1 Motivation

Viele Beziehungen in der realen Welt lassen sich durch Graphen darstellen, wie beispielsweise Straßenbahnnetz, soziale Kontakte oder Zitate von wissenschaftlichen Artikeln. Um diese Netzwerke zu untersuchen, kann man diese als Graphen darstellen und dann Algorithmen aus der Netzwerkanalyse darauf anwenden. In der Situation des Straßenbahnnetzes wäre es beispielsweise interessant, wie resistent dieses gegen Störungen ist.

Um derartige Algorithmen zu testen und sonstige Simulationen durchzuführen, benötigt man eine größere Menge an Testgraphen. Da die Graphenbeispiele aus der realen Welt begrenzt sind, versucht man, zufällige Graphen automatisch zu erstellen. Das einfachste dieser Modelle ist das Erdős–Rényi-Modell, das als Parameter die Anzahl der Knoten n und eine Wahrscheinlichkeit p bekommt. Es generiert einen Graphen mit n Knoten, wobei jede Kante unabhängig von den anderen mit Wahrscheinlichkeit p hinzugefügt wird (es gibt noch eine zweite Art des Modells, in dem man statt p die Anzahl der zufällig einzufügenden Kanten als Parameter hat; dieses wird jedoch deutlich seltener verwendet).

Reale Graphen haben allerdings charakteristische Eigenschaften. Bekannt ist die Hypothese, dass zwei Menschen auf der Welt sich stets über höchstens 6 Ecken kennen. Auch wenn diese nicht exakt zutrifft, ist dies eine gute Näherung. Reale Netzwerke haben üblicherweise einen kleinen Durchmesser. Außerdem tendieren sie zur Bildung von "Hubs", d.h. Knoten mit außergewöhnlich hohem Grad, wobei die Verteilung durch ein Potenzgesetz beschrieben wird. Zusätzlich besitzen die Graphen eine hohe Clusterung. Diese Eigenschaften werden gemäß der Analogie auch als Kleine-Welt-Eigenschaften bezeichnet. Da das Erdős–Rényi-Modell diese nicht beachtet, sucht man für diese Anwendung nach anderen Modellen.

2 Algorithmen

Das Watts-Strogatz-Modell (siehe https://en.wikipedia.org/wiki/Watts_and_Strogatz_model) bekommt drei Parameter ($nNodes$, $nNeighbors$ und p) und startet mit einem regulären Ringgitter, also einem Ring aus $nNodes$ Knoten, wobei jeder Knoten mit den $nNeighbors$ Nachbarn auf beiden Seiten verbunden wird. Danach werden die Kanten $\{u, v\}$ durchgegangen und mit Wahrscheinlichkeit p wie folgt geändert (sei hierfür $u < v$): Es wird ein zufälliger Knoten $w \neq u$ gewählt, der nicht zu u adjazent ist. Die Kante $\{u, v\}$ wird entfernt und die Kante $\{u, w\}$ neu eingefügt.

Das Dorogovtsev-Mendes-Modell (arXiv:cond-mat/0106144) startet mit einem Dreieck. Nun werden n Knoten nacheinander eingefügt, indem eine Kante gewählt wird, und der neue Knoten mit den beiden Endpunkten verbunden wird.

Das Forest-Fire-Modell (arXiv:physics/0603229) startet mit einem einzelnen Knoten. Die Knoten werden hintereinander eingefügt. Dabei wird der neue Knoten mit einem zufälligen Knoten verbunden. Von dort aus werden weitere Knoten gewählt: Für jeden gewählten Knoten wird über eine geometrische Verteilung (mit Mittelwert $\frac{p}{1-p}$) bestimmt, wie viele noch nicht besuchte Nachbarn zufällig gewählt werden. Mit den gewählten Knoten geht das ganze so weiter. Der neu hinzugefügte Knoten wird dann mit allen verbunden.

3 Bemerkungen zur Implementierung

Die bisherigen Generatoren sind entweder statisch oder dynamisch. Die Generatoren einer Gruppe haben grundsätzlich die gleiche Signatur. Die Parameter werden im Konstruktor übergeben. Statische Generatoren haben die Methode "Graph generate()", während dynamische Generatoren die Methode "std::vector<GraphEvent> generate(count nSteps)" haben. An diese Konvention halten sich auch die neuen Klassen.

Für das Watts-Strogatz-Modell wurde die Generierung des regulären Ringgitters in eine eigene Klasse ausgelagert, da diese auch ohne das Watts-Strogatz-Modell sinnvoll ist. Für das Ändern der Kanten werden für jeden Knoten zuerst die inzidenten Kanten gespeichert, weil es sonst durch das

Ändern zu Konflikten kommen kann. Eine Iteration über alle Kanten eines Knotens kann dabei fehlerhaft sein.

Das Dorogovtsev-Mendes-Modell war schon implementiert und sollte nur noch in den experimentellen Vergleich mit einbezogen werden.

Beim Forest-Fire-Modell musste entschieden werden, wie die Knoten weiterverfolgt werden. Das Modell gibt dazu keine Aussage. Um der Analogie eines Feuers möglichst gerecht zu werden, sollte dies für alle in einem Schritt gewählten Knoten parallel passieren. Da dies nicht sinnvoll mit einfachen Mitteln zu implementierten ist, wurde eine modifizierte Breitensuche verwendet. Die geometrische Verteilung wird nicht explizit generiert, stattdessen wird mit ein neuer Knoten mit Wahrscheinlichkeit p gewählt. Bei einem Fehlschlag ist der Knoten abgearbeitet. Dadurch ergibt sich eine geometrische Verteilung. Für das Auswählen der Nachbarn werden alle noch nicht besuchten Nachbarn in einen std::vector geschrieben. Dann wird ein Index gewählt, dieser Knoten genommen und durch den letzten Eintrag des vectors ersetzt.

4 Experimentelle Auswertung der Algorithmen

Durchschnittliche Pfadlänge ist von einem fest gewählten Knoten zu allen anderen berechnet (da die Berechnung für jedes Paar zu lange gehen würde). Außerdem ist angegeben, wie gut die Gradverteilung einem Potenzgesetz folgt (1 wäre exakte Übereinstimmung). Laufzeiten sind geschätzt; das Programm wurde auf der NetworKit-VM laufen gelassen.

Watts-Strogatz (nNeighbors = 4, p = 0.3)

n	m	Durchschnittl. Pfadlänge	Potenzgesetz (Korrelation)	Laufzeit
100	400	3.05	0.211	$<$ 1 s
1000	4000	4.16	0.384	$<$ 1 s
10000	40000	5.32	0.319	$<$ 1 s
100000	400000	6.77	0.450	$<$ 1 s
1000000	4000000	8.26	0.152	3 s

Statisches Dorogovtsev-Mendes

n	m	Durchschnittl. Pfadlänge	Potenzgesetz (Korrelation)	Laufzeit
100	197	3.98	0.9081278604241279	$<$ 1 s
1000	1997	4.41	0.979	$<$ 1 s
10000	19997	6.85	0.983	$<$ 1 s
100000	199997	12.47	0.984	$<$ 1 s
1000000	1999997	12.96	0.984	2 s

Dynamisches Dorogovtsev-Mendes

n	m	Durchschnittl. Pfadlänge	Potenzgesetz (Korrelation)	Laufzeit
100	197	1.96	0.998	$<$ 1 s
1000	1997	1.99	0.999	$<$ 1 s
10000	19997	1.99	0.999	$<$ 1 s
100000	199997	1.99	0.999	$<$ 1 s
1000000	1999997	1.99	0.999	2 s

Forest Fire (p = 0.3)

n	m	Durchschnittl. Pfadlänge	Potenzgesetz (Korrelation)	Laufzeit
100	129	6.61	0.932	$<$ 1 s
1000	1527	11.90	0.942	$<$ 1 s
10000	15407	8.98	0.951	$<$ 1 s
100000	156831	17.25	0.957	1 s
1000000	1562826	14.17	0.961	7 s

5 Fazit und Ausblick

Die drei Modelle zur Generierung von Graphen verhalten sich grundlegend unterschiedlich.

Das Watts-Strogatz-Modell ist hauptsächlich ein statisches Modell. Die Anzahl der Knoten und Kanten ist von vorne herein fest, es können keine neuen Knoten hinzugefügt werden. Die entstehenden Graphen zeigen eine hohe Clusterung und kurze durchschnittliche Pfadlänge, allerdings bildet sich eine zu gleichmäßige Gradverteilung. Damit generiert dieses Modell keine guten Kleine-Welt-Graphen.

Das Dorogovtsev-Mendes-Modell erstellt aufgrund seiner Konstruktion stets planare Graphen. Die durchschnittliche Pfadlänge ist relativ kurz, der Clusterkoeffizient ist hoch. Zudem zeigt das Modell die für reale Graphen typische Gradverteilung nach dem Potenzgesetz. Es generiert also tatsächlich Kleine-Welt-Graphen. Allerdings ist die Kantenzahl m stets $2n - 3$, also nicht zufällig.

Das Forest-Fire-Modell erstellt wie das Dorogovtsev-Mendes-Modell auch Kleine-Welt-Graphen. Im Gegensatz zu diesem jedoch wächst die Kantenzahl erwartet superlinear in der Anzahl Knoten. Außerdem ist der Graph nicht zwingend planar, und die Kantenzahl ist ebenfalls zufällig. Im Gegensatz zum Dorogovtsev-Mendes-Modell kann beim Forest-Fire-Modell der Abstand zwischen zwei Knoten über die Zeit kürzer werden.

Als zukünftige Erweiterung bietet es sich insbesondere an, das Forest-Fire-Modell auf gerichtete Graphen zu erweitern. Dies ist jedoch erst dann möglich, nachdem der Projekt-Fork in den dev-Branch von NetworKit gemergt wird, da NetworKit zur Zeit des Forks noch keine funktionierende Datenstruktur für gerichtete Graphen besessen hat.