Praktikumsbericht

Partielles Zoomen in großen Netzwerkgraphen

Julian Walter

Abgabedatum:	21. September 2019
Betreuer:	Prof. Dr. Alexander Wolff
	Johannes Zink M.Sc.
	Dr. David Hock
	Fabian Lipp M.Sc.

Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen, Komplexität und wissensbasierte Systeme

Zusammenfassung

Dieser Praktikumsbericht befasst sich mit Möglichkeiten zum partiellen Zoomen in Netzwerkgraphen. Im Rahmen des zugehörigen Praktikums wurden Ideen für Algorithmen hierzu und mögliche Visualisierungsvarianten gesammelt und verglichen. Hiervon wurde eine Variante ausgewählt und mit Hilfe der Programmbibliothek yFiles implementiert.

Inhaltsverzeichnis

1 Einleitung **4**

2 Literaturüberblick **6**

3 Visualisierung **13**
3.1 Zoomen, dann clustern 13
3.2 Hierarchisches Fisheye 15
3.3 Fokusbereich + Randbereich 16
3.4 Ebenen durch Rahmen 18
3.5 Fokusebene + Verbindungsknoten 19
3.6 Zu allen Ansätzen 21

4 Umsetzung **22**
4.1 Änderung der sichtbaren Knoten 22
4.2 Änderung der sichtbaren Kanten 23
4.3 Visualisierung 24
4.4 Tests 25

5 Ausblick **27**

1 Einleitung

Ziel der Arbeit ist es, einen Algorithmus zum partiellen Zoomen – ähnlich einer Fisheye-Ansicht – in einen großen Graphen zu entwickeln und zu implementieren. In der Fotografie wird mit einem Fisheye-Objektiv ein großer Bereich abgebildet – meist 180°-Ansichten. Hierbei wird der mittlere Bereich, welcher im Fokus ist, nahezu unverzerrt dargestellt und das Bild zu den Seiten hin gestaucht um den großen Bereich abbilden zu können. Angewendet auf einem Graphen kann das beispielsweise bedeuten, den fokussierten Bereich übersichtlich und detailliert zu halten und die Randbereiche zu stauchen oder zu clustern. Hintergrund war die Überlegung, einen Graphen zu visualisieren, welcher das komplette IT-Netzwerk eines Unternehmens beinhaltet, also alle Rechner und deren Verkabelung. Bei global agierenden Unternehmen kann das allein für einen Standort schon weit mehr Knoten und Kanten bedeuten, als auf einem 1080p-Display oder einer DIN A4-Seite übersichtlich dargestellt werden können.

Hierbei war der Wunsch, eine Möglichkeit zu finden, in Teile des Graphen hineinzoomen zu können, wobei jederzeit der komplette Graph zur Orientierung sichtbar bleibt. Der Gedanke dahinter ist, dass sich die „mental map“ des Benutzers beim Zoomen erhält. Damit zusammenhängend ist das Problem des Clusterns der Knoten, wobei auch Attribute von Knoten oder Kanten eine Rolle spielen können, welche vom Benutzer festgelegt sein können. Der Benutzer sollte angeben können, wie weit er einen Superknoten oder eine Teilmenge von Knoten vergrößern oder verkleinern möchte. Dies soll durch die Repräsentation von Teilen des Graphen durch einzelne Knoten – wie z. B. einen Knoten, repräsentativ für Europa, während ein Standort in den USA näher betrachtet wird – oder eine passend beschriftete, aus dem angezeigten Bereich herausführende Kante erreicht werden. Eine weitere Option für den Nutzer könnte ein „Knoten-Klick“ darstellen, wobei beim Klick auf einen Superknoten dieser ausgeklappt und auf dessen Unterknoten gezoomt wird.

Ebenfalls ist es wichtig darauf zu achten, dass die Gesamtmenge der sichtbaren Knoten nicht zu groß wird, um die Übersichtlichkeit zu gewährleisten. Hierzu könnte beispielsweise ein Schwellwert eingeführt werden; wird dieser beim Ausklappen eines Knotens überschritten, müssen am Rand Knoten zusammengefasst werden. Der Maßstab soll sich also beim Zoomen nicht gleichmäßig auf dem ganzen Graphen verändern, stattdessen sollten die wichtigen Gebiete zulasten der unwichtigen vergrößert bzw. detaillierter dargestellt werden.

Spezifizierung der Aufgabenstellung. Die Aufgabe des Praktikums bestand im Finden und Implementieren eines Algorithmus, welcher bei Eingabe eines großen Graphen eine übersichtliche Darstellung dieses Graphen durch Zusammenfassen der Knoten des nicht fokussierten Bereichs zu Clustern berechnet. Ein großer Graph bedeutet hier, ein Graph mit mehr Knoten als übersichtlich auf einem 1080p-Display oder auf einer DIN A4 Seite dargestellt werden können. Eine übersichtliche Darstellung beinhaltet, dass einzelne Knoten und Kanten mit bloßem Auge eindeutig identifiziert werden können und genügend Platz für eine lesbare Beschriftung vorhanden ist.

Übersicht. In Abschnitt 2 „Literaturüberblick“ werden einige Paper vorgestellt, welche sich mit diesem Thema auseinandersetzen. Abschnitt 3 „Visualisierung“ beschäftigt sich mit Möglichkeiten der Darstellung des Zoomens – dazu zählen unter anderem die Repräsentation der zusammengefassten Knoten und Übergangsanimationen. In Abschnitt 4 „Umsetzung“ ist die Implementierung des ausgewählten Modells näher beschrieben. In Abschnitt 5 „Ausblick“ werden Ideen vorgestellt, wie auf diese Arbeit aufgebaut werden kann.

2 Literaturüberblick

Tominski et al. befassen sich in ihrem Paper „Fisheye Tree Views and Lenses for Graph Visualization“ [TAVHS06] damit, wie ein Teilbereich eines großen Graphen durch Zoomen übersichtlich dargestellt werden kann. Hierzu haben sich die Autoren mit unterschiedlichen Lupenfunktionen auseinandergesetzt, welche jeweils den fokussierten Bereich auf unterschiedliche Art und Weise übersichtlicher darzustellen versuchen. Diese sind die „Local Edge Lens“, die „Bring Neighbors Lens“ und eine „Composite Lens“, welche beide anderen Lupenfunktionen und zusätzlich gleichzeitig eine traditionelle Fisheye-Lupenfunktion bereitstellt.

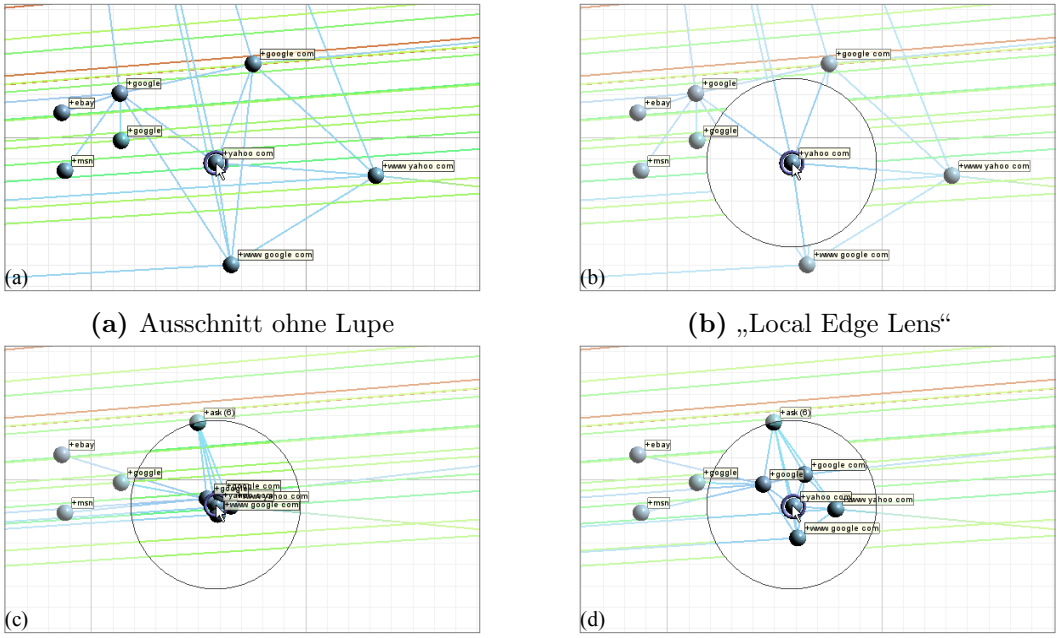
Zu sehen ist deren Wirkungsweise am besten in der Graphik aus ihrem Paper, welche in Abbildung 1 zu sehen ist. Abbildung 1b zeigt die „Local Edge Lens“, die alle Kanten, welche den fokussierten Bereich schneiden, ausblendet. Nur Kanten, welche von und zu Knoten innerhalb des fokussierten Bereichs führen, werden angezeigt. Die „Bring Neighbors Lens“, zu sehen in Abbildung 1c verschiebt temporär alle Endknoten von Kanten, welche aus dem fokussierten Bereich heraus führen, in den fokussierten Bereich hinein. In Abbildung 1d ist die „Composite Lens“ dargestellt, welche die beiden anderen Lupenfunktionen und die Fisheye-Ansicht kombiniert.

Eine andere Variante „Bring Neighbors Lens“ wird in dem Paper „Topology-Aware Navigation in Large Networks“ [MCH+09] von Moscovich et al. vorgestellt. Deren Lupenfunktion nennt sich „Bring & Go“, zu sehen in Abbildung 2, welche eine Graphik von Moscovich et al. zeigt. Hier ist die „Bring & Go“-Funktion am Beispiel von Flügen von und nach Sydney visualisiert. Die anderen Start- bzw. Zielflughäfen werden in den Bereich um Sydney herum zusammen gezogen. Diese Funktion unterscheidet sich von der „Bring Neighbors Lens“ dahingehend, dass hier die Knoten auf festen Radien um den Fokusknoten herum angeordnet werden, welche gleichzeitig den relativen Abstand zeigen (zu sehen in Abbildung 3). Flughäfen, welche sehr weit von Sydney entfernt liegen, werden auf den äußersten Ring verlegt, während Knoten, welche Flughäfen nahe Sydney repräsentieren auf den innersten Ring gezeichnet werden.

Eine Übersicht über einige ähnliche Arbeiten, wie die zwei bereits hier vorgestellten, wurde von Tominski et al. in deren Artikel „A Survey on Interactive Lenses in Visualization“ [TGK+14] zusammengetragen. In diesem sind einige weitere Arbeiten vorgestellt, welche sich mit unterschiedlichen Lupenfunktionen befassen.

Eine andere, verbreitete Herangehensweise ist, zu versuchen den Graphen in Cluster zu unterteilen und durch Nutzung dieser Informationen den gesamten Graphen übersichtlicher darzustellen. Hierzu gibt es Arbeiten wie „Large Graph Visualization by Hierarchical Clustering“ [HN08] von Huang und Nguyen, die klassisch die Knoten clustern sowie Arbeiten wie „Geometry-Based Edge Clustering for Graph Visualization“[CZQ+08] von Cui et al. welche sich mit dem Zusammenfassen von Kanten beschäftigen, um den Graph übersichtlicher zu machen.

In ihrem Paper „Interactive Visualization of Small World Graphs“ [VHVW04] haben sich die Autoren Frank van Ham und Jarke J. van Wijk mit dem Problem der dreidimensionalen Darstellung eines Graphen befasst, der zu groß ist um ihn übersichtlich mit allen Knoten und Kanten zu visualisieren. Ihre Herangehensweise lässt sich aber unverändert



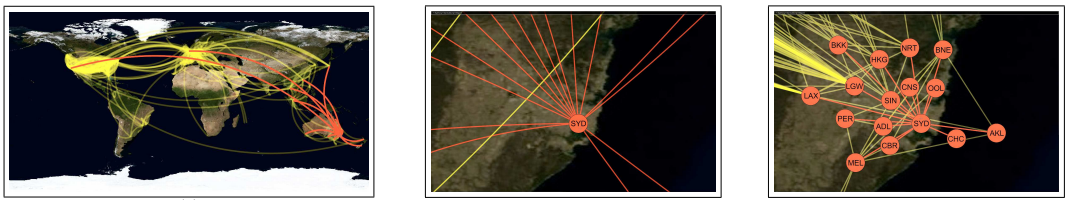
(a) Ausschnitt ohne Lupe

(b) „Local Edge Lens“

(c) „Bring Neighbors Lens“

(d) „Composite Lens“

Abb. 1: Graphiken aus dem Paper „Fisheye Tree Views and Lenses for Graph Visualization“ [TAVHS06]; Auswirkungen der verschiedenen vorgestellten Lupen am Ausschnitt eines Beispielgraphen



(a) Gesamte Weltkarte

(b) Ausschnitt ohne Lupe

(c) mit „Bring & Go“

Abb. 2: Graphiken aus dem Paper „Topology-Aware Navigation in Large Networks“ [MCH+09]; Beispiel eines Graphen welcher Flugrouten repräsentiert; „Bring & Go“-Funktion am Beispiel des Kartenausschnitts Sydney

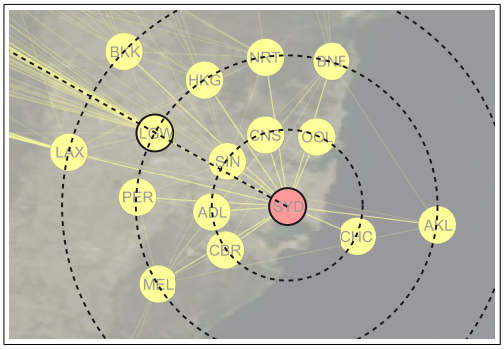


Abb. 3: „Bring & Go“ am Beispiel Sydney[MCH+09]; herangezogene Knoten werden auf festen Radien um den Sydney-Knoten herum angeordnet

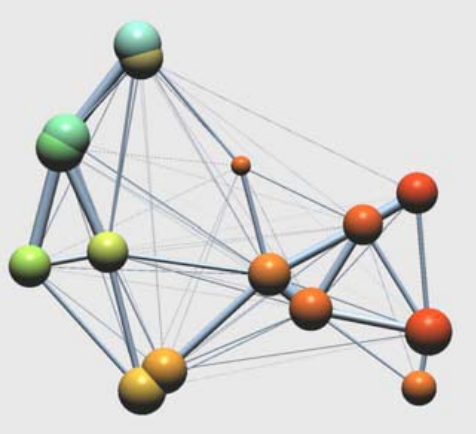
auch im Zweidimensionalen anwenden; sie ist die Folgende: Zuerst werden alle Knoten des Graphen zu hierarchischen Clustern zusammengefasst. Danach wird der Graph nur mit den Superknoten, welche die größten Cluster repräsentieren, visualisiert. Für die Details wurde eine Art Lupe entworfen, genannt „fisheye lens“, welche eine Fisheye-Ansicht auf der Abstraktionsebene realisiert. Das bedeutet, dass Superknoten, die „unter die Lupe genommen werden“, entpackt werden und somit im mittleren Bereich der Lupe alle Knoten zu sehen sind. In einem festgelegten Gebiet um diesen Bereich steigt die Abstraktion nach außen hin an, sodass weiter innen kleinere Cluster niedrigerer Hierarchiestufen abgebildet sind, welche weniger Knoten repräsentieren, und nach außen hin immer größere. Außerhalb dieses Bereichs sind nur noch Superknoten der höchsten Hierarchiestufe dargestellt.

Der theoretische Aufbau dieser Bereiche ist in Abbildung 4b zu sehen. Der Fokuspunkt ist durch f markiert. Um ihn herum gibt es den hellblau markierten, inneren Bereich, gekennzeichnet durch den Buchstaben A, in welchem Knoten des ursprünglichen Graphen zu sehen sind. Danach kommt der Übergangsbereich B in welchem die Abstraktion ansteigt und nach außen hin immer mehr Knoten zu Clustern zusammengefasst werden. Der Rest ist der Bereich C, in welchem ausschließlich Superknoten dargestellt sind.

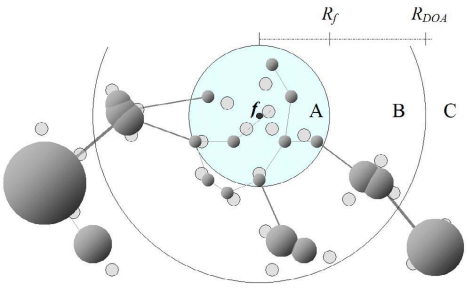
Anschaulich dargestellt sind die Auswirkungen der „fisheye lens“ am Beispielgraphen in Abbildung 4a. Abbildung 4c zeigt, wie sich dieser Graph unter Verwendung der Lupenfunktion bei unterschiedlichen Fokuspunkten verändert.

In vielen Arbeiten zum Thema Fisheye-Ansicht wird der Artikel „Generalized Fisheye Views“ [Fur86] von George W. Furnas zitiert. Auf dem darin vorgestellten Berechnungsansatz baut auch das Programm Semnet [FPF13] von Fairchild, Poltrock und Furnas auf, welches ebenfalls oft erwähnt wird. In seinem Artikel beschreibt Furnas zum einen ausführlich warum eine Fisheye-Ansicht sinnvoll ist und stellt dann eine Berechnung einer solchen Ansicht mithilfe des „Degree of Interest“ (DOI) vor. Der DOI ist ein Wert, der jedem Knoten zugeordnet wird. Je größer er ist, desto wichtiger ist der zugehörige Knoten für den Betrachter.

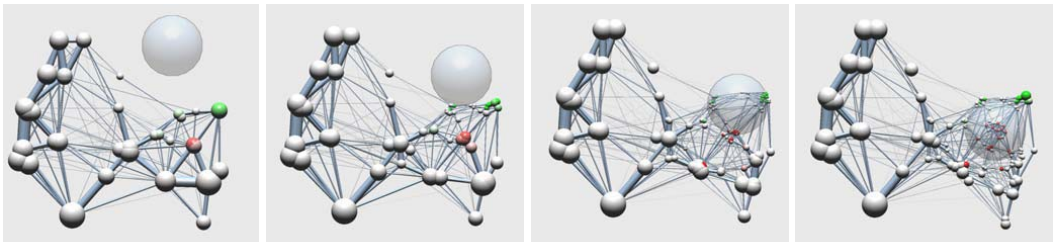
Zu sehen ist das ganze in Abbildung 5; hier ist die Berechnung des DOI am Beispiel eines Baumgraphen dargestellt. Die Berechnung ist simpel, es werden für jeden Knoten zwei



(a) Geclusterter Graph ohne Lupe



(b) Bereiche der Lupenfunktion



(c) Serie von Bildern welche den Effekt der „fisheye lens“ bei unterschiedlichen Fokuspunkten zeigt

Abb. 4: Graphiken aus dem Paper „Interactive Visualization of Small World Graphs“ [VHVW04]; Auswirkungen der „fisheye lens“ auf einen Beispielgraphen (4a & 4c) und theoretischer Aufbau der Funktion (4b)

Werte bestimmt und miteinander verrechnet. Einer der Werte, die a Priori Wichtigkeit, ergibt sich aus der Baumstruktur, abgebildet in Abbildung 5b. Hier hat der Wurzelknoten den Wert 0 und alle anderen ihre negative Distanz zum Wurzelknoten. Es wird also davon ausgegangen, dass Knoten niedrigerer Hierarchieebenen allgemein von geringerer Bedeutung für den Betrachter sind und erhalten deshalb einen Malus. Der zweite, für die Berechnung des DOI benötigte Wert ist die Distanz innerhalb des Hierarchiebaumes von einem Knoten zum Fokusknoten (zu sehen in Abbildung 5a). Der DOI eines Knotens ergibt sich aus seiner Wichtigkeit − seine Distanz zum Fokusknoten (Abbildung 5c).

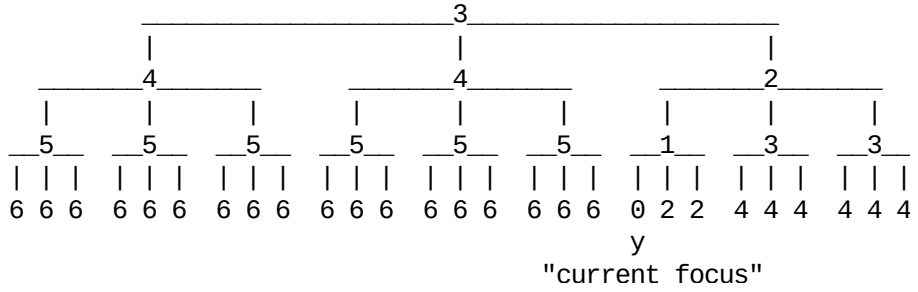
Dieser DOI kann nun verwendet werden, um bei gegebener Hierarchie und einem gegebenen Fokusknoten zu ermitteln, welche Knoten für den Nutzer vermutlich am wichtigsten sind. Das sind dann die Knoten mit dem höchsten DOI. Hierbei ist unerheblich, ob der DOI positiv oder negativ ist. Auch können die beiden Werte unterschiedlich gewichtet werden; ist es beispielsweise wichtiger mehr Knoten der unmittelbaren Nachbarschaft des Fokusknotens zu zeigen, als Knoten höherer Hierarchieebenen, könnten z. B. die Werte in Abbildung 5a verdoppelt werden oder sogar quadratisch mit mehr Abstand zum Fokusknoten wachsen. Auch muss sich die Wichtigkeit nicht aus der Baumstruktur ergeben sondern könnte auch durch Auswertung anderer Informationen zu den Knoten, zum Beispiel aus Metadaten, ermittelt werden.

Ein Ansatz, bei dem die Anzahl der sichtbaren Objekte reduziert wird, um den Graphen darstellen zu können, und gleichzeitig versucht wird die Struktur zu erhalten, wurde von Gansner, Koren und North entwickelt und in ihrem Artikel „Topological Fisheye Views for Visualizing Large Graphs“ [GKN05] beschrieben. Die Autoren gehen von Graphen mit tausenden oder Millionen zu visualisierender Objekte aus. Ihr Algorithmus arbeitet mit einer gegebenen Darstellung in Form von Knotenkoordinaten, welche durch einen beliebigen anderen Algorithmus bereitgestellt werden kann. Sie empfehlen hierbei ein kräftebasiertes Verfahren zu verwenden.

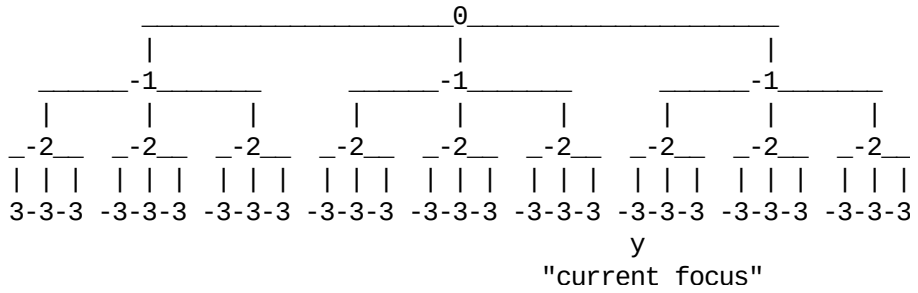
Im ersten Schritt ihres Algorithmus berechnen sie eine stufenweise Abstraktion, wobei darauf geachtet wird, dass Cluster ähnlicher Größe entstehen und nur Knoten zusammengefasst werden, welche durch wenige Schritte (maximal 2 oder 3) im Originalgraphen erreicht werden können. Sie bilden also Cluster, indem sie immer zwei Knoten auswählen und kontrahieren. Für die Auswahl der Kontraktionskandidaten werden nur Knoten berücksichtigt, welche in der gegebenen Darstellung nahe beieinander liegen. Hierzu wird eine Mischung aus „Delaunay triangulation“ und „relative neighborhood graph“ verwendet (relative neighborhood graph: p_i und p_j sind Nachbarn, wenn kein anderer Punkt näher an p_i ist als p_j und umgekehrt).

Anschließend wird eine Fisheye-Ansicht – von den Autoren „hybrid graph“ genannt – aus den Graphen der unterschiedlichen Abstraktionsstufen um einen Fokuspunkt herum zusammengebaut. Knoten nahe dem Fokuspunkt werden aus dem ursprünglichen Graphen entnommen, je weiter entfernt desto höher wird die Abstraktionsstufe. Damit der eingezoomte Bereich nicht zu unübersichtlich wird, wird jeder Knoten zunächst an seine Positionen in seinem Ursprungsgraphen gesetzt und anschließend verschoben, sodass eine gleichmäßige Dichte des Graphen entsteht.

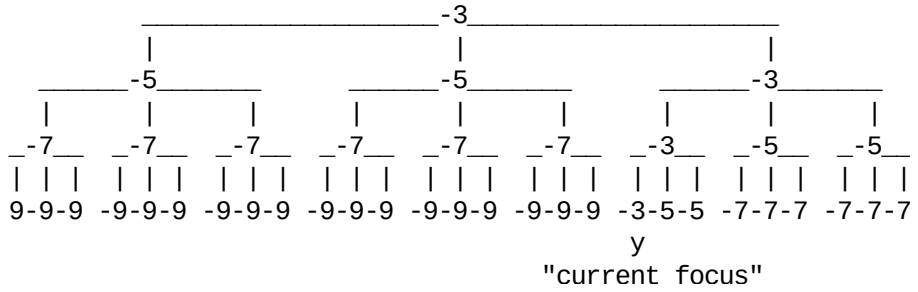
Die Vorgehensweise ist in Abbildung 6 am Beispiel des 4elt-Graphen zu sehen. Abbildung 6a zeigt den Graphen in unterschiedlichen Abstraktionsstufen mit von links nach



(a) Distanz der Knoten $x \in V$ vom Fokusknoten y; $dist(x|y) := d_{tree}(x, y)$

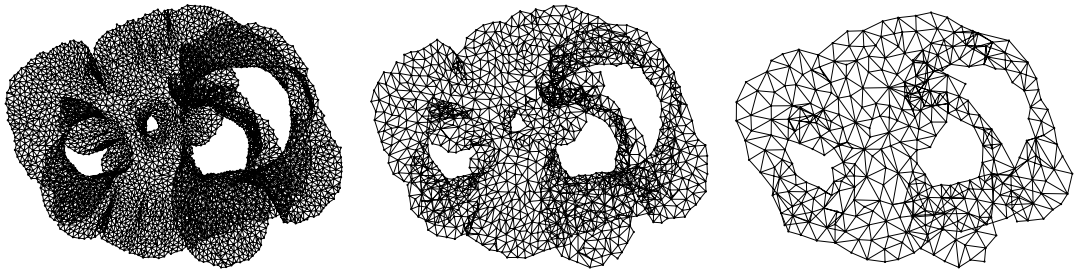


(b) A Priori Wichtigkeit der Knoten $x \in V$ im Baum: $Imp(x) := -d_{tree}(x, root)$

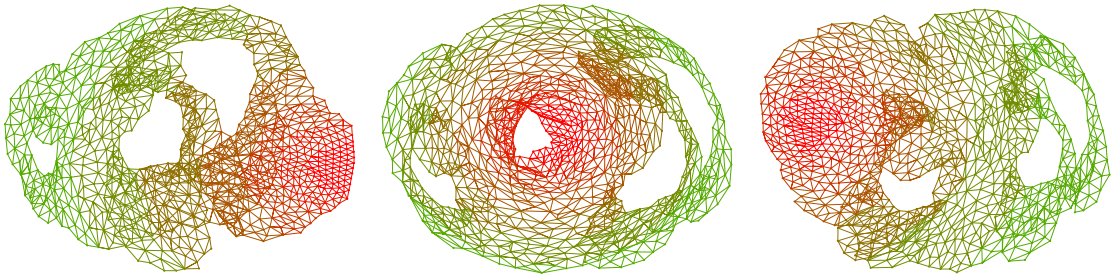


(c) Fisheye-DOI der Knoten $x \in V$ bei Fokus y; $DOI(x|y) = Imp(x) - dist(x|y)$

Abb. 5: Graphiken aus dem Paper „Generalized Fisheye Views“ [Fur86]; Dargestellt sind die drei Schritte zur Berechnung des DOI; Distanz (5a), A Priori Wichtigkeit (5b) und der DOI (5c) mit Berechnungsformeln



(a) 4elt-Graph in unterschiedlichen Abstraktionsstufen der Vorberechnung



(b) Fisheye-Ansichten des 4elt-Graphen; Fokusbereich rot markiert – Abstraktionslevel nimmt von rot nach grün ab – von links nach rechts: Fokuspunkt rechts unten, kleines Loch in der Mitte, links oben

Abb. 6: Graphiken aus dem Paper „Topological Fisheye Views for Visualizing Large Graphs“ [GKN05]; Abstraktionsstufen und Fisheye-Ansichten des 4elt-Graphen

rechts 4.394, 1.223 und 341 Knoten. In Abbildung 6b sind Fisheye-Ansichten des 4elt-Graphen mit unterschiedlichen Fokuspunkten abgebildet.

3 Visualisierung

In diesem Kapitel werden einige Vorschläge zur Visualisierung eines partiellen Zoomalgorithmus vorgestellt. In der beigefügten PowerPoint Präsentation [Wal19] sind alle Ansätze visuell dargestellt. Durch Animationen bei den Folienübergängen ist visualisiert, wie sich der Graph beim Zoomen verhalten soll. An dieser Stelle wird auf die Hintergründe, Vor- und Nachteile sowie die mögliche Umsetzung näher eingegangen.

Alle Ansätze verbindet, dass eine gegebene Hierarchie – welche unabhängig der Kanten sein kann – vorausgesetzt ist. Ebenfalls starten alle Ansätze auf der höchsten Hierarchieebene und es wird schrittweise an einen bestimmten Knoten herangezoomt. In der Präsentation und den Beispielen ist dies der Knoten, welcher Würzburg repräsentiert.

In Abbildung 7 ist der verwendete Beispielgraph ganz zu sehen. In Abbildung 7a ist die Hierarchie des Graphen erkennbar. Abbildung 7b zeigt die Startansicht für alle Ansätze, wobei der Graph bis auf die oberste Ebene zusammengepackt ist.

Einige Ansätze arbeiten mit einem Schwellwert $tmax$, der ebenfalls als gegeben vorausgesetzt wird. Bei ihnen wird die Anzahl der sichtbaren Knoten auf $tmax$ reduziert und somit eine zu unübersichtliche Darstellung vermieden.

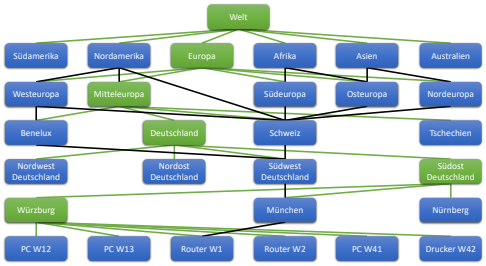
Vorgestellt werden fünf Vorschläge in folgender Reihenfolge:

- **Zoomen, dann clustern**
  Der fokussierte Knoten wird ausgeklappt; die restlichen Knoten werden zu Clustern zusammengefasst, bis weniger als $tmax$ Knoten vorhanden sind
- **Hierarchisches Fisheye**
  Der fokussierte Knoten wird ausgeklappt; die restlichen Knoten werden wieder zugeklappt, bis weniger als $tmax$ Knoten vorhanden sind
- **Fokusbereich + Randbereich**
  Der fokussierte Knoten wird vergrößert und dann in seine Kindknoten aufgespalten; die restlichen Knoten wandern in den Randbereich
- **Ebenen durch Rahmen**
  Der fokussierte Knoten wird vergrößert, sodass seine Kindknoten in ihm angezeigt werden können; ist nicht genug Platz wird der hierarchisch höchste Knoten zu einem Rahmen und seine direkten Kinder verschwinden
- **Fokusebene + Verbindungsknoten**
  Der fokussierte Knoten wird ausgeklappt; alle Knoten, welche keine direkten Nachbarn der neu hinzugekommenen Knoten sind, verschwinden

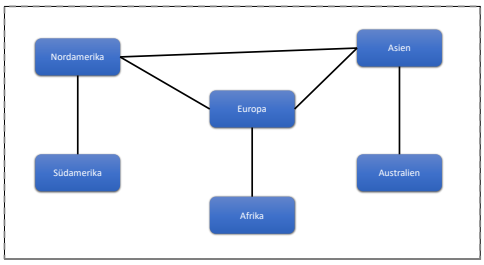
Am Ende des Kapitels werden einige mögliche Features vorgestellt, welche für alle Ansätze anwendbar sind. Ebenfalls werden Adaptionsmöglichkeiten der in der beigefügten Präsentation [Wal19] visualisierten Ansätze diskutiert.

3.1 Zoomen, dann clustern

Bei diesem Ansatz wird in jedem Schritt der fokussierte Knoten ausgeklappt. Sind anschließend zu viele Knoten sichtbar, werden Cluster gebildet und Knoten zusammengefasst, bis die Anzahl der sichtbaren Knoten kleiner als $tmax$ ist. Beim Bilden der Cluster



(a) Gesamtgraph der Beispiele; Knoten geordnet nach Hierarchieebenen



(b) Startansicht für alle Ansätze; Zoomebene Welt

Abb. 7: Graph für die Beispiele mit gegebener Hierarchie (links) und Startansicht aller Ansätze mit Zoom auf den Wurzelknoten *Welt*; Grüne Kanten zeigen Eltern-Kind-Beziehung zwischen Knoten; Schwarze Kanten entsprechen tatsächlich vorhandener Kanten im Graph; links sind Kanten auf einer Hierarchieebene weggelassen

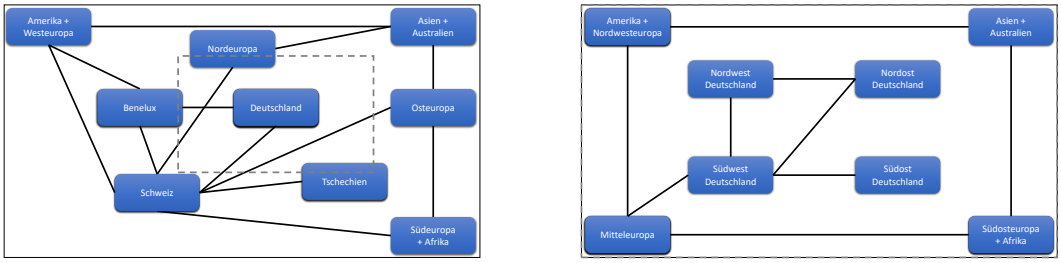
wird darauf geachtet, dass keine der durch das Ausklappen neu hinzugekommenen Knoten verwendet werden. Somit ist die aktuelle Zoomebene immer vollständig angezeigt.

Zu sehen ist die Vorgehensweise in Abbildung 8. Beim Zoomen auf Deutschland wird der „Deutschland-“Knoten zu den vier mittleren Knoten ausgepackt. Aus den restlichen Knoten und Cluster werden neue Clustern gebildet. So werden z. B. der Nordeuropa-Knoten dem Amerika-und-Westeuropa-Cluster hinzugefügt oder die Knoten „Benelux“, „Schweiz“ und „Tschechien“ zu einem „Rest-“Mitteleuropa-Cluster zusammengefasst.

Der Ansatz ist sehr flexibel, da jeder beliebige Algorithmus zum Bilden der Cluster verwendet werden kann. Um die Nutzerfreundlichkeit und vor allem Übersichtlichkeit zu gewährleisten ist es von Vorteil, einen Algorithmus zu wählen, welcher in Abhängigkeit der Knotenpositionen Cluster bildet. Dann würden bei der Übergangsanimation nicht alle Knoten durcheinander fliegen und der Benutzer könnte dem ganzen besser folgen. Ebenfalls ist es für diesen Ansatz empfehlenswert, einen stufenlosen Zoom zu verwenden. Hierdurch wird zuerst der Bereich um den fokussierten Knoten vergrößert, indem die anderen Knoten Richtung Rand verschoben werden. Hierbei kann darauf geachtet werden, dass sich Knoten, die beim Auspacken des fokussierten Knotens zusammengefasst werden sollen, aufeinander zu bewegen. Auch könnten in jedem Zoomschritt zuerst die neuen Cluster gebildet werden und dann der fokussierte Knoten ausgeklappt werden, dann würden nicht alle Animationen auf einmal geschehen.

Es ist denkbar, den Ansatz auf mehrere Fokuspunkte zu erweitern. Hierbei kann es zu Problemen kommen, wenn alle Fokusknoten gemeinsam zu viele Unterknoten haben. Entweder wird es dann sehr unübersichtlich, wenn die Restknoten auf eine sehr geringe Anzahl reduziert werden müssen, oder die Anzahl $tmax$ der sichtbaren Knoten wird überschritten, wenn die Menge der Kindknoten größer als $tmax$ ist. Dem kann entgegengewirkt werden, indem die Anzahl der Fokusknoten und die Anzahl an Kindknoten für einen Knoten in der vorgegebenen Hierarchie beschränkt wird.

Ein Nachteil ist, dass theoretisch jeder Knoten mit jedem anderen zusammengefasst werden kann. Dies macht die passende Beschriftung der Knoten sehr schwierig. Auch kann



(a) Zoomebene Mitteleuropa; Fokus auf Deutschland

(b) Zoomebene Deutschland; Zustand nach Zoom

Abb. 8: *Zoomen, dann clustern*: Veränderungen der Ansicht bei Zoom auf Deutschland; links vorherige Ansicht mit Fokus dargestellt durch gestricheltes Rechteck; rechts Ansicht nach dem Zoomen

es passieren, dass Knoten zusammengefasst werden, die aus Sicht des Benutzers nichts miteinander zu tun haben, was zu Verwirrung und Unübersichtlichkeit führen kann. Die Wahl eines passenden Algorithmus zum Clustern ist für die Nutzerfreundlichkeit von entscheidender Bedeutung.

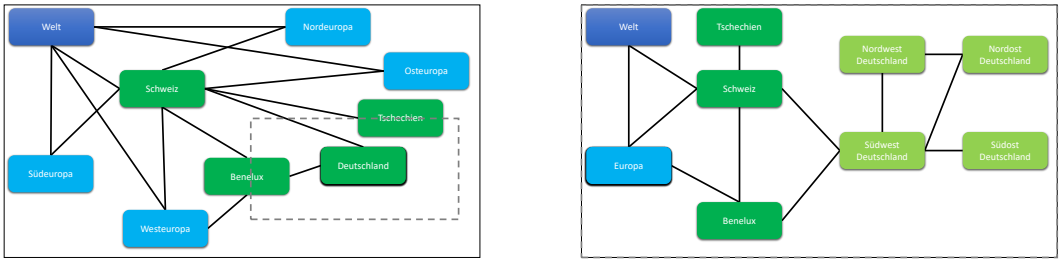
Eine farbliche Markierung der Knoten der unterschiedlichen Hierarchieebenen ist nur erschwert umsetzbar, sofern nicht verhindert wird, dass Knoten unterschiedlicher Ebenen zusammengefasst werden. Das würde wiederum die Flexibilität des Ansatzes deutlich einschränken und er würde sich dann nicht mehr nennenswert vom Ansatz *Hierarchisches Fisheye* unterscheiden.

3.2 Hierarchisches Fisheye

Die Vorgehensweise und Visualisierung dieses Ansatzes ist grundsätzlich die gleiche wie bei *Zoomen, dann clustern*. Der Unterschied der beiden Ansätze besteht in der Bildung der Cluster. Wo beim Ansatz *Zoomen, dann clustern* in jedem Zoomschritt neue Cluster gebildet wurden, sind die Cluster beim *Hierarchischen Fisheye* schon von vornherein festgelegt. Beim Zoomen werden der fokussierte Knoten ausgepackt und dessen Kindknoten sichtbar. Wären anschließend zu viele Knoten sichtbar, werden die hierarchisch höchsten Knoten zu einem „Restknoten“ zusammengefasst. Hierbei bleibt für jede höhere Hierarchieebene mindestens ein Cluster-Knoten übrig. Die Ansicht für den Zoom auf einen Knoten ist also ausschließlich von $tmax$ und der vorgegebenen Hierarchie abhängig.

In Abbildung 9 ist das beim Zoomen auf den grünen Deutschland-Knoten zu sehen. Beim Zoom wird der grüne Deutschland-Knoten aus 9a in die vier hellgrünen Deutschland-Knoten von Abbildung 9b entpackt. Gleichzeitig werden die vier zugeklappten Europa-Knoten zu einem „Rest-“Europa-Knoten zusammengefasst. Die anderen Knoten bleiben unverändert.

Der Ansatz bietet den Vorteil, dass jede Ansicht eindeutig und schnell berechnet werden kann. Durch die Beschränkung, dass nur Knoten der selben Hierarchie und wenn dann alle dieser Knoten zusammengefasst werden, ist sowohl eine farbliche Kennzeichnung der Knoten anhand ihrer Hierarchie möglich als auch eine passende Beschriftung der



(a) Zoomebene Mitteleuropa; Fokus auf Deutschland

(b) Zoomebene Deutschland; Zustand nach Zoom

Abb. 9: *Hierarchisches Fisheye*: Veränderungen der Ansicht bei Zoom auf Deutschland; Hierachieebenen farbig dargestellt; links vorherige Ansicht mit Fokus dargestellt durch gestricheltes Rechteck; rechts Ansicht nach dem Zoomen

entstehenden Cluster. Unterschiedliche Knotenfarben für die Hierarchiestufen, bestenfalls mit einem Farbverlauf wie in Abbildung 9b, sind für einen Nutzer intuitiv und einfach zu verstehen und sorgen für mehr Übersichtlichkeit. Auch wenn viele Knoten in einem Schritt erscheinen und zu Clustern zusammengefasst werden kommt ein Nutzer nicht so schnell durcheinander, da schon vor dem Zoomen klar ist, welche Knoten zu einem Cluster zusammengefasst werden könnten und wie dieses dann beschriftet sein würde.

Eine Adaptionsmöglichkeit wäre, die Beschränkung, dass wenn dann immer alle Knoten einer Hierarchiestufe zusammengefasst oder entpackt werden, aufzuheben. Dies würde zu mehr Flexibilität des Ansatzes führen, allerdings geht hierdurch der Vorteil der einfachen Clusterbeschriftungen verloren. Mehrere Fokuspunkte sind hier ebenfalls denkbar; hierbei könnte es hilfreich sein sowohl die Regionen um die Fokuspunkte als auch die Hierarchieebenen um deren ersten gemeinsamen Elterknoten detaillierter darzustellen.

Probleme dieses Ansatzes können entstehen, wenn der gegebene Hierarchiebaum nicht ausbalanciert ist. Dann kann es beim Zoom auf einen Knoten dazu kommen, dass sehr viele andere Hierarchieebenen durch einen Knoten repräsentiert werden und nur wenig Platz für Details in der Region um den Fokusknoten bleibt oder im Extremfalls sogar mehr als $tmax$ Knoten angezeigt werden müssen. Es ist folglich wichtig auf einen ausbalancierten Hierarchiebaum zu achten.

3.3 Fokusbereich + Randbereich

Dieser Ansatz bietet eine neue Herangehensweise an das Verbergen von Informationen. Hierbei wird die sichtbare Region in zwei Bereiche, den Fokus- und den Randbereich, unterteilt. Bei den Ansätzen *Zoomen, dann clustern* und *Hierarchisches Fisheye* wurden bei einer Überschreitung der Anzahl von $tmax$ sichtbaren Knoten einige Knoten zu Clustern zusammengefasst. Bei *Fokusbereich + Randbereich* werden diese hingegen in einen Randbereich verschoben um die Übersichtlichkeit im Fokusbereich zu gewährleisten. Der Wert $tmax$ wird hier also nicht als die Anzahl von sichtbaren Knoten insgesamt sonder als die Anzahl von sichtbaren Knoten im Fokusbereich definiert. Beim Heranzoomen an einen Fokusknoten wird also dieser ausgepackt und seine Kindknoten im Fokusbereich

sichtbar. Anschließend wird, falls *tmax* überschritten wurde, der irrelevanteste Knoten innerhalb des Fokusbereiches bestimmt – standardmäßig der hierarchisch höchste – und in den Randbereich verschoben. Dieser Vorgang wird so lange wiederholt, bis die Anzahl der Knoten im Fokusbereich wieder unter *tmax* liegt.

Abbildung 10 zeigt dieses Vorgehen beim Zoomen auf den Deutschland-Knoten. Die Knoten im Randbereich sind dadurch zu erkennen, dass sie ausgegraut und verkleinert sind. In Abbildung 10a ist der Graph vor dem Zoom zu sehen, Knoten höherer Hierarchieebenen als der fokussierten Ebene befinden sich bereits im Randbereich. Beim Zoomen werden dann der Deutschland-Knoten ausgepackt und „Benelux“, „Schweiz“ und „Tschechien“ in den Randbereich verschoben.

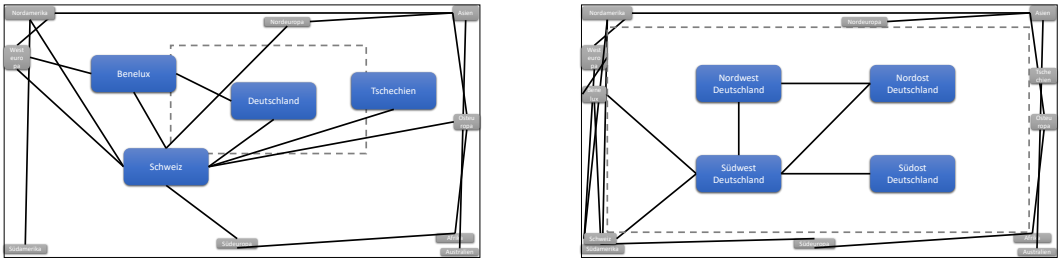
Ein Vorteil des Ansatzes besteht darin, dass für einen Nutzer sehr leicht nachvollziehbar ist, wohin gezoomt wird, da sich alle Knoten von diesem Punkt wegbewegen. Auch kann dies beispielsweise durch das Anzeigen eines Rahmens, wie dem gestrichelten Rahmen in Abbildung 10, welcher die beiden Bereiche trennt und sich automatisch verschiebt, visuell unterstützt werden.

Der größte Vorteil und gleichzeitig Nachteil des Ansatzes ist, dass alle Knoten höherer Hierarchien sichtbar bleiben. Dies bietet den Vorteil, dass keinerlei Cluster gebildet oder berechnet werden müssen, nur der fokussierte Knoten wird entpackt. Auch entstehen hierbei keine unübersichtlichen Animationen. Nachteile dabei sind, dass auch alle Kanten sichtbar bleiben, was schnell unübersichtlich werden kann. Der Randbereich im Allgemeinen wird schnell sehr unübersichtlich und unter der – da unbeschränkt – gegebenenfalls hohen Anzahl von sichtbaren Knoten kann die Performance leiden.

Einbußen bei der Performance lassen sich vermeiden, indem bei einer neuen Ansicht verhindert wird, dass alle Knoten neu gezeichnet werden. Es verändern sich in jedem Schritt nur Knoten, welche sich vor oder nach dem Zoom im Fokusbereich befinden und deren Anzahl ist durch *tmax* beschränkt. Zum anderen sollten bei einem ausbalancierten Hierarchiebaum bei wenigen Fokuspunkten nicht so viele Knoten sichtbar sein, dass dies Auswirkungen auf die Performance hat.

Da es keinerlei Beschränkung für die Knoten im Randbereich gibt, kann es auch sein, dass einige Kanten mitten durch den Fokusbereich gehen und diesen dadurch sehr unübersichtlich machen. Möglichkeiten mit den vielen sichtbaren Kanten zu verfahren wären zum einen solche, welche den Fokusbereich kreuzen einfach auszublenden, wie es in Abbildung 10b z. B. bei den Kanten von der Schweiz zu Ost-, Nord- und Südeuropa der Fall ist. Alternativ könnten diese ebenfalls ausgebleicht werden und nur sehr blass im Hintergrund des Fokusbereiches sichtbar bleiben. Eine dritte Möglichkeit wäre, diese abzuknicken oder zu krümmen, sodass sie vollständig im Randbereich verlaufen, was allerdings zu noch mehr Unübersichtlichkeit in diesem führt.

Dass der Randbereich sehr unübersichtlich wird, ist einerseits nicht weiter schlimm, da dies ja gewollt ist um den Fokusbereich übersichtlich zu halten. Andererseits stellt sich die Frage, ob es dann überhaupt nötig ist den Randbereich zu erstellen und darin alle Knoten zu erhalten, wenn nichts zu erkennen ist. Daher hierzu auch einige Vorschläge, wie dieser sinnvoll genutzt werden kann. Es könnte zum einen eine Mouseover-Funktion eingebaut werden, welche entweder die Endknoten einer fokussierten Kante oder, bei Anvisieren eines Knotens, dessen ausgehende Kanten und gegebenenfalls deren Endpunkte



(a) Zoomebene Mitteleuropa; Fokus auf Deutschland

(b) Zoomebene Deutschland; Zustand nach Zoom

Abb. 10: *Fokusbereich + Randbereich*: Veränderungen der Ansicht bei Zoom auf Deutschland; links vorherige Ansicht mit Fokus dargestellt durch gestricheltes Rechteck; rechts Ansicht nach dem Zoomen – Fokus- und Randbereich getrennt durch gestricheltes Rechteck

hervorhebt. Zum anderen wäre es eine Möglichkeit, bei der Verwendung von mehreren Fokuspunkten, deren kürzesten Verbindungsweg im Randbereich sichtbar zu machen.

Da keine Cluster gebildet werden, lassen sich sowohl alle Knoten problemlos benennen also auch eine farbliche Markierung der Knoten abhängig von deren Hierarchie umsetzen. Hierbei könnte dazu übergegangen werden, Knoten im Randbereich nicht auszugrauen sondern nur zu verblassen, damit die Farben erkennbar bleiben. Auch sollte dann darauf geachtet werden, nur kräftige, dunklere Farben zu verwenden, damit sich Knoten aus dem Randbereich farblich eindeutig von denen aus dem Fokusbereich unterscheiden.

3.4 Ebenen durch Rahmen

Dieser Ansatz ähnelt dem Ansatz *Fokusbereich + Randbereich*. Er unterscheidet sich hauptsächlich in der Gestaltung des Randbereiches. Die Vorgehensweise ist die, dass immer der fokussierte Knoten ausgeklappt wird. Hierbei erscheinen alle neuen Knoten innerhalb eines Rahmens, welcher den fokussierten Knoten darstellt. Währen anschließend Knoten aus mehr als zwei Hierarchiestufen sichtbar, werden die Knoten der höchsten Hierarchiestufe in den Randbereich verschoben und verschwinden anschließend. Repräsentativ für die ausgeblendeten Knoten wird ein Rahmen eingefügt, in welchen weiterhin die Verbindungskanten zu anderen Sichtbaren Knoten führen. Kanten zwischen zwei Knoten, welche beide im selben oder in unterschiedlichen Rahmen verborgen sind, werden ausgeblendet.

Zu sehen ist das in Abbildung 11. Beim Heranzoomen an Deutschland wird der Deutschland repräsentierende Knoten ausgepackt und vergrößert sich zu einem Rahmen, welcher die Knoten der nächsten Hierarchieebene enthält. Um die Übersichtlichkeit weiterhin zu gewährleisten werden die Knoten, welche die unterschiedlichen Gebiete Europas repräsentieren zu einem Rahmen mit Beschriftung „Europa“ zusammengefasst. Da der Knoten „Schweiz“ weiterhin sichtbar ist, bleiben all seine Kanten zu Europa-Knoten erhalten. Diese führen jetzt in den neuen Europa-Rahmen an die Stellen, an denen die Europa-Knoten ausgeblendet wurden. Somit sind die Verbindungen weiterhin sichtbar; um aber sehen zu

können, wohin genau in Europa eine der Verbindungen führt, muss wieder herausgezoomt werden.

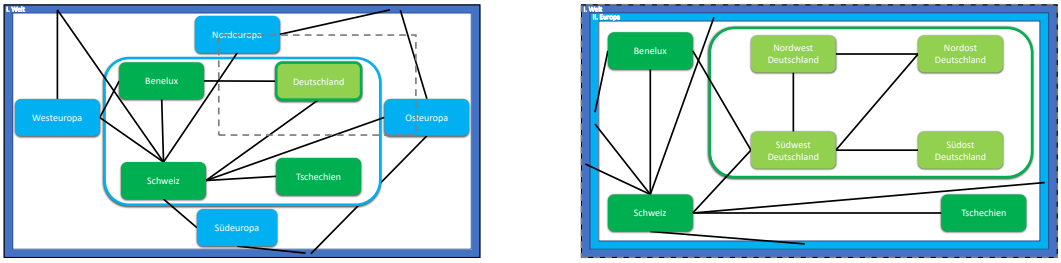
Der Ansatz ist also sehr beschränkt, da streng ebenenweise in den Graphen hineingezoomt wird. Dafür ist er sehr intuitiv und übersichtlich, da ständig alle höheren Ebenen eindeutig erkennbar sind. Die Übersichtlichkeit kann allerdings darunter leiden, wenn die Kindknotenanzahl unbeschränkt ist, da dann zum einen sehr viele Knoten gleichzeitig angezeigt werden können, da immer die aktuelle Hierarchieebene vollständig sichtbar ist, zum anderen werden die Übergänge sehr unübersichtlich, wenn auf einmal sehr viele Knoten gleichzeitig auftauchen oder verschwinden. Es sollte auch darauf geachtet werden, dass nach Möglichkeit keine Kanten zwischen zwei Knoten außerhalb des entpackten Knotens diesen schneiden. Lässt sich das nicht vermeiden besteht auch die Möglichkeit diese Kanten in den Hintergrund zu legen oder innerhalb des Rahmens auszugrauen, damit klar ist, dass sie nichts mit diesem Knoten zu tun haben. Führen viele Kanten eines Knotens in einen Rahmen, wie in Abbildung 11b von der Schweiz nach Europa, könnten diese auch durch eine Kanten ersetzt werden, da ja nur noch ein Knoten in Form des Rahmens vorhanden ist.

Ein Nachteil des Ansatzes könnte sein, dass bei zu tiefen Zoomstufen die Rahmen sehr viel Platz wegnehmen. Diese müssen eine gewisse Dicke haben, damit sie lesbar beschriftet werden können. Je gleichmäßiger und ausbalancierter der Hierarchiebaum ist, desto besser funktioniert der Ansatz.

Eine Erweiterung auf mehrere Fokuspunkte ist bei diesem Ansatz möglich, indem für jeden Fokus ein eigener Rahmen geöffnet wird. In Abbildung 11b kann man sich das so vorstellen, wie wenn der Benelux-Knoten ein ebensolcher entpackter Knoten wie der Deutschland-Knoten wäre, welcher dann z. B. die Knoten Niederlande Belgien und Luxemburg enthält. Ebenfalls eine nützliche Erweiterung für den Ansatz wäre, die entpackten Knoten, welche noch kein Außenrahmen sind, ebenfalls zu beschriften. Dies wäre vor allem bei mehreren Fokuspunkten sehr hilfreich. Eine denkbare Navigationsmöglichkeit für den Nutzer könnte ein Herauszoomen durch Anklicken eines Rahmens darstellen, wodurch auf dessen Ebene zu gezoomt wird. Auch eine Mouseover-Einblendung der Endknoten einer Kante, welche in einem Rahmen verborgen sind, wäre ein nützliches Feature. Auf ähnliche Weise könnte man auch bei Auswahl von zwei Knoten in unterschiedlichen Fokusgebieten alle Knoten, die sich auf dem kürzesten Weg zwischen den beiden befinden, einblenden lassen. Bei Ersetzung mehrerer Kanten durch nur eine in jeden Rahmen, könnten beim Überfahren mit der Maus alle ausgehenden Kanten eingeblendet werden, gegebenenfalls mit Endpunkten.

3.5 Fokusebene + Verbindungsknoten

Dieser Ansatz arbeitet ohne einen Schwellwert $tmax$, dafür mit einer maximalen Hopp-Distanz h_{dist}. Er ist recht simpel aufgebaut; es werden alle Knoten angezeigt, welche durch maximal h_{dist} Hopps vom Fokusknoten aus erreichbar sind. Wird also an einen Knoten v herangezoomt, welcher sich dann entpackt, bleiben alle Knoten sichtbar, welche in weniger als h_{dist} Hopps von v aus erreichbar waren. Zusätzlich werden die Kinder von v eingeblendet und alle übrigen Knoten ausgeblendet. Ebenfalls sichtbar bleiben



(a) Zoomtiefe Mitteleuropa; Fokus auf Deutschland

(b) Zoomtiefe Deutschland; Zustand nach Zoom

Abb. 11: *Ebenen durch Rahmen*: Veränderungen der Ansicht bei Zoom auf Deutschland; Hierachieebenen farbig dargestellt; links vorherige Ansicht mit Fokus dargestellt durch gestricheltes Rechteck; rechts Ansicht nach dem Zoomen

ausgehende Kanten zu Knoten, welche durch genau $h_{dist} + 1$ Hopps erreichbar sind.

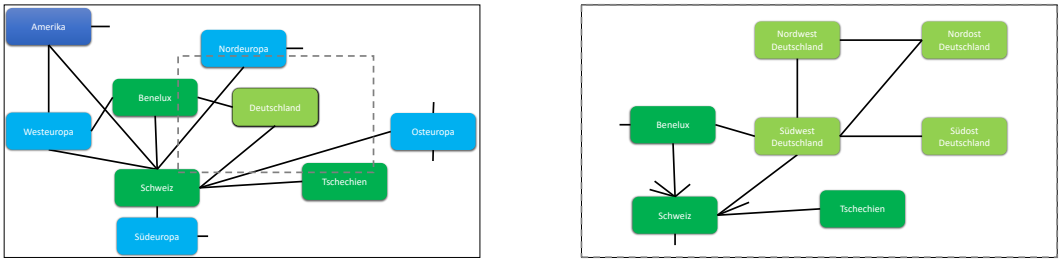
Zu sehen ist das Vorgehen in Abbildung 12, wieder am Beispiel des Zooms auf den Deutschland-Knoten. In Abbildung 12a zu sehen sind die Knoten der dunkelgrünen Hierarchieebene und alle, die einen Hopp von diesen entfernt sind. Beim Zoomen auf „Deutschland“ werden der Deutschland-Knoten in die hellgrünen Knoten entpackt und die blauen Knoten ausgeblendet, da diese mehr als einen Hopp vom Deutschland-Knoten entfernt sind. Ausgehende Kanten zu den ausgeblendeten Knoten bleiben im Ansatz erhalten, wie in Abbildung 12b am Benelux-Knoten zu sehen ist. Die Kante, welche in Abbildung 12a zum Westeuropa-Knoten führt, ist in 12b in Form der nach links ins Nichts führenden, ausgehenden Kante des Benelux-Knotens vorhanden.

Bei diesem Ansatz ist es besonders wichtig, h_{dist} nicht zu groß zu wählen, da sonst sehr viele Knoten angezeigt werden könnten. Eine denkbare Alternative wäre eine Verwendung von $tmax$ in der Form, dass h_{dist} dynamisch so gewählt wird, dass höchstens $tmax$ Knoten sichtbar sind. Somit ist es auch bei diesem Ansatz von Nutzen, wenn der Hierarchiebaum möglichst ausbalanciert ist.

Fokusebene + Verbindungsknoten unterscheidet sich von den anderen Ansätzen deutlich, da hier der vollständige Graph nicht immer repräsentiert ist. Auch sind nicht immer alle höheren Hierarchieebenen in Form mindestens eines Knotens vertreten. Hierin besteht die Gefahr, dass der Nutzer den Überblick über das Gesamtbild verliert. Dieses Problem kann umgangen werden, wenn beispielsweise an der Seite in Form einer Ordnerstruktur – wie sie z. B. aus dem Windows-Explorer bekannt ist – der Gesamtgraph abgebildet wird. Somit bleibt der Gesamtzusammenhang ersichtlich und der vollständige Graph bei jeder Ansicht repräsentiert.

Ein mögliches Feature wäre eine Mouseover-Einblendung von Endpunkten ins Nichts führender Kanten. Also wenn der Nutzer den Cursor auf eine ins Nichts führende Kante bewegt, wird der Endknoten eingeblendet. Sobald sich der Cursor nicht mehr über der Kante befindet, verschwindet der zusätzliche Knoten wieder.

Einer der größten Vorteile des Ansatzes ist, dass der Graph über alle Zoomstufen übersichtlich bleibt. Er ist ebenfalls sehr intuitiv für den Benutzer, da leicht nachvollziehbar



(a) Zoomstufe Mitteleuropa; Fokus auf Deutschland

(b) Zoomstufe Deutschland; Zustand nach Zoom

Abb. 12: *Fokusebene + Verbindungsknoten*: Veränderungen der Ansicht bei Zoom auf Deutschland; Hierachieebenen farbig dargestellt; links vorherige Ansicht mit Fokus dargestellt durch gestricheltes Rechteck; rechts Ansicht nach dem Zoomen

ist, wie der Algorithmus beim Übergang von einer Ansicht zur nächsten vorgegangen ist, da Knoten nicht einfach verschwinden, sondern über die ins Nichts führenden Kanten nachvollziehbar ist, wo sich auf der letzten Stufe noch Knoten befunden haben. Auch ist die direkte Nachbarschaft sehr detailliert, allerdings auf Kosten der Gesamtübersicht.

3.6 Zu allen Ansätzen

Angenehmer für den Benutzer könnte es sein, einen stufenlosen Zoom zur Verfügung zu haben, da hierbei besser nachvollziehbar ist, was mit den Knoten passiert. Dieser lässt sich theoretisch bei allen Ansätzen problemlos einbauen, indem zunächst an einen Fokuspunkt herangezoomt wird und eine bestimmte Schwelle eingebaut wird, bei deren Überschreitung der Übergang von einer Ansicht in die nächste passiert. Eine Möglichkeit hereinzuzoomen wäre, die Knoten auf strahlenförmigen Bahnen vom Fokuspunkt wegzubewegen. Hierbei könnte man Knoten, welche näher am Fokuspunkt sind, um eine weitere Strecke bewegen als Knoten, welche sich näher am Rand befinden. Als Schwelle wäre denkbar, dass ein Übergang stattfindet, sobald sich zwei Knoten zu Überschneiden drohen oder ein Knoten aus dem Sichtfeld gerät.

4 Umsetzung

Zur Umsetzung wurde der Visualisierungsansatz *Ebenen durch Rahmen* ausgewählt, da er allen Beteiligten am intuitivsten und übersichtlichsten erschien. Implementiert wurde der Ansatz in Java unter Verwendung der yFiles for Java (Swing) Bibliothek [yfi19].

Das Herzstück dieser Arbeit ist die Zoomfunktion. Durch scrollen kann der Nutzer in einen Teil des Graphen hineinzoomen bzw. wieder aus dem Graphen herauszoomen. Ebenfalls ist es möglich durch Klicken auf einen bestimmten Knoten auf dessen Ebene zu zoomen. Hierbei werden alle Unterknoten ausgeblendet und anschließend die direkten Kinder des angeklickten Knotens eingeblendet. Da ansonsten keine andere Interaktion mit dem Graphen durch den Nutzer vorgesehen ist, wurden die Zoomfunktionen durch Sperren der von yFiles vorgesehenen Interaktionsfunktionen und eine eigene Verarbeitung dieser Eingaben umgesetzt.

Im Folgenden werden die Algorithmen zur Auswahl der sichtbaren Objekte ausführlich erklärt und die hierzu benötigten Datenstrukturen vorgestellt. Zuerst wird näher auf die Berechnung der sichtbaren Knoten (Unterabschnitt 4.1) und im Anschluss auf die der Kanten (Unterabschnitt 4.2) eingegangen. Anschließend wird auf die Darstellung des Graphen und den verwendeten Layoutalgorithmus eingegangen (Unterabschnitt 4.3). Zum Schluss werden Testergebnisse vorgestellt, wobei die Software mit verschiedenen Graphen getestet wurde (Unterabschnitt 4.4).

4.1 Änderung der sichtbaren Knoten

Es wird beim Zoomen nach folgendem Muster vorgegangen. Zuerst wird mithilfe der Knotenlayouts sowie der Cursorposition ermittelt, welcher Knoten anvisiert ist. Dies ist standardmäßig immer der Wurzelknoten. Dann wird für alle seine Kindknoten überprüft, ob sich der Cursor über einem dieser befindet. Diese Vorgehensweise wird wiederholt, bis der hierarchisch niedrigste Knoten gefunden wurde, in dessen visuellen Grenzen sich der Cursor befindet. Ab hier unterscheidet sich die Vorgehensweise.

Hineinzoomen. Wurde ein Knoten ausgewählt, von welchem bereits alle Kinder sichtbar sind, gibt es keine Änderung bei den sichtbaren Knoten. Bei allen anderen gilt, es werden zuerst alle noch nicht sichtbaren Kinder des Kandidaten den sichtbaren Knoten hinzugefügt. Anschließend wird getestet, ob mehr als $tmax$ Knoten sichtbar wären. Solang dies der Fall ist, werden andere Knoten zusammengepackt, bis der Grenzwert nicht mehr überschritten wird.

Alle Knoten, die zum Zusammenpacken zur Verfügung stehen, sind in einer Liste abgespeichert. Die Kandidaten werden in der aktuellen Implementierung nach dem FIFO-Prinzip verwaltet – ein hinzukommender Kandidat wird immer hinten an die Liste angehängt und bei Bedarf wird immer der erste Knoten der Liste ausgewählt. Für das Einsortieren von Knoten in diese Liste und das Ermitteln des nächsten Kandidaten sind eigene Methoden angelegt. Hierdurch können auf einfache Art und Weise die Prioritäten für die Auswahl der Knoten geändert werden.

Herauszoomen. Das Herauszoomen funktioniert ähnlich dem Hineinzoomen mit dem Unterschied, dass der ausgewählte Knoten zusammengepackt wird. Anschließend sind wieder Kapazitäten frei und es werden Knoten entpackt, bis alle sichtbaren Knoten entpackt sind oder $tmax$ beim nächsten Entpacken überschritten werden würde.

Hier existiert ebenfalls eine Kandidatenliste, die gleich aufgebaut ist, wie die für die Kandidaten beim hinein Zoomen – sie arbeitet nach dem FIFO-Prinzip. Auch hier existieren Methoden, mit welchen die Reihenfolge adaptiert werden kann.

Knotenklick. Beim Anklicken eines Knotens, wird zunächst auf diesen herausgezoomt. Anschließend wird an ihn herangezoomt. Es ist also eine Hintereinanderausführung von Herauszoomen und Hineinzoomen. Nach der Ausführung sind keine der Enkelknoten des angeklickten Knotens mehr sichtbar. Dafür sind aber der angeklickte Knoten sowie alle seine Kindknoten unter den sichtbaren Knoten.

4.2 Änderung der sichtbaren Kanten

Die Auswahl der sichtbaren Kanten aus dem Ursprungsgraphen ist recht simpel, yFiles zeigt standardmäßig nur Kanten zwischen sichtbaren Knoten an. Eine größere Herausforderung ist die Darstellung von repräsentativen Kanten für solche Kanten, von welchen nicht beide Endknoten sichtbar sind. Eine recht simple Lösungsmöglichkeit wäre, über alle Kanten zu iterieren und für jede den passenden Repräsentanten zu bestimmen. Diese hat jedoch den offensichtlichen Nachteil einer sehr langen Laufzeit. Gerade bei sehr großen Graphen ist es von Vorteil, wenn die Laufzeit der Berechnung von einer Zoomansicht zur nächsten unabhängig von der Größe des Eingabegraphen ist.

Eine alternative Möglichkeit besteht darin, die Gruppenknotenfunktion von yFiles zu verwenden. Beim Zusammenpacken von Gruppenknoten fügt yFiles automatisch eine Kante von diesem Gruppenknoten zu einem anderen Knoten ein, wenn zuvor eine Kante von einem Kind des Gruppenknotens zu dem anderen Knoten existierte. Jedoch unterstützt yFiles nicht das „Teilzusammenpacken“ von Gruppenknoten, wobei ein Teil der Kinder sichtbar bleibt und die anderen ausgeblendet werden. Es ist nur möglich diese Kinder auszublenden, wobei allerdings auch alle Kanten verloren gehen und nicht auf den Gruppenknoten umgelegt werden.

Dem könnte durch den Einbau von „Fake-Gruppenknoten“ abgeholfen werden. Diese würden beim Teilzusammenpacken eines Gruppenknoten hierarchisch zwischen den Gruppenknoten und die auszublendenden Knoten eingebaut und anschließend zusammengepackt. Der Fake-Gruppenknoten würde somit alle repräsentativen Kanten erhalten. Wird dieser Fake-Gruppenknoten auf die Größe des Gruppenknotens gesetzt und in den Hintergrund dessen, ist das visuell so, als ob die Kanten zum eigentlichen Gruppenknoten führten. Hierbei bleibt jedoch noch das Problem, dass Kanten zwischen ineinander geschachtelten Rahmen noch manuell ausgeblendet werden müssen. Ebenfalls muss darauf geachtet werden, dass die Kanten nach wie vor im Vordergrund verlaufen. Der größte Nachteil besteht darin, dass die „Collapse-“ und „Expand-“ Funktionen eine Laufzeit in Abhängigkeit der Größe des Eingabegraphen haben.

Map möglicher Kanten. Die Möglichkeit der Wahl war dementsprechend eine, bei der die Berechnungszeit unabhängig der Größe des Eingabegraphen ist. Diese arbeitet mit einem zusätzlichen Vorberechnungsschritt, bei welchem eine HashMap der möglichen Kanten erzeugt wird. Es werden für jede Kante alle Möglichkeiten berechnet, wie diese repräsentiert werden könnte. Für jede Repräsentation wird dann ein Eintrag in der Map angelegt. Den Schlüssel stellt das Knotenpaar dar (im Folgenden *Schlüsselknotenpaar* genannt). Hierfür wurde eine neue Klasse erstellt, deren .hash() und .equals() Methoden unabhängig der Reihenfolge der Knoten eines Knotenpaares ist. Ein Eintrag beinhaltet eine Sammlung von Knotenpaaren (im Folgenden *Eintragsknotenpaare* genannt).

Durch Überprüfung der Sichtbarkeit der *Eintragsknotenpaare*, kann überprüft werden, ob eine Kante zwischen den *Schlüsselknoten* benötigt wird oder nicht. Ein *Eintragsknotenpaar* enthält von jedem Knoten des *Schlüsselknotenpaares* genau den Kindknoten zu welchem die Kante führen würde, wäre dieser sichtbar. Hat einer der *Schlüsselknoten* keine Kinder, besteht das *Eintragsknotenpaar* aus zwei Mal dem selben Knoten, nämlich dem entsprechenden Kindknoten des anderen *Schlüsselknotens*. Ein *Eintrag* entspricht also einer oder mehrerer Kanten des Eingabegraphen und ein *Schlüssel* genau einer Repräsentationsmöglichkeit dieser. Eine Repräsentation ist also immer dann nötig, wenn von mindestens einem *Eintragsknotenpaar* beide Knoten nicht sichtbar sind. Mit Hilfe dieses Tests werden die Repräsentationskanten berechnet.

4.3 Visualisierung

Die Darstellung der Gruppenknoten als Rahmen ist ohne großen Aufwand durch Erstellen eines eigenen Layouts möglich und wurde auch auf diesem Weg umgesetzt. Für Blattknoten wurde ebenfalls eine eigene Style-Klasse erstellt. Beide Klassen implementieren ein Interface, welches erlaubt unterschiedliche Farben für die Darstellung eines Knotens als Blatt- oder Gruppenknoten zu speichern.

Knotenbeschriftungen werden immer innerhalb eines Knotens oben links angezeigt. Farben für Beschriftungen lassen sich frei wählen. Für Knoten werden automatisch hierarchieabhängige Farben in Form eines Farbübergangs mittels des HSB-Farbraums berechnet. Sättigung und Helligkeit sowie Start- und Endfarbwert können frei gewählt werden. Kanten werden ungerichtet als einfache Linien dargestellt.

Bei Verwendung der Layoutalgorithmen von yFiles werden Kindknoten innerhalb des Rechtecks des Elternknoten platziert und der Elternknoten entsprechend groß genug dargestellt. Zur Berechnung der Knotenpositionen wurde der von yFiles vorimplementierte Algorithmus für organische Layouts verwendet. Dieser Algorithmus bietet jedoch keine Möglichkeit Kanten in die Rahmen führen zu lassen, da Kanten automatisch immer zum Mittelpunkt eines jeden Knoten führen. Um dies dennoch zu erreichen, werden die Kanten nach Berechnung des Layouts verschoben, sodass sie in den Rahmen führen, sofern sie von einem angezeigten Knoten zu einem seiner Vorfahren verlaufen.

4.4 Tests

Die Software wurde mit vier verschiedenen Graphen getestet. Schon während der Entwicklung wurde der in Abschnitt 3 vorgestellte Worldgraph zum Testen verwendet. Er wurde lediglich dahingehend abgewandelt, dass jeder Gruppenknoten, welcher bisher noch keine Unterknoten besaß, noch einen solchen erhalten hat, zu welchem die eigentlichen Kanten führen. Beispielsweise der Asien-Knoten hat noch einen Unterknoten namens RouterAs erhalten, z. B. die Kante zwischen Asien und Australien verläuft hier zwischen RouterAs und RouterAu. Beim Ausführen mit dem Worldgraph treten keine Performanceprobleme auf. Ansichten gehen problemlos ineinander über.

Nachteile sind, was allgemein für alle Graphen gilt, dass Knoten nicht an ihren Positionen bleiben und neue Knoten immer an derselben Stelle eingefügt werden. Außerdem springen nach dem Morphen die Kanten, welche in die Rahmen führen an die richtige Position. Dies macht alles etwas unübersichtlich und chaotisch, was aber im Vorfeld zu erwarten war, da ein passender Layoutalgorithmus fehlt. Die Ansicht sieht dafür nach dem Übergang genau so aus wie sie gedacht ist. Die Position, an welcher neue Knoten eingefügt werden, wurde nach diesem Test auf die Position ihres Elterknotens geändert und dieses Problem behoben.

Der zweite Test war mit einem zufällig generierten Graphen mit folgenden Parametern: Er hat 100.000 Knoten, jeder Knoten hat 10 Kindknoten, es gibt 80.000 zufällig generierte Kanten zwischen Blattknoten. Die Kanten werden so generiert, dass 99 von 100 zwischen Knoten mit gemeinsamem Elternknoten verlaufen, sodass der Graph zumindest ganz grob einem Netzwerkgraphen ähnelt. Die restlichen Kanten verlaufen komplett zufällig zwischen allen Blattknoten.

Dieser Test hat zu folgenden Ergebnissen geführt: Die Vorberechnung dauert entsprechend länger, jedoch insgesamt nicht länger als den Graphen zu konstruieren. Beides zusammen dauert in etwa 30 Sekunden. Die Übergänge zu berechnen geht, wie erwartet, genau so schnell wie bei dem kleinen Worldgraph. Das Morphen, also die Übergangsanimation sowie die Berechnung des Layouts mit dem Layoutalgorithmus von yFiles, dauert verhältnismäßig lang. Es ist deutliches Ruckeln bemerkbar und teilweise findet nahezu keine Animation statt, sondern die alte Ansicht springt in wenigen Schritten auf die neue Ansicht. Dieselben Probleme treten auch ohne Verwendung von Morphen als Übergangsanimation auf; es betrifft also vermutlich ausschließlich den Layoutalgorithmus. Der von mir implementierte Code zur Berechnung der neuen Ansicht läuft effizient – es werden bei $tmax = 70$ im Normalfall weniger als 100 ms für diesen Teil der Berechnung benötigt.

Ein dritter Test wurde mit einem kleineren Graphen aus einem Praxisbeispiel durchgeführt. Hierbei läuft alles effizient und es treten keine Probleme auf. Alles funktioniert einwandfrei, wie bei den Tests mit dem Worldgraph.

Der vierte Test wurde ebenfalls mit einem Graphen aus der Praxis durchgeführt. Hierbei handelte es sich um das Netzwerk der Firma BMW. Der verwendete Graph beinhaltet ca 31.000 Knoten. Die Ergebnisse sind, wie erwartet, ähnlich denen des zweiten Tests. Animationen funktionieren etwas besser, aber dennoch meist nicht flüssig. Die eigenen Berechnungen machen auch hier keine Probleme; laufzeitkritisch sind die vorimplementierten Funktionen von yFiles. Neu hinzu kommt bei diesem Graphen das Problem, dass

die Eingabe teilweise sehr viele Knoten mit gemeinsamem Elterknoten enthält. Dies führt zu sehr unübersichtlichen Zwischenansichten, da beim Auspacken eines Knotens immer alle Kindknoten angezeigt werden. Um dennoch mit einem solchen Graphen arbeiten zu können, wurde im Anschluss an diesen Test ein Eingabefeld für ein variables $tmax$ sowie eine zusätzliche Zoom-Möglichkeit hinzugefügt, bei welcher nur vergrößert und nichts aus- oder zusammengepackt wird.

5 Ausblick

Die Implementierung bietet die Möglichkeit, den in Abschnitt 3 vorgestellten Ansatz *Ebenen durch Rahmen* graphisch darzustellen, wozu die Eingabe des Graphen in Form einer IGraph-Instanz von yFiles benötigt wird. Hierauf kann auf mehreren Wegen aufgebaut werden:

Einbindung. Da die Implementierung bisher nur aus einem Paket besteht, ist der nächste wichtige Schritt, dieses in ein praktisch genutztes Visualisierungsprogramm einzubauen.

Zum einen wird natürlich ein passender Parser benötigt, der Graphen in das gewünschte Format umwandelt. Dieser muss dafür sorgen, dass eine vollständige Hierarchie mit einem Wurzelknoten vorhanden ist, von welchem aus alle Knoten erreicht werden können. Zum anderen sollten keine Kanten zwischen Gruppenknoten und insbesondere von Blatt- zu Gruppenknoten existieren, da diese nicht vorgesehen sind. Die Gruppenknoten stellen lediglich Cluster von Blattknoten dar.

Die Einbindung in eine graphische Oberfläche sollte unkompliziert sein, da die bereitgestellte Graphkomponente in Java Swing wie jedes andere graphische Objekt verwendet werden kann. Dies wurde für den allgemeinen Fall jedoch noch nicht getestet.

Erweiterungen. Es können die anderen Visualisierungsvarianten aus Abschnitt 3 implementiert werden. Wird dies umgesetzt, können die Vorschläge im direkten Vergleich getestet werden. Des Weiteren bietet es die Möglichkeit, mehrere Varianten anzubieten und dem Nutzer die Wahl zu überlassen. Auch kann es interessant sein, weitere Ideen einzuholen oder Verbesserungsvorschläge zu den hier vorgestellten Varianten zu sammeln und umzusetzen.

Es existieren im Code bisher zwei Listen von Knoten: die „ExpandList“, mit Kandidaten, von welchen ein Teil der Kindknoten beim Heranzoomen an einen anderen Knoten ausgeblendet wurde, und die „CollapseList“, welche Knoten enthält, von denen Kindknoten ausgeblendet werden können. Wird herausgezoomt, werden Kapazitäten frei und es werden Kandidaten der ExpandList entpackt, sofern möglich. Beim Hineinzoomen werden umgekehrt zu Gunsten der neu sichtbaren Knoten Kandidaten der CollapseList zusammengepackt, bis die Anzahl sichtbarer Knoten wieder stimmt oder die Liste leer ist.

Zum Hinzufügen von Kandidaten zu diesen Listen gibt es jeweils eine eigene Methode sowie für die Auswahl des nächsten Kandidaten. Diese arbeiten nach der aktuellen Implementierung nach dem FiFo-Prinzip. Es besteht hier die Möglichkeit, diese Methoden zu ändern oder zu überschreiben und somit eine Logik einzubauen, welche Kandidaten ermittelt. Der nächste Kandidat zum Zusammenpacken kann z. B. in Abhängigkeit des aktuellen Fokusknotens ausgewählt werden, der im Hierarchiebaum am weitesten entfernte Knoten könnte ein passender Kandidat sein.

Laufzeit. Bei der Visualisierung von großen Graphen spielt die Laufzeit eine entscheidende Rolle. In diesem Fall die Zeit, welche benötigt wird um die Mengen der sichtbaren

Knoten und Kanten sowie deren Postionen zu berechnen. In der bisherigen Implementierung wird eine Zeit von 500 ms veranschlagt um von einer Ansicht in die nächste zu morphen. Die Laufzeit zur Berechnung des neuen Layouts sollte also in einem ähnlichen Zeitrahmen abgeschlossen sein, um eine angenehme Benutzung zu gewährleisten. Da bei den Tests mit größeren Graphen Performanceprobleme bei den von yFiles zur Verfügung gestellten Klassen aufgetreten sind, muss hier noch etwas gemacht werden.

Zunächst kann ein zusätzlicher Test hilfreich sein, bei dem die Implementierung abgewandelt wird, sodass zwei getrennte Graphinstanzen gehalten werden – eine, die den vollständigen Graphen enthält und eine kleine Kopie welche nur die anzuzeigenden Knoten enthält. Hierdurch kann getestet werden, ob die Probleme beim FilteredGraphWrapper von yFiles liegen – dann sollte bei diesem Test alles effizient laufen – oder, ob die Performanceprobleme andere Ursachen haben – dann sollten sie auch bei diesem Test auftreten.

Eine diskutierte Alternative war auch, die yFiles-Funktion zum Zuklappen von Gruppenknoten zu verwenden. Diese kommt jedoch nicht in Frage, da zum einen ein Teilzuklappen von Knoten nicht möglich ist und zum anderen das Zu- und Aufklappen von Gruppenknoten bei großen Graphen sehr lang dauert. Es sind auf diesem Weg zwar Performanceverbesserungen beim OrganicLayout-Algorithmus zu erwarten, jedoch auf Kosten von deutlichen Laufzeitverlängerungen in der restlichen Berechnung. Das Morphen würde vermutlich deutlich besser funktionieren, jedoch erst nach mehreren Sekunden Berechnungszeit. Daher wurde von vornherein versucht, die Nutzung dieser Funktion zu vermeiden.

Laufzeitkritisch in der aktuellen Implementierung ist also die Verwendung des FilteredGraphWrappers in Kombination mit dem OrganicLayout von yFiles. Der eben genannte zusätzliche Test mit den zwei getrennten Graphinstanzen stellt eine Alternative zur Verwendung des FilteredGraphWrappers dar. Eine Alternative für den OrganicLayout-Algorithmus von yFiles besteht in der Implementierung eines eigenen Layoutalgorithmus, der noch zusätzliche Vorteile bringen kann. Es besteht die Vermutung, dass das Ersetzen einer oder beider yFiles-Klassen zur Behebung der Performanceprobleme bei großen Eingabgraphen führt. Näheres zu einem geeigneten Layoutalgorithmus wird im Folgenden beschrieben.

Layout. Die bereitgestellte Implementierung hat folgende, nicht ausgereifte Eigenschaften, was die graphische Darstellung der einzelnen Ansichten angeht:

Die Ansicht startet momentan immer mit dem Wurzelknoten. Es kann also noch überlegt werden, ob dies die passende Startansicht ist oder ob eine andere Ansicht besser wäre. Eine Möglichkeit hierzu ist, den Wurzelknoten direkt zu entpacken, da die einzig mögliche Interaktion eines Nutzers zu Beginn das Heranzoomen an diesen darstellt. Auch könnte die Ansicht der letzten Verwendung abgespeichert werden und beim erneuten Öffnen angezeigt werden oder die Möglichkeit bereitgestellt werden, vorab einen Startknoten zu wählen und mit Fokus auf diesen Knoten zu starten.

Das komplexeste Problem stellt der Layoutalgorithmus dar. Der verwendete, vorimplementierte Algorithmus für organische Layouts ist nicht in gewünschter Weise adaptierbar.

Es ist zwar möglich ihn deterministisch arbeiten zu lassen, sodass bei der selben Eingabe auch immer die selbe Ausgabe erzeugt wird. Aber sobald nur ein Knoten hinzugefügt oder weggenommen wird, können sich alle Knotenpositionen in einem neuen Layout beliebig ändern. Es fehlt also die Möglichkeit, Knotenpositionen von weiterhin sichtbaren Knoten in die Berechnung des neuen Layouts einfließen zu lassen. Es wird jedoch ein Layoutalgorithmus nach dem Interface ILayoutAlgorithm von yFiles benötigt, damit für den Übergang von einer Ansicht auf die nächste die vorimplementierte Möglichkeit zum Morphen genutzt werden kann. Es wurden im Rahmen dieser Arbeit Versuche hierzu unternommen, es war jedoch aus Zeitgründen nicht möglich, eine praktikable Lösung zu finden und umzusetzen. Es sollte also optimalerweise das Layout auf folgende Art und Weise berechnet werden:

Eine Ausgangsansicht sollte in zwei Schritten auf eine neue gemorpht werden. Zuerst wird auf eine Übergangsansicht gemorpht und anschließend auf die finale Ansicht.

Für die finale Ansicht wird ein eigener Layoutalgorithmus benötigt, wobei sich vermutlich ein kräftebasierter Algorithmus am besten eignet. Dieser sollte die aktuelle Knotenposition berücksichtigen und Knoten möglichst wenig verschieben, also eine Kraft beinhalten, welche Knoten auf ihren Positionen hält. Der Algorithmus sollte die Knoten gleichmäßig auf die zur Verfügung stehende Fläche verteilen, sodass der Wurzelknoten immer maximale Größe hat. Es muss darauf geachtet werden, dass Rahmen nach wie vor ineinander geschachtelt werden, sich also an allen Seiten berühren, sofern keine ihrer Geschwister sichtbar sind. Es sollte berücksichtigt werden, das Knoten, welche Kanten in einen Rahmen haben, möglichst weit außen platziert werden, sodass deren Kanten möglichst keine anderen Knoten schneiden. Ebenfalls sollte für jede dieser Kanten ein passender Endpunkt im Rahmen ermittelt werden.

Diese finale Ansicht darf nicht sofort angezeigt werden, sie muss gespeichert werden und dient als Information zur Berechnung einer Übergangsansicht. Die Übergangsansicht hat dieselben sichtbaren Knoten, wir die Ausgangsansicht. Es muss nun nur folgendes getan werden: Es müssen alle Kanten ermittelt werden, welche in der Ausgangsansicht noch zu einem Knoten und in der finalen Ansicht in einen Rahmen führen. Für jede dieser Kanten, muss der Endknoten an die Stelle in den Rahmen verschoben werden, an der in der finalen Ansicht die Kante endet.

Literatur

[CZQ+08] Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong und Xiaoming Li: Geometry-based edge clustering for graph visualization. IEEE Transactions on Visualization and Computer Graphics, 14(6):1277–1284, 2008.

[FPF13] George W Furnas, Steven E Poltrock und Kim M Fairchild: SemNet: Three-Dimensional Graphic Representations of Large Knowledge Bases. In: Cognitive Science and Its Applications for Human-computer Interaction, Seiten 215–248. Psychology Press, 2013.

[Fur86] George W Furnas: Generalized fisheye views. Human Factors in Computing Systems CHI '86 Conference Proceedings, 17(4):16–23, 1986.

[GKN05] Emden R Gansner, Yehuda Koren und Stephen C North: Topological fisheye views for visualizing large graphs. IEEE Transactions on Visualization and Computer Graphics, 11(4):457–468, 2005.

[HN08] Mao Lin Huang und Quang Vinh Nguyen: Large graph visualization by hierarchical clustering. Journal of Software, 19(8):1933–1946, 2008.

[MCH+09] Tomer Moscovich, Fanny Chevalier, Nathalie Henry, Emmanuel Pietriga und Jean Daniel Fekete: Topology-aware navigation in large networks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Seiten 2319–2328. ACM, 2009.

[TAVHS06] Christian Tominski, James Abello, Frank Van Ham und Heidrun Schumann: Fisheye tree views and lenses for graph visualization. In: Tenth International Conference on Information Visualisation (IV'06), Seiten 17–24. IEEE, 2006.

[TGK+14] Christian Tominski, Stefan Gladisch, Ulrike Kister, Raimund Dachselt und Heidrun Schumann: A survey on interactive lenses in visualization. EuroVis State-of-the-Art Reports, 3(2), 2014.

[VHVW04] Frank Van Ham und Jarke J Van Wijk: Interactive visualization of small world graphs. In: IEEE Symposium on Information Visualization, Seiten 199–206. IEEE, 2004.

[Wal19] Julian Walter: Visualisierungsmöglichkeiten. Beigefügte PowerPoint Präsentation. 2019.

[yfi19] yFiles for Java, 2019. `https://www.yworks.com/products/yfiles-for-java`, besucht: 2019-06-27.