Beating the Worst Case
Wintersemester 2023/2024
Jean-Pierre von der Heydt & Marcus Wilhelm

Karlsruher Institut für Technologie

Übungsblatt 2: Mehr Graphen

Ausgegeben am 29.11.2023. Geschätzter Zeitaufwand: 2 Wochen

Auf den vorherigen Übungsblättern haben wir zwei Netzwerkeigenschaften kennengelernt, welche helfen die Performance unserer Algorithmen vorherzusagen. Auf diesem Übungsblatt geht es darum mehr Graphen zu sammeln, die ein breiteres Spektrum an Eigenschaften abdecken. Dafür wollen wir sowohl Graphen generieren, als auch echte Netzwerke analysieren.

Aufgabe 1: Graphen generieren

Da Instanzen aus der realen Welt selten sind, ist es oft nützlich, einen Algorithmus auf zufällig erzeugten Instanzen auszuführen, um seine Skalierbarkeit zu untersuchen. Dies hat auch den Vorteil, dass die Parameter des Modells (z.B. Anzahl der Knoten und mittlerer Grad) selbst bestimmt werden können. In dieser Aufgabe wollen wir unsere Sammlung von Testinstanzen erweitern, um robustere Aussagen treffen zu können.

Es gibt eine Vielzahl von Modellen zur Erzeugung von Graphen, die je nach Anwendungsfall Vor- und Nachteile haben. Informiere dich über die gängigsten Modelle und erzeuge mit ihnen Testinstanzen. Ein guter Ausgangspunkt dafür sind die Graphengeneratoren verschiedener Frameworks, wie zum Beispiel NetworKit[1], NetworkX[2] oder igraph[3].

Suche dir fünf Netzwerkmodell heraus und generiere jeweils 100 zufällige Graphen mit diesen. Untersuche die Netzwerke entsprechend den vorherigen Übungsblättern.

Aufgabe 2: GIRG Generator

Im Framework habe wir bereits einen Generator[4] für sogenannte GIRGs (geometric inhomogenous random graphs) eingebunden. Der Generator kommt als C++ Library mit Command Line Interface für GIRGs und für Hyperbolische Zufallsgraphen. Das GIRG-Modell hat verschiedene Parameter. Welchen Einfluss haben die Parameter `ple` und `alpha` auf die Performanz der Algorithmen? Kannst du Netzwerke mit hoher Heterogenität und wenig Lokalität generieren? Gibt es Parameter für die sich GIRGs ähnlich wie die Generatoren aus Aufgabe 1 verhalten? Kannst du vielleicht sogar herausfinden, wie wir die Graphen auf dem nullten Übungsblatt generiert haben?

Hinweis 1: `ple` ist in dem Command Line Interface auf das Intervall $(2, 3]$ beschränkt. Werte die größere als 3 sind, sind aber auch sinnvoll. Um größere Werte zu verwenden musst du ggf. das

[1] `https://networkit.github.io/`
[2] `https://networkx.org/`
[3] `https://igraph.org/`
[4] `https://github.com/chistopher/girgs/`

Command Line Interface anpassen oder die C++ Bibliothek verwenden.

Hinweis 2: Der Parameter `alpha` ist auf $(1, \infty]$ beschränkt. Wenn du hier verschiedene Werte ausprobiert, dann denk eher über `1/alpha` nach. Also: Es ist zunächst hilfreich Werte zu wählen, sodass die zugehörigen `1/alpha` halbwegs gleichmäßig in $[0, 1)$ verteilt sind.

Aufgabe 3: Echtweltnetzwerke

In der Praxis hat man es natürlich nicht oft mit generierten Graphen zu tun und die reale Welt ist selten so ideal wie unsere Modelle. Verschieden gute Sammlungen von Netzwerken findet man zum Beispiel bei Network Repository[5], Konect[6] und auf einer Webseite des Lehrstuhls[7].

Was ist mit der Heterogenität und der Lokalität dieser Graphen? Können wir die Erkenntnisse über Graphparameter und Zufallsgraphen auch auf Realweltgraphen übertragen? Wiederhole die bisherigen Experimente auf einer großen Menge von Echtweltnetzwerken.

[5] `https://networkrepository.com/`
[6] `http://konect.cc/networks/`
[7] `https://external-validity.iti.kit.edu/`