Universität Oldenburg

Fortgeschrittene Computerorientierte Physik

Verteilung der kürzesten Pfade in skalenfreien Graphen

Author:
Jan Kämper
Florian Börgel

Supervisor:
Alexander Hartmann

March 28, 2016

Contents

1 Problemstellung

Ziel dieses Projektes war es statistische Beobachtungen über die Verteilung von kürzesten Pfaden in skalenfreien Graphen zu machen. In skalenfreien Graphen sind die Kanten pro Knoten nach einem Potenzgesetz verteilt:

$$P(k) \sim k^{-\alpha}$$

Die Erstellung von skalenfreien Graphen erfolgt anhand von speziellen Algorithmen. Der hier verwendete Algorithmus folgt dem Barabasi-Albert Model und nutzt die Methode Preferential Attachment. Dabei ist der Parameter m maßgeblich, der die Anzahl an Nachbarn eines neu hinzugefügten Knotens beschreibt.

Die Größe des Graphens wird über die Anzahl der Knoten n definiert. Um eine Aussage über die statistische Verteilung der kürzesten Pfade in unterschiedlichen Graphfamilien treffen zu können, müssen mehrere Simulationsläufe in Abhängigkeit der Parameter n und m durchgeführt werden, über die dann gemittelt wird. Zur Berechnung der kürzesten Wege wurde auf den Floyd-Warshall Algorithmus zurückgegriffen, der in kubischer Laufzeit kürzeste Wege für alle Knotenpaare in dem ungerichteten Ausgangsgraphen liefert. Experimente mit unterschiedlichen parametrisierten Graphen wurden dann durch ein Fitting an Normalverteilung und Potenzgesetze in Gnuplot ausgewertet.

In den Extraaufgaben wurde zum einen ein allgemeines preferential attachment umgesetzt, das zu einer veränderten Gradverteilung pro Knoten führt. Zum anderen wurde eine weitere Familie von Graphen betrachtet. Hierbei handelt es sich um ebene Zufallsgraphen in der $[0, 1]^2$ Ebene, wobei Kanten zwischen zweien der gleichmäßig in der Ebene verteilten Knoten mit einer Wahrscheinlichkeit hinzugefügt werden, die kubisch in der euklidischen Distanz der Knoten abnimmt. Hier wurde analog zu den skalenfreien Graphen die Verteilung der kürzesten Wege Längen untersucht, zusätzlich aber auch die mittlere Länge der kürzesten Wege in Abhängigkeit von der Größe des Graphen.

2 Programmentwurf

Die wichtigsten Bestandteile des Programms werden hier kurz erläutert. Auf vollständige Methodensignaturen wird dabei verzichtet. Alle Metoden mit Ausnahme der Main-Funktionen sind in der Datei *shortest_path_fragment.c* zu finden.

Generierung der Graphen Um die Problemstellung wie gefordert bearbeiten zu können, werden zunächst Methoden benötigt, die Graphen des gewünschten Typs (skalenfreier Graph oder ebener

gleichverteilter Graph) erzeugen. Die dafür benötigten Datenstrukturen für ungerichtete Graphen wurden aus der Vorlesung übernommen (Dateien *graph_lists.h, lists.h, lists.c*). Für die Zusatzaufgabe wurde lediglich die Datenstruktur für Knoten um die x- und y-Koordinate ergänzt. Weitere wesentliche Graphoperationen (*gs_insert_edge, gs_create_graph, gs_edge_exists, gs_preferential_attachment*) konnten in der Datei *shortest_path_fragment.c* ebenfalls übernommen werden. Für die Erzeugung der ebenen Zufallsgraphen wurde die Methode *gs_create_planar_graph* geschrieben, welche n zufällig verteilte Knoten und Kanten entsprechend der Wahrscheinlichkeit $p_{ij} = f \cdot (1 + \frac{\sqrt{N\Pi} \cdot d_{ij}}{\alpha})^{-\alpha}$ erzeugt.

Ausgabe Zur Kontrolle der einzelnen Berechnungsschritte und Ausgabe der Simulationsergebnisse wurden mehrere Methoden implementiert, die Graphen, Matrizen oder Histogramme auf der Konsole oder in Dateien ausgeben. Die Methode *exportGraphDot* schreibt eine Ausgabedatei, welche von dem Programm *dot* gelesen werden kann und durch die der Graph visualisiert wird.So konnten in frühen Stadien der Entwicklung bereits einfach Programmfehler erkannt und behoben werden. Zum gleichen Zweck dient *printedges*, die alle Kanten des Graphs in einfacher Form in eine Ausgabedatei schreibt. Um die berechnete Distanzmatrix mit allen kürzesten Weglängen im passenden Format auszugeben kann die Methode *printDistances* verwendet werden.

Die Ergebnisse der Simulationsläufe zur weiteren statistischen Analyse werden mit den Methoden *printHistogram* und *printHistogramNormed* in Dateien geschrieben. Die nicht-normierte Variante schreibt die einzelnen Einträge des Histograms zeilenweise in eine Datei. Die normierte Variante führt zusätzlich eine Normierung des Gesamtgewichts des Histograms auf 1 durch. Der Parameter *startindex* kann verwendet werdeno, um zum Beispiel die Histogram-Einträge mit Distanz 0 auszublenden.

Berechnung der kürzesten Wege Als Schritt nach der Graph-Erzeugung müssen für den angelegten Graph die kürzesten Wege berechnet werden. D.h. für den gegebenen Graphen muss für jede Kombination von zwei Knoten der jeweilige Abstand bestimmt werden. Das Ergebnis ist eine quadratisch symmetrische Matrix, die in jeder Zelle des oberen Dreiecks den Abstand des Knotenpaars (Spalte und Zeile) enthält. Um überhaupt eine Aussage über die Distanz treffen zu können, muss den Kanten zunächst eine Wertigkeit bzw. Länge zugeschrieben werden. Im Falle der skalenfreien Graphen wurde überall der Abstand $d_{ij} = 1$ angenommen, sodass die Länge eines Weges der Anzahl an Kanten auf dem Weg entspricht. Im Falle der ebenen Graphen wurde die euklidische Distanz betrachtet. Für die Berechnung der kürzesten Wege sollte laut Aufgabenstellung der Floyd-Warshall Algorithmus implementiert werden. Dieser ist in der Methode *gs_all_pair_shortest_paths* implementiert. Zunächst wird darin die Initialisierung der Distanzmatrix vorgenommen. Diese kann gewichtet (euklidisch) oder ungewichtet sein (skalenfreier Graph). Danach folgt die dreifach verschachtelte Schleife von Floyd-Wahrshall, welche die als Parameter vorhandene Distanzmatrix ausfüllt.

Aufbereitung der Simulationsdaten in Histogrammen Wie bereits im vorherigen Kapitel beschrieben, müssen mehrere Durchläufe mit immer neu generierten Graphen durchgeführt werden, um eine statistische Aussage über die Verteilung der kürzesten Wege treffen zu können. Für das Programm bedeutet das, dass jeder Durchlauf und die dabei berechneten kürzesten Wege gespeichert und in ein Histogram sortiert werden müssen. Für die Sortierung wird gezählt wie häufig eine berechnete Strecke innerhalb des Graphen vorkommt. Bei den skalenfreien Graphen sind alle Distanzen diskrete Werte (Kantengewichte 1) und damit jede Ganzzahl ein Histogram-Bin. Bei den ebenen Zufallsgraphen können beliebige Längen vorkommen. Somit wird die Anzahl an Bins mit der aus der Literatur bekannten Vorschrift $numBins = \sqrt{n}$ angenommen und die kürzesten Wege dementsprechend in Bins sortiert. Auf beide Weisen wird ein Histogramm erstellt, welches die Verteilung der kürzesten Wege enthält. Die Methoden *fillHistogramDiscrete* und *fillHistogramContinuous* tun genau dies.

Im letzten Teil der Zusatzaufgabe soll anstelle der Verteilung aller kürzesten Wege die mittlere kürzeste Weglänge in Abhängigkeit der Größe n untersucht werden. Dies übernimmt die Funktion *computeMeanShortestPath*, welche alle existierenden Wege im gegebenen Graphen zählt und deren Längen mittelt.

Simulationssteuerung Die Steuerung der Simulationsläufe erfolgt in den Methoden *runExperiments* (skalenfreie Graphen), *runExperimentsPlanar* (ebene Zufallsgraphen) und *runMeanExperimentsPlanar* (mittlere Weglänge). Der Ablauf ist in allen Fällen gleich: Eine Schleife führt eine festgelegte Anzahl an Simulationsläufen durch. In jedem Lauf wird zunächst einer neuer Graph erzeugt und dann die kürzesten Wege berechnet, und die Ergebnisse zwischengespeichert. Im Falle der Verteilungsuntersuchung wird zur Zwischenspeicherung der Histogram Array verwendet und über die Simulationsläufe hinweg hochgezählt. Im Falle der Mittelwertuntersuchung wird eine Double-Variable genutzt. Ausserdem gibt es hier noch eine äußere Schleife die über unterschiedliche Graph-Größen iteriert. Nachdem alle Simulationsläufe beendet sind erfolgt die Ausgabe der Ergebnisse in Dateiform.

Automatisierung Um die unterschiedlich parametrisierten Simulationen schnell neu anzustoßen (unterschiedliche Werte für die Parameter n und m) existieren die Skripte *simulationScalefree.scr* und *simulationPlanar.scr*. Diese rufen das kompilierte Programm mit der entsprechenden Main und allen benötigten Kommandozeilenparametern auf (n, ggf m und Dateinamen für die Ergebnisse).

Auswertung mit Gnuplot Basierend auf denen durch die Skripte erzeugten Dateien mit Endung .dat können Gnuplot Skripte verwendet werden, die die vorliegenden Daten graphisch darstellen und ein Fitting and die gewünschte Verteilung durchführen. Diese Skripte befinden sich im Ordner *Plotscripts* und können in gnuplot mit dem *load* Befehl geladen werden.

3 Resultate

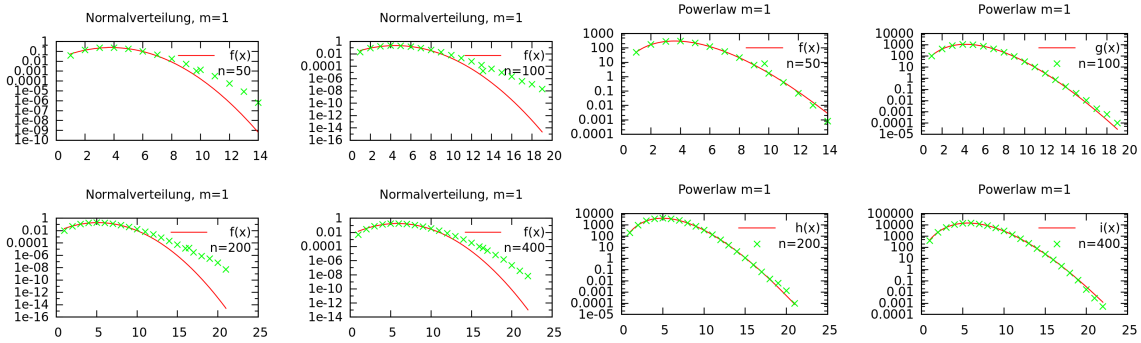
3.1 Skalenfreie Graphen

Für die skalenfreien Graphen wurde folgende Simulationen durchgeführt:

Größe n: 50,100,200,400

Parameter m: 1,2

Für $m = 1$ haben wir die Ergebnisse in Abbildung 1 erhalten. Die unterschiedliche Beschriftung der y-Achse ist der Tatsache geschuldet, dass für die Normalverteilung eine Normierung durchgeführt werden musste, was für das Potenzgesetz hier nicht notwendig war. Die Kurven sind aber in allen Abbildungen, die folgen, durch Skalierung identisch.



(a) Fitting an eine Normalverteilung

(b) Fitting an Potenzgesetz

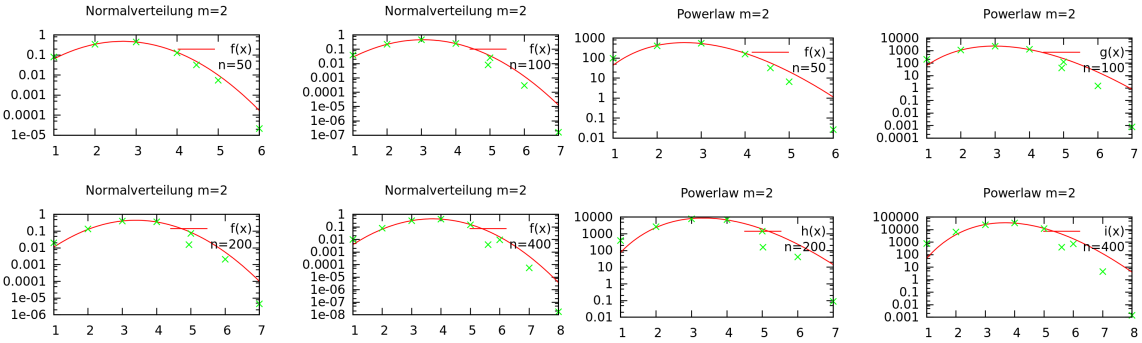
Figure 1: Skalenfreie Graphen mit $m = 1$

Unabhängig von der Anzahl an Knoten scheint der Fit an das Potenzgesetzt hier wesentlich besser zu sein. Dies ist im Falle $m = 2$ umgekehrt. Hier lässt sich feststellen, dass für ein Fitting an eine Normalverteilung geringere Abweichungen vorliegen. Die Ergebnisse sind in Abbildung 2 zu sehen.

Somit lässt sich also schlussfolgern, dass für $m = 1$ die Verteilung der kürzesten Wege im Graphen unabhängig von der Größe dem gleichen Gesetz folgt wie es für die Gradverteilung der Knoten gilt, nämlich dem Potenzgesetz. Ändert man den für das preferential attachment massgeblichen Parameter auf $m = 2$ so gilt dies nicht mehr, jetzt sind die kürzesten Weglängen im Graph eher normalverteilt.

3.2 Ebene Zufallsgraphen

Für ebene Zufallsgraphen wurden die gleichen Simulationen wie für skalenfreie Graphen durchgeführt. Der Parameter m spielt hier keine Rolle.

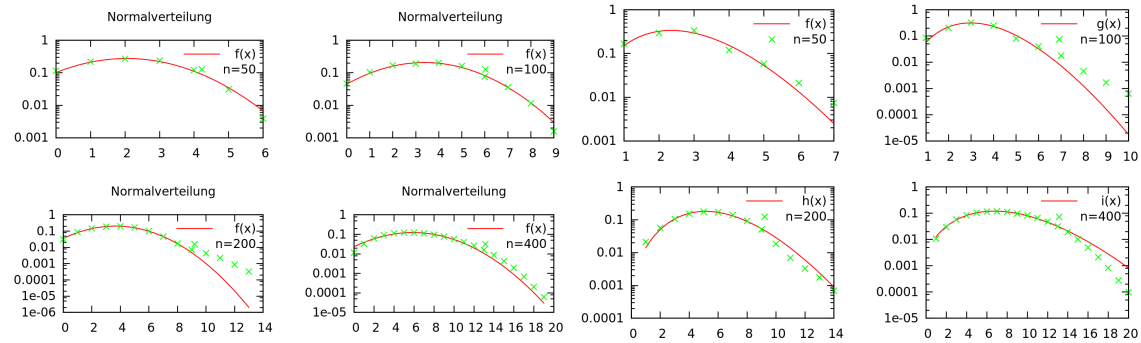


(a) Fitting an eine Normalverteilung (b) Fitting an Potenzgesetz

Figure 2: Skalenfreie Graphen mit $m = 2$

Größe n: 50,100,200,400

Es zeigt sich, dass auch hier eine Normalverteilung die bessere Approximation liefert (siehe Abbildung 3).



(a) Fitting an eine Normalverteilung (b) Fitting an Potenzgesetz

Figure 3: Ebene Zufallsgraphen

3.3 Mittlere Weglänge in ebenen Zufallsgraphen

Für ebene Zufallsgraphen wurde zusätzlich untersucht, wie sich die mittlere kürzeste Wegelänge in Abhängigkeit von der Anzahl der Knoten verhält.

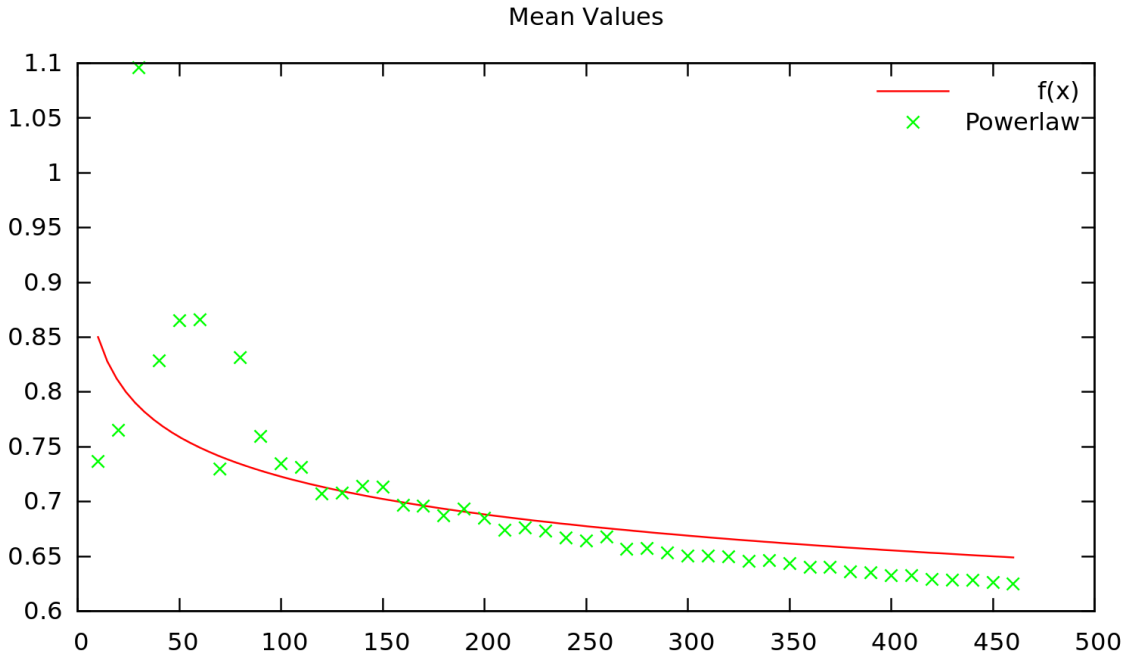


Figure 4: Fitting Power Law