The latest research on race and microblogging. `http://www.slate.com/id/2263462` – last visited 23th September 2010, August 2010.

[12] R. Milo, N. Kashtan, S. Itzkovitz, M. E. J. Newman, and U. Alon. On the uniform generation of random graphs with prescribed degree sequences, May 2004.

[13] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science (New York, N.Y.)*, 298(5594):824–827, October 2002.

[14] M. E. J. Newman. The structure and function of complex networks, Mar 2003.

[15] Michael Nollert. Soziale Netzwerke - Theoretische Konzepte, Analyseinstrumente und empirische Befunde. Technical report, Universitäten Freiburg i. Ue und Zürich, 2006.

[16] Matthias Scholz. Komplexe Netzwerke - Erdös-Rényi Zufallsgraph-Modell. `http://www.network-science.org/SS2009.html` –last visited 23th September 2010, Mai 2009.

[17] twitter.com. shitmydadsays. `http://twitter.com/shitmydadsays` – last visited 23th September 2010, September 2010.

[18] Remco van der Hofstad. *Random Graphs and Complex Networks*, chapter Chapter 7: Configuration Model. Department of Mathematics and Computer Science Eindhoven, University of Technology, 2010.

[19] Kyle VanHemert. Black people call and text way more than everyone else. `http://gizmodo.com/5620989/black-people-call-and-text-way-more-than-everony-else` –last visited 9th Oktober 2010, August 2010.

[20] Fabien Viger and Matthieu Latapy. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In Lusheng Wang, editor, *Computing and Combinatorics*, volume 3595 of *Lecture Notes in Computer Science*, chapter 45, pages 440–449–449. Springer Berlin / Heidelberg, Berlin/Heidelberg, 2005.

[21] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[22] whatis.com. six degrees of separation. `http://whatis.techtarget.com/definition/0,,sid9_gci932596,00.html` – visited last 19th September 2010, März 2008.

[23] wikipeda. Socialanthropology - history. `http://en.wikipedia.org/wiki/Social_anthropology#History` –last visited 18th September 2010, September 2010.

[24] wikipedia. Algorithmus von Floyd und Warshall. `http://de.wikipedia.org/wiki/Algorithmus_von_Floyd_und_Warshall` –last visited 4th Oktober 2010, August 2010.

[25] wikipedia. Markov-Kette. http://de.wikipedia.org/wiki/Markow-Kette –last visited 4^{th} Oktober 2010, August 2010.

[26] wikipedia. PageRank. http://en.wikipedia.org/wiki/PageRank –last visited 23^{th} September 2010, September 2010.

[27] wikipedia. Small World Experiment. http://en.wikipedia.org/wiki/Small_world_experiment – visited last 18^{th} September 2010, September 2010.

[28] wikipedia. Social Network. http://en.wikipedia.org/wiki/Social_network –visited last 19^{th} September 2010, September 2010.

[29] wikipedia. Soziales Netzwerk (Soziologie). http://de.wikipedia.org/wiki/Soziales_Netzwerk_%28Soziologie%29 –last visited 21^{th} September 2010, September 2010.

Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Kaiserslautern, den

. . . . . .
(Unterschrift des Kandidaten)