Engineering Bipartite Global-Curveball

Marius Hagemann

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

1 Einleitung

Bei der Analyse komplexer Netzwerke, wie beispielsweise soziale Netzwerke, werden die zugrundeliegenden Graphen häufig mit zufälligen Graphen verglichen, um deren Struktur zu untersuchen [3]. Zum Erzeugen von zufälligen Graphen existieren diverse Modelle wie beispielsweise der Erdős-Rényi-Graph [4] oder der Gilbert-Graph [5]. Diese Graphen weisen jedoch in der Regel kaum eine Ähnlichkeit zu dem zu analysierenden Netzwerk auf. Deshalb verwendet man Zufallsgraphen, bei welchen **jeder Knoten denselben Grad wie im originalen Graphen hat**. Curveball [2] und Global-Curveball [3] sind Lösungsansätze hierfür.
Ziel dieser Bachelorarbeit ist die Anpassung von Global-Curveball an bipartite Graphen zur Reduzierung der Laufzeit. Der entwickelte Algorithmus wird Teil des OpenSource Projekts NetworKit [1] werden.

2 Grundlagen

Curveball ist ein Prozess, bei dem Kanten zufällig getauscht werden. Bei einem Curveball-Tausch werden zwei verschiedene Knoten u und v zufällig uniform verteilt ausgewählt und deren disjunkte Nachbarschaft zufällig durchmischt. Dabei muss im Allgemeinen darauf geachtet werden, dass durch das Tauschen keine Eigenschleifen oder Multikanten entstehen und dass die Bipartitheit nicht verletzt wird. Ein Beispiel ist in Abbildung 1 gegeben.

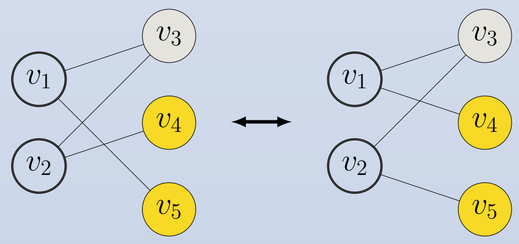

Abbildung 1: Auf einem der beiden Graphen wird ein Curveball-Tausch auf den Knoten v_1 und v_2 ausgeführt. Die gemeinsame Nachbarschaft ist in grau gekennzeichnet, die disjunkte Nachbarschaft in gelber Farbe. In diesem Beispiel gibt es nur die zwei dargestellten Graphen, die durch Tauschen der disjunkten Nachbarschaft entstehen können.

Ein **Global-Curveball Tausch** besteht aus mehreren Curveball-Tauschen, wobei möglichst jeder Knoten Teil eines solchen Curveball-Tausches sein soll. Im Fall von bipartiten Graphen werden die Curveball-Tausche lediglich auf den Knoten aus einer der beiden Partitionsklassen (diese wird **aktiv** genannt) ausgeführt. Somit wird verhindert, dass durch einen Global-Curveball Tausch Eigenschleifen entstehen oder die Bipartitheit verletzt wird. Auch werden dadurch unnötige Curveball-Tausche vermieden, welche den Graph nicht verändern würden. Auf Grund der Bipartitheit überschneiden sich die einzelnen Curveball-Tausche nicht und können daher vollständig parallel ausgeführt werden.

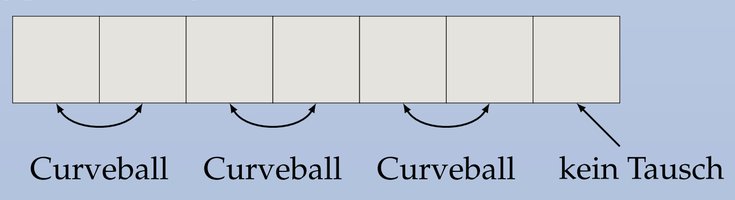

Abbildung 2: Global-Curveball auf dem zufällig permutierten Array der aktiven Partition. Bei ungerader Anzahl wird ein Knoten ausgelassen.

3 Implementierung eines Curveball-Tausches

Für einen Curveball-Tausch muss also die gemeinsame und disjunkte Nachbarschaft bestimmt und die disjunkte Nachbarschaft zufällig durchmischt werden. Die Nachbarschaften eines jeden Knotens sind jeweils in einem Array gespeichert. Dies ist in Abbildung 3 skizziert.

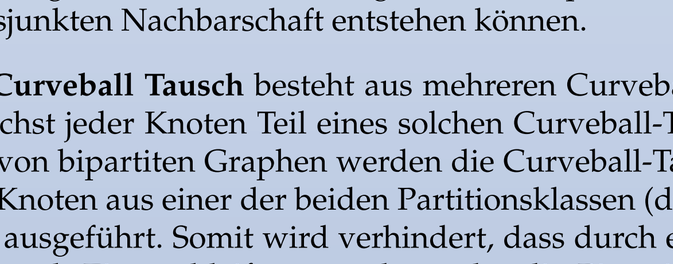

Abbildung 3: Skizze eines Curveball-Tausches auf den Arrays

Zur Bestimmung der gemeinsamen und disjunkten Nachbarschaft werden insgesamt 7 verschiedene Methoden betrachtet, zum zufälligen Tauschen 2 weitere. Weiterhin wird geprüft, ob es zu einem Laufzeitvorteil führt, die Arrays der Nachbarschaften immer sortiert zu halten. Auf diese Weise entstehen insgesamt 28 verschiedene Methoden einen Curveball-Tausch durchzuführen. Durch eine experimentelle Untersuchung auf verschiedenen Instanzen wird bestimmt, welche Methode das beste Laufzeitverhalten aufweist.

4 Experimentelle Untersuchung

In Abbildung 4 lassen sich die Laufzeiten der verschiedenen Varianten vergleichen. Dabei wird für jede Instanz **das Verhältnis der Laufzeit der jeweiligen Variante zur geringsten Laufzeit berechnet.** Die einzelnen Slowdowns sind für jede Variante aufsteigend sortiert und als Kurve eingezeichnet. Bei einer guten Methode verläuft die Kurve möglichst lange nahe der 1 und steigt nicht stark an. Die beiden besten Methoden sind **markiert**.

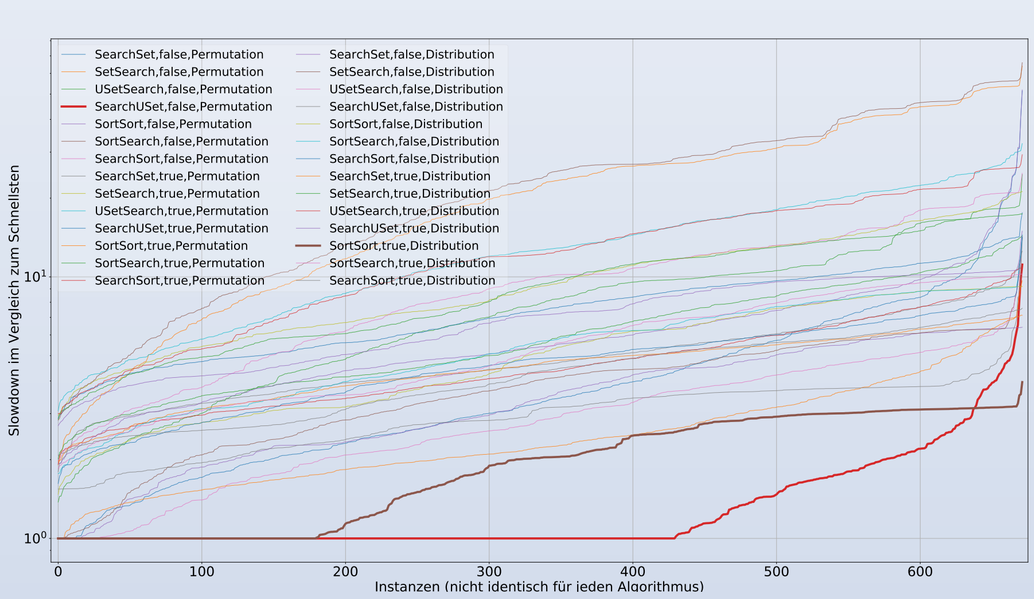

Abbildung 4: Slowdown der einzelnen Varianten im jeweiligen Vergleich zur Variante mit der geringsten Laufzeit.

Die Abbildung 5 zeigt einen direkten Vergleich der zwei besten Methoden auf ausgewählten Instanzen. Während die Laufzeiten bei den meisten Instanzen nahezu identisch sind, existieren einige Instanzen, bei denen die in blau gekennzeichnete Variante eine deutlich geringere Laufzeit aufweist. Deswegen wird diese Variante ausgewählt.

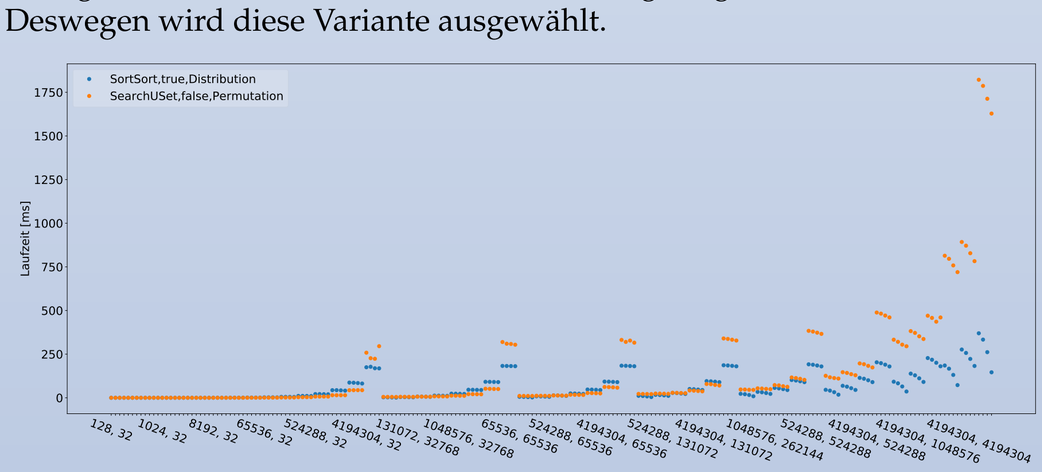

Abbildung 5: Vergleich der Laufzeiten der zwei besten Varianten. Auf der horizontalen Achse sind die Instanzen mit der jeweiligen Größe der beiden Arrays aufgetragen.

5 Ergebnis

Abschließend wird der neu entwickelte bipartite Global-Curveball mit dem schon existierenden Curveball **auf verschiedenen Graphen** verglichen. Die gemessenen Laufzeiten sind in Abbildung 6 dargestellt. Auf den Instanzen mit weit mehr als 10.000 Knoten zeigt der bipartite Global-Curveball eine deutlich bessere Laufzeit. Es wird ein Speedup von bis zu 17 erreicht. Dies liegt vor allem an der Parallelität im bipartiten Global-Curveball. Aber auch ohne die Parallelisierung erreicht der sequenzielle bipartite Global-Curveball einen Speedup von ungefähr 2.

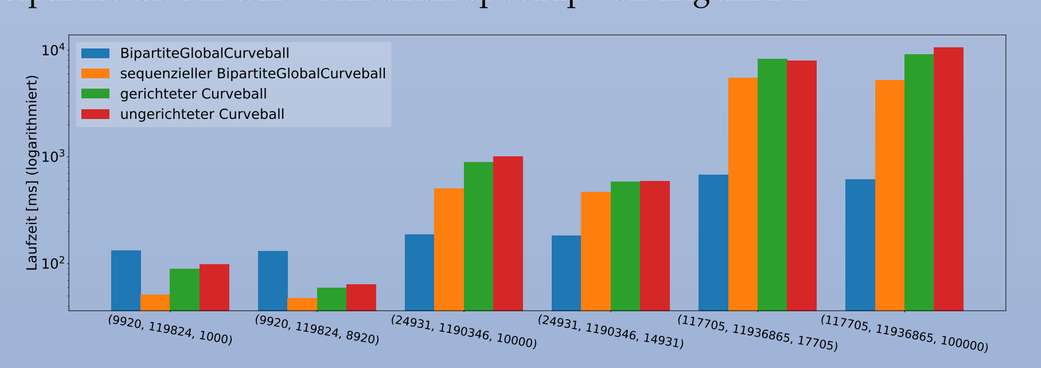

Abbildung 6: Vergleich von bipartitem Global-Curveball und Curveball. Als Beschriftung dient die Anzahl an Knoten, Kanten und aktiven Knoten.

Literatur

[1] NetworKit. `https://networkit.github.io/`. Abgerufen am 04.03.2020.

[2] C. J. Carstens, A. Berger, and G. Strona. Curveball: a new generation of sampling algorithms for graphs with fixed degree sequence. *CoRR*, abs/1609.05137, 2016.

[3] C. J. Carstens, M. Hamann, U. Meyer, M. Penschuck, H. Tran, and D. Wagner. Parallel and I/O-efficient randomisation of massive networks using global curveball trades. 112:11:1–11:15, 2018.

[4] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 1959.

[5] E. N. Gilbert. Random graphs. *Annals of Mathematical Statistics*, 30(4):1141–1144, 12 1959.