Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Informatik I
(Informationsstrukturen und wissensbasierte Systeme)

Diplomarbeit

Zentralitätsmaße in komplexen Netzwerken auf Basis kürzester Wege

Martin Fink

Abgabe: 10. März 2009

Betreuer:
Prof. Dr. Hartmut Noltemeier
Dipl.-Inf. Joachim Spoerhase

Inhaltsverzeichnis

1 Einleitung

1.1 Komplexe Netzwerke

1.1.1 Was sind komplexe Netzwerke?

Im alltäglichen Leben treffen wir auf etliche verschiedene Netzwerke, wie Caldarelli und Vespignani in [CV07, Einleitung] bemerken. Man betrachte dazu nur seinen eigenen Bekanntenkreis. Dieser bildet offenbar einen Teil eines großen *sozialen Netzwerkes*, in welchem je zwei Personen durch die Eigenschaft der „gegenseitigen Bekanntschaft“ verbunden sein können. Ebenso stellt das Straßennetz zwischen verschiedenen Städten ein Netzwerk dar.

Dagegen ist eine Enzyklopädie, in der ein Eintrag jeweils auf verschiedene andere Einträge verweisen kann, ein Beispiel für ein Netzwerk mit asymmetrischen Verbindungen, da die zwei verbundenen Einträge durch die Richtung der Referenzierung unterschieden werden können. Im Falle einer Online-Enzyklopädie, wie Wikipedia, ist das gut zu erkennen, da das Anklicken einer Referenz als Link eine Art Bewegung in Richtung des referenzierten Eintrages darstellt. Eine ähnliche Situation mit asymmetrischen Verbindungen gibt es auch bei einem Netzwerk zwischen wissenschaftlichen Artikeln, die sich gegenseitig zitieren. Ein solches Netzwerk wird häufig als „Citation Network“ bezeichnet.

Offensichtlichere Beispiele bilden oft direkt als solche bezeichnete Netzwerke, wie das Computernetz eines Unternehmens. Gerade das Internet stellt für viele das wohl bekannteste Netzwerk dar.

Doch was ist jetzt die Gemeinsamkeit, die es erlaubt, alle genannten Beispiele als Netzwerke zu bezeichnen? In allen Fällen gibt es zwei Arten von Grundelementen: Einerseits eine Menge von Objekten, andererseits eine Menge von Verbindungen zwischen je zwei dieser Objekte. Je nach der Art des Netzwerkes basieren die Verbindungen offensichtlich auf verschiedenen Eigenschaften, wie der „gegenseitigen Bekanntschaft“ beim sozialen Netzwerk oder der Eigenschaft „Artikel A zitiert Artikel B“. Auch die Art der Verbindungen kann unterschieden werden. Die letztgenannte Eigenschaft führt offenbar zu *gerichteten* Verbindungen, d.h. Verbindungen, in denen die beiden involvierten Objekte unterschieden werden können, da sie eine unterschied-

liche Rolle spielen. Im Falle des genannten Straßennetzes wären die Verbindungen dagegen *ungerichtet*, da die Straßen zwischen zwei Städten in beide Richtungen genutzt werden können, wobei wir hier annehmen, dass es zwischen zwei Städten keine Einbahnstraßen gibt.

Die Unterscheidung in Objekte und Verbindungen zwischen diesen legt eine *Modellierung* solcher Netzwerke als *Graph* nahe. Daher kommen auch Caldarelli und Vespignani zu dem Schluss:

> „Ein Netzwerk ist im Wesentlichen alles, was durch einen Graphen repräsentiert werden kann.“ [1]

Dabei entsprechen die Knoten des Graphen den Objekten des zugehörigen Netzwerkes, während die Verbindungen zwischen den Objekten durch Kanten – bzw. Pfeile im Falle gerichteter Verbindungen – dargestellt werden, die die zu den Objekten gehörigen Knoten verbinden. Zusatzeigenschaften von Graphen, wie zum Beispiel Knoten- oder Kantengewichte, können dabei genutzt werden, um erweiterte Eigenschaften von Netzwerken zu modellieren. Im Falle des erwähnten Städtenetzes könnte man so beispielsweise die Größe von Städten und die Distanzen zwischen ihnen auch im zugehörigen Graphen darstellen. Da wir komplexe Netzwerke also stets durch Graphen modellieren, werden wir im Folgenden häufig ein komplexes Netzwerk mit dem dieses modellierenden Graphen identifizieren.

Es bleibt noch zu klären, warum man solche Netzwerke als *komplexe Netzwerke* bezeichnet. Der Grund ist einfach, dass sie durch die Anzahl ihrer Knoten und Kanten (bzw. Objekte und Verbindungen zwischen diesen) die Größe, bis zu der sie für einen Menschen zu überblicken wären, im Normalfall um ein Vielfaches überschreiten. Allein durch die Anzahl der Knoten ist also die Komplexität der Netzwerke „das Hauptproblem bei ihrer Charakterisierung, Modellierung und ihrem Verständnis.“ [2]

1.1.2 Die Analyse komplexer Netzwerke und ihre Motivation

Komplexe Netzwerke werden durch Graphen modelliert, deshalb liefert die Graphentheorie die Hilfsmittel zur Analyse komplexer Netzwerke. Daher lässt sich die Netzwerk-Analyse auch als „angewandte Graphentheorie“[3] beschreiben. Bei der Analyse komplexer Netzwerke dienen also graphentheoretische Konstrukte wie Distanzen, kürzeste Wege oder die Breitensuche als

[1] [CV07, Einleitung, S. 2]
[2] [CV07, Vorwort, S. v]
[3] [BE05, S. 1]

bloße Hilfsmittel zur Beantwortung von Fragen wie „Welches ist das einflussreichste Objekt?“, „Welches ist die wichtigste Gruppe von Objekten im Netzwerk?“ oder „Was ist eine sinnvolle Aufteilung des Netzwerkes in Teilkomponenten?“.

Wir wollen einige Fragestellungen betrachten, die in konkreten komplexen Netzwerken auftreten. Dazu führen wir drei Beispiele von komplexen Netzwerken ein.

Beispiel 1 (Flughafennetz)*:* Wir betrachten das weltweite Flughafennetz. Mit einer Modellierung der Flughäfen als Knoten und der Direktverbindungen zwischen zwei Flughäfen als Kanten erhalten wir ein Modell des Flughafennetzes als Graph. Dabei genügt eine Direktverbindung zwischen zwei Flughäfen, um diese als verbunden zu betrachten.

Beispiel 2 (Städtenetz)*:* Wir wollen uns das schon als Beispiel genannte Städtenetz näher ansehen. Zur Modellierung sehen wir auch Kreuzungen von Verbindungsstraßen als Städte ohne Einwohner an. Im Modell stellen die Knoten die Städte und die Kanten die Straßen dar. Um die Distanzen zwischen Städten zu modellieren, kann man die Kanten gewichten. Zusätzlich lassen sich durch Knotengewichte die Einwohnerzahlen der Städte modellieren, um das Verkehrsaufkommen zwischen zwei Städten simulieren zu können. Bei einer genaueren Betrachtung wären weitere Dinge zu berücksichtigen, wie z.B. die Tatsache, dass zwischen weit entfernten Städten eher kein Berufsverkehr stattfindet.

Beispiel 3 (Computernetzwerk)*:* Auch das Computernetz, z.B. einer Universität, stellt ein komplexes Netzwerk dar. Wollen wir dieses als Graph modellieren, so werden die einzelnen Rechner und anderen Komponenten, wie Router oder WLAN-Komponenten, zu Knoten. Eine Kante zwischen zwei Knoten existiert, wenn die zugehörigen Komponenten ohne Umweg über eine dritte Komponente miteinander kommunizieren können.

Eine auftretende Fragestellung ist die, wie zentral gelegen oder wie „wichtig“ ein Knoten bzw. ein Objekt des Netzwerkes ist, was zum Begriff der *Zentralitätsmaße* führt, die wir in Kapitel 2 näher betrachten werden. Ebenso lässt sich diese Frage für die Kanten des Graphen bzw. die Verbindungen des Netzwerkes stellen. Im Falle des Flughafennetzes ist es interessant, die *Zentralität* der Flughäfen zu betrachten, da ein sehr zentraler Flughafen oft als Zwischenstation und Verteilungsknoten genutzt werden wird und somit für eine größere Menge an Reisenden und Flugzeugen ausgestattet sein muss.

Für das Städtenetz ist vor allem die Frage nach der Wichtigkeit der Straßen interessant. Hierbei sollte die Zentralität mit dem zu erwartenden Verkehrs-

aufkommen korrelieren. Durch ein geeignetes Maß für die Zentralität ließe sich also z.B. schon vor dem Bau einer neuen Verbindungsstraße das zu erwartende Verkehrsausmaß ermitteln und man könnte somit bestimmen, ob ein mehrspuriger Ausbau nötig ist.

Als Erweiterung und Verallgemeinerung der vorherigen Fragestellung wollen wir die Frage nach der wichtigsten/zentralsten Gruppe – d.h. Menge – von Knoten bzw. Kanten stellen, wobei die gewünschte Größe der Gruppe festgelegt oder zumindest nach oben begrenzt ist. In Kapitel 3 werden wir ein entsprechendes Maß kennenlernen und die Frage für dieses Maß in Kapitel 4 untersuchen. Auf das Beispiel des Städtenetzes bezogen, ließe sich die Frage nach einer Gruppe von höchstens k Verbindungsstraßen stellen, über die zusammengenommen der höchste Anteil an der Gesamtfahrzeugzahl fährt. Dies wäre z.B. interessant, um mit begrenztem Polizeipersonal bei einer Fahndung möglichst viele Fahrzeuge im Auge haben zu können. Da ein Fahrzeug im Normalfall mehrere Straßen benutzen wird, ist diese Frage nicht damit beantwortbar, dass man die k meistbefahrenen Straßen auswählt, weil man auf diese Weise unter Umständen einen großen Teil der Fahrzeuge mehrfach kontrolliert.

Kommen wir jetzt zum Computernetz. Hier stellt sich die Frage nach einer Gruppe von k Rechnern (oder anderen Komponenten), über die der größte Anteil am gesamten Datenverkehr verläuft. Diese Gruppe ist interessant, um mit dem Schutz weniger Rechner die Virenverbreitung im Netzwerk möglichst gut einzudämmen. Hier, wie auch bei anderen Beispielen, spielt also die Robustheit des Netzwerkes eine Rolle. Die wichtigsten Knoten oder Kanten sollen besonders geschützt oder in ihrer Kapazität verstärkt werden, um damit das ganze Netzwerk zu stützen. Puzis, Elovici und Dolev testen in [PED07b] und [PED07a] die Leistung des von ihnen entwickelten Approximationsalgorithmus zur Suche einer Gruppe mit einer hohen Zentralität anhand des Virenschutz-Beispiels experimentell. Wir werden diesen Algorithmus in Kapitel 4 kennenlernen und dann auch näher untersuchen.

1.2 Untersuchte Themen

Im Folgenden wird kurz vorgestellt, welche Teilaspekte der Analyse komplexer Netzwerke in dieser Arbeit untersucht werden sollen.

Zuerst werden wir in Kapitel 2 die sogenannten *Zentralitätsindizes* kennenlernen, mit deren Hilfe wir die Knoten (oder auch die Kanten) in einem komplexen Netzwerk charakterisieren. Dabei soll der Zentralitätsindexwert eines Knotens messen, wie *zentral* dieser ist. Da es auf den konkreten Anwendungsfall ankommt, was man unter „zentral“ zu verstehen hat, gibt es viele ver-

schiedene solche Indizes. Näher kennenlernen werden wir die *Shortest Path Betweenness Centrality*, ein Zentralitätsmaß, das auf dem Zählen von *kürzesten Wegen* im Graphen basiert. Dabei wird für den Knoten, dessen Zentralität bestimmt werden soll, ermittelt, welchen Anteil er an den kürzesten Wegen zwischen je zwei Knoten des Graphen hat.

Wir werden dabei auch einen Algorithmus zur schnellen Berechnung dieses Zentralitätsmaßes kennenlernen. Anschließend werden wir die Definition der *Shortest Path Betweenness Centrality* verallgemeinernd modifizieren, um nur Wege zwischen gewissen Paaren von Knoten zu berücksichtigen.

In Kapitel 3 werden wir die *Group Betweenness Centrality* kennenlernen, welche die *Shortest Path Betweenness Centrality* verallgemeinert, so dass wir die Zentralität von Mengen von Knoten messen können und dabei konsistent zum Maß für einzelne Knoten bleiben. Auch hier soll ein Algorithmus zur schnellen Berechnung vorgestellt und anschließend die *Group Betweenness Centrality* auch für Kanten definiert werden.

Mit Hilfe unserer Verallgemeinerung der Shortest Path Betweenness Centrality soll gezeigt werden, dass der Algorithmus zur Berechnung der Group Betweenness Centrality leicht auch zur Berechnung derselbigen für Gruppen von Kanten genutzt werden kann. Da diese einfache Technik allerdings zu einer Verlangsamung des Algorithmus führt, soll anschließend eine stärkere Modifikation des originalen Algorithmus entwickelt werden, so dass die Group Betweenness Centrality für Gruppen von Kanten genauso schnell wie für Gruppen von Knoten berechnet werden kann.

Im Kapitel 4 wird das Problem behandelt, die k-elementige Menge mit der höchsten Group Betweenness Centrality zu finden. Das entsprechende Problem, den Knoten mit der höchsten Shortest Path Betweenness Centrality (oder einem beliebigen anderen Zentralitätsmaß) zu finden, ist offenbar durch Berechnung aller Werte im Graphen leicht zu lösen. Da es allerdings exponentiell viele Teilmengen der Knoten gibt, ist dies hier schwieriger. Wir werden sehen, dass sich der Algorithmus zur Berechnung der Group Betweenness Centrality leicht in einen Greedy-Algorithmus umwandeln lässt, der das Problem approximiert.

Zum Schluss werden wir zeigen, dass dieser Algorithmus eine Approximationsgüte von $1 - \frac{1}{e}$ aufweist. Durch Konstruktion von Negativbeispielen wird außerdem gezeigt werden, dass die Abschätzung des Approximationsfaktors scharf ist, der Algorithmus das Problem also nicht mit einem Faktor c mit $c > 1 - \frac{1}{e}$ approximiert.

1.3 Vorhandene Literatur

Im Folgenden wird ein kurzer Überblick über die vorhandene und verwendete Literatur zu komplexen Netzwerken gegeben.

Eine gute Grundlage bildet [BE05]. Insbesondere auf das dritte Kapitel dieses Werkes sei hier verwiesen, in dem die Autoren neben einem Einstieg in die Zentralitätsindizes etliche Beispiele für solche angeben.

1.3.1 Verwendete Ergebnisse

Die in Kapitel 2 hauptsächlich betrachtete *Shortest Path Betweenness Centrality* wurde in [Fre77] eingeführt. Der Algorithmus zu ihrer schnellen Berechnung wurde unabhängig voneinander von Newman in [New01b] und von Brandes in [Bra01] entwickelt. Die Darstellung des Algorithmus, seiner Varianten und der Herleitung in dieser Arbeit sind an der Version von Brandes orientiert. Da wir aus Gründen der Konsistenz zu späteren Kapiteln eine Definition der Shortest Path Betweenness Centrality verwenden, die sich leicht von der von Brandes verwendeten unterscheidet, war es nötig, den Algorithmus, sowie die Herleitung und den zugehörigen Beweis anzupassen. Grundidee und Laufzeit bleiben aber erhalten.

In [Bra08] stellt Brandes einige Variationen der Shortest Path Betweenness vor und gibt Modifikationen seines Algorithmus an, mit denen die Variationen berechnet werden können. Hier findet sich auch die Modifikation zur Berechnung der Shortest Path Betweenness von Kanten mit gleicher Laufzeit wie für Knoten. Zudem werden auch einige weitere Varianten betrachtet, wie die Berücksichtigung von Knoten- und Kantengewichten.

Die *Group Betweenness Centrality* wird in [EB99] definiert. Der Algorithmus zu ihrer schnellen Berechnung stammt aus [PED07a] von Puzis, Elovici und Dolev. Dort und vor allem in [PED07b] behandeln diese auch das von ihnen definierte Problem KPP-Com, die Frage nach einer k-elementigen Gruppe von Knoten mit möglichst hoher Group Betweenness. Der von ihnen abgeleitete Greedy-Algorithmus zur Approximation dieses Problems soll später näher untersucht werden.

Die verwendete Definition des Maximum-Coverage-Problems und des zugehörigen Greedy-Algorithmus befindet sich in [JM08, Kapitel 4]. Ebenso findet man dort den Beweis zur Approximationsgüte des Greedy-Algorithmus. In [Fei98] wird schließlich gezeigt, dass sich Maximum-Coverage (dort Max k-Cover genannt) unter der Voraussetzung $\mathrm{P} \neq \mathrm{NP}$ in Polynomialzeit nicht besser als mit einer Güte von $1 - \frac{1}{e}$ approximieren lässt.

1.3.2 Weitere Arbeiten

Eine Übersicht über verschiedene Zentralitätsindizes findet sich in [BE06].

Newman und Girvan nutzen die Shortest Path Betweenness Centrality in [NG04], um komplexe Netzwerke möglichst gut in Teilkomponenten zu zerlegen. Dabei entfernen sie nacheinander die Kanten mit der höchsten Zentralität, wobei die Zentralitätswerte nach jedem Entfernungsschritt neu berechnet werden müssen. Auf diese Weise erhalten sie in $\mathcal{O}(nm^2)$ eine Reihenfolge der Entfernung von Kanten. Die entstehenden Aufteilungen des Graphen in Teilkomponenten vergleichen sie durch ein *Modularität* genanntes Maß hinsichtlich ihrer Güte. Newman gibt in [New04] einen alternativen Algorithmus dazu an, der direkt auf der Definition dieses Maßes basiert. Für diese Suche nach sogenannten *Gemeinschaftsstrukturen* (Community Structures) finden sich in [GN02] weitere Beispiele.

Ebenso von Newman stammt [New01a]. Dort führt er anhand *sozialer Netzwerke* in die Netzwerkkonstruktion ein, bevor er im zweiten Teil [New01b] seinen Algorithmus zur schnellen Berechnung der Shortest Path Betweenness Centrality vorstellt.

In [New05] wird eine Alternative zur Shortest Path Betweenness Centrality vorgestellt, die sogenannte *Random-Walk Betweenness*. Diese berücksichtigt nicht nur die kürzesten Wege zwischen zwei Knoten, sondern alle möglichen Wege zwischen ihnen. Es wird also angenommen, dass eine Information von einem Knoten zu einem anderen nicht auf kürzesten Wegen übertragen, sondern zufällig weitergeleitet wird, bis sie das Ziel erreicht. Auch in [WS03] findet man Alternativen zur Shortest Path Betweenness, die andere Mengen von Wegen als nur die kürzesten Wege zulassen, z.B. alle Wege, die nicht länger als ein $k \in \mathbb{N}$ sind.

In [ACEP09] wird ein Problem, ähnlich dem KPP-Com-Problem, aufgeworfen und für dünn besetzte Graphen – d.h. Graphen mit wenigen Kanten im Vergleich zur Anzahl der Knoten – untersucht.

1.4 Grundlagen

Weil komplexe Netzwerke durch Graphen modelliert werden, arbeiten wir stets mit solchen. Daher sind einige graphentheoretische Grundlagen zum Verständnis nötig, wofür auf [KN05] verwiesen sei. Um Missverständnissen vorzubeugen, sollen einige wichtige Definitionen angegeben werden, so wie wir sie später verwenden.

1.4.1 Graphentheorie

Komplexe Netzwerke können durch verschiedene Arten von Graphen dargestellt werden. Die in dieser Arbeit vorgestellten Konzepte und Algorithmen funktionieren meist auch mit gerichteten oder gewichteten Graphen, wobei dazu einige Modifikationen nötig sein können. Aus Gründen der Verständlichkeit wird hier aber stets mit einfachen, ungerichteten und ungewichteten Graphen gearbeitet.

Definition 1.1 (Graph)**:** Ein *Graph* ist ein Paar $G = (V, E)$ mit

$$E \subseteq \{\{u, v\} \mid u, v \in V \wedge u \neq v\}$$

Die Elemente $v \in V$ nennt man *Knoten* oder *Ecken*. $e = \{u, v\} \in E$ nennen wir eine *Kante* zwischen den Knoten u und v. Zur besseren Unterscheidbarkeit von allgemeinen Teilmengen von V schreiben wir auch $e = [u, v]$. u und v sind die beiden *Endknoten* von e.

Wir bezeichnen mit $n := |V|$ die Anzahl der Knoten des Graphen und die Anzahl der Kanten mit $m := |E|$. Sofern nicht anders angegeben, verwenden wir n und m stets in dieser Bedeutung.

Da wir für gewöhnlich reale Netzwerke modellieren, sind die betrachteten Graphen endlich, d.h. es gilt $n, m \in \mathbb{N}$.

Einige weitere Grundbegriffe für Graphen:

Definition 1.2 (Adjazenz, Inzidenz)**:** Zwei Knoten $u, v \in E$ heißen *adjazent* oder *benachbart*, wenn es eine Kante $e \in E$ gibt mit $e = [u, v] = [v, u]$. Man nennt u und v dann auch *Nachbarn*.

Ein Knoten $v \in V$ heißt *inzident* zu einer Kante $e \in E$, falls $v \in e$ gilt. Zwei Kanten $e, e' \in E$ heißen *inzident*, falls $e \cap e' \neq \emptyset$ ist.

Alle wichtigen Konzepte, die hier vorgestellt und untersucht werden, basieren auf dem Zählen von Wegen im Graphen.

Definition 1.3 (Weg)**:** Sei $P = (v_0, v_1, \ldots, v_k)$ eine endliche Liste von, nicht notwendigerweise verschiedenen, Knoten $v_0, \ldots, v_k \in V$ mit $k \geq 0$. Falls

$$\forall_{i=1}^{k} [v_{i-1}, v_i] \in E$$

gilt, so nennen wir P einen *Weg* oder *Pfad* im Graphen G. Wir schreiben dann auch $P = [v_0, v_1, \ldots, v_k]$, um zu verdeutlichen, dass P ein Weg und keine gewöhnliche Liste von Knoten ist.

v_0 ist der *Startknoten* und v_k der *Endknoten* des Weges P. Wir nennen P auch einen v_0-v_k-Weg. $|P| := k$ heißt die *Länge* des Weges.

Es lässt sich jetzt die Erreichbarkeit in Graphen über Wege definieren:

Definition 1.4 (Erreichbarkeit, Zusammenhang)**:** Seien $u, v \in V$. u heißt von v aus erreichbar, wenn es einen v-u-Weg P gibt. G heißt *zusammenhängend*, wenn für alle Paare $u, v \in V$ der Knoten u vom Knoten v aus erreichbar ist.

Bemerkung 1.5: Die Erreichbarkeit ist eine transitive Eigenschaft: Ist u von v aus über einen Weg P erreichbar und w von u über einen Weg P', so kann man P und P' zu einem v-w-Weg hintereinanderlegen.

Wenn also ein Graph nicht zusammenhängend ist, so muss es zu jedem Knoten v einen Knoten u geben, so dass es keinen v-u-Weg gibt.

Bemerkung 1.6: Alle im Weiteren betrachteten Graphen werden wir als zusammenhängend annehmen.

Durch den Vergleich von Weglängen kommen wir zum Begriff der Distanz im Graphen, sowie zu den kürzesten Wegen.

Definition 1.7 (Distanz, kürzester Weg)**:** Seien $u, v \in V$. Wir nennen

$$d_G(u, v) := \inf \{|P| \mid P \text{ ist } u\text{-}v\text{-Weg in } G\}$$

die *Distanz* von u und v in G. Wenn klar ist, welcher Graph gemeint ist, so schreiben wir einfach $d(u, v)$ statt $d_G(u, v)$.

Einen u-v-Weg P mit $|P| = d(u, v)$ nennen wir einen *kürzesten u-v-Weg*.

Bemerkung 1.8: Es gilt für alle Paare $u, v \in V$ offenbar $d(u, v) \in \mathbb{N}_0$, da wir nur zusammenhängende Graphen betrachten, in welchen stets ein u-v-Weg existiert.

Für jedes $v \in V$ ist $d(v, v) = 0$, denn der v-v Weg $P = [v]$ der Länge 0 existiert.

1.4.2 Optimierungsprobleme

In Abschnitt 4 wird das KPP-Com-Problem betrachtet und ein Approximationsalgorithmus für dieses untersucht. Daher sollen auch die für die Untersuchung von Optimierungsproblemen nötigen Grundbegriffe definiert werden.

Definition 1.9 (Optimierungsproblem)**:**
Ein Optimierungsproblem ist ein 4-Tupel

$$\mathcal{O} = (\mathcal{I}, \mathcal{L}, l, c)$$

bestehend aus

- einer Menge $\mathcal{I}$ von *Instanzen*
- einer Menge $\mathcal{L}$, dem *Lösungsraum*

- einer totalen Funktion $l : \mathcal{I} \to \mathcal{P}(\mathcal{L})$, die jeder Instanz $I \in \mathcal{I}$ eine Menge $l(I) \subseteq \mathcal{L}$ von *zulässigen Lösungen* zuweist
- einer *Zielfunktion* $c : \mathcal{L} \to \mathbb{R}$, die jeder Lösung L einen Wert $c(L)$ zuweist

Bei einem *Maximierungsproblem* ist es das Ziel, zu jeder Instanz I eine zulässige Lösung $L \in l(I)$ zu finden, so dass $c(L)$ maximal ist. Analog ist bei einem Minimierungsproblem zu jeder Instanz I eine Lösung $L \in l(I)$ mit $c(L)$ minimal gesucht.

In dieser Arbeit werden nur Maximierungsprobleme eine Rolle spielen.

Definition 1.10 (α-Approximationsalgorithmus)**:**
Sei ein Maximierungsproblem

$$\mathcal{O} = (\mathcal{I}, \mathcal{L}, l, c)$$

gegeben. Für jede Instanz $I \in \mathcal{I}$ sei $\mathrm{OPT}(I)$ eine *optimale Lösung*, d.h. eine zulässige Lösung $L \in l(I)$, für die $c(L)$ maximal ist.

Ein Algorithmus $\mathcal{A}$ *approximiert* das Maximierungsproblem $\mathcal{O}$, wenn er jeder Instanz I eine zulässige Lösung $\mathcal{A}(I) \in l(I)$ zuweist. Sei $0 \leq \alpha \leq 1$. Gilt

$$\forall_{I \in \mathcal{I}} c(\mathcal{A}(I)) \geq \alpha \cdot c\,(\mathrm{OPT}(I))$$

so heißt $\mathcal{A}$ ein *α-Approximationsalgorithmus* für das Maximierungsproblem $\mathcal{O}$. Man sagt auch „$\mathcal{A}$ approximiert $\mathcal{O}$ mit dem *Approximationsfaktor* α“ und nennt α auch die *Approximationsgüte* von $\mathcal{A}$.

1.4.3 Weitere Grundlagen

Zur Angabe der Laufzeit von Algorithmen werden wir die $\mathcal{O}$-Notation verwenden.

Definition 1.11 ($\mathcal{O}$-Klasse)**:** Sei $f : \mathbb{N} \to \mathbb{R}$ gegeben. Dann sei

$$\mathcal{O}(f) := \{g \mid g : \mathbb{N} \to \mathbb{R} \wedge \exists_{c \in \mathbb{R}} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} g(n) \leq c \cdot f(n)\}$$

Für weitere Begriffe wie die Laufzeit eines Algorithmus, polynomielle Laufzeit, die Klassen P und NP, sowie NP-Vollständigkeit sei wieder auf [KN05] oder auch auf [Wag03, Kapitel 3] – für eine genauere Einführung – verwiesen.

Ein weiterer Begriff, der später eine Rolle spielen wird, ist der des *Single-Source-Shortest-Path-Problems*. Dabei geht es darum, von einem ausgezeichneten Knoten s kürzeste Wege zu allen anderen Knoten des Graphen zu finden. So wie dieses Problem z.B. in [KN05] definiert ist, wird nur nach einem kürzesten Weg zu jedem Knoten des Graphen oder sogar nur nach der Länge

eines kürzesten Weges gefragt. Da wir in Kapitel 2 aber zumindest die Anzahl aller kürzesten Wege von s zu jedem Knoten v benötigen, soll es hier so definiert sein, dass nach allen kürzesten Wegen von s aus gefragt wird.

Definition 1.12 (Single-Source-Shortest-Path-Problem)**:**
Gegeben: Ein Graph $G = (V, E)$; ein $s \in V$
Gesucht: Für alle $v \in V$ die Distanz $d(s, v)$ und alle kürzesten s-v-Wege

Die kürzesten Wege können durch eine Menge von Vorgängern $P[v]$ für jeden Knoten $v \neq s$ ausgegeben werden, da sich jeder kürzeste s-v-Weg aus einem kürzesten Weg zu einem Knoten $w \in V$ und der Kante $[w, v]$ zusammensetzt. Mit Hilfe der *Breitensuche* kann das Single-Source-Shortest-Path-Problem in $\mathcal{O}(m)$ gelöst werden. Diese durchläuft alle Knoten in zunehmendem Abstand zu s und betrachtet ihre Nachbarn, vergleiche Algorithmus 10 in Anhang A.1. Bei den verschiedenen Variationen der Shortest Path Betweenness Centrality in Kapitel 2 wird die Breitensuche als Grundgerüst zur Berechnung dienen.

2 Zentralitätsmaße und ihre Berechnung

2.1 Zentralität

Eine wichtige Fragestellung in einem komplexen Netzwerk ist es, wie *zentral* ein Knoten (oder auch eine Kante) gelegen ist. Darüber, was genau „zentral“ bedeutet, herrscht allerdings keine Einigkeit, weshalb es viele verschiedene Maße für die *Zentralität* gibt, welche man *Zentralitätsindizes* oder *Zentralitätsmaße* nennt. Daher gibt es nur eine schwache gemeinsame Forderung, nämlich, dass jeder Zentralitätsindex ein *Strukturindex* sein muss, vergleiche [BE05, Abschnitt 3.2].

Definition 2.1 (Strukturindex)**:** Sei $\mathcal{G}$ eine Menge von Graphen und entweder $X = V$ oder $X = E$. Sei s eine Abbildung $s : G \in \mathcal{G} \mapsto s_G$, die jedem Graph G eine Funktion $s_G : X(G) \to \mathbb{R}$ zuweist. Falls

$$\forall_{G,H\in\mathcal{G}} : G \cong H \text{ mit Isomorphismus } \varphi \Rightarrow \left(\forall_{x\in X(G)} s_G(x) = s_H(\varphi(x))\right)$$

so ist s ein *Strukturindex*. Ist G fest, so schreibt man auch $s(x)$ statt $s_G(x)$.

Das bedeutet, dass jeder Strukturindex unabhängig von „Umbenennung“ des Graphen sein muss.

Wir fordern also, dass jeder Zentralitätsindex ein Strukturindex ist. Zudem soll ein Knoten umso zentraler sein, je höher sein Zentralitätsindex ist.

2.2 Beispiele für Zentralitätsindizes

Wir wollen einige einfache Zentralitätsindizes betrachten. Dabei benutzen wir den Graphen aus Abbildung 1, um die Maße zu vergleichen. In [BE05, Kapitel 3] werden, neben den hier genannten, auch einige weitere Indizes vorgestellt.

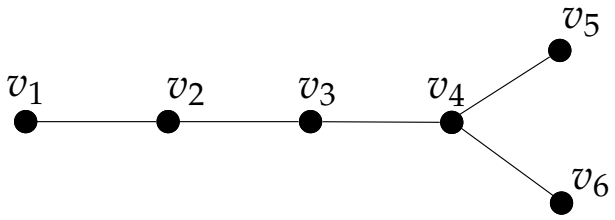

Abbildung 1: Beispielgraph

2.2.1 Grad

Unserem ersten Beispielmaß liegt die Annahme zu Grunde, dass zentrale Knoten mehr Nachbarn haben als weniger zentrale. Durch einfaches Zählen der Nachbarn der Knoten erhalten wir somit einen ersten Zentralitätsindex.

Definition 2.2 (Grad)**:** Sei $G = (V, E)$ gegeben und $v \in V$. Dann heißt

$$g(v) := |\{u \in V \mid v \text{ und } u \text{ sind in } G \text{ adjazent}\}|$$

der *Grad* von v.

Der Grad eines Knotens ist offenbar in isomorphen Graphen gleich. Somit bildet der Grad einen Strukturindex nach Definition 2.1 und kommt damit als Zentralitätsindex in Frage. In Abbildung 2 sehen wir die Grade der Knoten des Beispielgraphen eingetragen. Nach den Graden zu urteilen ist somit der Knoten v_4 derjenige mit der höchsten Zentralität.

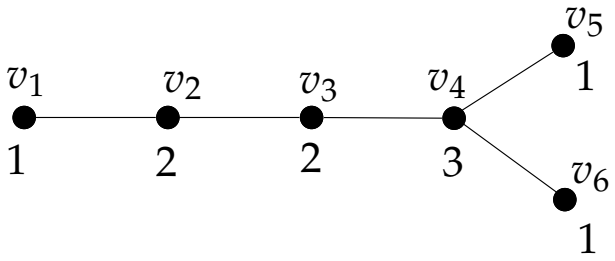

Abbildung 2: Grade im Beispielgraphen

2.2.2 Exzentrizität

Eine weitere Idee für die Messung der Zentralität ist es, den Abstand von einem Knoten v zu einem von diesem am weitesten entfernten Knoten zu messen. Ist dieser Abstand gering, so nehmen wir an, dass der Knoten recht zentral im Graphen liegt. Dies wird über die Exzentrizität gemessen.

Definition 2.3 (Exzentrizität)**:** Sei $G = (V, E)$ und $v \in V$. Dann heißt

$$e(v) := \sup_{u \in V} d(u, v)$$

die *Exzentrizität* von v.

Bemerkung 2.4: Da wir nur endliche, zusammenhängende Graphen betrachten, gibt es zu jedem Knoten v einen von diesem am weitesten entfernten Knoten u und es gilt $e(v) = d(v, u) \in \mathbb{N}_0$.

Die Menge $C := \{v \in V \mid e(v) = \inf_{u \in V} e(u)\}$ heißt das *Zentrum* des Graphen.

Da wir wollen, dass hohe Werte für eine hohe Zentralität stehen, müssen wir zum Kehrwert der Exzentrizität übergehen.

Definition 2.5: Sei G ein Graph mit mindestens zwei Knoten. Für $v \in V$ heißt

$$c_e(v) := \frac{1}{e(v)}$$

die *Exzentrizitäts-Zentralität* von v.

Bemerkung 2.6: Es ist stets $0 < c_e(v) \leq 1$. Die Knoten aus dem Zentrum weisen die höchste Zentralität auf.

Auch die Exzentrizitäts-Zentralität ist offensichtlich ein Strukturindex. In Abbildung 3 sehen wir die entsprechenden Werte für den Beispielgraphen. In diesem Falle weist der Knoten v_3 die höchste Zentralität auf.

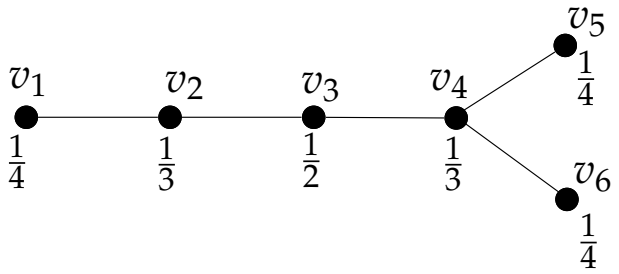

Abbildung 3: Exzentrizitäts-Zentralität im Beispielgraphen

2.2.3 Distanzsumme

Anstatt nur die höchste Distanz im Graphen zu beachten, wollen wir die Distanzen zu allen Knoten berücksichtigen. Dies erreichen wir durch einfaches

Aufsummieren der Distanzen. Der Knoten, für den die Summe der Distanzen minimal ist, hat minimalen Durchschnittsabstand zu allen Knoten des Graphen und wird daher zentral gelegen sein.

Definition 2.7 (Distanzsumme)**:** Sei $G = (V, E)$ und $v \in V$. Dann ist

$$s(v) := \sum_{u \in V} d(v, u)$$

die *Distanzsumme* von v.

Bemerkung 2.8: Da die betrachteten Graphen zusammenhängend sind, gilt natürlich $s(v) \in \mathbb{N}_0$ für alle Knoten $v \in V$.

Damit die Knoten mit minimaler Distanzsumme den höchsten Zentralitätswert aufweisen, müssen wir wiederum zum Kehrwert übergehen.

Definition 2.9: Sei G ein Graph mit mindestens zwei Knoten. Für $v \in V$ heißt

$$c_s(v) := \frac{1}{s(v)}$$

die *Distanzsummen-Zentralität* von v.

Bemerkung 2.10: Auch hier gilt für alle Knoten $v \in V$: $0 < c_s(v) \leq 1$

Die Strukturindex-Eigenschaft ist wiederum klar. Wie in Abbildung 4 zu sehen, haben diesmal die Knoten v_3 und v_4 gemeinsam die höchste Zentralität.

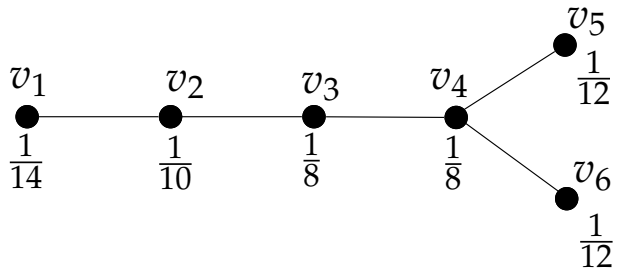

Abbildung 4: Distanzsummen-Zentralität im Beispielgraphen

2.2.4 Zusammenfassung

Wir haben drei Beispiele für Zentralitätsindizes gesehen, die jeweils auf verschiedenen Eigenschaften von Knoten basieren. Schon in dem gewählten kleinen Beispiel liefern alle drei Indizes eine verschiedene Menge von Knoten mit

der höchsten Zentralität. Aber auch zwei andere Knoten, die bezüglich des einen Indizes dieselbe Zentralität besitzen, können bei einem anderen Zentralitätsmaß verschiedene Werte aufweisen. Es ist also offenbar nicht möglich, sich auf einen universellen Zentralitätsindex zu einigen. Vielmehr haben – je nach Anwendung – verschiedene Indizes ihre Daseinsberechtigung.

2.3 Zentralität von Kanten

2.3.1 Zentralität für Knoten und Kanten

Die meisten Zentralitätsmaße sind für Knoten definiert. Allerdings ist es oft auch sinnvoll, die Zentralität von Kanten zu betrachten, wobei man meist in einem Graphen auf das gleiche beziehungsweise ein ähnliches Maß für Knoten und Kanten zurückgreifen möchte. Daher stellt sich die Frage, mit welchem Verfahren man aus einem Zentralitätsmaß für Knoten eines für Kanten gewinnen kann.

Eine Idee ist es, dabei die *Line-Graph-Konstruktion* zu verwenden, um einen zweiten Graphen zu konstruieren, in dem die ehemaligen Kanten die neuen Knoten sind und umgekehrt. Die Rollen von Knoten und Kanten werden also vertauscht.

Definition 2.11 (Line-Graph, vgl. [KN05])**:** Sei $G = (V, E)$ ein einfacher, ungerichteter Graph mit $E \neq \emptyset$. Setze $L(V) := E$ und

$$L(E) := \big\{[e, e'] \mid e \in E \text{ und } e' \in E \text{ sind in } G \text{ inzident}\big\}$$

Dann heißt $L(G) := (L(V), L(E))$ der *Line-Graph* zu G.

Dies liefert allerdings oft nicht das gewünschte Ergebnis, da viele Zentralitätsmaße auf dem Zählen kürzester Wege basieren und diese im Linegraphen andere sind, siehe z.B. Satz 1 in [Bra08].

2.3.2 Der Inzidenzgraph

Ein anderer Ansatz ist der sogenannte *Inzidenzgraph* (vgl. [BE05, Abschnitt 3.5]). Dabei wird auf jeder Kante des Originalgraphen ein zusätzlicher „Kantenknoten“ platziert, dessen Zentralität dann die Zentralität der Kante ist.

Definition 2.12 (Inzidenzgraph)**:** Sei $G = (V, E)$ gegeben. Setze $V' := V \cup E$ und

$$E' := \{[v, e] \mid e \in E \wedge v \in V \wedge e \text{ und } v \text{ sind in } G \text{ inzident}\}$$

Dann heißt $G' := (V', E')$ der *Inzidenz-Graph* zu G.

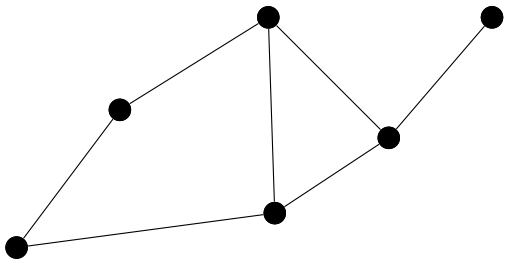

Abbildung 5: Beispielgraph G

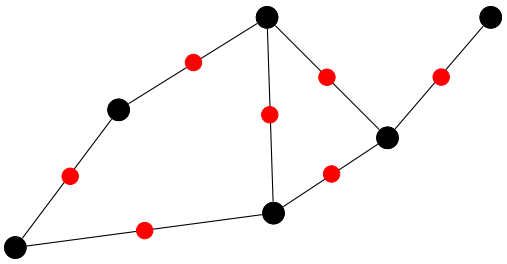

Abbildung 6: Inzidenzgraph $L(G)$ zu G

Je nach Zentralitätsmaß muss man dazu aber im neuen Inzidenzgraphen die originalen Knoten von den neuen unterscheiden, wie wir in Abschnitt 2.6.3 sehen werden, wo diese Konstruktion explizit angewendet werden wird. In vielen Fällen bietet sich auch eine implizite Nutzung an, bei der der Inzidenzgraph nicht wirklich erzeugt wird, sondern nur die Grundidee bildet. Solche Fälle werden wir in 2.5 und 3.4 kennenlernen beziehungsweise selbst herleiten.

2.4 Die Shortest Path Betweenness Centrality

Eine fortgeschrittenere Idee, die Zentralität von Knoten zu messen, ist es, zu bestimmen, wie stark ein Knoten an der Übertragung von irgendeiner Art von Information im Netzwerk beteiligt wäre. Für die im Weiteren vorgestellte *Shortest Path Betweenness Centrality* wird dabei die Vorstellung zu Grunde gelegt, dass eine solche Informationsübertragung immer auf kürzesten Wegen stattfindet. Im Gegensatz dazu wird bei manchen anderen Ansätzen eine größere Menge erlaubter Wege unterstellt, wie z.B. alle Wege, die eine Maximallänge nicht überschreiten, vergleiche die in 1.3.2 genannten Werke [New05] und [WS03].

Die *Shortest Path Betweenness Centrality* ist ein Zentralitätsmaß, das auf dem Zählen kürzester Wege basiert. Für einen Knoten $v \in V$ wird angenommen, dass er umso zentraler liegt, je mehr er an möglichen Kommunikationswegen zwischen Paaren von Knoten beteiligt ist. Dabei geht man davon aus, dass als Kommunikationswege jeweils nur *kürzeste Wege* gewählt werden.

Für jedes Paar $s, t \in V$ von Knoten wird dabei der Einfluss von v für die Kommunikation zwischen s und t gemessen als der Anteil aller kürzesten s-t-Wege, die v berühren. Durch Aufsummieren des Einflusses von v auf die Kommunikation zwischen allen Paaren von Knoten erhält man die *Shortest Path Betweenness Centrality*.

Die Shortest Path Betweenness Centrality wurde in [Fre77] eingeführt. Es

gibt einige Variationsmöglichkeiten bei ihrer Definition, vor allem die Möglichkeit, s-t-Wege bei der Zentralität von s und t mitzuzählen oder nicht. Daher verwenden wir aus Konsistenzgründen im Folgenden stets die auch in [PED07b] verwendete Definition und Notation.

2.4.1 Definition

Gegeben sei ein Graph $G = (V, E)$.

Definition 2.13: Für $s, t \in V$ sei

$$\sigma_{s,t} := |\{P \mid P \text{ ist kürzester } s\text{-}t\text{-Weg in } G\}|$$

die Anzahl kürzester Wege zwischen s und t in G.

Bemerkung 2.14: Für alle $v \in V$ ist $\sigma_{v,v} = 1$.

Wir wollen den Anteil, den v an kürzesten s-t-Wegen hat, messen. Dazu benötigen wir noch die Anzahl der kürzesten s-t-Wege, die den Knoten v benutzen.

Definition 2.15: Mit

$$\sigma_{s,t}(v) := |\{P \mid P \text{ ist kürzester } s\text{-}t\text{-Weg in } G \ \wedge v \in P\}|$$

sei für ein beliebiges $v \in V$ die Anzahl kürzester s-t-Wege, auf denen v liegt, bezeichnet.

Bemerkung 2.16: Es ist $\sigma_{s,t}(s) = \sigma_{s,t}(t) = \sigma_{s,t}$.

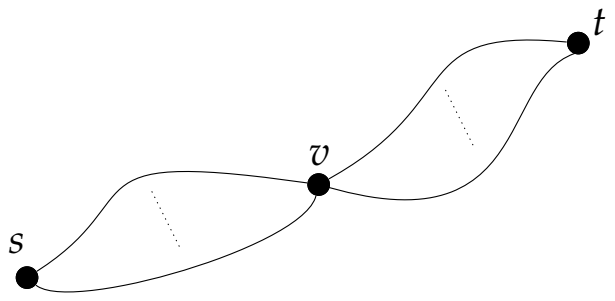

Abbildung 7: Kombination von s-v-Wegen und v-t-Wegen zu s-t-Wegen

Falls v überhaupt auf einem kürzesten s-t-Weg liegt, so muss

$$d(s,t) = d(s,v) + d(v,t)$$

gelten. In diesem Fall ist, wie in Abbildung 7 dargestellt, jeder s-t-Weg durch v aus einem s-v-Weg und einem v-t-Weg kombiniert. Analoges gilt natürlich auch für kürzeste Wege. Man erhält also:

Eigenschaft 2.17: Für beliebige $s, t, v \in V$ gilt:

$$\sigma_{s,t}(v) = \begin{cases} \sigma_{s,v} \cdot \sigma_{v,t} & \text{falls } d(s,t) = d(s,v) + d(v,t) \\ 0 & \text{sonst } (v \text{ liegt auf keinem kürzesten } s\text{-}t\text{-Weg}) \end{cases}$$

Wie in Abbildung 8 dargestellt, wissen wir jetzt, dass der Anteil der s-t-Wege, die durch v laufen, $\frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ beträgt. Jetzt müssen wir nur noch diesen Anteil für alle Paare s, t addieren.

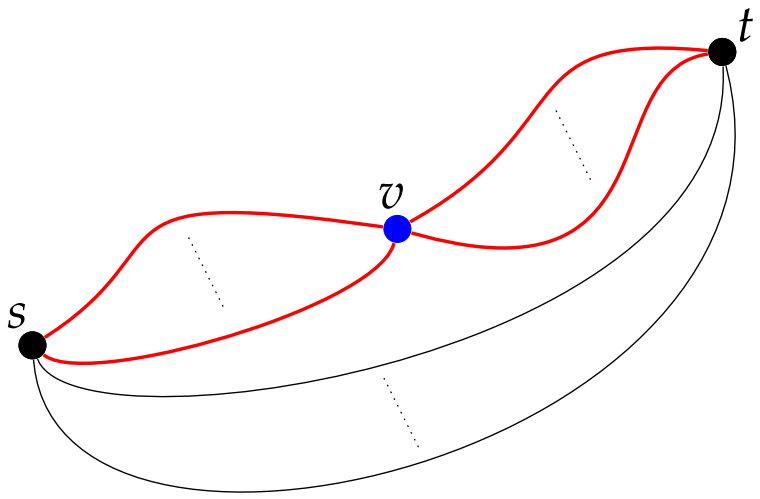

Abbildung 8: Anteil $\frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ der kürzesten s-t-Wege läuft durch v

Definition 2.18 (Shortest Path Betweenness Centrality)**:** Die *Shortest Path Betweenness Centrality* von $v \in V$ ist definiert als

$$\mathrm{B}(v) := \sum_{s,t \in V | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

Wir wollen die Doppelsumme umschreiben, wozu wir noch eine Definition benötigen.

Definition 2.19: Für $s, v \in V$ sei $\delta_{s,\bullet}(v)$ der Einfluss von v auf kürzesten Pfade, die in s beginnen:

$$\delta_{s,\bullet}(v) := \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

Mit Hilfe der Teilsummen $\delta_{s,\bullet}(v)$ erhalten wir jetzt:

Eigenschaft 2.20: Für $v \in V$ gilt:

$$\mathrm{B}(v) = \sum_{s,t \in V | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} = \sum_{s \in V} \delta_{s,\bullet}(v)$$

2.4.2 Berechnung der Shortest Path Betweenness Centrality

Durch Lösen jeweils eines *Single-Source-Shortest-Path-Problems* (vergleiche Abschnitt 1.4.3) für jeden Knoten $s \in V$ lassen sich alle Distanzen $d(s,t)$ im Graphen sowie die Anzahlen $\sigma_{s,t}$ der kürzesten Wege zwischen allen Paaren von Knoten mit Hilfe der Breitensuche in $\mathcal{O}(nm)$ berechnen. Mit Eigenschaft 2.17 lassen sich dann die einzelnen Summenglieder $\frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ jeweils in $\mathcal{O}(1)$ berechnen. Durch einfaches Aufsummieren erhält man anschließend in $\mathcal{O}(n^2)$ die Shortest Path Betweenness für einen Knoten v bzw. in $\mathcal{O}(n^3)$ für alle Knoten, vergleiche auch [BE05, Kapitel 4.2]. In Anhang A.2 findet sich der entsprechende Algorithmus 11 in Pseudocode.

2.4.3 Schnelle Berechnung der Shortest Path Betweenness Centrality

Brandes hat einen neuen Algorithmus entwickelt, der nur $\mathcal{O}(nm)$ benötigt, um die Shortest Path Betweenness Centrality für alle Knoten gleichzeitig zu berechnen ([Bra01]). Unabhängig davon hat Newman in [New01b] dasselbe Verfahren entwickelt. Wir werden uns im Folgenden an der Version von Brandes orientieren. Allerdings ist zu berücksichtigen, dass Brandes die Shortest Path Betweenness dabei etwas anders definiert hat, er setzt nämlich $\sigma_{s,t}(s) = \sigma_{s,t}(t) = 0$. Aus diesem Grunde wird hier eine modifizierte Version seines Algorithmus vorgestellt. Auch die Aussage von Satz 2.21 und der zugehörige Beweis unterscheiden sich von Brandes Versionen. Die Idee bleibt allerdings dieselbe.

Die Berechnung verläuft dabei nicht über das Bilden der vollen Summe der Werte $\frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ über alle $s,t \in V$, sondern es werden die Teilsummen $\delta_{s,\bullet}(v)$ jeweils für alle v in einem Durchlauf gebildet und dann wie in Eigenschaft 2.20 über alle s summiert. Der Trick dabei ist die Nutzung einer Rekursionsformel für die Teilsummen:

Satz 2.21: *Seien $s, v \in V$ mit $s \neq v$. Es sei für $w \in V$*

$$P_s(w) := \{u \in V \mid \exists_P : P = [s, \ldots, u, w] \textit{ ist kürzester s-w-Weg}\}$$

die Menge der Vorgänger von w auf kürzesten Wegen von s aus. Dann gilt:

$$\delta_{s,\bullet}(v) = 1 + \sum_{w|v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta_{s,\bullet}(w)$$

Beweis. Ist $d(s,v)$ maximal unter den von s aus erreichbaren Knoten, so liegt v auf keinen kürzesten s-t Wegen, außer den s-v-Wegen. Wegen $\sigma_{s,v}(v) = \sigma_{s,v}$ gilt

$$\delta_{s,\bullet}(v) = \frac{\sigma_{s,v}(v)}{\sigma_{s,v}} = 1 = 1 + 0 = 1 + \sum_{w|v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta_{s,\bullet}(w)$$

da es kein $w \in V$ mit $v \in P_s(w)$ gibt.

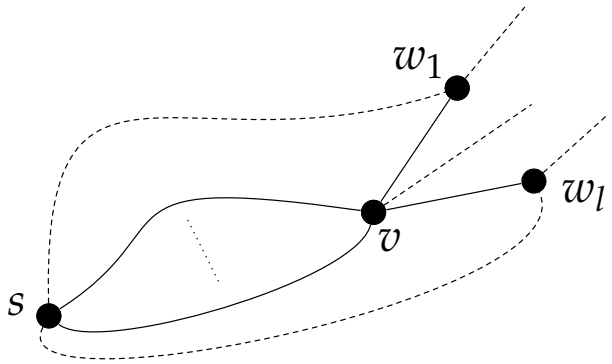

Abbildung 9: Situation für $v \in V$

Es existiere jetzt also mindestens ein $w \in V$, so dass $v \in P_s(w)$. Wir befinden uns in einer Situation wie in Abbildung 9 dargestellt. Schreibe $\delta_{s,t}(v) := \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ und damit $\delta_{s,\bullet}(v) = \sum_{t \in V|s \neq t} \delta_{s,t}(v)$. Für $e \in E$ sei zusätzlich $\sigma_{s,t}(v,e)$ die Anzahl der kürzesten s-t-Wege, die v und e enthalten, sowie $\delta_{s,t}(v,e) := \frac{\sigma_{s,t}(v,e)}{\sigma_{s,t}}$.

Wir müssen für $t = v$ die kürzesten s-v-Wege beachten. Ist $t \neq v$, so liegt v nur auf kürzesten s-t-Wegen, wenn t von s aus gesehen „hinter“ v liegt. Es muss also außer v noch mindestens eine Kante zu einem Nachfolger w von v (also einem Knoten mit $v \in P_s(w)$) auf einem relevanten s-t-Weg liegen. Damit gilt:

$$\delta_{s,t}(v) = \begin{cases} 1 & \text{falls} \quad t = v \\ \sum_{w|v \in P_s(w)} \delta_{s,t}(v,[v,w]) & \text{sonst} \end{cases}$$

Da für $w \in V$ mit $v \in P_s(w)$ auch $\delta_{s,v}(v,[v,w]) = 0$ ist, erhalten wir:

$$\begin{aligned}
\delta_{s,\bullet}(v) &= \delta_{s,v}(v) + \sum_{t \in V \setminus \{v\} | s \neq t} \delta_{s,t}(v) \\
&= 1 + \sum_{t \in V \setminus \{v\} | s \neq t} \sum_{w | v \in P_s(w)} \delta_{s,t}(v,[v,w]) \\
&= 1 + \sum_{t \in V | s \neq t} \sum_{w | v \in P_s(w)} \delta_{s,t}(v,[v,w]) \\
&= 1 + \sum_{w | v \in P_s(w)} \sum_{t \in V | s \neq t} \delta_{s,t}(v,[v,w])
\end{aligned}$$

Wegen $[v,w] \in E$ sind die kürzesten s-t-Wege, die v und $[v,w]$ benutzen, gerade die, die die Kante $[v,w]$ in dieser Reihenfolge durchlaufen. Für $w = t$ liegt t natürlich auf allen kürzesten s-t-Wegen, von denen wiederum ein Anteil von

$$\frac{\sigma_{s,w}(v)}{\sigma_{s,w}} = \frac{\sigma_{s,v}}{\sigma_{s,w}} = \frac{\sigma_{s,v}}{\sigma_{s,t}}$$

vorher durch v verläuft. Ist $w \neq t$, so verläuft ebenfalls ein Anteil von $\frac{\sigma_{s,v}}{\sigma_{s,w}}$ der kürzesten s-t-Wege durch w vorher durch v. Zusammen mit $\frac{\sigma_{s,w}(w)}{\sigma_{s,w}} = 1$ gilt:

$$\begin{aligned}
\delta_{s,t}(v,[v,w]) &= \begin{cases} \frac{\sigma_{s,v}}{\sigma_{s,w}} & \text{für } t = w \\ \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \frac{\sigma_{s,t}(w)}{\sigma_{s,t}} & \text{sonst} \end{cases} \\
&= \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \frac{\sigma_{s,t}(w)}{\sigma_{s,t}}
\end{aligned}$$

Es gilt also

$$\begin{aligned}
\sum_{t \in V | s \neq t} \delta_{s,t}(v,[v,w]) &= \sum_{t \in V | s \neq t} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \frac{\sigma_{s,t}(w)}{\sigma_{s,t}} \\
&= \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(w)}{\sigma_{s,t}} \\
&= \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta_{s,\bullet}(w)
\end{aligned}$$

und man erhält:

$$\begin{aligned}
\delta_{s,\bullet}(v) &= 1 + \sum_{w | v \in P_s(w)} \sum_{t \in V | s \neq t} \delta_{s,t}(v,[v,w]) \\
&= 1 + \sum_{w | v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta_{s,\bullet}(w)
\end{aligned}$$

□

Bemerkung 2.22: Im verbleibenden Fall $v = s$ gilt $\delta_{s,\bullet}(s) = n - 1$.

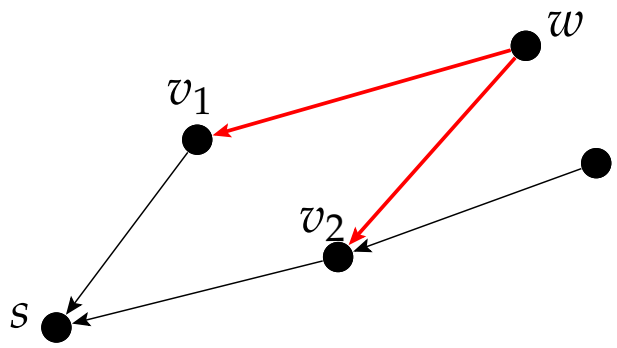

Abbildung 10: $\frac{\sigma_{s,v_i}}{\sigma_{s,w}} \cdot \delta_{s,\bullet}(w)$ wird an Vorgänger v_i von w weitergegeben

Der Algorithmus berechnet nacheinander für jedes $s \in V$ die Teilsummen $\delta_{s,\bullet}(v)$ für alle $v \in V$ gleichzeitig und summiert diese auf. Dabei wird zuerst in $\mathcal{O}(m)$ das Single-Source-Shortest-Path-Problem für s mittels Breitensuche gelöst, wobei man die Distanzen aller Knoten zu s erhält, sowie die Vorgängermengen $P_s(v)$. Zusätzlich wird ein Stack aufgebaut, um anschließend nacheinander in absteigendem Abstand zu s auf alle Knoten zugreifen zu können. Dabei wird für jedes betrachtete $w \in V$ gemäß Satz 2.21 zuerst 1 zu $\delta_{s,\bullet}(w)$ addiert und anschließend ein Wert von $\frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta_{s,\bullet}(w)$ an jeden Vorgänger $v \in P_s(w)$ weitergegeben. Da die Knoten in abnehmender Distanz zu s durchlaufen werden, wurden alle Knoten, die w als Vorgänger haben, schon vorher behandelt. Der Wert $\delta_{s,\bullet}(w)$ ist nach der Addition von 1 also korrekt. Für die Knoten w' mit maximalem Abstand zu s erhält man dabei richtigerweise den Wert $\delta_{s,\bullet}(w') = 1$. Zum Schluss muss noch $\delta_{s,\bullet}(s) = n - 1$ berücksichtigt werden.

In Algorithmus 1 sehen wir das resultierende Verfahren zum Einsammeln der Werte. Für eine vollständige Version des Algorithmus, einschließlich dem Lösen des Single-Source-Shortest-Path-Problems, siehe Algorithmus 12 in Anhang A.3.

Laufzeit Beim Sammeln der $\delta_{s,\bullet}$-Werte werden alle Kanten nochmals rückwärts in Richtung s durchlaufen. Daher wird auch hierfür $\mathcal{O}(m)$ benötigt. Da insgesamt n Single-Source-Shortest-Path-Probleme zu lösen sind, benötigt der neue Algorithmus also nur $\mathcal{O}(nm)$ um die Shortest Path Betweenness Centrality für alle Knoten auf einmal zu berechnen.

Algorithmus 1 : Shortest Path Betweenness Centrality

Eingabe : $G = (V, E)$
Ausgabe : $\mathrm{B}(v)$ für alle $v \in V$

```
1  ∀_{v∈V} B(v) := 0;
2  foreach s ∈ V do
3      ∀_{v∈V} δ_{s,•}(v) := 0;
4      Löse Single-Source-Shortest-Path-Problem für s
5      S ist Stack zum Zugriff auf die Knoten in abnehmender Entfernung
       zu s
6      while S nicht leer do
7          pop w ← S;
8          δ_{s,•}(w) := δ_{s,•}(w) + 1;
9          for v ∈ P_s(w) do δ_{s,•}(v) := δ_{s,•}(v) + (σ_{s,v} / σ_{s,w}) · δ_{s,•}(w);
10         if w ≠ s then B(w) := B(w) + δ_{s,•}(w);
11     B(s) := B(s) + n − 1;
12 return B;
```

2.5 Die Shortest Path Betweenness Centrality für Kanten

Für Kanten lässt sich ebenfalls die Shortest Path Betweenness Centrality definieren. Dabei werden völlig analog zum Fall der Knoten die Anteile an kürzesten Wegen gezählt, die durch eine Kante verlaufen.

2.5.1 Definition

Definition 2.23: Für eine Kante $e = [u, v] \in E$ und Knoten $s, t \in V$ ist analog zur Definition für Knoten

$$\sigma_{s,t}(e) := |\{P \mid P \text{ ist kürzester } s\text{-}t\text{-Weg in } G \ \wedge e \in P\}|$$

als die Anzahl der kürzesten s-t-Wege, die die Kante e benutzen, definiert.

Hierbei setzen sich kürzeste s-t-Wege durch e aus Teilwegen vor und nach e zusammen, wie in Abbildung 11. Im Unterschied zu Eigenschaft 2.17 ist aber noch zu beachten, dass e in zwei verschiedenen Richtungen durchlaufen werden kann. Man erhält:

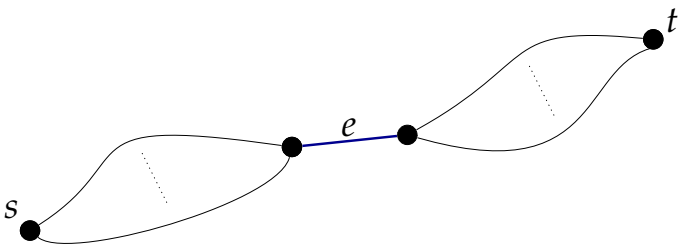

Abbildung 11: Kombination kürzester s-t-Wege durch die Kante e

Eigenschaft 2.24: Für $s, t \in V, e = [u, v] \in E$ gilt:

$$\sigma_{s,t}(e) = \begin{cases} \sigma_{s,u} \cdot \sigma_{v,t} & \text{falls } d(s,t) = d(s,u) + d(v,t) + 1 \\ \sigma_{s,v} \cdot \sigma_{u,t} & \text{falls } d(s,t) = d(s,v) + d(u,t) + 1 \\ 0 & \text{sonst } (e \text{ liegt auf keinem kürzesten } s\text{-}t\text{-Weg}) \end{cases}$$

Der Anteil der kürzesten s-t-Wege, die die Kante e benutzen ist also $\frac{\sigma_{s,t}(e)}{\sigma_{s,t}}$. Durch Betrachtung aller Paare $s, t \in V$ erhalten wir die *Shortest Path Betweenness Centrality von Kanten*, die analog zu der von Knoten definiert ist:

Definition 2.25: Die *Shortest Path Betweenness Centrality der Kante* $e \in E$ ist

$$B(e) := \sum_{s,t \in V | s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}}$$

Wir definieren wieder die Teilsummen:

Definition 2.26: Für $s \in V, e \in E$ ist

$$\delta_{s,\bullet}(e) := \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}}$$

der Einfluss kürzester Wege durch e, die in s beginnen.

Damit erhalten wir:

Eigenschaft 2.27:

$$B(e) = \sum_{s,t \in V | s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}} = \sum_{s \in V} \delta_{s,\bullet}(e)$$

2.5.2 Berechnung der Shortest Path Betweenness Centrality von Kanten

Natürlich lässt sich auch für Kanten mit einem naiven Verfahren, wie in Abschnitt 2.4.2 für Knoten erwähnt, die Shortest Path Betweenness Centrality

aller Kanten auf einmal berechnen. Da hierbei aber für die m Kanten jeweils über n^2 Paare von Knoten summiert wird, liegt die Laufzeit bei $\mathcal{O}(n^2m)$. Aber auch hier geht es schneller.

In [Bra08] wird eine einfache Modifikation von Algorithmus 1 vorgestellt, die auch die Shortest Path Betweenness für alle Kanten auf einmal in $\mathcal{O}(nm)$ berechenbar macht. Dabei wird implizit die Inzidenzgraphkonstruktion aus Abschnitt 2.3.2 ausgenutzt, die sich auch in expliziter Form verwenden lässt, siehe Abschnitt 2.6.3.

Dazu wird einfach im zweiten Schritt, dem „Aufsammeln" der jeweiligen Teilsummen $\delta_{s,\bullet}(v)$, beim Durchlaufen einer Kante e der weitergegebene Wert zur Zentralität der Kante addiert. Brandes folgert dies bei seiner Version nur kurz aus dem Beweis des Satzes zur Berechnung im Grundalgorithmus. Da dieser Algorithmus und der entsprechende Satz hier etwas anders lauten, und um dies nochmals genau zu überprüfen, geben wir den folgenden Satz an:

Satz 2.28: *Sei $G = (V, E)$ gegeben. Für $s, w \in V$ sei*

$$P_s(w) := \{u \in V \mid \exists_P : P = [s, \ldots, u, w] \text{ ist kürzester } s\text{-}w\text{-Weg}\}$$

die Menge der Vorgänger von w auf kürzesten Wegen von s aus.
Für $s \in V$, $e = [v, w] \in E$ gilt

$$\delta_{s,\bullet}(e) = \begin{cases} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta_{s,\bullet}(w) & \text{falls} \quad v \in P_s(w) \\ \frac{\sigma_{s,w}}{\sigma_{s,v}} \cdot \delta_{s,\bullet}(v) & \text{falls} \quad w \in P_s(v) \\ 0 & \text{sonst} \end{cases}$$

Beweis. Schreibe $\delta_{s,t}(e) := \frac{\sigma_{s,t}(e)}{\sigma_{s,t}}$ und damit $\delta_{s,\bullet}(e) = \sum_{t \in V | s \neq t} \delta_{s,t}(e)$. Für $e' \in E$ sei zusätzlich $\sigma_{s,t}(v, e')$ die Anzahl der kürzesten s-t-Wege, die v und e' enthalten, sowie $\delta_{s,t}(v, e') := \frac{\sigma_{s,t}(v,e')}{\sigma_{s,t}}$.

Natürlich kann nur einer der Fälle eintreten. Im dritten Fall verläuft kein kürzester Weg von s aus durch e und die Behauptung stimmt offenbar. Durch Umbenennung können wir also o.E. $v \in P_s(w)$ annehmen. Die Situation ist daher wie in Abbildung 12 dargestellt.

Es gilt wie im Beweis zu Satz 2.21:

$$\sigma_{s,t}(v, [v, w]) = \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \frac{\sigma_{s,t}(w)}{\sigma_{s,t}}$$

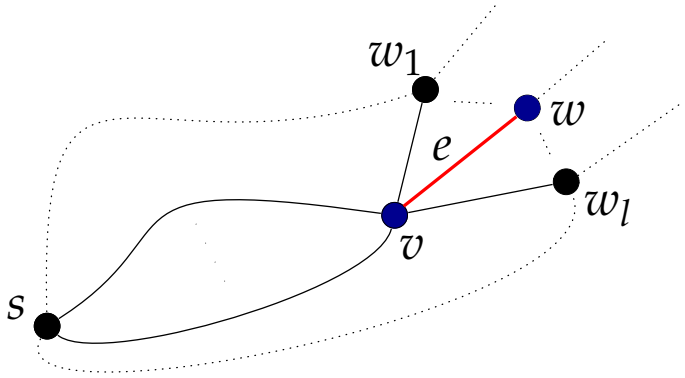

Abbildung 12: Situation für $e = [v, w]$

Damit erhalten wir:

$$\begin{aligned}
\delta_{s,\bullet}(e) &= \delta_{s,\bullet}(v, [v, w]) \\
&= \sum_{t \in V | s \neq t} \delta_{s,t}(v, [v, w]) \\
&= \sum_{t \in V | s \neq t} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \frac{\sigma_{s,t}(w)}{\sigma_{s,t}} \\
&= \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(w)}{\sigma_{s,t}} \\
&= \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta_{s,\bullet}(w)
\end{aligned}$$

□

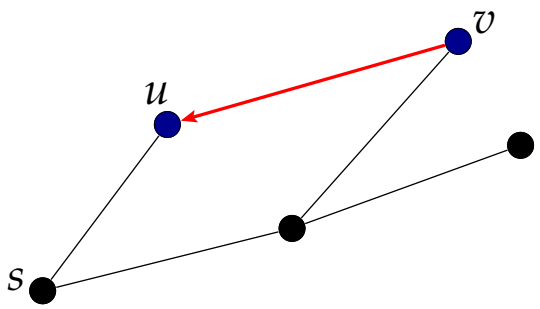

Abbildung 13: $\delta_{s,\bullet}([u, v])$ wird über die Kante $[u, v]$ übertragen

Der Beitrag kürzester Wege durch e, die in s starten, ist also gerade der beim schnellen Algorithmus (Algorithmus 1) für Knoten sowieso über e weitergegebene Wert, wie in Abbildung 13 dargestellt. Wir erhalten also Algorithmus

2 als einfache Modifikation von Algorithmus 1. Der vollständige Algorithmus findet sich wiederum in Anhang A.4.

Algorithmus 2 : Shortest Path Betweenness Centrality für Kanten

Eingabe : $G = (V, E)$
Ausgabe : $\mathrm{B}(e)$ für alle $e \in E$

```
1  ∀_{e∈E} B(e) := 0;
2  foreach s ∈ V do
3      Löse Single-Source-Shortest-Path-Problem für s
4      S ist Stack zum Zugriff auf die Knoten in abnehmender Entfernung
       zu s
5      while S nicht leer do
6          pop w ← S;
7          δ_{s,•}(w) := δ_{s,•}(w) + 1;
8          for v ∈ P_s(w) do
9              c := (σ_{s,v} / σ_{s,w}) · δ_{s,•}(w);
10             δ_{s,•}(v) := δ_{s,•}(v) + c;
11             B([v, w]) := B([v, w]) + c;
12 return B;
```

2.6 Die Shortest Path Betweenness Centrality mit ausgezeichneter Quell- und Zielmenge

Zur vorherigen *Shortest Path Betweenness Centrality* führen wir eine neue Modifikation ein, indem wir zwei Teilmengen der Knotenmenge V auszeichnen: Eine Quellmenge S und eine Zielmenge T. Wir betrachten danach nur noch kürzeste Wege, die von einem Knoten aus S zu einem Knoten aus T verlaufen. Brandes stellt in [Bra08] eine Modifikation von Algorithmus 1 vor, so dass er mit einer ähnlichen Variation der Shortest Path Betweenness Centrality arbeitet, welche aus [FFSN04] stammt. Bei dieser Variante bilden S, T eine Partition von V. Da wir später aber gerade $S = T \subsetneq V$ benötigen werden, ist diese Variante hier ungeeignet. Es ist daher nötig, Algorithmus 1 selbst zu modifizieren.

Durch unsere Modifikation wird simuliert, dass nur manche Knoten Informationen versenden und nur manche sie empfangen, während alle anderen Knoten nur als Mittler fungieren. Eine solche Unterscheidung ist nötig, um

z.B. auch auf dem Inzidenzgraphen korrekt zu arbeiten, vergleiche Abschnitt 2.6.3.

2.6.1 Definition

Sei wieder $G = (V, E)$ gegeben. Ferner seien $S, T \subseteq V$ zwei nichtleere Mengen von Knoten.

Definition 2.29: Die *verallgemeinerte Shortest Path Betweenness Centrality* mit Quellmenge S und Zielmenge T von $v \in V$ ist definiert als

$$\mathrm{B}^{S,T}(v) := \sum_{s \in S} \sum_{t \in T | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

Setze ferner für ein $s \in S$

$$\delta^{T}_{s,\bullet}(v) := \sum_{t \in T | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

Dann ist

$$\mathrm{B}^{S,T}(v) = \sum_{s \in S} \delta^{T}_{s,\bullet}(v)$$

Bemerkung 2.30: Für $S := T := V$ gilt für alle $v \in V$:

$$\mathrm{B}^{V,V}(v) = \sum_{s \in V} \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} = \mathrm{B}(v)$$

Somit ist die gewöhnliche Shortest Path Betweenness Centrality nur ein Spezialfall der verallgemeinerten Shortest Path Betweenness Centrality.

2.6.2 Schnelle Berechnung der verallgemeinerten Shortest Path Betweenness Centrality

Der schnelle Algorithmus aus Abschnitt 2.4.3 soll für die verallgemeinerte Definition angepasst werden.

Ähnlich zu Satz 2.21 lässt sich zeigen:

Satz 2.31: *Sei $s \in S$ und $v \in V$ mit $v \neq s$. Für $w \in V$ sei*

$$P_s(w) := \{u \in V \mid \exists P : P = [s, \ldots, u, w] \text{ ist kürzester } s\text{-}w\text{-Weg}\}$$

die Menge der Vorgänger von w auf kürzesten Wegen von s aus.

Dann gilt:

$$\delta^{T}_{s,\bullet}(v) = \begin{cases} 1 + \sum_{w | v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^{T}_{s,\bullet}(w) & \text{falls} \quad v \in T \\ \sum_{w | v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^{T}_{s,\bullet}(w) & \text{sonst} \end{cases}$$

Beweis. Gibt es keinen Knoten w mit $v \in P_s(w)$, so liegt v nur auf kürzesten s-v-Wegen. Diese sind aber nur relevant, falls $v \in T$ gilt. In diesem Falle erhalten wir:

$$\delta^T_{s,\bullet}(v) = \frac{\sigma_{s,v}(v)}{\sigma_{s,v}} = 1 = 1 + 0 = 1 + \sum_{w \in V | v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^T_{s,\bullet}(w)$$

Andernfalls gilt:

$$\delta^T_{s,\bullet}(v) = 0 = \sum_{w \in V | v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^T_{s,\bullet}(w)$$

Jetzt gebe es mindestens einen Knoten w mit $v \in P_s(w)$. Wir haben also wieder die Situation wie in Abbildung 9 in Abschnitt 2.4.3.

Schreibe $\delta_{s,t}(v) := \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ und damit $\delta^T_{s,\bullet}(v) = \sum_{t \in T | s \neq t} \delta_{s,t}(v)$. Für $e \in E$ sei wieder $\sigma_{s,t}(v,e)$ die Anzahl der kürzesten s-t-Wege, die v und e enthalten, sowie $\delta_{s,t}(v,e) := \frac{\sigma_{s,t}(v,e)}{\sigma_{s,t}}$.

Gilt $t = v$, so müssen wir natürlich kürzeste s-v-Wege beachten. Diese zählen aber nur, falls $v \in T$ gilt. Für $t \neq v$ liegt v nur auf kürzesten s-t-Wegen, wenn t von s aus gesehen „hinter" v liegt. Es muss also außer v noch mindestens eine Kante zu einem Nachfolger w von v (also einem Knoten mit $v \in P_s(w)$) auf einem relevanten s-t-Weg liegen.

Damit gilt:

$$\begin{aligned}
\delta_{s,t}(v) &= \begin{cases} 1 & \text{falls } t = v \text{ und } v \in T \\ 0 & \text{falls } t = v \text{ und } v \notin T \\ \sum_{w | v \in P_s(w)} \delta_{s,t}(v,[v,w]) & \text{sonst} \end{cases} \\
&= \begin{cases} 1 & \text{falls } t = v \text{ und } v \in T \\ \sum_{w | v \in P_s(w)} \delta_{s,t}(v,[v,w]) & \text{sonst} \end{cases}
\end{aligned}$$

Wie im Beweis zu Satz 2.21 gilt dabei

$$\delta_{s,t}(v,[v,w]) = \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \frac{\sigma_{s,t}(w)}{\sigma_{s,t}}$$

und wir erhalten:

$$\begin{aligned}
\sum_{t \in T | s \neq t} \delta_{s,t}(v,[v,w]) &= \sum_{t \in T | s \neq t} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \frac{\sigma_{s,t}(w)}{\sigma_{s,t}} \\
&= \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \sum_{t \in T | s \neq t} \frac{\sigma_{s,t}(w)}{\sigma_{s,t}} \\
&= \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^T_{s,\bullet}(w)
\end{aligned}$$

Damit ergibt sich:

$$\begin{aligned}
\delta^T_{s,\bullet}(v) &= \delta_{s,v}(v) + \sum_{t \in T \setminus \{v\} | s \neq t} \delta_{s,t}(v) \\
&= \delta_{s,v}(v) + \sum_{t \in T \setminus \{v\} | s \neq t} \sum_{w | v \in P_s(w)} \delta_{s,t}(v, [v,w]) \\
&= \delta_{s,v}(v) + \sum_{t \in T | s \neq t} \sum_{w | v \in P_s(w)} \delta_{s,t}(v, [v,w]) \\
&= \delta_{s,v}(v) + \sum_{w | v \in P_s(w)} \sum_{t \in T | s \neq t} \delta_{s,t}(v, [v,w]) \\
&= \delta_{s,v}(v) + \sum_{w | v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^T_{s,\bullet}(w) \\
&= \begin{cases} 1 + \sum_{w | v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^T_{s,\bullet}(w) & \text{falls } v \in T \\ \sum_{w | v \in P_s(w)} \frac{\sigma_{s,v}}{\sigma_{s,w}} \cdot \delta^T_{s,\bullet}(w) & \text{sonst} \end{cases}
\end{aligned}$$

□

Bemerkung 2.32: Es ist $\delta^T_{s,\bullet}(s) = |T \setminus \{s\}|$.

Somit lässt sich Algorithmus 1 einfach anpassen: Das Single-Source-Shortest-Path-Problem wird nur noch für alle Knoten $s \in S$ gelöst und beim Aufsammeln der Werte müssen in der while-Schleife die beiden Fälle unterschieden werden. Zudem muss beachtet werden, dass alle Wege zu Knoten aus $T \setminus \{s\}$ für $\delta_{s,\bullet}(s)$ zählen. Wir erhalten Algorithmus 3 (siehe Algorithmus 14 in Anhang A.5 für eine vollständige Version).

Laufzeit Die Laufzeit des neuen Algorithmus ist $\mathcal{O}(|S|m)$ statt $\mathcal{O}(nm)$, da nur noch $|S|$ viele Single-Source-Shortest-Path-Probleme betrachtet werden.

2.6.3 Anwendung zur Berechnung der Shortest Path Betweenness von Kanten

Zu $G = (V, E)$ ist der *Inzidenz-Graph* $G' := (V', E')$ definiert durch $V' := V \cup E$ und

$$E' := \{[v, e] \mid e \in E \wedge v \in V \wedge e \text{ und } v \text{ sind in } G \text{ inzident}\}$$

wie in Definition 2.12. Setzt man $S := T := V$ und verwendet den Algorithmus für die verallgemeinerte Shortest Path Betweenness Centrality auf G', so liefert er für $e \in E \subseteq V'$ die Werte $\mathrm{B}(e)$ für die Kanten im Originalgraphen,

Algorithmus 3 : verallgemeinerte Shortest Path Betweenness Centrality

Eingabe : $G = (V, E)$ und $S, T \subseteq V$
Ausgabe : $\mathrm{B}^{S,T}(v)$ für alle $v \in V$

```
1  ∀_{v∈V} B^{S,T}(v) := 0;
2  foreach s ∈ S do
3      Löse Single-Source-Shortest-Path-Problem für s
4      S ist Stack zum Zugriff auf die Knoten in abnehmender Entfernung
       zu s
5      while S nicht leer do
6          pop w ← S;
7          if w ∈ T then δ^T_{s,•}(w) := δ^T_{s,•}(w) + 1;
8          for v ∈ P_s(w) do δ^T_{s,•}(v) := δ^T_{s,•}(v) + (σ_{s,v}/σ_{s,w}) · δ^T_{s,•}(w);
9          if w ≠ s then B^{S,T}(w) := B^{S,T}(w) + δ^T_{s,•}(w);
10     B^{S,T}(s) := B^{S,T}(s) + |T \ {s}|;
11 return B^{S,T};
```

denn es gilt:

$$\begin{aligned}
\mathrm{B}^{S,T}(e) &= \sum_{s \in S} \sum_{t \in T | s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}} && \text{(in } G'\text{)} \\
&= \sum_{s \in V} \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}} && \text{(in } G'\text{)} \\
&= \sum_{s \in V} \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}} && \text{(in } G\text{)} \\
&= \mathrm{B}(e) && (\text{ im Graphen } G)
\end{aligned}$$

Die Laufzeit ist dabei $\mathcal{O}(|S| \cdot |E'|) = \mathcal{O}(n \cdot 2m) = \mathcal{O}(nm)$ wie bei der Berechnung der Shortest Path Betweenness Centrality für Kanten mit Algorithmus 2.

2.7 Die Shortest Path Betweenness Centrality für Mengen von Knoten

Aus dem Bestreben, nicht nur für einzelne Knoten, sondern für ganze Teilmengen (Gruppen) von Knoten $C \subseteq V$ ein Maß für die Zentralität zu erhalten, lässt sich die Definition der Shortest Path Betweenness Centrality ver-

allgemeinern, wodurch man die *Group Betweenness Centrality* erhält, die im nächsten Kapitel betrachtet werden soll.

3 Die Group Betweenness Centrality

3.1 Die Group Betweenness Centrality für Gruppen von Knoten

Durch Erweiterung der Definition der Shortest Path Betweenness Centrality auf ganze Gruppen von Knoten erhält man die *Group Betweenness Centrality*. Dabei wird für eine ganze Menge von Knoten der Anteil der kürzesten Wege zwischen Paaren von Knoten gezählt, die mindestens einen Knoten der Menge benutzen. Die Group Betweenness Centrality wird in [EB99] eingeführt. Wir verwenden sie, wie sie in [PED07a] und [PED07b] definiert wird.

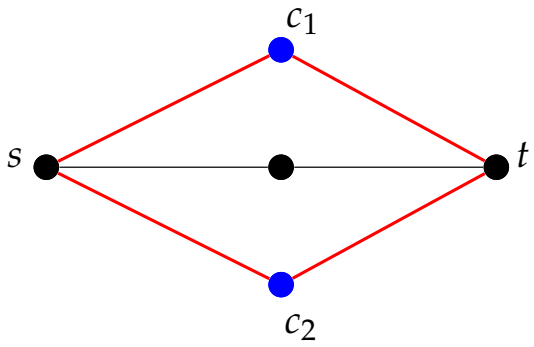

Abbildung 14: $C = \{c_1, c_2\}$ überdeckt zwei s-t-Wege

3.1.1 Definition

Sei $G = (V, E)$ gegeben. Wir wollen analog zur Shortest Path Betweenness jetzt auch für eine ganze Menge von Knoten messen, wie viele kürzeste s-t-Wege diese überdeckt.

Definition 3.1: Für eine Teilmenge $C \subseteq V$ von Knoten sei

$$\ddot{\sigma}_{s,t}(C) := |\{P \mid P \text{ ist kürzester } s\text{-}t\text{-Weg in } G \wedge \exists_{v \in C} v \in P\}|$$

die Anzahl kürzester s-t-Wege, die mindestens einen Knoten aus C benutzen.

Um die Group Betweenness Centrality zu erhalten, ist es jetzt nur noch nötig, über alle Paare $s, t \in V$ zu summieren.

Definition 3.2 (Group Betweenness Centrality)**:** Die *Group Betweenness Centrality* von $C \subseteq V$ ist

$$\ddot{\mathrm{B}}(C) := \sum_{s,t \in V | s \neq t} \frac{\ddot{\sigma}_{s,t}(C)}{\sigma_{s,t}}$$

Bemerkung 3.3: Für $v \in V$ ist $\sigma_{s,t}(v) = \ddot{\sigma}_{s,t}(\{v\})$ und damit $\mathrm{B}(v) = \ddot{\mathrm{B}}(\{v\})$. Also stellt die Group Betweenness Centrality wie gewünscht eine Verallgemeinerung der Shortest Path Betweenness Centrality dar.

3.1.2 Path Betweenness Centrality

Als Hilfsmittel zur Berechnung der Shortest Path Betweenness wird uns die Path Betweenness Centrality dienen. Diese verallgemeinert die Shortest Path Betweenness Centrality nicht auf Mengen, sondern auf geordnete Listen von Knoten.

Definition 3.4: Sei $S = (v_1, \ldots, v_k)$ eine geordnete Liste von Knoten. Dann bezeichne

$$\tilde{\sigma}_{s,t}(S)$$

die Anzahl der kürzesten s-t-Wege, die die Knoten aus S in der gegebenen Reihenfolge (evtl. mit weiteren Zwischenknoten) durchlaufen.

Abbildung 15: Von $S = (v_1, v_2, v_3)$ überdeckter s-t-Weg

Wiederum müssen wir nur noch Wege zwischen allen Paaren von Knoten berücksichtigen, um ein Maß für die Zentralität von Listen von Knoten zu erhalten.

Definition 3.5 (Path Betweenness Centrality)**:** Die *Path Betweenness Centrality* einer Liste S von Knoten ist definiert als:

$$\tilde{\mathrm{B}}(S) := \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}_{s,t}(S)}{\sigma_{s,t}}$$

Wir schreiben dabei für $S = (v_1, \ldots, v_k)$ auch

$$\tilde{\sigma}_{s,t}(v_1, \ldots, v_k) = \tilde{\sigma}_{s,t}(S) \text{ und } \tilde{B}(v_1, \ldots, v_k) = \tilde{B}(S).$$

Bemerkung 3.6: Für $v \in V$ gilt natürlich

$$\tilde{B}(v) = B(v) = \ddot{B}(\{v\})$$

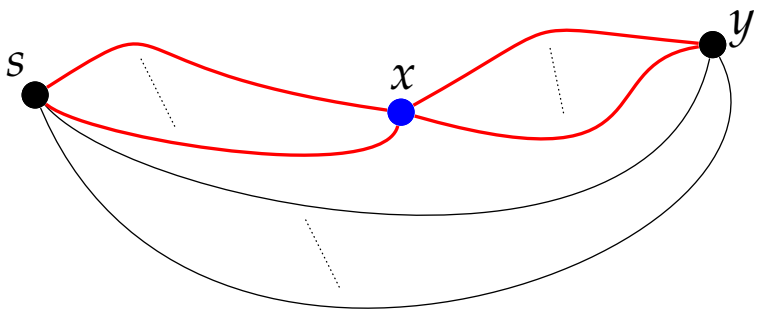

Abbildung 16: s-y-Wege durch x

Für den Sonderfall einer zweielementigen Liste $S = (x, y)$ lässt sich vereinfachen: Für ein festes $s \in V$ läuft ein Anteil von $\frac{\sigma_{s,y}(x)}{\sigma_{s,y}}$ der kürzesten s-y-Wege durch x, vergleiche Abbildung 16. Da für ein $t \in V$ ein Anteil $\frac{\sigma_{s,t}(y)}{\sigma_{s,t}}$ der kürzesten s-t-Wege durch v verläuft, gilt:

$$\frac{\tilde{\sigma}_{s,t}(x,y)}{\sigma_{s,t}} = \frac{\sigma_{s,y}(x)}{\sigma_{s,y}} \cdot \frac{\sigma_{s,t}(y)}{\sigma_{s,t}}$$

Man erhält:

Eigenschaft 3.7:

$$\begin{aligned}
\tilde{B}(x,y) &= \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}_{s,t}(x,y)}{\sigma_{s,t}} \\
&= \sum_{s,t \in V | s \neq t} \frac{\sigma_{s,y}(x)}{\sigma_{s,y}} \cdot \frac{\sigma_{s,t}(y)}{\sigma_{s,t}} \\
&= \sum_{s \in V} \frac{\sigma_{s,y}(x)}{\sigma_{s,y}} \cdot \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(y)}{\sigma_{s,t}} \\
&= \sum_{s \in V} \frac{\sigma_{s,y}(x)}{\sigma_{s,y}} \cdot \delta_{s,\bullet}(y)
\end{aligned}$$

3.2 Berechnung der Group Betweenness Centrality

In [Bra08] wird eine Modifikationsmöglichkeit für Algorithmus 1 vorgestellt, durch die bei einem Durchlauf des Algorithmus die Group Betweenness einer vorgegebenen Gruppe mitberechnet wird. Es wird also auf diese Weise $\mathcal{O}(nm)$ benötigt, um die Group Betweenness Centrality von nur einer Gruppe zu berechnen. Da es sehr viele solcher Gruppen gibt, ist eine schnellere Berechnung erwünscht.

Ein Algorithmus zur schnellen Berechnung der Group Betweenness für eine Gruppe von Knoten wird in [PED07a] und [PED07b] vorgestellt. Dieser Algorithmus soll in diesem Abschnitt vorgestellt werden. Er benötigt einen einmaligen Preprocessing-Schritt mit höherem Aufwand. Anschließend kann die Group Betweenness Centrality für jede Knotengruppe schnell berechnet werden.

3.2.1 Preprocessing

Es werden drei $n \times n$-Matrizen berechnet:

1. d speichert die Distanzen $d(s,t)$.
2. σ speichert die Anzahlen $\sigma_{s,t}$ der kürzesten s-t-Wege.
3. $\tilde{\mathrm{B}}$ speichert die Path Betweenness $\tilde{\mathrm{B}}(x,y)$ für alle Paare $x, y \in V$.

Dabei können d und σ in $\mathcal{O}(nm)$ berechnet werden, indem für jedes $s \in V$ ein Single-Source-Shortest-Path-Problem gelöst wird. Für alle s, y auf einmal kann außerdem auch $\delta_{s,\bullet}(y)$ in $\mathcal{O}(nm)$ ermittelt werden. Hierzu müssen nur beim Durchlauf von Algorithmus 1 die entsprechenden Werte dauerhaft gespeichert werden. Jeder der n^2 Einträge von $\tilde{\mathrm{B}}$ wird anschließend mittels Eigenschaft 3.7 (und Eigenschaft 2.17) in $\mathcal{O}(n)$ berechnet und man erhält eine Gesamtlaufzeit von $\mathcal{O}(n^3)$ für das Preprocessing.

3.2.2 Eigentliche Berechnung

Die Idee zur Berechnung von $\ddot{\mathrm{B}}(C)$ für eine k-elementige Teilmenge $C \subseteq V$ ist es, mit $M := \emptyset$ zu starten und dann schrittweise die Elemente von C zu M hinzuzunehmen, bis $M = C$ gilt. Dabei werden drei Datenstrukturen verwaltet und in jedem Schritt aktualisiert, mit deren Hilfe sich die Zunahme der Group Betweenness bei Hinzunahme eines neuen Knotens zu M ermitteln lässt. Wir brauchen zur Erklärung noch zusätzliche Definitionen.

Definition 3.8: Seien $M \subseteq V$ und $s, t, v \in V$, sowie S eine Liste von Knoten aus V. Es sei

$\sigma_{s,t}^{M}$ die Anzahl der kürzesten s-t-Wege, die keinen Knoten aus M berühren

$\sigma_{s,t}^{M}(v)$ die Anzahl der kürzesten s-t-Wege durch v, die keinen Knoten aus M berühren

$\tilde{\sigma}_{s,t}^{M}(S)$ die Anzahl der kürzesten s-t-Wege, die die Knoten aus S in dieser Reihenfolge durchlaufen und keinen Knoten aus M berühren

$\tilde{\mathrm{B}}^{M}(S) := \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}_{s,t}^{M}(S)}{\sigma_{s,t}}$ berücksichtigt dann nur die noch nicht überdeckten kürzesten Wege für die Path Betweenness.

Dies folgenden Datenstrukturen werden im Algorithmus verwaltet:

1. $\ddot{\mathrm{B}}(M)$: Die aktuelle Group Betweenness Centrality von M. Der Anfangswert ist $\ddot{\mathrm{B}}(\emptyset) = 0$.

2. σ^{M}: Eine $k \times k$-Matrix. Für jedes Paar von Knoten $s, t \in C$ wird die Anzahl $\sigma_{s,t}^{M}$ der kürzesten s-t-Wege gespeichert, die keinen Knoten aus M berühren. Die Eingangswerte $\sigma_{s,t}^{\emptyset}$ entsprechen den Werten $\sigma_{s,t}$ aus dem Preprocessing-Schritt.

3. $\tilde{\mathrm{B}}^{M}$: Eine zweite $k \times k$-Matrix. In dieser wird für jedes Paar $x, y \in C$ die Path Betweenness für (x, y) gespeichert, wobei nur der Beitrag von kürzesten s-t-Wegen durch (x, y) gezählt wird, die keinen Knoten aus M berühren. Die Anfangswerte sind wegen $\tilde{\mathrm{B}}^{\emptyset}(x, y) = \tilde{\mathrm{B}}(x, y)$ wieder die vorberechneten Werte.

Diese Datenstrukturen lassen sich bei jeder Hinzunahme zu M in $\mathcal{O}(k^2)$ aktualisieren, wie wir sehen werden.

Update von $\ddot{\mathrm{B}}(M)$ Es gilt für $x \in V$:

$$\tilde{\mathrm{B}}^{M}(x, x) = \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}_{s,t}^{M}(x, x)}{\sigma_{s,t}} = \sum_{s,t \in V | s \neq t} \frac{\sigma_{s,t}^{M}(x)}{\sigma_{s,t}}$$

Daher gilt für jedes $v \in C$, dass $\tilde{\mathrm{B}}^{M}(v, v)$ durch Hinzunahme von v zu M für die Group Betweenness gewonnen werden kann, also $\ddot{\mathrm{B}}(M \cup \{v\}) = \ddot{\mathrm{B}}(M) + \tilde{\mathrm{B}}^{M}(v, v)$.

Update von σ^M Bei Hinzunahme von $v \in C$ zu M werden die kürzesten Wege durch v von $M \cup \{v\}$ überdeckt. Deswegen gilt:

Eigenschaft 3.9:

$$\sigma_{x,y}^{M\cup\{v\}} = \sigma_{x,y}^{M} - \sigma_{x,y}^{M}(v)$$

Durch Nichtbeachtung kürzester Wege, die Knoten aus M benutzen, erhält man analog zu Eigenschaft 2.17:

Eigenschaft 3.10:

$$\sigma_{x,y}^{M}(v) = \begin{cases} \sigma_{x,v}^{M} \cdot \sigma_{v,y}^{M} & \text{falls } d(x,y) = d(x,v) + d(v,y) \\ 0 & \text{sonst} \end{cases}$$

Somit lässt sich jeder neue Wert $\sigma_{x,y}^{M\cup\{v\}}$ in $\mathcal{O}(1)$ berechnen, wenn die Werte aus σ^M vorher korrekt waren.

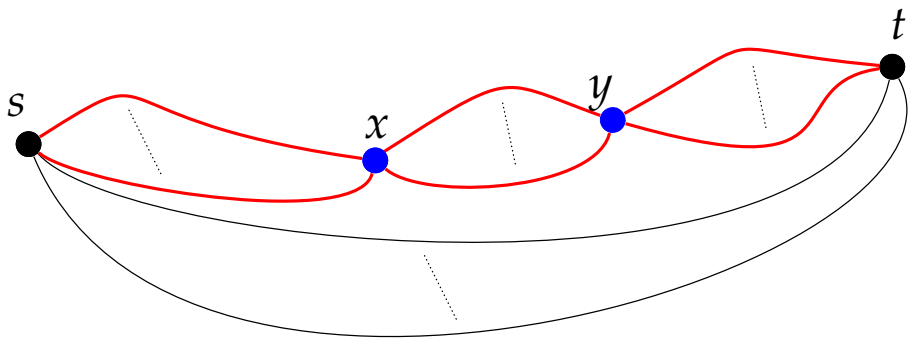

Abbildung 17: kürzeste s-t-Wege durch (x, y)

Update von $\tilde{B}^M$ Seien x, y zwei Knoten, die in dieser Reihenfolge auf einem kürzesten s-t-Weg liegen. Dann sieht man durch abschnittsweise Betrachtung wie in Abbildung 17, dass

$$\tilde{\sigma}_{s,t}^{M}(x,y) = \sigma_{s,x}^{M} \cdot \sigma_{x,y}^{M} \cdot \sigma_{y,t}^{M}$$

gilt. Liegt noch v auf einem kürzesten x-y-Weg, so gilt weiter:

$$\begin{aligned} \tilde{\sigma}_{s,t}^{M}(x,v,y) &= \sigma_{s,x}^{M} \cdot \sigma_{x,v}^{M} \cdot \sigma_{v,y}^{M} \cdot \sigma_{y,t}^{M} = \sigma_{s,x}^{M} \cdot \sigma_{x,v}^{M} \cdot \sigma_{v,y}^{M} \cdot \sigma_{y,t}^{M} \cdot \frac{\sigma_{x,y}^{M}}{\sigma_{x,y}^{M}} \\ &= \sigma_{s,x}^{M} \cdot \sigma_{x,y}^{M} \cdot \sigma_{y,t}^{M} \cdot \frac{\sigma_{x,v}^{M} \cdot \sigma_{v,y}^{M}}{\sigma_{x,y}^{M}} \\ &= \tilde{\sigma}_{s,t}^{M}(x,y) \cdot \frac{\sigma_{x,y}^{M}(v)}{\sigma_{x,y}^{M}} \end{aligned}$$

Damit erhält man:

Eigenschaft 3.11: Liegt v auf einem kürzesten x-y-Weg, so gilt:

$$\begin{aligned}\tilde{\mathrm{B}}^M(x,v,y) &= \sum_{s,t\in V} \frac{\tilde{\sigma}^M_{s,t}(x,v,y)}{\sigma^M_{s,t}} = \frac{\sigma^M_{x,y}(v)}{\sigma^M_{x,y}} \cdot \sum_{s,t\in V} \frac{\tilde{\sigma}^M_{s,t}(x,y)}{\sigma^M_{s,t}} \\ &= \frac{\sigma^M_{x,y}(v)}{\sigma^M_{x,y}} \cdot \tilde{\mathrm{B}}^M(x,y)\end{aligned}$$

Für $x, y \in V$ definiere das Hinzufügen von $z \in V$ zum geordneten Paar (x,y) in sinnvoller Reihenfolge, falls es einen kürzesten Weg gibt, auf dem alle drei Knoten liegen:

Definition 3.12:

$$(x,y) \circ z := \begin{cases} (z,x,y) & \text{falls} \quad d(z,y) = d(z,x) + d(x,y) \\ (x,z,y) & \text{falls} \quad d(x,y) = d(x,z) + d(z,y) \\ (x,y,z) & \text{falls} \quad d(x,z) = d(x,y) + d(y,z) \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

Bemerkung 3.13: Falls mehr als eine der Bedingungen erfüllt ist, stimmen mindestens zwei der drei Knoten überein.

Wenn es keinen gemeinsamen kürzesten Weg gibt, also $(x,y) \circ z$ undefiniert ist, sei $\tilde{\mathrm{B}}^M((x,y) \circ z) = 0$.

Es sollen jetzt alle neuen Werte $\tilde{\mathrm{B}}^{M\cup\{v\}}(x,y)$ berechnet werden. Dazu müssen wir überlegen, welche von M nicht überdeckten kürzesten Wege durch (x,y) von $M \cup \{v\}$ überdeckt werden. Zuerst gilt:

Eigenschaft 3.14: Seien $x, y, v \in C$. Im Fall $x = y \neq v$ gilt

$$\tilde{\mathrm{B}}^{M\cup\{v\}}(x,x) = \tilde{\mathrm{B}}^M(x,x) - \tilde{\mathrm{B}}^M(x,v) - \tilde{\mathrm{B}}^M(v,x)$$

da v sowohl vor als auch nach x durchlaufen werden kann.

In den anderen Fällen muss, unter Berücksichtigung der Reihenfolge, der Anteil der kürzesten Wege, die (x,y) durchlaufen und auch durch v gehen, abgezogen werden, also:

Eigenschaft 3.15:

$$\tilde{\mathrm{B}}^{M\cup\{v\}}(x,y) = \tilde{\mathrm{B}}^M(x,y) - \tilde{\mathrm{B}}^M((x,y) \circ v)$$

Hierin ist auch der Spezialfall $x = y = v$ enthalten.

Die Werte $\sigma^M_{x,y}$ sind ebenso wie die Werte $\tilde{\mathrm{B}}^M(x,y)$ bekannt. Die Berechnung von $\tilde{\mathrm{B}}^M((x,y) \circ v)$ ist somit mit Eigenschaft 3.11 in $\mathcal{O}(1)$ möglich. Damit kann auch jeder neue Wert $\tilde{\mathrm{B}}^{M\cup\{v\}}(x,y)$ in $\mathcal{O}(1)$ berechnet werden.

3.2.3 Algorithmus

Aus dem beschriebenen Vorgehen mit Initialisierung und schrittweisem Update der Werte erhält man Algorithmus 4.

Algorithmus 4 : Group Betweenness Centrality

Eingabe : $G = (V, E), C \subseteq V$ und die vorberechneten Werte $d, \sigma, \tilde{\mathrm{B}}$
Ausgabe : $\ddot{\mathrm{B}}(C)$

```
1  M := ∅;
2  ∀_{x,y∈C} σ^M_{x,y} := σ_{x,y};
3  ∀_{x,y∈C} B̃^M(x,y) := B̃(x,y);
4  B̈ := 0;
5  foreach v ∈ C do
6      B̈ := B̈ + B̃^M(v,v);
7      foreach x, y ∈ C do
8          σ^{M∪{v}}_{x,y} := σ^M_{x,y} − σ^M_{x,y}(v);
9          if x = y ≠ v then
10             B̃^{M∪{v}}(x,x) := B̃^M(x,x) − B̃^M(v,x) − B̃^M(x,v);
11         else
12             B̃^{M∪{v}}(x,y) := B̃^M(x,y) − B̃^M((x,y) ∘ v);
13     M := M ∪ {v};
14 return B̈;
```

Laufzeit Ohne das Preprocessing ($\mathcal{O}(n^3)$) zu zählen, wird die äußere Schleife k mal durchlaufen. Die innere Schleife zum Update der vorgehaltenen Werte wird k^2 mal durchlaufen, wobei ein einzelnes Update jeweils in konstanter Zeit möglich ist, wie gezeigt wurde. Damit läuft der Algorithmus ohne den Preprocessingschritt in $\mathcal{O}(k^3)$.

3.3 Die Group Betweenness Centrality für Gruppen von Kanten

Weder Everett und Borgatti in [EB99] noch Puzis et. al. in [PED07b] und in [PED07a] definieren und untersuchen die Group Betweenness Centrality von Gruppen von Kanten. Dennoch stellt diese eine logische Erweiterung der

Shortest Path Betweenness Centrality von Kanten dar, analog zur Group Betweenness Centrality von Gruppen von Knoten.

Wir wollen daher die Definition der Group Betweenness Centrality entsprechend anpassen. Außerdem werden wir sehen, dass es dann auch möglich ist, den Algorithmus zur schnellen Berechnung entsprechend anzupassen, so dass er auch die Group Betweenness von Gruppen von Kanten berechnen kann.

3.3.1 Definition

Definition 3.16: Auch für eine Teilmenge $C \subseteq E$ der Kanten sei

$$\ddot{\sigma}_{s,t}(C) := |\{P \mid P \text{ ist kürzester } s\text{-}t\text{-Weg in } G \ \wedge \exists_{e \in C} e \in P\}|$$

die Anzahl kürzester s-t-Wege, die mindestens eine Kante der Menge benutzen.

Es ergibt sich wieder völlig analog die *Group Betweenness Centrality* von Gruppen von Kanten:

Definition 3.17 (Group Betweenness Centrality für Gruppen von Kanten)**:** Für eine Teilmenge $C \subseteq E$ der Kanten ist

$$\ddot{\mathrm{B}}(C) := \sum_{s,t \in V | s \neq t} \frac{\ddot{\sigma}_{s,t}(C)}{\sigma_{s,t}}$$

die *Group Betweenness Centrality*.

Path Betweenness Centrality

Die Path Betweenness Centrality muss man für Kanten nicht neu definieren. Werden die beiden Endknoten einer Kante in dieser Reihenfolge auf einem kürzesten Weg durchlaufen, so sind sie auch direkt hintereinander, da sich sonst durch die Kante eine Abkürzung ergäbe.

3.3.2 Berechnung

Wir wollen überlegen, wie wir die Group Betweenness Centrality von Gruppen von Kanten berechnen können. Ein Ansatzpunkt hierfür ist die Inzidenzgraphkonstruktion aus Abschnitt 2.3.2, mit deren Hilfe man weiterhin Algorithmus 4 nutzen könnte. Durch Verwendung des Algorithmus zur Berechnung der verallgemeinerten Shortest Path Betweenness ließen sich die im Preprocessingschritt berechneten Werte $d, \sigma, \tilde{\mathrm{B}}$ auch für die neuen Knoten im

Inzidenzgraphen berechnen. In der anschließenden Berechnung der Group Betweenness würde sich nichts ändern, da nur noch mit den korrekt vorberechneten Werten weitergerechnet wird.

Leider wird das Preprocessing langsamer. Es müssen Distanzen und Anzahl der kürzesten Wege zwischen allen m^2 Paaren von „Knoten“ $e \in E$ im Inzidenzgraphen berechnet werden, was in $\mathcal{O}(|E| \cdot |E'|) = \mathcal{O}(m \cdot 2m) = \mathcal{O}(m^2)$ möglich ist. Außerdem muss m^2 mal die Path Betweenness zwischen zwei neuen Knoten $e \in E$ berechnet werden, wozu es nötig ist, jeweils n Werte zu summieren. Insgesamt weist also das Preprocessing nun eine erhöhte Laufzeit von $\mathcal{O}(nm^2)$ gegenüber der Knotenversion mit $\mathcal{O}(n^3)$ auf.

Zudem wird auch der resultierende Greedy-Algorithmus (vergleiche den Abschnitt 4.1.1) langsamer. Der Algorithmus muss statt zwei $n \times n$-Matrizen zwei $m \times m$-Matrizen mit Daten zu allen Paaren von Kanten verwalten. Dadurch verschlechtert sich die Laufzeit ohne das vorherige Preprocessing auf $\mathcal{O}(km^2)$ statt $\mathcal{O}(kn^2)$.

Durch diese verschlechterte Laufzeit ergibt sich die Fragestellung, ob nicht durch ein nur implizites Nutzen der Inzidenzgraphkonstruktion, ähnlich wie in Abschnitt 2.5.2, die Laufzeit wieder auf die gleiche Klasse wie im Falle von Knotengruppen gebracht werden kann.

3.4 Schnelle Berechnung der Group Betweenness Centrality für Gruppen von Kanten

In diesem Abschnitt soll Algorithmus 4 modifiziert werden, so dass er die Group Betweenness Centrality von Gruppen von Kanten berechnet. Dabei werden wir die Grundstruktur des Algorithmus einschließlich der verwendeten Datenstrukturen beibehalten. Wir werden sehen, dass man mit Hilfe der Path Betweenness Centrality den jeweiligen Gewinn für die bisherige Group Betweenness ermitteln kann, wenn die vorherigen Werte korrekt waren. Dann müssen wir nur noch das Update der Werte bei Hinzunahme einer Kante zur bisherigen Gruppe lösen.

3.4.1 Preprocessing

Am Preprocessing muss nichts geändert werden, da keine anderen Daten benötigt werden als im Falle der Group Betweenness Centrality für Gruppen von Knoten.

3.4.2 Berechnung

Das Vorgehen bleibt hier im Wesentlichen dasselbe wie bei der Berechnung der Group Betweenness Centrality für Knoten:

Eine anfangs leere Menge $M \subseteq E$ wird durch schrittweise Hinzunahme von Kanten aus $C \subseteq E$ an die k-elementige Menge C angenähert, wobei die vorgehaltenen Daten aktualisiert werden. Diese sind praktisch dieselben wie im Algorithmus für Knoten. Dazu sei

$$C' := \{v \in V \mid \exists_{e \in C} e \text{ und } v \text{ sind inzident } \}$$

und $k' := |C'|$.

Analog zu Definition 3.8 definieren wir für Mengen von Kanten:

Definition 3.18: Seien $M \subseteq E$ und $s, t, v \in V, e \in E$ und S eine Liste von Knoten aus V. Es sei

$\boldsymbol{\sigma_{s,t}^{M}}$ die Anzahl der kürzesten s-t-Wege, die keine Kante aus M benutzen

$\boldsymbol{\sigma_{s,t}^{M}(v)}$ die Anzahl der kürzesten s-t-Wege durch v, die keine Kante aus M benutzen

$\boldsymbol{\sigma_{s,t}^{M}(e)}$ die Anzahl der kürzesten s-t-Wege durch e, die keine Kante aus M benutzen

$\boldsymbol{\tilde{\sigma}_{s,t}^{M}(S)}$ die Anzahl der kürzesten s-t-Wege, die die Knoten aus S in dieser Reihenfolge durchlaufen und keine Kante aus M benutzen

$\tilde{\mathrm{B}}^{M}(S) := \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}_{s,t}^{M}(S)}{\sigma_{s,t}}$ berücksichtigt dann nur die von M noch nicht überdeckten kürzesten Wege für die Path Betweenness.

Es werden wiederum drei Datenstrukturen verwaltet:

1. $\ddot{\mathrm{B}}(M)$: Die aktuelle Group Betweenness Centrality. Der Anfangswert ist $\ddot{\mathrm{B}}(\emptyset) = 0$.
2. σ^{M}: Eine $k' \times k'$-Matrix. Für jedes Paar $s, t \in C'$ wird die Anzahl $\sigma_{s,t}^{M}$ der kürzesten s-t-Wege gespeichert, die keine Kante aus M berühren, vergleiche Definition 3.18. Die Eingangswerte $\sigma_{s,t}^{\emptyset}$ entsprechen den Werten $\sigma_{s,t}$ aus dem Preprocessing-Schritt.
3. $\tilde{\mathrm{B}}^{M}$: Eine zweite $k' \times k'$-Matrix. In dieser wird für jedes Paar $x, y \in C'$ die Path Betweenness für (x, y) gespeichert, wobei nur der Beitrag von kürzesten s-t-Wegen durch (x, y) gezählt wird, die keine Kante aus M berühren, wie in Definition 3.18. Die Anfangswerte sind wegen $\tilde{\mathrm{B}}^{\emptyset}(x, y) = \tilde{\mathrm{B}}(x, y)$ die vom Preprocessing vorberechneten Werte.

Wir werden sehen, dass sich auch hier die Datenstrukturen in $\mathcal{O}(k'^2) = \mathcal{O}(k^2)$ aktualisieren lassen.

Update von $\ddot{\mathrm{B}}(M)$ Da eine Kante $e = [u, v]$ in beide Richtungen durchlaufen werden kann, lässt sich durch ihre Hinzunahme zu M insgesamt $\tilde{\mathrm{B}}^M(u, v) + \tilde{\mathrm{B}}^M(v, u)$ für die Group Betweenness Centrality gewinnen. Es gilt also:

$$\ddot{\mathrm{B}}(M \cup \{e\}) = \ddot{\mathrm{B}}(M) + \tilde{\mathrm{B}}^M(u, v) + \tilde{\mathrm{B}}^M(v, u)$$

Update von σ^M Bei Hinzunahme von $e \in C$ werden die kürzesten Wege durch diese Kante abgedeckt. Also gilt:

Eigenschaft 3.19:

$$\sigma_{x,y}^{M \cup \{e\}} = \sigma_{x,y}^M - \sigma_{x,y}^M(e)$$

Analog zu Eigenschaft 2.24 gilt dabei:

Eigenschaft 3.20:

$$\sigma_{s,t}^M(e) = \begin{cases} \sigma_{s,u}^M \cdot \sigma_{v,t}^M & \text{falls } d(s,t) = d(s,u) + d(v,t) + 1 \\ \sigma_{s,v}^M \cdot \sigma_{u,t}^M & \text{falls } d(s,t) = d(s,v) + d(u,t) + 1 \\ 0 & \text{sonst} \end{cases}$$

Damit ist das Update von $\sigma_{x,y}^M$ zu $\sigma_{x,y}^{M \cup \{v\}}$ in $\mathcal{O}(1)$ möglich.

Update von $\tilde{\mathrm{B}}^M$ Da für $e = [u, v]$ zwei Knoten beteiligt sind, gibt es mehr Kombinationen, in der diese auf irgendeinem kürzesten Weg zu der Knotenreihenfolge (x, y) hinzugefügt werden können. Sei also analog zu Definition 3.12:

Definition 3.21:

$$(x,y) \circ e := \begin{cases} (u,v,x,y) & \text{falls} \quad d(u,y) = d(v,x) + d(x,y) + 1 \\ (v,u,x,y) & \text{falls} \quad d(v,y) = d(u,x) + d(x,y) + 1 \\ (x,u,v,y) & \text{falls} \quad d(x,y) = d(x,u) + d(v,y) + 1 \\ (x,v,u,y) & \text{falls} \quad d(x,y) = d(x,v) + d(u,y) + 1 \\ (x,y,u,v) & \text{falls} \quad d(x,v) = d(x,y) + d(y,u) + 1 \\ (x,y,v,u) & \text{falls} \quad d(x,u) = d(x,y) + d(y,v) + 1 \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

Bemerkung 3.22: Mehr als eine Bedingung kann nur erfüllt sein, wenn mindestens zwei der Knoten (und höchstens drei, da $u \neq v$) übereinstimmen.

Falls es keinen kürzesten Weg gibt, auf dem alle vier Knoten liegen, so ist wieder $(x, y) \circ e$ undefiniert mit $\tilde{\mathrm{B}}^M((x, y) \circ e) = 0$.

Seien $x, y \in C'$ und $e = [u, v] \in C \setminus M$ die Kante, die zu M hinzugenommen werden soll. Wir betrachten mehrere Fälle für das Update von $\tilde{\mathrm{B}}^M(x, y)$ zu $\tilde{\mathrm{B}}^{M \cup \{e\}}(x, y)$.

- Ist $x = y = u$, so gilt

  $$\tilde{\mathrm{B}}^{M \cup \{e\}}(u, u) = \tilde{\mathrm{B}}^M(u, u) - \tilde{\mathrm{B}}^M(u, v) - \tilde{\mathrm{B}}^M(v, u)$$

  da Wege durch e wegfallen. Analog gilt für $x = y = v$:

  $$\tilde{\mathrm{B}}^{M \cup \{e\}}(v, v) = \tilde{\mathrm{B}}^M(v, v) - \tilde{\mathrm{B}}^M(u, v) - \tilde{\mathrm{B}}^M(v, u)$$

- Ist $x = y$, aber $u \neq x \neq v$, so ist zu unterscheiden:
  - Gibt es keinen kürzesten Weg, auf dem sowohl x, als auch e liegen, d.h. gilt $d(x, u) = d(x, v)$, so ist

    $$\tilde{\mathrm{B}}^{M \cup \{e\}}(x, x) = \tilde{\mathrm{B}}^M(x, x)$$

  - Gibt es einen gemeinsamen kürzesten Weg, so ist

    $$\tilde{\mathrm{B}}^{M \cup \{e\}}(x, x) = \tilde{\mathrm{B}}^M(x, x) - \tilde{\mathrm{B}}^M(u, v, x) - \tilde{\mathrm{B}}^M(x, v, u)$$

    falls $d(x, v) < d(x, u)$. Gilt dagegen $d(x, u) < d(x, v)$, so ist

    $$\tilde{\mathrm{B}}^{M \cup \{e\}}(x, x) = \tilde{\mathrm{B}}^M(x, x) - \tilde{\mathrm{B}}^M(v, u, x) - \tilde{\mathrm{B}}^M(x, u, v)$$

    In beiden Fällen wird Eigenschaft 3.11 zur Berechnung benötigt, um jeweils $\tilde{\mathrm{B}}^{M \cup \{e\}}(x, x)$ in $\mathcal{O}(1)$ zu berechnen.

- Es bleibt nur $x \neq y$.
  - Sind höchstens drei der Knoten verschieden, d.h. $\{x, y\} \cap \{u, v\} \neq \emptyset$, so liefert $(x, y) \circ e$, falls definiert, eine eindeutige Reihenfolge von zwei oder drei Knoten. Es gilt dann:

    $$\tilde{\mathrm{B}}^{M \cup \{e\}}(x, y) = \tilde{\mathrm{B}}^M(x, y) - \tilde{\mathrm{B}}^M((x, y) \circ e)$$

    Dies kann entweder direkt berechnet werden oder, falls es drei verschiedene Knoten gibt, unter Verwendung von Eigenschaft 3.11 in konstanter Zeit.

– Es bleibt der Fall, dass alle vier Knoten verschieden sind. Falls definiert, erhält man eine eindeutige Reihenfolge

$$(x, y) \circ e = (v_1, v_2, v_3, v_4)$$

von vier verschiedenen Knoten. Analog zur Herleitung von Eigenschaft 3.11 erhält man durch abschnittsweise Betrachtung:

$$\begin{aligned}\tilde{\sigma}^M_{s,t}(v_1, v_2, v_3, v_4) &= \sigma^M_{s,v_1} \cdot \sigma^M_{v_1,v_2} \cdot \sigma^M_{v_2,v_3} \cdot \sigma^M_{v_3,v_4} \cdot \sigma^M_{v_4,t} \cdot \left(\frac{\sigma^M_{v_1,v_3}}{\sigma^M_{v_1,v_3}}\right) \\ &= \frac{\sigma^M_{v_1,v_3}(v_2)}{\sigma^M_{v_1,v_3}} \cdot \tilde{\sigma}^M_{s,t}(v_1, v_3, v_4) \\ &\overset{(3.11)}{=} \frac{\sigma^M_{v_1,v_3}(v_2)}{\sigma^M_{v_1,v_3}} \cdot \frac{\sigma^M_{v_1,v_4}(v_3)}{\sigma^M_{v_1,v_4}} \cdot \tilde{\sigma}^M_{s,t}(v_1, v_4)\end{aligned}$$

Es gilt also auch

$$\begin{aligned}\tilde{\mathrm{B}}^M(v_1, v_2, v_3, v_4) &= \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}^M_{s,t}(v_1, v_2, v_3, v_4)}{\sigma^M_{s,t}} \\ &= \frac{\sigma^M_{v_1,v_3}(v_2)}{\sigma^M_{v_1,v_3}} \cdot \frac{\sigma^M_{v_1,v_4}(v_3)}{\sigma^M_{v_1,v_4}} \cdot \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}^M_{s,t}(v_1, v_4)}{\sigma^M_{s,t}}\end{aligned}$$

und damit:

Eigenschaft 3.23:

$$\tilde{\mathrm{B}}^M(v_1, v_2, v_3, v_4) = \frac{\sigma^M_{v_1,v_3}(v_2)}{\sigma^M_{v_1,v_3}} \cdot \frac{\sigma^M_{v_1,v_4}(v_3)}{\sigma^M_{v_1,v_4}} \cdot \tilde{\mathrm{B}}^M(v_1, v_4)$$

Daher ist auch dieser Fall in konstanter Zeit berechenbar.

Es ist somit insgesamt möglich, die einzelnen Werte der verwalteten Datenstrukturen jeweils in $\mathcal{O}(1)$ zu aktualisieren.

3.4.3 Algorithmus

Der vorherige Algorithmus 4 für die Group Betweenness von Knotenmengen lässt sich jetzt einfach auf Kantenmengen anpassen, man erhält den Algorithmus 5. Die Fallunterscheidungen zum Update der Path Betweenness in Zeile 9 sind dabei zwecks Übersichtlichkeit in Algorithmus 6 ausgelagert.

Algorithmus 5 : Group Betweenness Centrality für Kanten

Eingabe : $G = (V, E), C \subseteq E$ und die vorberechneten Werte $d, \sigma, \tilde{\mathrm{B}}$
Ausgabe : $\ddot{\mathrm{B}}(C)$

```
0  C' := {v ∈ V | ∃_{e∈C} v ∈ e};
1  M := ∅;
2  ∀_{x,y∈C'} σ^M_{x,y} := σ_{x,y};
3  ∀_{x,y∈C'} B̃^M(x,y) := B̃(x,y);
4  B̈ := 0;
5  foreach e = [u,v] ∈ C do
6      B̈ := B̈ + B̃^M(u,v) + B̃^M(v,u);
7      foreach x,y ∈ C do
8          σ^{M∪{e}}_{x,y} := σ^M_{x,y} − σ^M_{x,y}(e);
9          B̃^{M∪{e}}(x,y) := newpc(G, d, σ^M, B̃^M, x, y, e);
10     M := M ∪ {e};
11 return B̈;
```

Laufzeit Wie beim Algorithmus für Knoten wird die äußere Schleife auch hier k mal durchlaufen. Allerdings sind jetzt durch die innere Schleife k'^2 Werte zu aktualisieren, was wieder für jeden Wert in konstanter Zeit möglich ist. Wegen $k \leq k' \leq 2k$ bleibt es bei der Wachstumsklasse $\mathcal{O}(k^3)$ für die Bestimmung der *Group Betweenness Centrality* auch für Gruppen von Kanten, wenn man die Laufzeit von $\mathcal{O}(n^3)$ des Preprocessings außer Acht lässt.

Algorithmus 6 : Prozedur newpc zum Update der Path Betweenness

Eingabe : $G, d, \sigma^M, \tilde{B}^M, x, y, e = [u, v]$
Ausgabe : $\tilde{B}^{M \cup \{e\}}(x, y)$

if $x = y = u$ **or** $x = y = v$ **then**
  return $\tilde{B}^M(x, x) - \tilde{B}^M(u, v) - \tilde{B}^M(v, u)$;
else if $x = y$ **and** $u \neq x \neq v$ **then**
  if $d(x, u) > d(x, v)$ **then**
    return $\tilde{B}^M(x, x) - \tilde{B}^M(u, v, x) - \tilde{B}^M(x, v, u)$;
  else if $d(x, u) < d(x, v)$ **then**
    return $\tilde{B}^M(x, x) - \tilde{B}^M(v, u, x) - \tilde{B}^M(x, u, v)$;
  else `/* kein gemeinsamer kürzester Weg */`
    return $\tilde{B}^M(x, x)$;
else
  return $\tilde{B}^M(x, y) - \tilde{B}^M((x, y) \circ e)$;

4 Das KPP-Com-Problem der Group Betweenness Centrality

4.1 Das KPP-Com-Problem und der Approximationsalgorithmus ItrK

Wie wir in Abschnitt 1.1.2 gesehen haben, stellt sich oft die Frage nach der Gruppe von Knoten mit der höchsten Zentralität. Da die Group Betweenness Centrality ein Maß hierfür darstellt, ist für uns die Gruppe mit der höchsten Zentralität die, die die höchste Group Betweenness Centrality aufweist.

In [PED07b] wird diese Frage als **KPP-Com**-Problem (Key Player Problem in Communication) aufgeworfen, da die Gruppe mit der höchsten Group Betweenness Centrality als eine Gruppe von Knoten mit hohem Einfluss auf die Kommunikation im Netzwerk angesehen wird. Genauer formuliert geht es darum, für ein gegebenes Netzwerk $G = (V, E)$ und ein $k \in \mathbb{N}$ eine k-elementige Knotenmenge $C \subseteq V$ zu finden, die die höchste Group Betweenness unter allen k-elementigen Mengen von Knoten aufweist.

Definition 4.1 (KPP-Com-Problem)**:**
Gegeben: Ein Graph $G = (V, E)$; ein $k \in \mathbb{N}$
Gesucht: Eine Teilmenge $C' \subseteq V$ mit $|C'| = k$, so dass

$$\ddot{\mathrm{B}}(C') := \sum_{s,t \in V | s \neq t} \frac{\ddot{\sigma}(C')}{\sigma_{s,t}}$$

maximal ist.

Bemerkung 4.2: Wie die Autoren in [PED07b] bemerken, ist KPP-Com NP-hart, da sich das Problem eine *minimale Eckenüberdeckung* (vgl. [KN05, Seite 69]) zu finden, das NP-vollständig ist, auf KPP-Com reduzieren lässt: Eine Gruppe von Knoten, deren Group Betweenness gleich $\ddot{\mathrm{B}}(V)$ ist muss jeden kürzesten Weg der Länge ≥ 1 im Graphen überdecken. Insbesondere muss sie einen Endknoten jeder Kante enthalten, da jede Kante auch einen kürzesten Weg darstellt. Umgekehrt überdeckt jede Eckenüberdeckung auch jeden kürzesten Weg im Graphen. Eine Eckenüberdeckung ist also eine Teilmenge

$C \subseteq V$ mit $\ddot{\mathrm{B}}(C) = \ddot{\mathrm{B}}(V)$. Indem man nacheinander für $k = n, n-1, \ldots$ das KPP-Com-Problem löst, findet man daher eine minimale Eckenüberdeckung.

Dass KPP-Com schwer zu lösen ist, ist auch informell leicht klar, da es exponentiell viele Teilmengen von V gibt.

4.1.1 Der Greedy-Algorithmus ItrK zur Approximation von KPP-Com

Da KPP-Com NP-hart ist, stellt sich die Frage nach einem Algorithmus, der dieses Problem in Polynomialzeit approximiert. In [PED07b] und [PED07a] wird dazu der Greedy-Algorithmus **ItrK** mit k Iterationsschritten vorgeschlagen. Dieser ergibt sich direkt durch Umwandlung des schnellen Group-Betweenness-Centrality-Algorithmus (Algorithmus 4) in einen Greedy-Algorithmus (Algorithmus 7): Die Gruppe wird wieder schrittweise aufgebaut. Anstatt aber in jedem Schritt einen neuen der schon bekannten Knoten hinzuzunehmen, wird aus den noch vorhandenen Knoten des gesamten Graphen derjenige gewählt, der den höchsten Zugewinn zur bisher ermittelten Group Betweenness bringt. Da jeweils alle Knoten in Frage kommen, müssen dazu natürlich auch die Werte $\sigma^M_{x,y}$ und $\tilde{\mathrm{B}}^B(x,y)$ für alle $x, y \in V$ verwaltet werden. $\sigma^M, \tilde{\mathrm{B}}^M$ sind also $n \times n$-Matrizen.

4.1.2 Laufzeit des Greedy-Algorithmus

Ohne Beachtung des Preprocessing-Schrittes wird die äußere Schleife k mal durchlaufen. Die innere Schleife zum Update der vorgehaltenen Werte muss jetzt allerdings n^2 mal durchlaufen werden, wobei die einzelnen Werte wiederum in konstanter Zeit berechnet werden. Hinzu kommt vorher noch die Suche nach dem Knoten, der die größte Verbesserung von $\ddot{\mathrm{B}}$ bringt. Da jeweils höchstens n Werte in Frage kommen, benötigt dieser Schritt $\mathcal{O}(n)$. Insgesamt kommt man für den Algorithmus auf eine Laufzeitklasse von $\mathcal{O}(kn^2)$, was bei einmaliger Ausführung vom Preprocessingschritt mit $\mathcal{O}(n^3)$ dominiert wird.

4.2 Das KPP-Com-Problem für Gruppen von Kanten

Ebenso wie im Fall der Knotengruppen ergibt sich auch für Gruppen von Kanten das KPP-Com-Problem, nämlich die Frage nach einer Gruppe von

Algorithmus 7 : Greedy-Algorithmus ItrK

Eingabe : G, k und die vorberechneten Werte $d, \sigma, \tilde{\mathrm{B}}$
Ausgabe : $C \subseteq V$ mit $|C| = k$ und möglichst hohem Wert $\ddot{\mathrm{B}}(C)$

```
1  M := ∅;
2  ∀_{x,y∈V} σ^M_{x,y} := σ_{x,y};
3  ∀_{x,y∈V} B̃^M(x,y) := B̃(x,y);
4  B̈ := 0;
5  for i := 1 to k do
6      wähle v_i ∈ V \ M mit B̃^M(v_i,v_i) maximal;
7      B̈ := B̈ + B̃^M(v_i,v_i);
8      foreach x,y ∈ V do
9          σ^{M∪{v_i}}_{x,y} := σ^M_{x,y} − σ^M_{x,y}(v_i);
10         if x = y ≠ v_i then
11             B̃^{M∪{v_i}}(x,x) := B̃^M(x,x) − B̃^M(v_i,x) − B̃^M(x,v_i);
12         else
13             B̃^{M∪{v_i}}(x,y) := B̃^M(x,y) − B̃^M((x,y) ∘ v_i);
14     M := M ∪ {v_i};
15 return M;
```

Kanten mit einer möglichst hohen Group Betweenness Centrality. Wir werden sehen, dass es wiederum wie in Abschnitt 3.3 möglich ist, den schon vom Fall der Knotengruppen bekannten Algorithmus zu modifizieren.

4.2.1 Der Greedy-Algorithmus ItrK zur Approximation der wichtigsten Kantengruppe

Analog wie im Falle von Knotengruppen, lässt sich auch der von uns abgewandelte Algorithmus 5 zur Berechnung der Group Betweenness von Kantengruppen anpassen, um jetzt eine Gruppe von Knoten mit möglichst hoher Group Betweenness zu finden. Auch hier wird der Algorithmus in einen Greedy-Algorithmus **ItrK'** umformuliert, indem in jedem Schritt die Kante gewählt wird, die die bisherige Group Betweenness maximal erhöht. Der resultierende Algorithmus 15 findet sich in Anhang A.6.

4.2.2 Laufzeit

Lässt man wiederum das Preprocessing außen vor, so gibt es k Durchläufe der äußeren Schleife. In dieser werden die m Werte nach der Kante mit der größten Verbesserung für die bisherige Group Betweenness Centrality durchsucht. Anschließend müssen die $2n^2 + 1$ gespeicherten Werte jeweils in $\mathcal{O}(1)$ aktualisiert werden. Man erhält eine Gesamtlaufzeit von $\mathcal{O}(k(m + n^2))$. Da G ein einfacher Graph ist, gilt $m \leq n^2$, und die Laufzeit ist wiederum $\mathcal{O}(kn^2)$, dominiert vom Preprocessing mit $\mathcal{O}(n^3)$.

4.3 Approximationsgüte von ItrK

4.3.1 KPP-Com und das Maximum-Coverage-Problem

In [PED07b] und [PED07a] wird der Approximationsalgorithmus ItrK für das Problem KPP-Com nur experimentell untersucht, es wird keine Aussage über seine Approximationsgüte gemacht. Wir werden diese im Folgenden untersuchen und dabei sehen, dass es aufgrund der Ähnlichkeit zum Maximum-Coverage-Problem, wie es z.B. in [JM08] definiert wird, möglich ist, einen Approximationsfaktor anzugeben. Beim Maximum-Coverage-Problem gibt es eine Grundmenge von Grundelementen und verschiedene Teilmengen der Grundmenge. Gesucht sind k dieser Teilmengen, die zusammen möglichst viele der Grundelemente enthalten. Zusätzlich werden dabei die Grundelemente noch gewichtet.

KPP-Com (vergleiche Definition 4.1)

Gegeben: Ein Graph $G = (V, E)$; ein $k \in \mathbb{N}$
Gesucht: Eine Teilmenge $C' \subseteq V$ mit $|C'| = k$, so dass

$$\ddot{\mathrm{B}}(C') := \sum_{s,t \in V | s \neq t} \frac{\ddot{\sigma}(C')}{\sigma_{s,t}}$$

maximal ist.

Maximum Coverage Problem ([JM08])

Definition 4.3 (Maximum-Coverage-Problem)**:**
Gegeben: Eine Grundmenge S mit Gewichtsfunktion $w : S \to \mathbb{R}_0^+$; eine Mengenfamilie $\mathcal{F} = \{S_1, \ldots, S_m\}$ mit $\forall_{i=1}^{m} S_i \subseteq S$; ein $k \in \mathbb{N}$

Gesucht: Eine Gruppe $C \subseteq \mathcal{F}$ von Mengen mit $|C| = k$, so dass mit $S' := \bigcup_{\tilde{S} \in C} \tilde{S}$

$$w(S') := \sum_{s \in S'} w(s)$$

maximal ist.

4.3.2 Zusammenhang zwischen KPP-Com und Maximum-Coverage

Wir werden zeigen, dass sich KPP-Com auf Maximum-Coverage zurückführen lässt. Des Weiteren wird man sehen, dass der Approximationsalgorithmus ItrK (Algorithmus 7) nach demselben Prinzip arbeitet wie der Greedy-Algorithmus AMC für das Maximum-Coverage-Problem, den wir im Folgenden noch kennenlernen werden.

Definition 4.4: Sei eine KPP-Com-Instanz $(G = (V, E), k)$ mit den Knoten $V = \{v_1, \ldots, v_n\}$ gegeben. Dann sei für $s, t \in V$ mit $s \neq t$:

$$S_{s,t} := \{P \mid P \text{ ist kürzester } s\text{-}t\text{-Weg}\}$$

Setze

$$S := S(G) := \{P \mid \exists_{s,t \in V} : s \neq t \wedge P \in S_{s,t}\} = \bigcup_{s,t \in V | s \neq t} S_{s,t}$$

als Menge der kürzesten Wege des ganzen Graphen. Für einen Weg $P \in S_{s,t}$ sei $w(P) := \frac{1}{\sigma_{s,t}}$ sein Gewicht. Schließlich sei

$$\mathcal{F} := \mathcal{F}(G) := \{S(v_1), \ldots, S(v_n)\}$$

mit

$$S(v_i) := \{P \in S(G) \mid \exists_{s,t \in V} : s \neq t \wedge P \in S_{s,t} \wedge v_i \in P\}$$

Bemerkung 4.5: $(S, w, \mathcal{F}, k) = (S(G), w, \mathcal{F}(G), k)$ bildet jetzt eine Maximum-Coverage-Instanz.

Für ein $C' = \{c_1, \ldots, c_k\} \subseteq V$ sei

$$C(C') := \{S(c_1), \ldots, S(c_k)\} \subseteq \mathcal{F}(G)$$

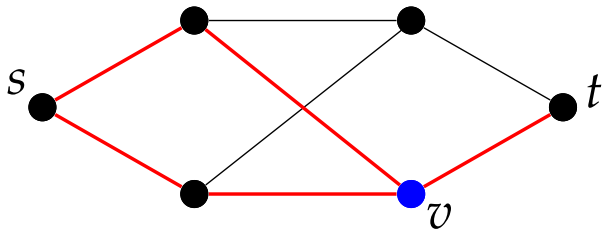

Abbildung 18: Zwei s-t-Wege in $S(v)$

Dann gilt mit $S(C') := \bigcup_{A \in C(C')} A = \bigcup_{i=1}^{k} S(c_i)$:

$$\begin{aligned}
w(S(C')) &= \sum_{P \in S(C')} w(P) \\
&= \sum_{s,t \in V | s \neq t} \sum_{P \in S_{s,t} \cap S(C')} w(P) \\
&= \sum_{s,t \in V | s \neq t} \frac{1}{\sigma_{s,t}} \sum_{P \in S_{s,t} \cap S(C')} 1 \\
&= \sum_{s,t \in V | s \neq t} \frac{1}{\sigma_{s,t}} \cdot |\{P \mid P \text{ kürzester } s\text{-}t\text{-Weg } \wedge \exists_{c \in C'} P \in S(c)\}| \\
&= \sum_{s,t \in V | s \neq t} \frac{1}{\sigma_{s,t}} \cdot |\{P \mid P \text{ kürzester } s\text{-}t\text{-Weg } \wedge \exists_{c \in C'} c \in P\}| \\
&= \sum_{s,t \in V | s \neq t} \frac{1}{\sigma_{s,t}} \cdot \ddot{\sigma}_{s,t}(C') \\
&= \ddot{\mathrm{B}}(C')
\end{aligned}$$

Damit entspricht die Group Betweenness Centrality einer Menge von Knoten genau dem Gewicht der entsprechenden Menge kürzester Wege in der zugehörigen Maximum-Coverage-Instanz, insbesondere gilt für ein $v \in V$ auch

$$\mathrm{B}(v) = \ddot{\mathrm{B}}(\{v\}) = w(S(v))$$

Daher entspricht das Lösen einer KPP-Com-Instanz genau dem Lösen der zugehörigen Maximum-Coverage-Instanz.

4.3.3 Die Algorithmen

Der Greedy-Algorithmus AMC (Approximate Maximum Coverage) zur Approximation des Maximum-Coverage-Problems wird in der Form, in der er in Algorithmus 8 zu sehen ist, in [JM08, Kapitel 4] angegeben.

Algorithmus 8 : Greedy-Algorithmus AMC$(S, w, \mathcal{F}, k)$ für Maximum-Coverage

```
1 U := S;
2 C := ∅;
3 for l = 1 to k do
4 |   wähle S̃_l ∈ F mit w(S̃_l ∩ U) maximal;
5 |   U := U \ S̃_l;
6 |   C := C ∪ {S̃_l};
7 return C;
```

Es gilt dabei folgender Satz:

Satz 4.6: *Sei $C \subseteq \mathcal{F}$ die Ausgabe von* AMC$(S, w, \mathcal{F}, k)$ *und $C^* \subseteq \mathcal{F}$ diejenige k-elementige Teilmenge von $\mathcal{F}$ für die $w(\bigcup_{\tilde{S} \in C^*} \tilde{S})$ maximal ist. Dann gilt:*

$$w(\bigcup_{\tilde{S} \in C} \tilde{S}) \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot w(\bigcup_{\tilde{S} \in C^*} \tilde{S}) \geq \left(1 - \frac{1}{e}\right) \cdot w(\bigcup_{\tilde{S} \in C^*} \tilde{S})$$

Beweis. siehe z.B. [JM08, Kapitel 4] □

Wir wollen den Greedy-Algorithmus ItrK (Algorithmus 7) für KPP-Com hier nochmals schematisch angegeben, um die starke Ähnlichkeit der Algorithmen zu sehen. Wir erhalten den umgeformten Algorithmus 9.

Algorithmus 9 : Greedy-Algorithmus ItrK$(G = (V, E), k)$ für KPP-Com

```
1 [U := S(G)];
2 C' := ∅;
3 for l = 1 to k do
4 |   wähle ṽ_l ∈ V mit B̈(C' ∪ {ṽ_l}) − B̈(C') maximal;
5 |   [U := U \ S(ṽ_l)];
6 |   C' := C' ∪ {ṽ_l};
7 return C';
```

4.3.4 Zusammenhang und Folgerungen

Wie man sieht, ist der einzige relevante Unterschied der beiden Algorithmen die Auswahl der jeweils nächsten Menge bzw. des nächsten Knotens in Zeile 4. Es lässt sich zeigen, dass in beiden Fällen derselbe Wert maximiert wird,

falls eine zur Maximum-Coverage-Instanz umgeformte KPP-Com-Instanz verwendet wird:

$$\begin{aligned}
\ddot{\mathrm{B}}(C' \cup \{\tilde{v}_l\}) - \ddot{\mathrm{B}}(C') &= \sum_{s,t \in V | s \neq t} \frac{\ddot{\sigma}_{s,t}(C' \cup \{\tilde{v}_l\})}{\sigma_{s,t}} - \sum_{s,t \in V | s \neq t} \frac{\ddot{\sigma}_{s,t}(C')}{\sigma_{s,t}} \\
&= \sum_{s,t \in V | s \neq t} \frac{1}{\sigma_{s,t}} \left(\ddot{\sigma}_{s,t}(C' \cup \{\tilde{v}_l\}) - \ddot{\sigma}_{s,t}(C') \right) \\
&= \sum_{s,t \in V | s \neq t} \frac{1}{\sigma_{s,t}} \cdot |\{P \in S_{s,t} \mid \tilde{v}_l \in P \wedge \neg \exists_{c \in C'} c \in P\}| \\
&= \sum_{s,t \in V | s \neq t} \frac{1}{\sigma_{s,t}} \cdot |\{P \in S_{s,t} \mid P \in S(\tilde{v}_l) \wedge \neg \exists_{c \in C'} P \in S(c)\}| \\
&= \sum_{s,t \in V | s \neq t} \frac{1}{\sigma_{s,t}} \cdot |\{P \in S_{s,t} \mid P \in S(\tilde{v}_l) \cap U\}| \\
&= \sum_{P \in U \cap S(\tilde{v}_l)} w(P) \\
&= w(U \cap S(\tilde{v}_l))
\end{aligned}$$

Damit folgt direkt aus Satz 4.6:

Satz 4.7: *Sei $C' \subseteq V$ die von ItrK zu einer KPP-Com-Instanz $(G = (V, E), k)$ ermittelte Knotenmenge und sei $C^* \subseteq V$ die k-elementige Knotenmenge mit $\ddot{\mathrm{B}}(C^*)$ maximal, dann gilt:*

$$\ddot{\mathrm{B}}(C') \geq \left(1 - (1 - \frac{1}{k})^k\right) \ddot{\mathrm{B}}(C^*) \geq \left(1 - \frac{1}{e}\right) \ddot{\mathrm{B}}(C^*)$$

Wir wissen jetzt also, dass der Greedy-Algorithmus zur Bestimmung einer Menge von Knoten mit möglichst hoher Group Betweenness Centrality das Problem KPP-Com mit einem Approximationsfaktor von $1 - \frac{1}{e}$, bzw. von $1 - \left(1 - \frac{1}{k}\right)^k$ in Abhängigkeit von k, approximiert.

4.4 Schärfe der Abschätzung

Im vorherigen Abschnitt wurde ein Approximationsfaktor von $1 - \frac{1}{e}$ für den Greedy-Algorithmus 7 zur Approximation von KPP-Com nachgewiesen. Es ist aber noch nicht klar, wie gut die gemachte Abschätzung war. Im Folgenden soll daher diese Abschätzung weiter untersucht werden.

Wie in [Fei98] gezeigt wird, ist es unter der Voraussetzung P $\neq$ NP nicht möglich, in Polynomialzeit Maximum-Coverage mit einer Güte von $1 - \frac{1}{e} + \varepsilon$ für irgendein festes $\varepsilon > 0$ zu approximieren. Insbesondere ist die für den zugehörigen Greedy-Algorithmus AMC gezeigte Abschätzung der Approximationsgüte scharf. Es stellt sich die Frage, ob die Abschätzung auch im Spezialfall *KPP-Com* gilt, oder ob hier der Greedy-Algorithmus besser arbeitet. Wir werden sehen, dass auch für ItrK die gezeigte Abschätzung scharf ist.

Dazu soll für jedes $k \in \mathbb{N}$ eine Familie von KPP-Com-Instanzen konstruiert werden, mit deren Hilfe man dem Approximationsfaktor $1 - \left(1 - \frac{1}{k}\right)^k$ im negativen Sinne beliebig nahe kommt, nämlich bis auf ein beliebig kleines $\varepsilon > 0$.

Bemerkung 4.8: Da die endgültige Konstruktion recht kompliziert ist, werden wir bei der Herleitung auf gerichtete Graphen zurückgreifen. Diese verwenden wir aber eher intuitiv ohne explizite Definition, um kürzeste Wege nur in eine Richtung betrachten zu müssen.

In den im Weiteren dargestellten Graphen stehen Zahlen x an gestrichelten Kanten für eine, wie auch immer geartete, Konstruktion von Knoten auf den durch die gestrichelten Kanten dargestellten Wegen. Diese Konstruktion ermöglicht dabei x verschiedene Wege beim Durchlaufen der gestrichelten Kante. Zudem können diese Wege so konstruiert werden, dass die Distanzen zwischen s und t auf allen Wegen gleich sind.

Wird im Folgenden von Wegen gesprochen, so sind stets kürzeste Wege gemeint.

4.4.1 Überlegungen für kleines k

Wir betrachten den Fall $k = 1$. Hier ist $1 - \left(1 - \frac{1}{k}\right)^k = 1$. Da die Group Betweenness Centrality einer einelementigen Gruppe gleich der Shortest Path Betweenness Centrality des enthaltenen Knotens ist, wird stets eine optimale Lösung gefunden. In jedem Graph wird also die behauptete Approximationsgüte für $k = 1$ erreicht.

Betrachten wir jetzt den Fall $k = 2$. Der Einfachheit halber beschränken wir uns für die vorbereitenden Überlegungen auf gerichtete Graphen. Durch das Ersetzen von ungerichteten durch gerichtete Wege funktioniert auch hier der Greedy-Algorithmus mit derselben Approximationsgüte. Des Weiteren betrachten wir vorerst nur Wege von einem ausgezeichneten Knoten s zu einem weiteren ausgezeichneten Knoten t. Außerdem wollen wir uns auf eine festgelegte Menge von „Kandidatenknoten“ einschränken, so dass alle anderen Knoten im Greedy-Algorithmus nicht gewählt werden können.

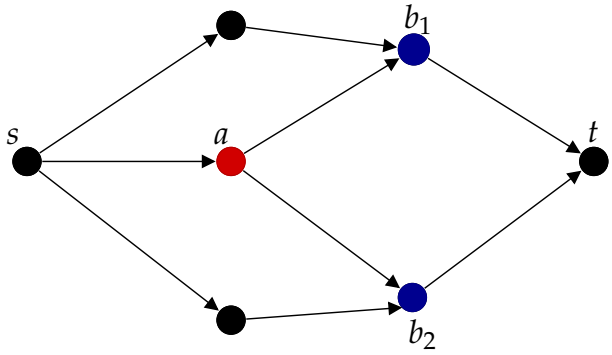

Abbildung 19: Idee für $k = 2$

Betrachten wir den Graphen aus Abbildung 19 für $k = 2$. Die drei Knoten a, b_1, b_2 seien dabei die Kandidaten. Es gibt 4 s-t-Wege, davon je zwei durch jeden der Kandidaten. Sei im ersten Schritt a ausgewählt. Dann wird im zweiten Schritt nur einer der Knoten b_1 und b_2 ausgewählt, ein s-t-Weg bleibt also unbelegt. In einer optimalen Lösung können dagegen durch Auswahl von b_1 und b_2 alle 4 s-t-Wege belegt werden. Man kommt somit auf einen Approximationsfaktor von $\frac{3}{4} = 1 - \frac{1}{4} = 1 - \left(1 - \frac{1}{2}\right)^2$.

4.4.2 Überlegungen für beliebiges $k \geq 2$

Wir betrachten jetzt den allgemeinen Fall $k \geq 2$. Dabei behalten wir die für $k = 2$ gemachten Einschränkungen bei. Um auf den Faktor von

$$1 - \left(1 - \frac{1}{k}\right)^k = 1 - \left(\frac{k-1}{k}\right)^k = 1 - (k-1) \cdot \frac{(k-1)^{k-1}}{k^k}$$

zu kommen, wollen wir einen Graphen so konstruieren, dass in einer optimalen Lösung alle von insgesamt k^k vielen s-t-Wegen überdeckt werden können. In der suboptimalen Lösung, die von ItrK geliefert werden soll, müssen dann $(k-1)^k = (k-1)(k-1)^{k-1}$ viele s-t-Wege unbelegt bleiben.

Diese Zahlen stimmen genau mit den Anzahlen der Wege im Beispiel für $k = 2$ überein. Hier bleiben nach der Wahl von $1 = k - 1$ Knoten noch $2 = k$ Knoten mit jeweils $1 = (k-1)^{k-1}$ unbelegten s-t-Wegen übrig, von denen nur einer gewählt werden kann.

Ein entsprechendes Schema soll auch für $k \geq 3$ entwickelt werden. Es gebe dabei genau $2k - 1$ Kandidatenknoten $a_1, \ldots, a_{k-1}, b_1, \ldots, b_k$, angeordnet in einer vorderen $(a_1, \ldots, a_{k-1})$ und einer hinteren $(b_1, \ldots, b_k)$ Schicht. Wie in Abbildung 20 zu erkennen, soll dabei jeder der Knoten a_i über einen Pfeil mit

jedem der Knoten b_j verbunden sein. Für die gestrichelt dargestellten Wege vergleiche Bemerkung 4.8.

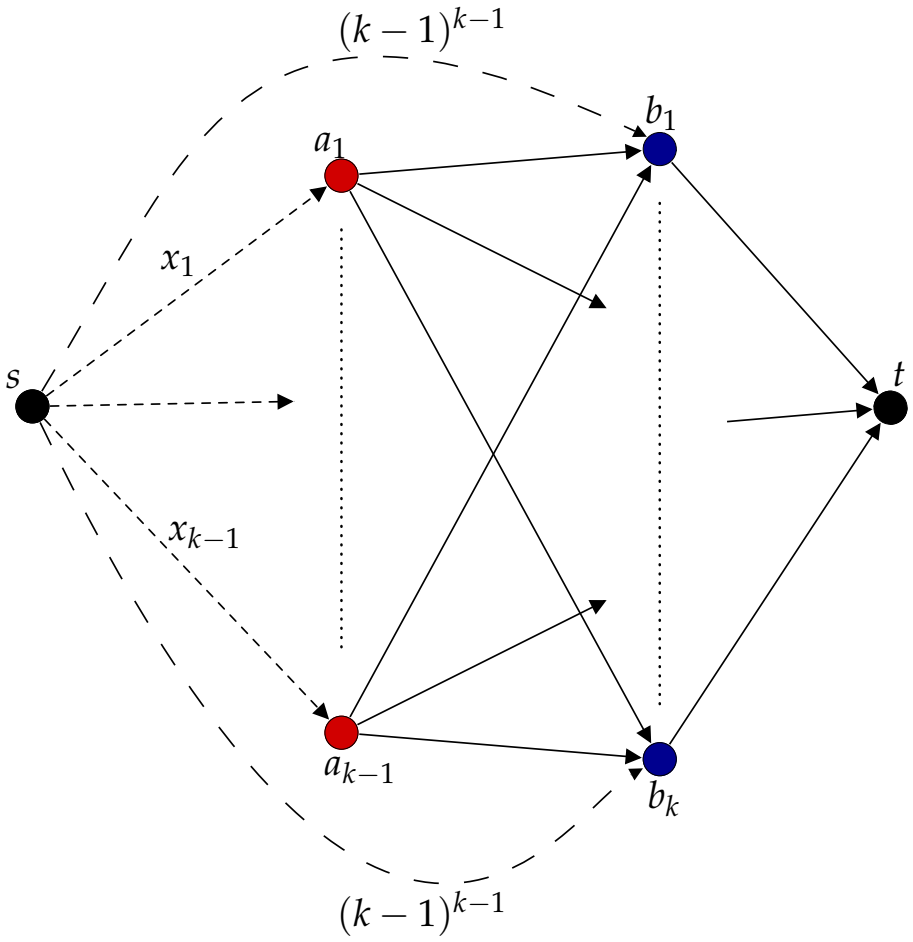

Abbildung 20: Schema für $k \geq 2$

Für jedes a_i bezeichne x_i die Anzahl der s-a_i-Wege, die wir noch bestimmen werden, so dass insgesamt $k \cdot x_i$ viele s-t-Wege durch a_i führen. Zusätzlich sollen von s zu jedem b_j genau $(k-1)^{k-1}$ viele Wege führen, die keinen der Knoten $a_1, \ldots, a_{k-1}$ berühren. Gesucht sind jetzt passende Anzahlen $x_1, \ldots, x_{k-1}$ von Wegen, so dass gilt:

(i) Es gibt insgesamt k^k viele s-t-Wege.

(ii) In den ersten $k-1$ Schritten des Greedy-Algorithmus sollen die vorderen Knoten $a_1, \ldots, a_{k-1}$ in dieser Reihenfolge ausgewählt werden (insbesondere also $x_1 \geq x_2 \geq \ldots \geq x_{k-1}$).

Da durch jedes b_j gleich viele der k^k vielen s-t-Wege (also je k^{k-1} viele)

führen, ergibt sich aus Forderung (i):

$$k^{k-1} = (k-1)^{k-1} + \sum_{i=1}^{k-1} x_i$$

Also muss $\sum_{i=1}^{k-1} x_i = k^{k-1} - (k-1)^{k-1}$ sein. Wir versuchen, die zweite Forderung zu erfüllen, indem wir für jedes x_i jeweils den gerade nötigen Minimalwert wählen, so dass durch a_i genauso viele s-t-Wege verlaufen, wie durch die b_j nach vorheriger Auswahl von $a_1, \ldots, a_{i-1}$ noch unüberdeckt sind:

- Durch a_1 müssen $x_1 k = k^{k-1}$ viele s-t-Wege gehen, wähle also $x_1 := k^{k-2}$. Es bleiben durch jedes b_j noch $k^{k-1} - k^{k-2} = k^{k-2}(k-1)$ viele unbelegte s-t-Wege.
- Durch a_2 müssen $x_2 k = k^{k-2}(k-1)$ viele s-t-Wege gehen, wähle also $x_2 := k^{k-3}(k-1)$. Es bleiben durch jedes b_j noch

  $$k^{k-2}(k-1) - k^{k-3}(k-1) = k^{k-3}(k-1)(k-1) = k^{k-3}(k-1)^2$$

  viele unbelegte s-t-Wege.

Wir wollen dies induktiv für $i \leq k-1$ formulieren:

- Durch a_i müssen $x_i k = k^{k-i}(k-1)^{i-1}$ viele s-t-Wege gehen, wähle also

  $$x_i := k^{k-1-i}(k-1)^{i-1}$$

  Es bleiben durch jedes b_j noch

  $$\begin{aligned} k^{k-i}(k-1)^{i-1} - k^{k-1-i}(k-1)^{i-1} &= k^{k-1-i}(k-1)^{i-1}(k-1) \\ &= k^{k-1-i}(k-1)^i \end{aligned}$$

  viele unbelegte s-t-Wege.
- Durch a_{i+1} müssen $x_{i+1} k = k^{k-1-i}(k-1)^i$ viele s-t-Wege gehen, wähle also

  $$x_i := k^{k-2-i}(k-1)^i$$

  Es bleiben durch jedes b_j noch

  $$\begin{aligned} k^{k-1-i}(k-1)^i - k^{k-2-i}(k-1)^i &= k^{k-2-i}(k-1)^i(k-1) \\ &= k^{k-2-i}(k-1)^{i+1} \end{aligned}$$

  viele unbelegte s-t-Wege.

Wir betrachten den vorletzten Schritt, die Auswahl von a_{k-1}:

- Durch a_{k-1} müssen $x_{k-1}k = k(k-1)^{k-2}$ viele s-t-Wege gehen, wähle also

  $$x_{k-1} := (k-1)^{k-2}$$

  Es bleiben durch jedes b_j noch

  $$\begin{aligned} k(k-1)^{k-2} - (k-1)^{k-2} &= (k-1)^{k-2}(k-1) \\ &= (k-1)^{k-1} \end{aligned}$$

  viele unbelegte s-t-Wege.

Man sieht also, dass mit der Wahl von $x_i := k^{k-1-i}(k-1)^{i-1}$ für jedes a_i die Knoten in der „richtigen“, d.h. der gewünschten, Reihenfolge vom Greedy-Algorithmus ausgewählt werden können. Es stellt sich aber noch die Frage, ob damit die erste Forderung erfüllt wird, also ob

$$\sum_{i=1}^{k-1} x_i = \sum_{i=1}^{k-1} k^{k-1-i}(k-1)^{i-1} = k^{k-1} - (k-1)^{k-1}$$

gilt. Implizit haben wir das durch das induktive Vorgehen schon gesehen. Das folgende Lemma zeigt, dass es sich leicht nachrechnen lässt.

Lemma 4.9: *Für $k \in \mathbb{N}$ gilt*

$$k^{k-1} - (k-1)^{k-1} = \sum_{i=1}^{k-1} k^{k-1-i} \cdot (k-1)^{i-1}$$

Beweis. Es gilt:

$$\begin{aligned} \sum_{i=1}^{k-1} k^{k-1-i}(k-1)^{i-1} &= \sum_{i=0}^{k-2} k^{k-2-i}(k-1)^{i} = k^{k-2} \cdot \sum_{i=0}^{k-2} \left(\frac{k-1}{k}\right)^i \\ &= k^{k-2} \cdot \frac{1 - \left(\frac{k-1}{k}\right)^{k-1}}{1 - \frac{k-1}{k}} = k^{k-2} \cdot \frac{1 - \frac{(k-1)^{k-1}}{k^{k-1}}}{\frac{1}{k}} \\ &= k^{k-1} - (k-1)^{k-1} \end{aligned}$$

□

Somit funktioniert dieses Verfahren. Der Greedy-Algorithmus kann also in den ersten $k-1$ Schritten die Knoten $a_1, \ldots, a_{k-1}$ auswählen. Im letzten Schritt wird noch einer der Knoten b_j ausgewählt. Es bleiben $k-1$ viele b_j

übrig, wobei durch jeden dieser Knoten genau $(k-1)^{k-1}$ viele unbelegte s-t-Wege verlaufen. Von der gefundenen Gruppe werden also

$$k^k - (k-1)(k-1)^{k-1} = k^k - (k-1)^k$$

viele s-t-Wege belegt. Dabei wäre es aber für eine optimale Lösung möglich, die k vielen Knoten b_j auszuwählen, womit alle k^k vielen s-t-Wege überdeckt wären. Es wird also genau ein Anteil von

$$\frac{k^k - (k-1)^k}{k^k} = 1 - \left(\frac{k-1}{k}\right)^k = 1 - \left(1 - \frac{1}{k}\right)^k$$

der möglichen kürzesten s-t-Wege überdeckt.

4.4.3 Zuleitungswege

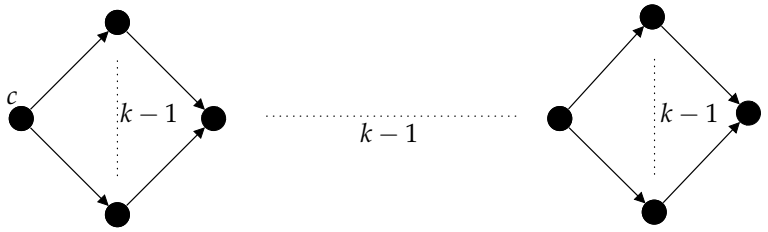

Abbildung 21: $(k-1)^{k-1}$ viele Wege zu den Knoten b_i

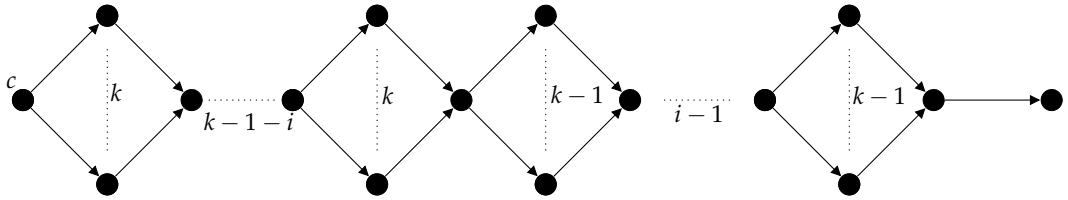

Abbildung 22: x_i viele Wege zum Knoten a_i

An dieser Stelle soll darauf eingegangen werden, wie man die x_i vielen Wege zu den Knoten a_i und die $(k-1)^{k-1}$ zu den Knoten b_j erzeugt. Wie in Abbildung 21 zu sehen ist, sind dabei zum Erzeugen der $(k-1)^{k-1}$ vielen Wege

$$(k-1) \cdot k = k^2 - k$$

viele Knoten nötig ohne den zugehörigen Verteilerknoten c. Im Falle der x_i vielen Wege zu a_i ist außer der eigentlichen Struktur noch ein zusätzlicher

Knoten nötig, damit die Distanz von c zu a_i um genau 1 geringer ist als die Distanz zu den b_j, siehe Abbildung 22. Ohne c werden hierfür

$$(k-1-i)\cdot(k+1)+(i-1)\cdot k+1 = k^2-k-i$$

Knoten benötigt. Für ein c sind dies für alle a_i zusammen

$$\sum_{i=1}^{k-1}\left(k^2-k-i\right) = (k-1)(k^2-k)-\sum_{i=1}^{k-1} i = k^3-2k^2+k-\frac{k(k-1)}{2}$$

Knoten.

4.4.4 Übergang zur vollständigen Konstruktion

Als Nächstes stellt sich die Frage, wie wir die diversen Einschränkungen, die wir zur Überlegung gemacht haben, wieder beseitigen.

Alle kürzesten Wege zählen

Kommen wir zuerst zur Beschränkung auf kürzeste Wege von einem Knoten s zu einem Knoten t. Lässt man diese fallen, so ist es fraglich, ob die Kandidatenknoten noch in der richtigen Reihenfolge ausgewählt werden (was z.B. im Beispiel für $k = 2$ nicht der Fall ist). Selbst wenn die gewünschten Knoten gewählt werden, wird sich sich die Group Betweenness Centrality der gefundenen Menge, sowie der optimalen Gruppe, ändern, so dass der gewünschte Faktor $1-\left(1-\frac{1}{k}\right)^k$ nicht mehr erreicht wird. Wir müssen also eine Möglichkeit finden, sowohl die gewünschte Reihenfolge, in der die Knoten ausgewählt werden, wiederherzustellen, als auch dem Faktor $1-\left(1-\frac{1}{k}\right)^k$, wenn er schon nicht erreicht wird, so doch beliebig nahe zu kommen.

Dazu ersetzen wir die Knoten s und t jeweils durch Mengen

$$S := \{s_1,\ldots,s_l\} \text{ und } T := \{t_1,\ldots,t_l\}$$

von Knoten, so dass es insgesamt l^2 viele Paare (s_i,t_j) gibt, für die die kürzesten s_i-t_j-Wege zur Group Betweenness Centrality beitragen. Dabei ist für alle s_i,t_j der Anteil der kürzesten s_i-t_j-Wege durch die Knoten $a_1,\ldots,a_{k-1}$ und $b_1,\ldots,b_k$ unabhängig von s_i und t_j. Zusätzlich wird ein Verteilerknoten c eingefügt, von dem aus die Wege aufgespannt werden, vergleiche Abbildung 23.

Für jeden Knoten v setzt sich in jedem Schritt des Greedy-Algorithmus der erreichbare Zugewinn für die Group Betweenness Centrality als

$$\alpha^M(v)l^2+\beta^M(v)l+\gamma^M(v)$$

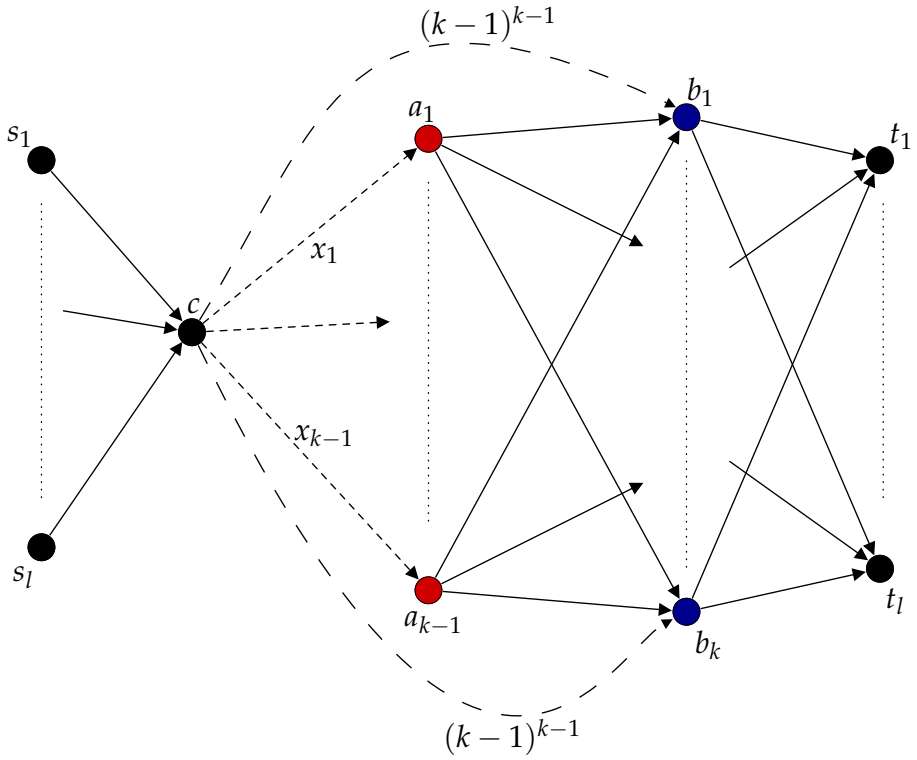

Abbildung 23: Vervielfachung von s und t

zusammen, wobei M die Menge aller bisher gewählten Knoten sei. Die einzelnen Werte $\alpha^M(v), \beta^M(v), \gamma^M(v)$ hängen dabei nur von dem Knoten v und den bisher gewählten Knoten M ab, nicht aber von der Anzahl l. Wir betrachten, wie $\alpha^M(v), \beta^M(v)$ und $\gamma^M(v)$ zu interpretieren sind:

- Der Wert $\alpha^M(v)$ gibt den für alle l^2 vielen Paare (s_i, t_j) identischen Anteil $\frac{\tilde{\sigma}^M_{s_i,t_j}(v)}{\sigma_{s_i,t_j}}$ der noch nicht von M überdeckten s_i-t_j Wege durch v unter allen s_i-t_j-Wegen an.

- $\beta^M(v)$ gibt für ein i mit $1 \leq i \leq l$ die Summe

$$\sum_{u \in V \setminus (S \cup T)} \left(\frac{\tilde{\sigma}^M_{s_i,u}(v)}{\sigma_{s_i,u}} + \frac{\tilde{\sigma}^M_{u,t_i}(v)}{\sigma_{u,t_i}} \right)$$

  an. Es wird also der Zugewinn für die Group Betweenness gemessen, der durch Paare von Knoten erreichbar ist, von denen nur genau einer in $S \cup T$ liegt, nämlich s_i oder t_i.

- Durch $\gamma^M(v)$ wird der Zugewinn der Group Betweenness bei Wahl von v gemessen, der von Wegen zwischen Paaren (u, w) mit $u, w \in V \setminus (S \cup$

T) stammt. Die Anzahl solcher Paare hängt also nicht von l ab, sondern ist für ein festes k konstant.

Durch die Wahl eines genügend großen Wertes für l lässt sich erreichen, dass die Werte $\alpha^M(v), \beta^M(v), \gamma^M(v)$ in dieser Reihenfolge festlegen, welcher Knoten jeweils gewählt wird. Wie wir vorher gesehen haben, ermöglichen die α-Werte die Auswahl in der richtigen Reihenfolge. Da allerdings in jedem der k Schritte mindestens die k Knoten b_i den gleichen α-Wert aufweisen (vergleiche 4.4.2), müssen hier auch noch die β- und möglicherweise auch γ-Werte den Ausschlag zur Bevorzugung der Knoten a_i geben.

Um zu erreichen, dass schon die β-Werte für die a_i sprechen, fügen wir eine Hilfskonstruktion wie in Abbildung 24 ein, d.h. an jedes a_i wird eine gleiche Anzahl $r(k)$ an zusätzlichen Knoten angehängt. Durch die Wege von den s_j aus zu diesen Knoten, die alle durch das jeweilige a_i gehen, erhöhen sich die β-Werte von a_i, so dass nur $r(k)$ genügend groß gewählt werden muss. Da die nötige Anzahl auch von weiteren Faktoren, die wir im Folgenden ändern werden, abhängt, wird auf diese Hilfsknoten erst im eigentlichen Beweis wieder eingegangen.

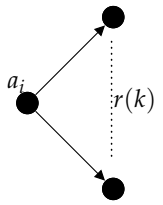

Abbildung 24: $r(k)$ viele zusätzliche Knoten an a_i

Alle Knoten als Kandidaten für den Algorithmus

Die nächste Einschränkung, die fallen soll, ist die auf die Knoten $a_1, \ldots, a_{k-1}$ und $b_1, \ldots, b_k$ als Kandidatenknoten für den Greedy-Algorithmus. Wir wollen erreichen, dass hier schon der Anteil $\alpha^M(v)$ an den s_i-t_j-Wegen für die l^2 vielen Paare (s_i, t_j) den Ausschlag gibt, so dass für genügend großes l doch nur die Knotenmengen $\{a_1, \ldots, a_{k-1}\}$ und $\{b_1, \ldots, b_k\}$ ernsthafte Kandidaten darstellen.

Offenbar gilt für jeden der Knoten $v \in \{s_1, \ldots, s_l, t_1, \ldots, t_l\}$ schon $\alpha^{\emptyset}(v) = 0$. Damit stören nur noch der Verteilerknoten c und die folgenden Knoten, die die richtige Anzahl an Wegen zu den Knoten $a_1, \ldots, a_{k-1}, b_1, \ldots, b_k$ erzeugen. Um dieses Problem zu beheben, vervielfachen wir auch diese Knoten, so dass sich jeweils ein gewisser Anteil an Knoten s_i die Zuleitungen teilt. Damit kön-

nen diese Zuteilungsknoten jeweils nur für einen entsprechenden Anteil an Paaren (s_i, t_j) einen Teil der kürzesten s_i-t_j-Wege überdecken.

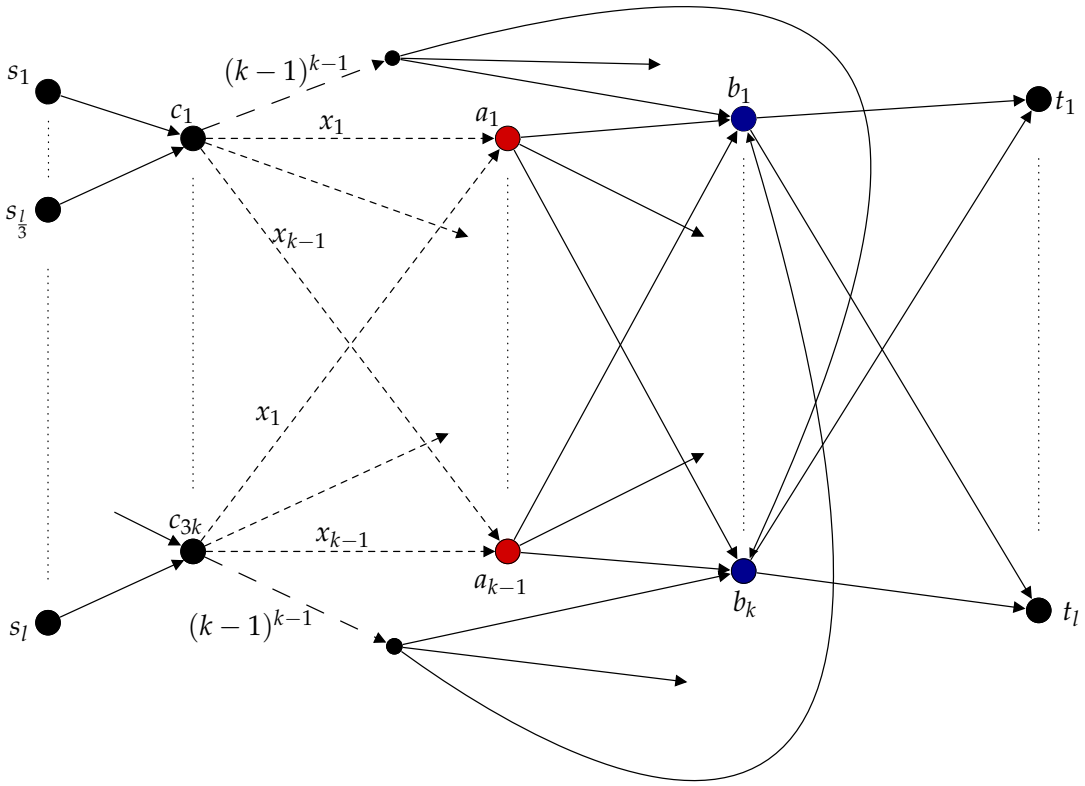

Abbildung 25: Konstruktion mit $3k$ vielen Verteilern

Wie in Abbildung 25 zu erkennen, wurden $3k$ viele Verteiler gewählt. Offenbar kann jeder s_i-t_j-Weg, der von einem der „Vervielfachungsknoten“ auf den Wegen zu a_i oder b_j überdeckt wird, auch vom zugehörigen Knoten $c_{i'}$ überdeckt werden, weshalb die c_i-Knoten an erster Stelle als störend für die Wahl der gewünschten Knoten $a_1, \ldots, a_{k-1}, b_1, \ldots, b_k$ in Frage kommen. Anfangs, und somit auch zu jedem Zeitpunkt des Greedy-Algorithmus, gilt für alle c_i

$$\alpha^M(c_i) l^2 \leq \frac{l}{3k} \cdot l = \frac{1}{3k} l^2$$

also $\alpha^M(c_i) \leq \frac{1}{3k}$. Weiter gilt:

Wurden in den bisherigen (bis zu $k-1$ vielen) Auswahlschritten nur Knoten aus $\{a_1, \ldots, a_{k-1}, b_1, \ldots, b_k\}$ ausgewählt, so bleibt mindestens ein Knoten b_j übrig. Für diesen gilt, weil von jedem s_i zu ihm mindestens $(k-1)^{k-1}$ kür-

zeste Wege unüberdeckt bleiben:

$$\begin{aligned}
\alpha^M(b_j) &\geq \frac{(k-1)^{k-1}}{k^k} = \frac{1}{k} \cdot \left(\frac{k-1}{k}\right)^{k-1} \\
&= \frac{1}{k} \cdot \frac{1}{\left(\frac{k}{k-1}\right)^{k-1}} = \frac{1}{k} \cdot \frac{1}{\left(1+\frac{1}{k-1}\right)^{k-1}} \\
&\overset{(1+\frac{1}{x})^x<e}{>} \frac{1}{k} \cdot \frac{1}{e} > \frac{1}{3k} \\
&\geq \alpha^M(c_i) \text{ für alle } c_i
\end{aligned}$$

Daher wird auch im nächsten Schritt ein Knoten aus $\{a_1, \ldots, a_{k-1}, b_1, \ldots, b_k\}$ ausgewählt.

Ungerichtete Graphen

Es bleibt nur noch der Übergang zu ungerichteten statt gerichteten Graphen. Ersetzt man einfach jeden Pfeil durch eine Kante, so werden Wege zwischen Paaren von Knoten, die schon im gerichteten Fall relevant sind, vorwärts und auch rückwärts gezählt. Wege, die im gerichteten Fall nicht möglich sind und neu hinzukommen, sind solche zwischen Knoten s_i, s_j und Knoten t_i, t_j. Von solchen Paaren gibt es jeweils $\mathcal{O}(l^2)$ viele. Damit diese nicht stören, wird einfach zwischen allen Paaren (s_i, s_j) und allen Paaren (t_i, t_j) jeweils eine Kante hinzugefügt. Auf diese Weise zählen Wege zwischen solchen Paaren nur für die beiden Endknoten. Ansonsten kommen nur für ein k konstant viele Wege hinzu, z.B. solche zwischen Knoten a_i und $a_{i'}$.

4.4.5 Ergebnis

Als Ergebnis dieser ganzen Konstruktionen erhalten wir Satz 4.10.

Satz 4.10: *Die gezeigte Approximationsgüte des Greedy-Algorithmus ItrK ist scharf, es gilt also:*

Sei $k \in \mathbb{N}$ beliebig. Dann gibt es für jedes $\varepsilon > 0$ einen Graphen $G = (V, E)$, in dem für die vom Greedy-Algorithmus gefundene Menge $C \subseteq V$ und die Menge $C^* \subseteq V$, für die $\ddot{\mathrm{B}}(C^*)$ maximal unter allen k-elementigen Teilmengen von V ist, gilt:

$$\ddot{\mathrm{B}}(C) \leq \left(1 - \left(1 - \frac{1}{k}\right)^k + \varepsilon\right) \cdot \ddot{\mathrm{B}}(C^*)$$

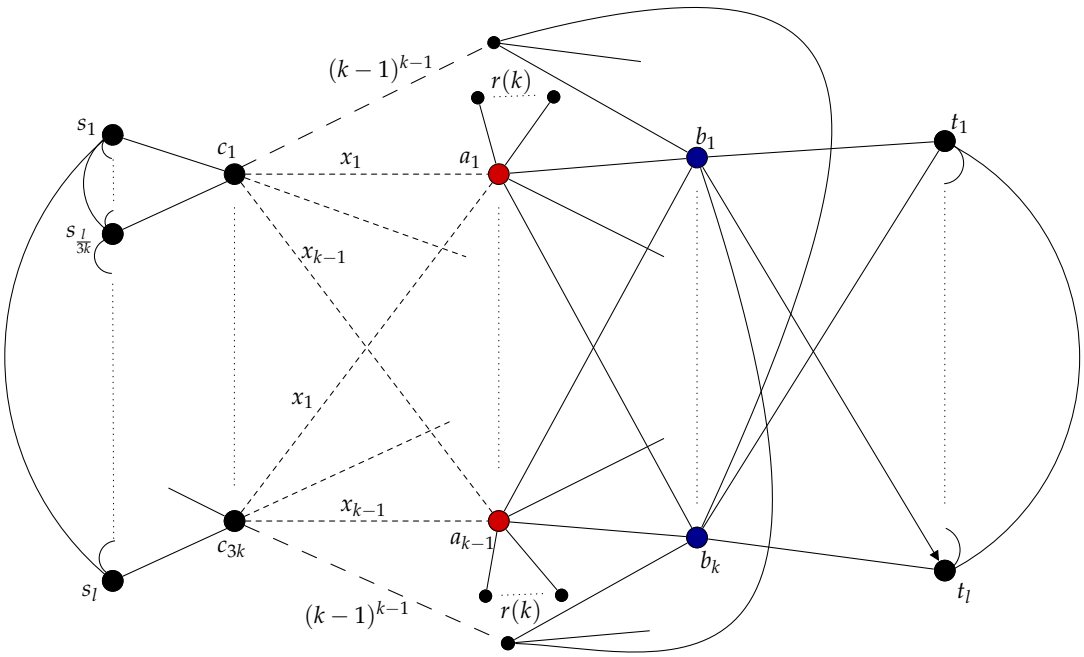

Abbildung 26: Konstruktion des Negativbeispiels für Satz 4.10

Beweis. Für $k = 1$ ist die Behauptung klar. Sei also $k \geq 2$. Wir verwenden die Konstruktion aus Abbildung 26, wobei $r(k) := 3k^3 + \frac{3}{2}k + 6$ sei. Wie wir später sehen, reicht diese Wahl für $r(k)$ aus. Außerdem ist $x_i := k^{k-1-i}(k-1)^{i-1}$, wobei die Zuleitungswege zu den Knoten a_i und b_j wie in Abschnitt 4.4.3 beschrieben aufgebaut seien. Für jeden Knoten $v \in V$ setzen wir $\overline{\mathrm{B}}^M(v) := \tilde{\mathrm{B}}^M(v, v)$ als erreichbaren Zuwachs der Group Betweenness Centrality bei Auswahl von v, wenn vorher im Verlauf des Greedy-Algorithmus schon die Knoten aus $M \subseteq V$ ausgewählt wurden (Es gilt $\overline{\mathrm{B}}^{\emptyset}(v) = \mathrm{B}(v)$). Wie vorher schon gesehen, können wir jeweils schreiben:

$$\overline{\mathrm{B}}^M(v) = \alpha^M(v) \cdot l^2 + \beta^M(v) \cdot l + \gamma^M(v)$$

Ebenfalls haben wir gesehen, dass wir nur die Knoten $a_1, \ldots, a_{k-1}, b_1, \ldots, b_k$ im Verlauf des Algorithmus betrachten müssen, da für relevante Größen von l nur sie als Kandidaten in Frage kommen. Im Folgenden soll gezeigt werden, dass für jeden Knoten a_i gilt, dass $\beta^{\emptyset}(a_i) > \beta^{\emptyset}(b_1) = \ldots = \beta^{\emptyset}(b_k)$ ist. Anschließend werden wir folgern, dass die β-Werte im Zweifelsfall den Ausschlag zur Wahl eines Knotens a_i geben. Zum Schluss wird durch Betrachtung der α-Werte gezeigt, dass der Greedy-Algorithmus die Menge

$$\{a_1, \ldots, a_{k-1}, b_j\}$$

für ein $j \leq k$ liefert.

Betrachten wir die β-Werte für die relevanten Knoten vor dem Start des Algorithmus, wobei mit $x \to y$ jeweils die Art der einzelnen betrachteten Wege angedeutet sei. Dabei sind $c_{i'}^{\to a}$ die Knoten hinter $c_{i'}$ auf dem Weg zu den $a_{j'}$-Knoten, $c_{i'}^{\to b}$ die Knoten hinter $c_{i'}$ auf dem Weg zu den $b_{j'}$-Knoten und $a_{i'}^{\to}$ die $r(k)$ vielen Zusatzknoten an $a_{i'}$. Zählen der Anteile an den kürzesten Wegen nach der Art ihrer Endknoten liefert:

$$
\begin{aligned}
\beta^{\emptyset}(b_1) &= \ldots = \beta^{\emptyset}(b_k) \\
&= 2 \cdot \Bigg(\underbrace{0}_{s_{i'} \to a_{j'}} + \underbrace{1}_{s_{i'} \to b_{j'}} + \underbrace{(k-1) \cdot \frac{1}{k}}_{a_{i'} \to t_{j'}} + \underbrace{1}_{b_{i'} \to t_{j'}} + \underbrace{3k \cdot \frac{1}{k}}_{c_{i'} \to t_{j'}} \\
&\qquad + \underbrace{3k(k^2-k) \cdot \frac{1}{k}}_{c_{i'}^{\to b} \to t_{j'}} + \underbrace{3k \left(k^3 - 2k^2 + k - \frac{k(k-1)}{2} \right) \cdot \frac{1}{k}}_{c_{i'}^{\to a} \to t_{j'}} \\
&\qquad + \underbrace{0}_{s_{i'} \to a_{j'}^{\to}} + \underbrace{(k-1) \cdot r(k) \cdot \frac{1}{k}}_{a_{i'}^{\to} \to t_{j'}} \Bigg) \\
&= \left(3k^3 - \frac{9}{2}k^2 + \frac{3}{2}k + 6 - \frac{1}{k} + \frac{k-1}{k} r(k) \right) \cdot 2 \\
&= 2 \cdot \left(3k^3 - \frac{9}{2}k^2 + \frac{3}{2}k + 6 - \frac{1}{k} \right) + \frac{k-1}{k} 2r(k)
\end{aligned}
$$

$$
\begin{aligned}
\beta^{\emptyset}(a_i) = 2 \cdot \Bigg(& \underbrace{1}_{s_{i'} \to a_{j'}} + \underbrace{k \cdot \frac{k^{k-1-i}(k-1)^{i-1}}{k^{k-1}}}_{s_{i'} \to b_{j'}} + \underbrace{1}_{a_{i'} \to t_{j'}} + \underbrace{0}_{b_{i'} \to t_{j'}} \\
& + \underbrace{3k \cdot \frac{k^{k-1-i}(k-1)^{i-1}}{k^{k-1}}}_{c_{i'} \to t_{j'}} + \underbrace{0}_{c_{i'}^{\to b} \to t_{j'}} \\
& + \underbrace{3k \cdot \left(k^2 - k - i \right)}_{c_{i'}^{\to a} \to t'_j} + \underbrace{r(k)}_{s_{i'} \to a_{j'}^{\to}} + \underbrace{r(k)}_{a_{i'}^{\to} \to t_{j'}} \Bigg)
\end{aligned}
$$

$$
\begin{aligned}
&= 2 \cdot \left(4 \left(\frac{k-1}{k} \right)^{i-1} + 3k^3 - 3k^2 - 3ik + 2 + 2r(k) \right) \\
&\geq 4r(k) = 2 \cdot \left(3k^3 + \frac{3}{2}k + 6 \right) + 2r(k) \\
&> 2 \cdot \left(3k^3 - \frac{9}{2}k^2 + \frac{3}{2}k + 6 - \frac{1}{k} \right) + \frac{k-1}{k} \cdot 2r(k) \\
&= \beta^{\emptyset}(b_1) = \ldots = \beta^{\emptyset}(b_k)
\end{aligned}
$$

Die für β relevanten Wege starten oder enden jeweils in einem Knoten s_i oder t_j. Jeder solche Weg geht durch höchstens einen Knoten aus $a_1, \ldots, a_{k-1}$. Wird daher im Verlauf des Greedy-Algorithmus ein Knoten a_i ausgewählt, so ändert sich der β-Wert für die anderen Knoten $a_{i'}$, $i \neq i'$ nicht. Andererseits können sich die β-Werte niemals erhöhen, insbesondere nicht für die Knoten b_j. Daher gilt für $M \subseteq \{a_1, \ldots, a_{k-1}\}, a_i \notin M, j \in \{1, \ldots, k\}$:

$$
\begin{aligned}
\beta^M(a_i) &= \beta^{\emptyset}(a_i) \\
&> \beta^{\emptyset}(b_j) \geq \beta^M(b_j)
\end{aligned}
$$

Also wird bei identischen α-Werten stets der Knoten a_i vor einem b_j bevorzugt. Wir wollen jetzt sehen, dass in den ersten $k-1$ Auswahlschritten die Knoten der Liste $(a_1, \ldots, a_{k-1})$ in dieser Reihenfolge gewählt werden. Dazu zeigen wir:

Wurde in den bisherigen $i-1$ Schritten ($i \leq k-1$) die Knotenmenge $M_i := \{x_1, \ldots, x_{i-1}\}$ ausgewählt, so gilt

$$
\alpha^{M_i}(b_1) = \ldots = \alpha^{M_i}(b_k) = 2\frac{(k-1)^{i-1}}{k^i}
$$

und a_i wird gewählt.

Für $i = 1$ stimmt die Behauptung offensichtlich, wegen

$$
\alpha^{\emptyset}(a_1) = 2\frac{1}{k} = \alpha^{\emptyset}(b_1) = \ldots = \alpha^{\emptyset}(b_k)
$$

Die Behauptung gelte also für ein $i < k-1$. Nach der Auswahl von a_i gilt wegen der gleichmäßigen Verteilung der $s_{i'}$-$t_{j'}$-Wege durch a_i auf die b_j:

$$\begin{aligned}
\alpha^{M_{i+1}}(b_1) &= \ldots = \alpha^{M_{i+1}}(b_k) \\
&= \alpha^{M_i}(b_1) - \frac{1}{k}\alpha^{M_i}(a_i) = 2\frac{(k-1)^{i-1}}{k^i} - \frac{1}{k} \cdot 2 \cdot \frac{(k-1)^{i-1}}{k^i} \\
&= 2\frac{(k-1)^{i-1}}{k^{i+1}}\,(k-1) \\
&= 2\frac{(k-1)^i}{k^{i+1}}
\end{aligned}$$

Andererseits gilt auch:

$$\begin{aligned}
\alpha^{M_{i+1}}(a_{i+1}) &= \alpha^{\emptyset}(a_{i+1}) \\
&= 2 \cdot \frac{k \cdot k^{k-1-(i+1)}(k-1)^{(i+1)-1}}{k^k} = 2\frac{k^{k-1-i}(k-1)^i}{k^k} \\
&= 2\frac{(k-1)^i}{k^{i+1}}
\end{aligned}$$

Da die α-Werte der a_j für $j > i+1$ geringer sind, wird also im Schritt $i+1$ der Knoten a_{i+1} gewählt, wie behauptet.

Damit bleibt nur der letzte Schritt übrig, in dem ein Knoten b_j ausgewählt wird, o.E. b_1. Der Greedy-Algorithmus liefert also als Ausgabe die Menge $C = \{a_1, \ldots, a_{k-1}, b_1\}$. Für diese gilt mit nur von k abhängigen Zahlen β_1, γ_1:

$$\begin{aligned}
\ddot{\mathrm{B}}(C) &= 2l^2\left(\frac{(k-1)^{k-1}}{k^k} + \sum_{i=1}^{k-1}\frac{(k-1)^{i-1}}{k^i}\right) + \beta_1 l + \gamma_1 \\
&= 2l^2\left(\sum_{i=1}^{k}\frac{(k-1)^{i-1}}{k^i}\right) + \beta_1 l + \gamma_1 \\
&= 2l^2\left(\frac{1}{k}\sum_{i=0}^{k-1}\left(\frac{k-1}{k}\right)^i\right) + \beta_1 l + \gamma_1 \\
&= 2l^2 \cdot \frac{1}{k} \cdot \frac{1-\left(\frac{k-1}{k}\right)^k}{1-\frac{k-1}{k}} + \beta_1 l + \gamma_1 \\
&= 2l^2 \cdot \frac{1-\left(\frac{k-1}{k}\right)^k}{k-(k-1)} + \beta_1 l + \gamma_1 \\
&= 2l^2 \cdot \left(1-\left(1-\frac{1}{k}\right)^k\right) + \beta_1 l + \gamma_1
\end{aligned}$$

Durch Setzen von $C^* := \{b_1, \ldots, b_k\}$ lässt sich in einer optimalen Lösung aber jeder kürzeste s_i-t_j-Weg besetzen. Damit gilt

$$\ddot{\mathrm{B}}(C^*) = 2l^2 + \beta_2 l + \gamma_2$$

wobei β_2 und γ_2 wiederum nur von k abhängig sind. Wir erhalten

$$\begin{aligned}
\frac{\ddot{\mathrm{B}}(C)}{\ddot{\mathrm{B}}(C^*)} &= \frac{2l^2 \cdot \left(1 - \left(1 - \frac{1}{k}\right)^k\right) + \beta_1 l + \gamma_1}{2l^2 + \beta_2 l + \gamma_2} \\
&= \frac{\left(1 - \left(1 - \frac{1}{k}\right)^k\right) + \frac{\beta_1}{2l} + \frac{\gamma_1}{2l^2}}{1 + \frac{\beta_2}{2l} + \frac{\gamma_2}{2l^2}} \\
&\longrightarrow 1 - \left(1 - \frac{1}{k}\right)^k \text{ für } l \to \infty
\end{aligned}$$

Sei $\varepsilon > 0$ vorgegeben. Dann gibt es einen Wert l, so dass

$$\frac{\ddot{\mathrm{B}}(C)}{\ddot{\mathrm{B}}(C^*)} \in \left[1 - \left(1 - \frac{1}{k}\right)^k - \varepsilon, 1 - \left(1 - \frac{1}{k}\right)^k + \varepsilon\right]$$

Insbesondere gilt für dieses l

$$\ddot{\mathrm{B}}(C) \leq \left(1 - \left(1 - \frac{1}{k}\right)^k + \varepsilon\right) \cdot \ddot{\mathrm{B}}(C^*)$$

□

Bemerkung 4.11: Wir zählen die Anzahl der Knoten in der verwendeten Konstruktion (Abbildung 26). Es ist:

$$\begin{aligned}
|V| = {} & \underbrace{2l}_{s_i, t_j} + \underbrace{2k - 1}_{a_i, b_j} \\
& + \underbrace{3k \cdot \left(1 + (k^2 - k) + \left(k^3 - 2k^2 + k - \frac{k(k+1)}{2}\right)\right)}_{\text{Zuführungswege}} \\
& + \underbrace{(k-1) \cdot \left(3k^3 + \frac{3}{2}k + 6\right)}_{\text{Hilfsknoten an } a_i} \\
= {} & 2l + 6k^4 - \frac{15}{2}k^3 + 3k^2 + \frac{19}{2}k - 7
\end{aligned}$$

Es gibt also schon im Grundgerüst ohne die je l vielen s_i und t_j mindestens $\mathcal{O}(k^4)$ viele Knoten. Für einen Graphen, der nahe an den gezeigten Approximationsfaktor herankommt, gilt also mit $n = |V|$ auf jeden Fall $k \leq \sqrt[4]{n}$. Da es normalerweise gerade die Aufgabe ist, nach wenigen kritischen Knoten zu suchen, ist das aber ein realistisches Verhältnis. Insbesondere basiert die Konstruktion nicht darauf, dass k im Verhältnis zu n sehr groß wird.

Auch dem Faktor $1 - \frac{1}{e}$ kommt man beliebig nahe, wenn k und l groß genug gewählt werden: Für ein genügend großes k erreicht man

$$1 - \left(1 - \frac{1}{k}\right)^k \leq 1 - \frac{1}{e} + \frac{\varepsilon}{2}$$

für ein beliebiges $\varepsilon > 0$. Ist zusätzlich l groß genug, so gilt auch:

$$\frac{\ddot{\mathrm{B}}(C)}{\ddot{\mathrm{B}}(C^*)} \leq 1 - \left(1 - \frac{1}{k}\right)^k + \frac{\varepsilon}{2} \leq 1 - \frac{1}{e} + \varepsilon$$

5 Zusammenfassung und Ausblick auf offene Probleme

Als Abschluss der Arbeit sollen noch zwei offene Probleme in Bezug auf die behandelten Themen genannt werden. Anschließend sollen die erarbeiteten Ergebnisse zusammengefasst werden.

5.1 Offene Probleme

5.1.1 Dynamisches Update der Shortest Path Betweenness Centrality

Eine offene Frage, die auch Brandes in [Bra08] als interessant nennt, ist die nach einer Aktualisierung der Zentralitätswerte aller Knoten im Graphen bei geringer Änderung desselben. Wir beschränken uns hier auf das Hinzufügen einer Kante zum Graphen.

Wird eine vorher nicht vorhandene Kante zwischen zwei Knoten s und t hinzugefügt, so ist jetzt die Direktverbindung der beiden Knoten der einzige kürzeste s-t-Weg. Daher fällt für alle Knoten v, die vorher auf mindestens einem kürzesten s-t-Weg lagen, der Beitrag $\frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ zur Shortest Path Betweenness weg. Dies können im Extremfall alle anderen Knoten des Graphen sein. Umgekehrt können durch die hinzugefügte Kante neue kürzeste Wege führen, auf denen auch weitere Knoten liegen. Betrachten wir den Graphen G, bei dem alle n Knoten auf einem Pfad liegen. Man überlegt sich, dass die Shortest Path Betweenness der Knoten zur Mitte des Pfades hin zunimmt. Aus Symmetriegründen haben maximal zwei Knoten dieselbe Shortest Path Betweenness. Fügt man eine Kante so zum Pfad hinzu, dass aus diesem ein Kreis wird, so haben dann natürlich alle Knoten dieselbe Shortest Path Betweenness Centrality. Der Zentralitätswert hat sich damit für mindestens $n-2$ Knoten geändert.

Also wird die Laufzeit eines Algorithmus, der die Shortest Path Betweenness Centrality aller Knoten bei Hinzunahme einer Kante aktualisiert, mindestens bei $\mathcal{O}(n)$ liegen. Umgekehrt sollte die Laufzeit aber besser als $\mathcal{O}(nm)$

sein, da sonst die Neuberechnung aller Werte ebenso schnell oder schneller wäre.

Gäbe es einen Algorithmus zum Update in $\mathcal{O}(n)$, so könnte dieser sogar den Algorithmus zur schnellen Berechnung aller Werte ersetzen, indem man mit einem kantenlosen Graphen startet und die Kanten schrittweise hinzufügt. Werden die Werte nach jedem Hinzufügen einer Kante aktualisiert, so ist nach dem Hinzufügen aller Kanten die Shortest Path Betweenness Centrality aller Knoten berechnet. Da es m Kanten hinzuzunehmen gibt, besitzt der resultierende Algorithmus eine Laufzeit von $\mathcal{O}(nm)$.

Eine Idee, bei dem Versuch einen Algorithmus zum dynamischen Update zu entwickeln, wäre es, bei der Grundidee des Algorithmus von Brandes zu bleiben, und diese für das dynamische Update anzupassen. Hierbei gibt es allerdings zwei Flaschenhälse zu beseitigen. Einerseits müssen die Daten für die Single-Source-Shortest-Path-Probleme von allen Knoten aus schneller als in $\mathcal{O}(nm)$ aktualisiert werden. Andererseits bleibt danach noch der zweite Schritt, das Aufsammeln der $\delta_{s,\bullet}$-Werte zu jedem Knoten s hin. Auch dies muss beim dynamischen Update schneller als in $\mathcal{O}(nm)$ (für alle s zusammen) gelöst werden.

5.1.2 Weitere Untersuchung von KPP-Com

Wir haben gezeigt, dass der Greedy-Algorithmus für das KPP-Com-Problem einen Approximationsfaktor von $1 - \frac{1}{e}$ aufweist, und diese Abschätzung sogar scharf ist. Es stellt sich jetzt die weitere Frage, ob es einen anderen Polynomialzeitalgorithmus gibt, der KPP-Com besser approximiert.

Wie in Abschnitt 4.4 erwähnt, ist es in Polynomialzeit nicht möglich, Maximum-Coverage mit einem besseren Approximationsfaktor in Polynomialzeit zu approximieren ([Fei98]). Möglicherweise könnte dies auch für KPP-Com gelten, da dieses Problem einen Spezialfall von Maximum-Coverage darstellt. Durch Reduktion von Maximum-Coverage-Instanzen auf KPP-Com-Instanzen ließe sich dies untersuchen.

5.2 Fazit

Zum Abschluss sollen noch einmal die Ergebnisse dieser Arbeit rekapituliert werden. In Kapitel 2 wurde die Shortest Path Betweenness Centrality vorgestellt. Anschließend wurde eine verallgemeinerte Variante entwickelt und der bestehende Algorithmus von Brandes zur Berechnung dieser Variante modifiziert. In Kapitel 3 wurde dann die Group Betweenness Centrality eingeführt

und der Algorithmus aus [PED07b] und [PED07a] zu ihrer Berechnung erklärt. Dieser braucht nach einmaligem Preprocessing in $\mathcal{O}(n^3)$ eine Laufzeit von $\mathcal{O}(k^3)$ zur Berechnung der Group Betweenness einer Gruppe von k Knoten.

Wir haben mit Hilfe unserer verallgemeinerten Shortest Path Betweenness Centrality gezeigt, wie sich der Algorithmus leicht nutzen lässt, um auch die Group Betweenness Centrality von Gruppen von Kanten zu berechnen. Durch diesen ersten Ansatz wurde leider das Preprocessing auf $\mathcal{O}(nm^2)$ verlangsamt. Da auch die Laufzeit des zugehörigen Greedy-Algorithmus zur Suche nach einer k-elementigen Menge von Knoten mit hoher Group Betweenness von $\mathcal{O}(kn^2)$ auf $\mathcal{O}(km^2)$ verschlechtert wurde, haben wir nach einer besseren Technik gesucht. Durch eine stärkere Modifikation des Algorithmus für Knotengruppen wurde die Idee des Inzidenzgraphen im neuen Algorithmus implizit genutzt. Damit war es möglich, wieder eine Laufzeit von $\mathcal{O}(n^3)$ für das Preprocessing und $\mathcal{O}(kn^2)$ für die Greedy-Variante zu erreichen.

In Kapitel 4 wurde das KPP-Com-Problem behandelt, bei dem es darum geht, eine k-elementige Menge von Knoten zu finden, die eine möglichst hohe Group Betweenness Centrality aufweist. Für den in [PED07b] und [PED07a] entwickelten, dort aber nur experimentell untersuchten Greedy-Algorithmus zur Approximation dieses Problems, konnte eine Approximationsgüte von $1 - \left(1 - \frac{1}{k}\right)^k$ – in Abhängigkeit von k – bzw. von $1 - \frac{1}{e}$ nachgewiesen werden. In [DEPZ07] wird ebenfalls eine Approximationsgüte von $1 - \frac{1}{e}$ für diesen Algorithmus gezeigt. Dieser unveröffentlichte Artikel wird allerdings ausschließlich in der nicht leicht zu findenden Ankündigung eines Workshop-Vortrags erwähnt ([Puz08]), der erst gegen Ende des Verfassens dieser Arbeit durch Zufall entdeckt wurde. [DEPZ07] lag daher zur Erstellungszeit der vorliegenden Arbeit nicht vor. Es ist deswegen auch nicht möglich, zu sagen, welche Technik die Autoren für ihren Beweis der Approximationsgüte verwendet haben.

Wir haben KPP-Com und den Greedy-Algorithmus ItrK auf das Maximum-Coverage-Problem und einen Greedy-Algorithmus zu dessen Approximation zurückgeführt. Abschließend konnte mit Worst-Case-Konstruktionen gezeigt werden, dass der Greedy-Algorithmus tatsächlich keinen besseren Approximationsfaktor als $1 - \frac{1}{e}$ aufweist.

Anhang

A Algorithmen

A.1 Breitensuche

Algorithmus 10 löst in einem Graphen $G = (V, E)$ das Single-Source-Shortest-Path-Problem für ein $s \in V$ in $\mathcal{O}(m)$ durch die *Breitensuche*, vergleiche Abschnitt 1.4.3.

Algorithmus 10 : Breitensuche BFS($G = (V, E), s$)

Eingabe : $G = (V, E), s \in V$
Ausgabe : für alle $v \in V$ den Abstand $d(s, v) = d[v]$, die Anzahl $\sigma_{s,v} = \sigma[v]$ und die Vorgänger $P[v]$ von v auf kürzesten Wegen von s aus

```
1  ∀_{w∈V} P[w] := leere Liste;
2  σ[s] := 1; ∀_{t∈V\{s}} σ[t] := 0;
3  d[s] := 0; ∀_{t∈V\{s}} d[t] := −1;
4  Q := leere Schlange;
5  enqueue s → Q;
6  while Q nicht leer do
7      dequeue v ← Q;
8      foreach [v, w] ∈ E do
9          if d[w] < 0 then
10             enqueue w → Q;
11             d[w] := d[v] + 1;
12         if d[w] = d[v] + 1 then
13             σ[w] := σ[w] + σ[v];
14             append v → P[w];
15 return (d, σ, P);
```

A.2 Naive Berechnung der Shortest Path Betweenness Centrality

Algorithmus 11 berechnet in einem Graphen $G = (V, E)$ für alle Knoten $v \in V$ die Shortest Path Betweenness Centrality, siehe Abschnitt 2.4.2. Er verwendet die Breitensuche aus Algorithmus 10.

Algorithmus 11 : naive Berechnung der Shortest Path Betweenness Centrality

Eingabe : $G = (V, E)$
Ausgabe : für alle $v \in V$ die Shortest Path Betweenness Centrality $\mathrm{B}(v) = \mathrm{B}[v]$

```
1  ∀_{v∈V} B[v] := 0;
2  foreach s ∈ V do
3      (d_s, σ_s, P_s) := BFS(G);
4      foreach v ∈ V do
5          d[s, v] := d_s[v];
6          σ[s, v] := σ_s[v];
7  foreach v ∈ V do
8      foreach s, t ∈ V, s ≠ t do
9          if d[s, t] = d(s, v) + d(v, t) then
10             B[v] := B[v] + σ[s,v]·σ[v,t] / σ[s,t];
11 return B;
```

A.3 Schnelle Berechnung der Shortest Path Betweenness Centrality

Algorithmus 12 berechnet in einem Graphen die Shortest Path Betweenness Centrality für alle Knoten auf einmal in $\mathcal{O}(nm)$. Er ist die vollständige Version von Algorithmus 2.

Algorithmus 12 : Shortest Path Betweenness Centrality

Eingabe : $G = (V, E)$
Ausgabe : $\mathrm{B}(v) = \mathrm{B}[v]$ für alle $v \in V$

```
1  ∀_{v∈V} B[v] := 0;
2  foreach s ∈ V do
3      S := leerer Stack; ∀_{w∈V} P[w] := leere Liste;
4      σ[s] := 1; ∀_{t∈V\{s}} σ[t] := 0; d[s] := 0; ∀_{t∈V\{s}} d[t] := −1;
5      Q := leere Schlange; enqueue s → Q;
6      while Q nicht leer do
7          dequeue v ← Q; push v → S;
8          foreach [v, w] ∈ E do
9              if d[w] < 0 then
10                 enqueue w → Q;
11                 d[w] := d[v] + 1;
12             if d[w] = d[v] + 1 then
13                 σ[w] := σ[w] + σ[v];
14                 append v → P[w];
15     ∀_{v∈V} δ[v] := 0;
16     while S nicht leer do
17         pop w ← S; δ[w] := δ[w] + 1;
18         for v ∈ P[w] do δ[v] := δ[v] + (σ[v]/σ[w]) · δ[w];
19         if w ≠ s then B[w] := B[w] + δ[w];
20     B[s] := B[s] + |V| − 1;
21 return B;
```

A.4 Schnelle Berechnung der Shortest Path Betweenness Centrality für Kanten

Algorithmus 13 berechnet in einem Graphen die Shortest Path Betweenness Centrality für alle Kanten auf einmal in $\mathcal{O}(nm)$. Er ist die vollständige Version von Algorithmus 2.

Algorithmus 13 : Shortest Path Betweenness Centrality für Kanten

Eingabe : $G = (V, E)$
Ausgabe : $\mathrm{B}(e) = \mathrm{B}[e]$ für alle $e \in E$

```
1  ∀_{e∈E} B[e] := 0;
2  foreach s ∈ V do
3      S := leerer Stack;
4      ∀_{w∈V} P[w] := leere Liste;
5      σ[s] := 1; ∀_{t∈V\{s}} σ[t] := 0;
6      d[s] := 0; ∀_{t∈V\{s}} d[t] := −1;
7      Q := leere Schlange; enqueue s → Q;
8      while Q nicht leer do
9          dequeue v ← Q; push v → S;
10         foreach [v, w] ∈ E do
11             if d[w] < 0 then
12                 enqueue w → Q;
13                 d[w] := d[v] + 1;
14             if d[w] = d[v] + 1 then
15                 σ[w] := σ[w] + σ[v];
16                 append v → P[w];
17     ∀_{v∈V} δ[v] := 0;
18     while S nicht leer do
19         pop w ← S;
20         δ[w] := δ[w] + 1;
21         for v ∈ P[w] do
22             c := (σ[v] / σ[w]) · δ[w];
23             δ[v] := δ[v] + c;
24             B[[v, w]] := B[[v, w]] + c;
25 return B;
```

A.5 Schnelle Berechnung der verallgemeinerten Shortest Path Betweenness Centrality

Algorithmus 14 berechnet in einem Graphen die verallgemeinerte Shortest Path Betweenness Centrality für alle Knoten auf einmal in $\mathcal{O}(|S|m)$.

Algorithmus 14 : verallgemeinerte Shortest Path Betweenness Centrality

Eingabe : $G = (V, E), S, T$
Ausgabe : $\mathrm{B}^{S,T}(v) = \mathrm{B}[v]$ für alle $v \in V$

1 $\forall_{v \in V}\, \mathrm{B}[v] := 0$;
2 **foreach** $s \in S$ **do**
3 $S :=$ leerer Stack;
4 $\forall_{w \in V} P[w] :=$ leere Liste;
5 $\sigma[s] := 1$; $\forall_{t \in V \setminus \{s\}} \sigma[t] := 0$;
6 $d[s] := 0$; $\forall_{t \in V \setminus \{s\}} d[t] := -1$;
7 $Q :=$ leere Schlange; enqueue $s \rightarrow Q$;
8 **while** Q *nicht leer* **do**
9 dequeue $v \leftarrow Q$;
10 push $v \rightarrow S$;
11 **foreach** $[v, w] \in E$ **do**
12 **if** $d[w] < 0$ **then**
13 enqueue $w \rightarrow Q$;
14 $d[w] := d[v] + 1$;
15 **if** $d[w] = d[v] + 1$ **then**
16 $\sigma[w] := \sigma[w] + \sigma[v]$;
17 append $v \rightarrow P[w]$;
18 $\forall_{v \in V} \delta[v] := 0$;
19 **while** S *nicht leer* **do**
20 pop $w \leftarrow S$;
21 **if** $w \in T$ **then** $\delta[w] := \delta[w] + 1$;
22 **for** $v \in P[w]$ **do** $\delta[v] := \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot \delta[w]$;
23 **if** $w \neq s$ **then** $\mathrm{B}[w] := \mathrm{B}[w] + \delta[w]$;
24 $\mathrm{B}[s] := \mathrm{B}[s] + |T \setminus \{s\}|$;
25 **return** B;

A.6 Greedy-Algorithmus zur Approximation einer Gruppe von Kanten mit hoher Group Betweenness Centrality

Algorithmus 15 findet eine Gruppe von Kanten mit hoher Group Betweenness Centrality. Er benötigt Algorithmus 6 zur Berechnung der neuen Werte in Zeile 10.

Algorithmus 15 : Greedy-Algorithmus ItrK' für Gruppen von Kanten

Eingabe : G, k und die vorberechneten Werte $d, \sigma, \tilde{\mathrm{B}}$
Ausgabe : $C \subseteq E$ mit $|C| = k$ und möglichst hohem Wert $\ddot{\mathrm{B}}(C)$

1 $M := \emptyset$;
2 $\forall_{x,y \in V} \sigma_{x,y}^{M} := \sigma_{x,y}$;
3 $\forall_{x,y \in V} \tilde{\mathrm{B}}^{M}(x,y) := \tilde{\mathrm{B}}(x,y)$;
4 $\ddot{\mathrm{B}} := 0$;
5 **for** $i := 1$ **to** k **do**
6 wähle $e_i = [u_i, v_i] \in E \setminus M$ mit $\tilde{\mathrm{B}}^{M}(u_i, v_i) + \tilde{\mathrm{B}}^{M}(v_i, u_i)$ maximal;
7 $\ddot{\mathrm{B}} := \ddot{\mathrm{B}} + \tilde{\mathrm{B}}^{M}(u_i, v_i) + \tilde{\mathrm{B}}^{M}(v_i, u_i)$;
8 **foreach** $x, y \in V$ **do**
9 $\sigma_{x,y}^{M \cup \{e_i\}} := \sigma_{x,y}^{M} - \sigma_{x,y}^{M}(e_i)$;
10 berechne $\tilde{\mathrm{B}}^{M \cup \{e_i\}}(x,y)$;
11 $M := M \cup \{e_i\}$;
12 **return** M;

B Wichtige Definitionen

$\sigma_{s,t}$ Anzahl der kürzesten s-t-Wege

$\sigma_{s,t}(v)$ Anzahl der kürzesten s-t-Wege, auf denen $v \in V$ liegt

$\delta_{s,\bullet}(v) := \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$

$\mathbf{B}(v) := \sum_{s,t \in V | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} = \sum_{s \in V} \delta_{s,\bullet}(v)$ Shortest Path Betweenness Centrality des Knotens $v \in V$

$\sigma_{s,t}(e)$ Anzahl der kürzesten s-t-Wege, auf denen $e \in E$ liegt

$\delta_{s,\bullet}(e) := \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}}$

$\mathbf{B}(e) := \sum_{s,t \in V | s \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}} = \sum_{s \in V} \delta_{s,\bullet}(e)$ Shortest Path Betweenness Centrality der Kante $e \in E$

$\delta^{T}_{s,\bullet}(v) := \sum_{t \in T | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ für $T \subseteq V$

$\mathbf{B}^{S,T}(v) := \sum_{s \in S} \sum_{t \in t | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} = \sum_{s \in S} \delta^{T}_{s,\bullet}(v)$ verallgemeinerte Shortest Path Betweenness Centrality mit Quellmenge $S \subseteq V$ und Zielmenge $T \subseteq V$ von $v \in V$

$\ddot{\sigma}_{s,t}(C) := |\{P \mid P \text{ ist kürzester } s\text{-}t\text{-Weg in } G \wedge \exists_{v \in C} v \in P\}|$ Anzahl der kürzesten s-t-Wege, die Knoten aus $C \subseteq V$ benutzen

$\ddot{\mathbf{B}}(C) := \sum_{s,t \in V | s \neq t} \frac{\ddot{\sigma}_{s,t}(C)}{\sigma_{s,t}}$ Group Betweenness Centrality von $C \subseteq V$

$\tilde{\sigma}_{s,t}(S)$ Anzahl der kürzesten s-t-Wege, die die Knoten der geordneten Liste S in der richtigen Reihenfolge durchlaufen

$\tilde{\mathbf{B}}(S) := \sum_{s,t \in V | s \neq t} \frac{\tilde{\sigma}_{s,t}(S)}{\sigma_{s,t}}$ Path Betweenness Centrality der Liste S von Knoten

$\ddot{\sigma}_{s,t}(C) := |\{P \mid P \text{ ist kürzester } s\text{-}t\text{-Weg in } G \wedge \exists_{e \in C} e \in P\}|$ Anzahl der kürzesten s-t-Wege, die Kanten aus $C \subseteq E$ benutzen

$\ddot{\mathbf{B}}(C) := \sum_{s,t \in V | s \neq t} \frac{\ddot{\sigma}_{s,t}(C)}{\sigma_{s,t}}$ Group Betweenness Centrality von $C \subseteq E$

C Abbildungsverzeichnis

Liste der Algorithmen

D Literaturverzeichnis

[ACEP09] ARULSELVAN, A., C.W. COMMANDER, L. ELEFTERIADOU und P.M. PARDALOS: *Detecting critical nodes in sparse graphs*. Computers and Operations Research, 36(7):2193–2200, 2009.

[BE05] BRANDES, ULRIK und THOMAS ERLEBACH (Herausgeber): *Network Analysis: Methodological Foundations*, Band 3418 der Reihe *Lecture Notes in Computer Science*. Springer, 2005.

[BE06] BORGATTI, S.P. und M.G. EVERETT: *A graph-theoretic perspective on centrality*. Social networks, 28(4):466–484, 2006.

[Bra01] BRANDES, U.: *A faster algorithm for Betweenness Centrality*. Journal of Mathematical Sociology, 25(2):163–177, 2001.

[Bra08] BRANDES, U.: *On variants of Shortest-Path Betweenness Centrality and their generic computation*. Social Networks, 30(2):136–145, 2008.

[CV07] CALDARELLI, G. und A. VESPIGNANI: *Large scale structure and dynamics of complex networks: from information technology to finance and natural science*. World Scientific, 2007.

[DEPZ07] DOLEV, S., Y. ELOVICI, R. PUZIS und P. ZILBERMAN: *Incremental deployment of network monitors based on group betweenness centrality*. submitted for publication, 2007.

[EB99] EVERETT, M.G. und S.P. BORGATTI: *The centrality of groups and classes*. Journal of Mathematical Sociology, 23:181–202, 1999.

[Fei98] FEIGE, U.: *A threshold of* $\ln n$ *for approximating set cover*. Journal of the ACM (JACM), 45(4):634–652, 1998.

[FFSN04] FLOM, P.L., S.R. FRIEDMAN, S. STRAUSS und A. NEAIGUS: *A new measure of linkage between two sub-networks*. Connections, 26(1):62–70, 2004.

[Fre77] FREEMAN, L.C.: *A set of measures of centrality based on betweenness*. Sociometry, 40(1):35–41, 1977.

[GN02] GIRVAN, M. und M. E. J. NEWMAN: *Community structure in social and biological networks*. Proceedings of the National Academy of Sciences, 99(12):7821, 2002.

[JM08] JANSEN, KLAUS und MARIAN MARGRAF: *Approximative Algorithmen und Nichtapproximierbarkeit*. de Gruyter, 2008.

[KN05] KRUMKE, S.O. und H. NOLTEMEIER: *Graphentheoretische Konzepte und Algorithmen*. Teubner, 2005.

[New01a] NEWMAN, M. E. J.: *Scientific collaboration networks. I. Network construction and fundamental results*. Physical Review E, 64:016131, 2001.

[New01b] NEWMAN, M. E. J.: *Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality*. Physical Review E, 64:016132, 2001.

[New04] NEWMAN, M. E. J.: *Fast algorithm for detecting community structure in networks*. Physical Review E, 69:066133, 2004.

[New05] NEWMAN, M. E. J.: *A measure of betweenness centrality based on random walks*. Social Networks, 27(1):39–54, 2005.

[NG04] NEWMAN, M. E. J. und M. GIRVAN: *Finding and evaluating community structure in networks*. Physical Review E, 69:026113, 2004.

[PED07a] PUZIS, R., Y. ELOVICI und S. DOLEV: *Fast algorithm for successive computation of Group Betweenness Centrality*. Physical Review E, 76(5):056709, 2007.

[PED07b] PUZIS, R., Y. ELOVICI und S. DOLEV: *Finding the most prominent group in complex networks*. AI Communications, 20(4):287–296, 2007.

[Puz08] PUZIS, R.: *Group Betweenness Centrality: Efficient Computations and Applications*, 2008. `http://www.cri.haifa.ac.il/events/2008/0504_graph/cri-puzis.pdf`.

[Wag03] WAGNER, K.W.: *Theoretische Informatik: Eine kompakte Einführung*. Springer, 2003.

[WS03] WHITE, S. und P. SMYTH: *Algorithms for estimating relative importance in networks*. In: *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, Seiten 266–275. ACM New York, NY, USA, 2003.

E Erklärung der selbständigen Bearbeitung

Hiermit versichere ich, die vorliegende Arbeit selbständig verfasst und dabei keine anderen Hilfsmittel und Quellen als die angegebenen benutzt zu haben.

Würzburg, den _____ , _____
(Martin Fink)