
# Untersuchung, Implementierungen und Bewertung von Graph-Metriken

## Studienarbeit

im Studiengang Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart, Campus Horb am Neckar

von

**Benedict Weichselbaum**

28. Januar 2021

| | |
|---|---|
| **Bearbeitungszeitraum** | 28.09.2020 - 31.05.2021 |
| **Matrikelnummer, Kurs** | 6275457, TINF2018 |
| **Betreuer & Gutachter** | Prof. Dr. ing. Olaf Herden |

# Erklärung

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Thema *Graphen: Metriken und Ähnlichkeit* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.
Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Nürnberg, 28. Januar 2021

---

Benedict Martin Weichselbaum

## Abstract

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

# 1 Einleitung

## 1.1 Motivation für die Studienarbeit

Graphen sind einer der wichtigsten Datenstrukturen der Informatik. Warum kann man das sagen? In seinem Buch „Algorithmische Graphentheorie“ nennt Volker Turau, Professor an der Universität Hamburg-Harburg, den Grund dafür:

> *Graphen sind die in der Informatik am häufigsten verwendete Abstraktion. Jedes System, welches aus diskreten Zuständen oder Objekten und Beziehungen zwischen diesen besteht, kann als Graph modelliert werden.* [Tur04]

Diese netzartigen Strukturen können dabei die verschiedensten Konstrukte repräsentieren. Dazu zählen Straßennetze, Computernetzwerke, elektrische Schaltungen aber auch zum Beispiel chemische Moleküle. [Tit19]

Um Graphen zu beschreiben und zu charakterisieren, haben sich über die Zeit zahlreiche Metriken, bzw. Eigenschaften für diese herausgebildet („graph properties“ [Lov12]). Das heißt, einem Graphen können gewisse Kennzahlen zugeordnet werden, die ihn auszeichnen. Auch diese Metriken sind, wie die Graphen selbst, oft praktisch anwendbar. Zum Beispiel in der Untersuchung von Netzwerken [EK13].

Diese Studienarbeit soll nun diese Metriken genauer untersuchen. Hierbei ist es zunächst wichtig die verschiedenste Metriken vorzustellen und zu erläutern. Dabei ist es auch wichtig herauszufinden, wie verbreitet diese Metriken sind und inwieweit die jeweiligen Kennzahlen zu bewerten sind. Des Weiteren soll auf Basis der Metriken auch der Begriff der Ähnlichkeit von Graphen aufgegriffen werden.

Neben einer theoretischen Betrachtung der Graphmetriken soll auch eine Implementierung stattfinden. Es ist dabei das Ziel, mithilfe von Graphdatenbanken die jeweiligen Metriken umzusetzen und diese miteinander zu Vergleichen.

In einem Weiteren Teil ist außerdem noch darauf einzugehen, welche Anwendung die gezeigten Metriken haben, um den praktischen Nutzen der Thematik aufzuzeigen.

## 1.2 Fragestellungen

Auf Basis dieser Motivation können nun auch die konkreten Fragestellungen formuliert werden, die diese Arbeit betrachten soll. Insgesamt sollen vier wissenschaftliche Fragen

beantwortet werden.

1. Welche Graph-Metriken gibt es und wie sind diese definiert und zu kategorisieren?

Hierzu gehört, wie bereits erwähnt die Vorstellung der einzelnen Metriken, aber auch eine Kategorisierung in Rubriken, um Metriken besser voneinander abzugrenzen, da diverse Metriken höchst unterschiedliche Aussagen über einen Graphen treffen. Es wird auch darauf eingegangen welche Motivation hinter den jeweiligen Metriken steht. Bei der Beantwortung dieser Frage soll außerdem auch darauf eingegangen werden, inwieweit die beschriebene Metrik in bestimmten Mathematikbibliotheken wie „Sage Math“ oder „Wolfram“ vorkommen.

2. Wie sind die vorgestellten Metriken zu bewerten?

Bei dieser Frage soll es vor allem darum gehen, die vorgestellten Metriken dahingehend zu bewerten, wie „schwer“ es ist, sie zu ermitteln. Außerdem wird bei der Bewertung auch auf die Verbreitung eingegangen. Speziell soll erwähnt werden, ob die jeweilige Metrik in bekannten Mathematikbibliotheken vertreten ist oder nicht.

3. Was beschreibt der Ähnlichkeitsbegriff bei Graphen?

Basierend auf Graph-Metriken lässt sich auch ermitteln, ob zwei Graphen Ähnlichkeiten aufweißen [WM19]. Auch auf diesen Aspekt soll die Arbeit bezug nehmen und dabei ein Anwendungs-Beispiel konstruieren.

4. Wie können die vorgestellten Metriken in Graphdatenbanken verwendet werden, bzw. implementiert werden?

Auf die theoretische Betrachtung der Graph-Metriken folgt dann ein praktischer Teil, der behandeln soll, wie sich die Metriken in bekannten Graphdatenbanken umsetzen lassen, bzw. umgesetzt wurden. Dabei ist es wichtig herauszufinden welche Graphkennzahlen bereits teil der Graphdatenbank-Lösungen sind, bzw. welche Metriken selbst umgesetzt werden müssen.

5. Wie sind die jeweiligen Implementierungen zwischen und innerhalb der Graphdatenbanken zu bewerten?

Folgend auf die Implementierung, ist es noch wichtig zu verstehen, wie diese Umsetzungen zu betrachten sind. Dabei wird vor allem ein Fokus auf das Thema Performance und Skalierung gelegt.

6. Welche Anwendungen gibt es für Graph-Metriken und den Vergleich von Graphen (Ähnlichkeit)?

Als letztes soll sich die Studienarbeit mit praktischen Beispielen beschäftigen. Es ist dabei wichtig zu verstehen, welchen konkreten Nutzen die gezeigten Kennzahlen für Graphen in modernen Anwendungsszenarien haben.

# 2 Graph-Metriken

Dieser erste Teil der Arbeit wird sich nun ausführlich mit einer weiten Reihe an Graph-Metriken beschäftigen. Hierbei sollen die ersten zwei Fragestellungen der Arbeit genau beantwortet werden. Zur jeweiligen Vorstellung einer Graph-Metrik sollen dabei die folgengen Punkte erläutert werden:

- Was drückt die Metrik aus (Definition)?
- Wie ist die Metrik im Bezug auf den Rechenaufwand zu bewerten?
- Inwieweit ist die Metrik verbreitet? Zum Beispiel in der Literatur oder in Mathematikbibliotheken.
- Was ist eine konkrete Motivation bzw. Anwendung für die Metrik, falls diese auszumachen ist?

Es ist noch zu erwähnen, dass alle im folgenden vorgestellten Metriken über die einzelnen Sektionen der Arbeit in Kategorien eingeteilt sind.

Darüber hinaus ist noch eine grundsätzliche Notationen während der Arbeit zu klären: Ein **Graph G** ist ein Paar bestehend aus **Knoten V** und **Kanten E**.

$$G = (V, E),\ wobei\ E \subseteq V \times V$$

Für V können wir auch V(G) schreiben, für E auch E(G). [Die00] V ist dabei Englisch und bedeutet „Vertices“, E steht für „Edges“. Wenn es um die Datenstrukturen von Graphen geht, kommen im Rahmen dieser Arbeit hauptsächlich Adjazenzmatrizen und Adjazenzlisten zum Einsatz. Allerdings können bei Bedarf auch Inzidenzen (Beziehung zwischen Knoten und Kanten) im Graphen eine Rolle spielen, wie Inzidenzmatrizen und Inzidenzlisten. [Kne19; Die00]

## 2.1 Grundlegende Metriken

Ein einem ersten Teil sollen grundlegende Graph-Kennzahlen vorgestellt werden. Diese beschreiben einen Graphen auf rudimentäre Art und Weise und zeigen die am einfachsten zu berechnenden Eigenschaften des Graphen.

### Ordnung und Größe eines Graphen

Die Frage danach, wie viele Knoten ein Graph hat lässt sich mit der **„Ordnung“** eines Graphen beantworten. Die Ördnung”beschreibt dabei einfach die Anzahl der Elemente in der Menge V. Man schreibt: $|V|\ oder\ |V(G)|\ oder\ auch\ |G|$. [Die00] Diese Eigenschaft ist essentiell zur allgemeinen Beschreibung und z.B. graphischen Darstellung eines Graphen. Sie lässt sich dabei in sämtlichen mathematischen Bibliotheken finden, wie SageMath, Matlab und Wolfram [Sag20b; Mat20a; Wol20a]. Die Komplexität zur Erfassung der Metrik gestaltet sich dabei äußerst einfach. Bei einer Adjazenzmatrix lässt sich die Anzahl der Knoten dadurch herausfinden, wie „lang“ eine Dimension des zweidimensionalen Arrays bzw. der zweidimensionalen Liste. Dies kann man je nach Implementierung der jeweiligen Datenstruktur in einer Komplexität von $O(n)$ oder $O(1)$ herausfinden.

Eine weitere grundlegende Kennzahl von Graphen ist dessen **„Größe“**. Die Größe beschreibt dabei die Anzahl der Kanten, die im Graphen vorkommen, also die Anzahl der Elemente in der Menge E. Man schreibt analog zur Größe des Graphen: $|E|\ oder\ |E(G)|\ oder\ auch\ ||G||$. [Bal97; Die00] Auch diese Metrik ist weit verbreitet. So lässt sie sich in vielen Büchern zur Graphentheorie finden, aber auch in den genannten Mathematikbibliotheken [Sag20b; Mat20a; Wol20a]. Die Anzahl der Kanten innerhalb eines Graphen herauszufinden, erweist sich nicht ganz so trivial wie das Herausfinden der Ordnung. Ist ein Graph nicht gerichtet, d.h. seine Kanten haben keine feste Richtung [Die00] so ist seine Adjazenzmatrix symmetrisch. Man kann also zählen wie viele Einträge es innerhalb der Matrix auf der Hauptdiagonalen und einer der Hälften gibt. Das ergäbe immer $\frac{1}{2}n^2$ Schritte, wenn $n$ die Ordnung des Graphen ist. Die Komplexität beträge also $O(n^2)$. Bei der Darstellung durch eine Inzidenzliste wäre das anders. Hier könnte einfach die Größe der Liste gesucht werden und man wüsste die Größe des Graphen. Die Komplexität wäre hier, wie schon erwähnt, je nach Implementierung $O(n)$ oder $O(1)$.

### Der Grad eines Knotens

Während die zwei ersten vorgestellten Metriken vor allem den Graphen als ganzes beschreiben, ist es auch noch wichtig zu wissen, was einen einzelnen Knoten auszeichnet, um einen Graphen besser zu beschreiben. Hierzu gibt es die grundlegende Metrik des **Grad** eines Knotens. Der Grad eines Knotens beschreibt die Anzahl der mit einem Knoten inzidenten Kanten [Die00]. D.h. es drückt aus, wie viele Kanten mit einem Knoten verbunden sind. Man kann dies z.B. durch eine Funktion ausdrücken, die einen Knoten v auf eine natürliche Zahl abbildet: $d(v)$.

Auf Basis dieser Metrik lässt sich auch andere verwandte Metriken ableiten. Hierzu gehört der **„Minimalgrad“** und der **„Maximalgrad“**. Der Minimalgrad ist der kleinste Knoten-Grad eines Graphen G: $\delta(G) := min\{d(v) \mid v\ \in\ V(G)\}$. Parallel dazu ist der Ma-

ximalgrad der größte Knoten-Grad eines Graphen G: $\Delta(G) := max\{d(v) \mid v \in V(G)\}$. Darüber hinaus kann man noch den **„Durchschnittsgrad“** eines Graphen bestimmen. Dieser bildet den Durchschnitt aller Knotengrade ab: $d(G) := \sum_{v \in V(G)} d(v)/|V|$. [Die00]

Des Weiteren gibt es bei der Betrachtung eines gerichteten Graphen zusätzliche Abwandlungen der Metrik. Da hier die Kanten immer zu einem Knoten gerichtet sind unterscheidet man speziell zwischen dem **„Eingangsgrad“** und dem **„Ausgangsgrad“**. Der Eingangsgrad eines Knoten beschreibt dabei die Anzahl der Kanten, die auf einen Knoten „zeigen“. Der Ausgangsgrad zeigt wie viele Kanten von einem Knoten „weggehen“. [Bal97]

Auch der Grad eines Knotens und die meisten seiner verwandten Metriken sind weit verbreitet. So sind der allgemeine Grad, der Eingangsgrad, der Ausgangsgrad in allen drei betrachteten Mathematikbibliotheken vorhanden. SageMath unterstützt sogar nativ die Metrik „Durchschnittsgrad“. [Sag20b; Mat20a; Wol20a]

Die Berechnung eines Grades über eine Adjazenzmatrix oder eine Adjazenzliste ist in linearer Zeit lösbar ($O(n)$). Bei der Adjazenzmatrix muss einfach nur die jeweilige Reihe des zugehörigen Knotens durchlaufen werden und gezählt werden, wie häufig ein Eintrag für eine Kante enthalten. Mit Hilfe der Adjazenzliste kann einfach die Größe der Liste als Grad genommen werden, die dem Knoten zugehörig ist.

### Anzahl der Zusammenhangskomponenten

Eine weitere Variante einen Graphen grundlegend zu beschreiben, besteht darin seine Zusammenhangskomponenten zu zählen. Hierfür ist es zunächst wichtig zu verstehen, was Zusammenhang bei Graphen bedeutet.

Ein Graph gilt dann als zusammenhängend, wenn gilt: $\forall a, b(a \in V \land b \in V \land a \neq b \implies Weg\_existiert(a, b))$. Anschaulich bedeutet das, dass es zwischen zwei beliebigen Knoten immer einen Weg geben muss, der die beiden Knoten miteinander verbindet. Graphisch erscheint ein zusammenhängender Graph so, dass sich keine verschiedenen, klar voneinander trennbaren Komponenten erkennen lassen. Diese einzelnen Komponenten oder Partitionen eines nicht-zusammenhängenden Graphen werden „Zusammenhangskomponenten“ genannt. Innerhalb der Zusammenhangskomponenten gilt natürlich wieder die Eigenschaft des Zusammenhangs. [Die00]

Die Anzahl der Zusammenhangskomponenten gibt nun an, wie viele Komponenten innerhalb eines Graphen vorhanden sind. Die Metrik ist sowohl in MatLab als auch in SageMath vertreten [Sag20b; Mat20b]. Zur Ermittlung der Anzahl wird sich eines einfachen Algorithmus bedient, der die Tiefen- oder Breitensuche nutzt, die jeweils jeden Knoten als „besucht“ markiert, den sie traversiert. Folgender Pseudocode beschreibt diesen Algorithmus:

```
1 Integer zaehleKomponenten (graph)
2   int komponentenanzahl = 0
```

```
    for (knoten in graph)
      if (knoten ist nicht besucht)
        tiefen_oder_breitensuche(graph, knoten)
      komponentenanzahl++
    return komponentenanzahl
```

Da der Algorithmus von einem der Suchalgorithmen abhängig ist, bestimmt dieser die Komplexität zur Ermittlung der Kennzahl. Jeder Knoten wird genau einmal in dieser Prozedur besucht. Zudem wird über alle Knoten in der äußeren Zählschleife iteriert. Es ergibt sich dadurch eine Komplexität von $O(|V| + |E|)$.

## 2.2 Distanzmetriken

Innerhalb der Graphentheorie gibt es den Begriff des Wegs. Ein Weg beschreibt dabei einen Graphen, der Knoten in einer Reihe hinterander Verbindet. Somit hat der Anfangs- und End-Knoten den Grad 1 und alle „mittleren“ Knoten den Grad 2. Meist sucht man aber einen bestimmten Weg innerhalb eines bestehenden Graphen. Der Weg ist hier dann ein Teilgraph des ursprünglichen Graphen. Anschaulicher lässt sich ein Weg also als eine Folge von Kanten beschreiben, in der kein Knoten zweimal besucht wird. Die Länge eines Weges ist dabei die Anzahl der Kanten, die in einem Weg vorhanden sind. [Die00] Auf Basis des Weges und seiner Längen-Eigenschaft lassen sich nun eine Reihe von Metriken definiert werden.

#### Abstand/Distanz

Der „Abstand“ ist eine Metrik, die auf Basis von zwei Knoten innerhalb eines Graphen definiert wird. Sie beschreibt die Länge des kürzesten Weges zwischen den zwei Knoten, von denen man den Abstand wissen will. Aufgeschrieben werden kann die Metrik mittels einer Funktion, die für den Graph G zwei Knoten x und y auf eine natürliche Zahl abbildet: $d_G(x, y)$ [Die00] Diese Metrik ist wichtig als Basis für andere Metriken. Wie die bisherigen Metriken ist auch diese in den jeweiligen Bibliotheken vertreten [Sag20b; Mat20d; Wol20a]. Zur Berechnung der Metrik kann auf verschiedene bekannte, graphtraversierende Algorithmen zurückgegriffen werden. Dazu zählen die Breitensuche, der Djikstra-Algorithmus oder der Bellman-Ford-Algorithmus [Sag20b]. Somit ist auch die Komplexität zum Herausfinden der Metrik gleich mit der der Algorithmen. So wäre bei einer Breitensuche eine Komplexität von $O(|V| + |E|)$ zu erwarten, da jede Kante abgegangen wird. Der Djikstra-Algorithmus hingegen hat eine Komplexität von $O(|V|^2)$ [Jun13].

### Extrenzität eines Knotens

Mit Hilfe des Abstandes lässt sich nun u.a. die **„Extrenzität“** eines Knotens bestimmen. Die „Extrenzität“ ist dabei der maximale Abstand den ein Knoten von einem anderen Knoten in einem Graphen G haben kann. Eine einfache formale Notierung für den Knoten x wäre: $ecc(x, G) = max_{x,y}\{d_G(x, y)\}$, wobei x der gegebene Knoten ist. [Har01] Herauszufinden ist diese Kennzahl beispielsweise über den Djikstra-Algorithmus, der den kürzesten Abstand zu jedem anderen Knoten sucht und dann einfach der größte zu wählen ist. D.h. das die Metrik auch die Komplexität hat wie bei der Metrik „Abstand“ erläutert wurde. Die Extrenzität eines Knotens ist anschließend noch für andere Metriken eine wichtige Basis und allgemein weit verbreitet in den genannten Bibliotheken [Sag20b; Mat20d; Wol20a].

### Durchmesser

Wie schon erwähnt, ist es nun möglich auf Basis des Abstands weitere Metriken zu definieren. Hierzu zählt unter anderem der **„Durchmesser“** eines Graphen. Der Durchmesser beschreibt dabei den größten Abstand zweier Knoten im Graphen G. [Die00] D.h. es ist der Abstand von allen Knoten zu allen Knoten zu berechnen und davon die größte Zahl auszuwählen. Formal notiert lässt sich die Metrik folgendermaßen beschreiben: $Durchmesser(G) = max_{x,y}\{d_G(x, y)\}$. Nimmt man zur Ermittlung der Abstände dabei den Djikstra-Algorithmus und wendet diesen dann jeweils auf jeden einzelnen Knoten an, kann eine Komplexität von $O(|V|^3)$ angenommen werden, um die Metrik zu extrahieren. Auch diese Metrik lässt sich z.B. in SageMath oder Wolfram finden. In Matlab kann der Durchmesser über die Distanzmatrix ermittelt werden, die Matlab erstellen kann. [Sag20b; Mat20d; Wol20a] Die Anwendung für diese Metrik kann z.B. sein, rein topologische Eigenschaften des Graphen herausfinden zu wollen. Allerdings kann auch in realen Problemen der Durchmesser als Metrik auftauchen. So ist z.B. der Abstand und damit der Durchmesser auch mit gewichteteten Kanten berechenbar. [Sag20b; GIT14] In einem Navigationssystem wäre der Durchmesser dann die längste fahrbare Strecke.

### Radius

Parallel zum Durchmesser eines Graphen kann man auch dessen **„Radius“** bestimmten. Hierfür wird die Metrik der Extrenzität wichtig und der Begriff der Zentralität. Eine Kante ist dann *zentral* bzw. im Zentrum eines Graphen, wenn dessen Extrenzität minimal ist. Dies kann nur einen Knoten, aber auch mehrere Knoten betreffen. Die minimale Extrenzität in einem Graphen, die die Knoten des Zentrums haben, nennt man dann auch den **„Radius“** des Graphen. Geschrieben wird $rad\ G = min_{x \in V(G)} max_{y \in V(G)} d_G(x, y)$. [Die00] Der Radius lässt sich grundsätzlich auf die gleiche Weise herausfinden, wie

der Durchmesser und hat dementsprechend die gleiche Komplexität. Des Weiteren ist diese Metrik auch weit verbreitet und lässt sich in allen untersuchten Bibliotheken finden [Sag20b; Wol20a; Mat20d]

## 2.3 Kreis-basierte Metriken

Ausgehend von einem Weg innerhalb eines Graphen können wir auch den Begriff des Kreises definieren. Ein Kreis ist dabei einfach ein Weg, der eine zyklische Eckenfolge hat. Sei $x_i$ ein Knoten, so hat ein Kreis folgende typische Eckenfolge: $x_0 ... x_{k-1},\ x_0$. [Die00] Auch auf Basis solcher Kreise lassen sich verschiedene Graph-Metriken definieren.

### Länge eines Kreises

Ist ein solcher Kreis, wie vorhin beschrieben, gegeben, kann unter anderen an diesem seine Länge abgelesen werden. Die Länge beschreibt dabei die Anzahl der Kanten, die ein Kreis enthält. Ist erst einmal ein Kreis gegeben lässt sich die Länge leicht berechnen, denn die Länge eines Kreises ist gleich mit der Anzahl der Knoten innerhalb eines Kreises. Somit lässt sich die Länge in konstanter Zeit berechnen. [Die00] Diese Metrik ist allerdings nicht direkt in referenzierten Mathematikbibliotheken vertreten, da grundsätzlich nur Graphen allgemein verarbeitet werden. Allerdings wäre es möglich, auf Basis des Wissens einen zyklischen Graphen zu erstellen und beispielsweise sich die Kantenanzahl zurückgeben zu lassen.

### Taillenweite und Umfang

Nimmt man die Länge eines Kreises als Basis, können weitere Kennzahlen für den gesamten Graphen ermittelt werden. Eine davon ist die „Taillenweite“ des Graphen. Die „Taillenweite“ ist so definiert, dass sie den Wert der Länge des kürzesten Kreises innerhalb des Graphen annimmt. Umgekehrt lässt sich auch der „Umfang“ des Graphen bestimmen. Der Umfang ist dabei so groß wie die Länge des größtmöglichen Kreises innerhalb eines Graphen. Hat ein Graph keinen Kreis, so ist es nicht möglich für beide Kennzahlen einen Wert zu ermitteln. Allerdings haben diese dann einen festen Wert. So beträgt die „Taillenweite“ in diesem Fall $\infty$ und der Umfang null. [Die00]

Um die Taillenweite und den Umfang eines Graphen herauszufinden, ist es grundsätzlich notwendig die jeweiligen Zyklen innerhalb des Graphen herauszufinden. Hierfür lässt sich unter anderem eine modifizierte Tiefensuche nutzen, die mittels Markierung der Knoten erkennt, ob sie bereits schon einmal bei einem Knoten war und infolgedessen einen Zyklus erkennt. Hierbei wird unterschieden, ob ein Knoten noch nicht bearbeitet wurde, in Bearbeitung ist oder bereits der Algorithmus auf ihm vollständig abgeschlossen wurde. Wird der Algorithmus auf einem Knoten aufgerufen, der sich noch in Bearbeitung

befindet, ist ein Zyklus gefunden. Mit einer richtigen Ausgabe kann dieser dann auch benannt werden. Daraus folgt auch, dass die Ermittlung der Taillenweite und des Umfangs entspricht, denn nach der Tiefensuche, muss einfach der größte bzw. der kleinste Zyklus gewählt werden, um die Metriken zu ermitteln. Die Komplexität beträgt damit $O(|V| + |E|)$. [Kne19; Vöc+08]

Zur Metrik „Umfang“ lassen sich in den genannten Bibliotheken, außer bei Wolfram, keine direkten Implementierungen finden. Allerdings ist es möglich in SageMath und in Wolfram die „Taillenweite“ direkt zu evaluieren. [Sag20b; Wol20c]

## 2.4 Zusammenhangsmetriken (Connectivity)

Neben der Definition von Metriken auf Basis von Distanzen ist es auch möglich, Kennzahlen zu ermitteln, die den Zusammenhang eines Graphen betrachten. Hierfür ist es wichtig zu verstehen, wann ein Graph als zusammehängend gilt und was eine Zusammenhangskomponente bzw. Partition eines Graphen ist. Dies wurde bei der Metrik „Anzahl der Zusammenhangskomponenten“ erläutert.

#### Dichte

Bei der Vorstellung grundlegender Graphmetriken wurden u.a. die Größe und die Ordnung eines Graphen erklärt. Darauf aufbauend kann eine Kennzahl ermittelt werden, die aussagt, wie stark vernetzt ein Graph ist. Die „Dichte“ eines Graphen gibt an, inwiefern der Graph so viele Kanten hat wie es ihm theoretisch möglich ist. Die „Dichte“ setzt also tatsächliche Kantenanzahl (Größe) und die mögliche Kantenanzahl in ein Verhältnis. Die Metrik kann darauffolgend einen Wert zischen 0 und 1 annehmen. Ist der Wert 0 hat der Graph keine Kanten. Ist der Wert hingegen 1 so hat man einen vollständigen Graphen vor sich liegen. Möchte man die Dichte eines Graphen ermitteln ist es nötig die Größe des Graphen durch die potenzielle Größe zu teilen. Dies ist mit folgender Gleichung möglich: $\frac{|E|}{\binom{|V|}{2}}$. Hat man statt einem ungerichteten Graphen einen gerichteten muss die Formel leicht abgewandelt werden, da für einen vollständigen Graphen nun doppelt so viele Kanten nötig sind: $\frac{|E|}{2\binom{|V|}{2}}$. [Sag20b]

Zur Implementierung der Metrik ist es infolgedessen nur nötig die Größe und die Ordnung des Graphen herauszufinden. Die Komplexität zur Berechnung der „Dichte“ ist deshalb gleich der Komplexität zur Berechnung von Größe und Ordnung addiert. Die Berechnung der „Dichte“ selbst erfolgt in konstanter Zeit. Durch die Ableitung der Metrik aus zwei grundlegenden Kennzahlen ist die Berechnung auch problemlos möglich, ohne dass es eine explizite Implementierung in einer Bibliothek gibt. Allerdings bieten Wolfram und SageMath spezielle Funktionen für die „Dichte“ eines Graphen. [Sag20b; Wol20c; Mat20a]

### Stärke

Oft repräsentieren Graphen ein Netzwerk. Im Rahmen von Netzwerken wird unter anderem von deren „Stärke“ gesprochen. Die „Stärke“ gibt dabei das minimale Verhältnis zwischen entfernten Kanten und dadurch erstellter Zusammenhangskomponenten an. Es muss dabei allerdings die Anzahl der Zusammenhangskomponenten insgesamt erhöht werden. Ist die „Stärke“ eines Netzwerks, bzw. eines Graphen, hoch, ist es u.a. schwieriger für einen Angreifer das Netzwerk stark zu beschädigen, greift dieser die Verbindungen, bzw. Kanten, des Netzwerks an. Zur Berechnung der Stärke $\sigma(G)$ seien folgende Annahmen gegeben: Sei $\Pi$ die Menge aller möglichen Partitionierungen der Knoten V und $\partial\pi$ die Menge an Kanten, die entfernt werden müssten, um die Partitionierung $\pi$ zu erreichen, gilt folgende Formel zur Errechnung der Stärke:

$$\sigma(G) = \min_{\pi \in \Pi} \frac{|\partial\pi|}{|\pi| - 1}$$

Es wird also jede mögliche Partitionierung der Knoten V betrachtet und ermittelt welche Kanten man entfernen müsste, um diese Partitionierung der Knoten zu erhalten. Die Anzahl der Elemente in der jeweiligen Menge werden dann in ein Verhältnis gesetzt. Dabei wird eine Zusammenhangskomponenten aus $\pi$ subtrahiert, da ein Graph immer zumindest aus einer Komponente besteht. Aus all diesen erstellten Verhältnissen ist nun das Minimum das Ergebnis. [Tru93; Cun85] Es kann auch anders gesagt werden, dass ein Graph bei dem die Entfernung weniger Kanten zu vergleichsweise vielen Zusammenhangskomponenten führt, ein sehr „schwacher“ Graph ist. Umgekehrt ist es bei einem „starken“ Graphen nicht möglich, selbst durch die Entfernung vieler Kanten (Zähler), eine vergleichsweise hohe Anzahl an Zusammenhangskomponenten (Nenner) zu erreichen.

Die Berechnung der „Stärke“ und die Verbesserung der Komplexität des Algorithmus war Thema mehrerer wissenschaftlicher Arbeiten. Die beste erreichte Komplexität erzielte dabei V. A. Trubin mit einer Komplexität von $O(\min(\sqrt{m}, n^{2/3})mn\log(n^2/m+2))$. $m$ ist hierbei die Anzahl an Kanten im Graphen, $n$ die Anzahl an Knoten. [Tru93]

In SageMath, Wolfram oder MatLab ist die „Stärke“ von Graphen nicht implementiert. Darüber hinaus ist es auch möglich statt über die Entfernung von Kanten die Metrik über die Entfernung von Knoten definieren. In diesem Fall spricht man über die „Härte“ oder „Zähigkeit“ (engl. „Toughness“) des Graphen. [Chv06]

### „Vertex Connectivity“/Zusammenhang

Die Stärke eines Graphen ist eine nicht ganzzahlige Metrik zur Beschreibung des allgemeinen Zusammenhangs innerhalb eines Graphen. Wie bei der Stärke schon erwähnt, ist diese Metrik in den einschlägigen Bibliotheken nicht zu finden. Die nächsten zwei Metriken sind sowohl in SageMath als auch in Wolfram zu finden und beschreiben

die stärke des Zusammenhangs des Graphen mittels einer ganzen Zahl [Sag20b; Wol20a]

Die erste dieser Metriken ist die „Vertex Connectivity“ oder „Zusammenhang“. Hierbei wird ein zusammenhängender oder auch nicht zusammenhängender Graph betrachtet und ermittelt wie viele Knoten aus dem Graphen mindestens entfernt werden müssen, sodass der Graph nicht mehr zusammehängend ist. *k* entspricht dieser minimalen Anzahl an Knoten. Formal lässt sich sagen, dass ein Graph *k-zusammenhängend* ist, wenn $|G| > k$ und der Graph für jede mögliche Knotenmenge X mit der Mächtigkeit $< k$ zusammenhängend bleibt, sobald man alle Knoten $X \subseteq V$ aus V entfernt $(G - X)$. Da ein Graph der *3-zusammenhängend* ist auch *4/5/...-zusammenhängend* ist, ist die absolute „Vertex Connectivity“ bzw. der „Zusammenhang“ das *k*, das für den Graph minimal ist. Der Zusammenhang ist dann 0, wenn der Graph von Anfang an nicht zusammenhängend ist oder der Graph nur aus einem Knoten besteht. [Die00] Auch bei dieser Metrik gilt wie bei der Stärke, dass der Graph schwerer zu „trennen“ ist, je höher die jeweilige Kennzahl ist. Daraus ist auch zu folgen, dass bei hohem Zusammenhang beispielweise ein Netzwerk weniger Anfällig für Angriffe ist.

Um den Zusammenhang eines Graphen algorithmisch herauszufinden, können zunächst zwei triviale Fälle abgedeckt werden. Ist ein Graph leer oder trivial (nur ein Knoten) ist der Zusammenhang, wie bereits erwähnt, 0. Ist hingegen der Graph vollständig, beträgt der Zusammenhang $|V|$. Allerdings lässt sich auch ein allgemeiner Algorithmus definieren, der die „Vertex Connectivity“ berechnet. Für den Algorithmus wird eine zusätzliche Funktion benötigt. $N(a, b)$ nimmt zwei Knoten entgegen. Eine Menge an Knoten, die bei Entfernung dafür sorgt, dass zwischen a und b kein Weg mehr existiert, wird „Knoten-Separator“ genannt. N gibt nun die Kardinalität des minimalen „Knoten-Separators“ von a und b zurück. Sind a und b direkt mit einer Kante verbunden, gibt es keinen „Knoten-Separator“. Sich diese Knotenpaare bei der Berechnung der „Vertex Connectivity“ anzusehen, ist unnötig und muss nicht betrachtet werden. Schlussendlich ist nämlich die Kardinalität des kleinsten minimalen „Knoten-Separators“ die gesuchte Kennzahl. Zur Berechnung gibt Shimon Even in seinem Buch „Graph Algorithms“ folgenden Algorithmus an [Eve12]:

```
1  Vertex-Connectivity(V, E)
2    Sortiere Knoten v_1, v_2, ..., v_|V| so, dass es von v_1 keine Kante
     zu irgendeinem v gibt
3    gamma = unendlich
4    i = 1
5    while i <= gamma
6      for each v, sodass v_i keine Kante zu irgendeinem v gibt
7        gamma = min{gamma, N(v_i, v)}
8    return gamma
```

Der gezeigte Algorithmus terminiert mit $\gamma$ gleich dem Zusammenhang. In seinen Ausführungen erläutert Even zudem, dass der Algorithmus eine Zeitkomplexität von

$O(|V|^{1/2} \cdot |E|^2)$ aufweist.

#### „Edge Connectivity“

Ähnlich zur „Vertex Connectivity“ ist auch die „Edge Connectivity“ ein ganzzahliges Zusammenhangsmaß für einen Graphen und ist auch ähnlich definiert. Wie die „Vertex Connectivity“ ist diese in SageMath und Wolfram enthalten [Sag20b; Wol20a]. Die „Edge Connectivity“ ist nur definiert, wenn der Graph mindestens 2 Knoten hat. Hat ein Graph nur einen Knoten oder ist von Anfang an nicht zusammenhängend, so ist der auch sogenannte „Kantenzusammenhang“ von G $\lambda(G)$ gleich null. Ansonsten wird der Kantenzusammenhang so definiert, dass es die minimale Zahl an Kanten ist, die aus einem Graphen entnommen werden kann, sodass dieser nicht mehr zusammenhängend ist. Präziser kann es folgendermaßen definiert werden: Ein Graph hat einen Kantenzusammenhang von $\lambda(G)$, wenn $G - F$ für alle Kantenmengen $F \subseteq E$ der Mächtigkeit $< \lambda(G)$ zusammehängend ist. Speziell ist damit das kleinstmögliche $\lambda(G)$ gemeint, das für den jeweiligen Graphen möglich ist. [Die00] Die „Edge Connectivity“ bildet also das genau Pendant zur „Vertex Connectivity“ und arbeitet komplett analog zu dieser Metrik.

## 2.5 Zentralitätsmetriken

Eine weitere wichtige Eigenschaft eines Graphen ist dessen Zentralität, bzw. dessen Zentralitäten. Bei der Untersuchung dieser Eigenschaften eines Graphen, möchte man herausfinden, welche Knoten oder allgemeiner Regionen eines Graphen besonders wichtig sind. Die Anwendungen für die Verwendung von Zentralitäts-Informationen ist dabei äußerst vielfältig. Vor allem die Analyse von realen sozialen Netzwerken wahr häufig der Ausgangspunkt für etwaige Untersuchungen. Aber auch bei Themengebieten wie Geographie, Stadtentwicklung und Organisationsaufbau wurde Zentralität zur Informationgewinnung herangezogen. Allgemeiner lässt sich sagen, dass diese Thematik bei allen möglichen Anwendungen interessant sein kann, die Graph-Daten sammeln. Infolgedessen soll nun eine Reihe an Zentralitäts-Metriken vorgestellt werden, die einen Graphen auf diese Eigenschaft auf unterschiedliche Art und Weise untersuchen. [Fre78]

#### Degree Centrality

Die erste Metrik, die vorgestellt werden soll, ist zugleich die einfachste. Die „Degree Centrality“ wird für einen Knoten eines Graphen definiert und basiert bzw. ist gleich zu einer schon vorgestellten Metrik. Die „Degree Centrality“ oder „Grad-Zentralität“ eines Knotens gleicht dessen Grad. D.h. $Degree\_centrality(v) = d(v); v \in V$. Die Berechnung der Zentralität erfolgt meist für jeden Knoten und kann im Fall der „Degree Centrality“ recht simpel berechnet werden. Liegt der Graph als Adjazenzmatrix dar, so muss nur

über diese vollständig iteriert werden und für jeden Knoten mitgezählt werden, wie viele Nachbarn er hat. Die Komplexität beläuft sich damit auf $O(|V(G)|^2)$. Die Metrik selbst ist damit auch, wie der Knotengrad in den angeführten Bibliotheken zu finden, bzw. implizit nativ berechenbar. Auch wenn die Metrik auf den ersten Blick recht rudimentär wirkt, so ist sie aber äußerst aussagekräftig z.B. bei der Analyse im Social-Media-Bereich. Geht man davon aus, dass jeder Knoten ein Nutzer oder eine Nutzerin ist und eine Kante eine Beziehung wie „Freund von“ oder „folgt“ gleichkommt, so gibt die „Degree Centrality“ an, wie wichtig der jeweilige User ist. Dies ist vor allem dann interessant, wenn man wissen möchte, ob eine Person besonders einflussreich ist oder nicht. Solche Informationen sind beispielsweise für Werbetreibene von Bedeutung.

#### Betweenness Centrality

Eine weitere Möglichkeit die Wichtigkeit eines Knoten in einem Graphen zu ermitteln ist über die sogenannte „Betweenness Centrality“. Auch diese Metrik wird häufig für die Analyse von sozialen Netzwerken genutzt. Besonders gibt die Kennzahl für einen Knoten an, wie viel Einfluss dieser hat für den Fluss der Information. Besonders wird es genutzt, um Knoten zu finden, die als Brücke von einem Graph-Teil zum anderen fungieren. [neo20a]

Zur Berechnung der Metrik ist zunächst wichtig zu verstehen, was unter dem Wert $\sigma_{st}$ und der Funktion $\sigma_{st}(v)$ zu verstehen ist, wobei gilt $s, t, c \in V(G)$. $\sigma_{st}$ gibt an, wie viele kürzeste Wege es zwischen den beiden Knoten $s$ und $t$ gibt. Die Funktion $\sigma_{st}(v)$ bildet wiederum einen Knoten $v$ aus $G$ auf die Anzahl der kürzesten Wege zwischen $s$ und $t$ ab, die durch $v$ gehen. Die Funktion ist dabei folgendermaßen definiert, bzw. kann so berechnet werden. [Bra01]

$$\sigma_{st}(v) = \begin{cases} 0, & \text{wenn } d_G(s,t) < d_G(s,v) + d_G(v,t) \\ \sigma_{sv} \cdot \sigma_{vt}, & sonst \end{cases}$$

Die erste Bedingung für $\sigma_{st}(v)$ gilt deshalb, weil $v$ nur dann auf einem der kürzesten Wege zwischen $s$ und $t$ sein kann, wenn die kürzeste Distanz zischen $s$ und $t$ gleich der kürzesten Distanz zischen $s$ und $v$ addiert mit der kürzesten Distanz zwischen $v$ und $t$ ist: $d_G(s,t) = d_G(s,v) + d_G(v,t)$.

Die „Betweenness Centrality“ $C_B(v)$ ist nun so definiert, dass sie die Aufsummierung der Fraktion zwischen $\sigma_{st}(v)$ und $\sigma_{st}$ für alle möglichen Paare $s, t$ angibt, wobei $s$ und $t$ nie gleich sind und auch nicht gleich $v$ sind:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Die Berechnung dieser Metrik ist in polynomialer Laufzeit möglich und kann mit einem

Algorithmus berechnet werden, der eine Komplexität von $O(|V|^3)$ aufweist. Grundsätzlich ist es für die Berechnung zunächst notwendig zwischen allen Knotenpaaren die Anzahl und die Länge der kürzesten Wege zu berechnen. Auf Basis dieser Daten kann dann mit den obigen Funktionen die Kennzahl berechnet werden. In seinem Artikel „A Faster Algorithm for Betweenness Centrality“ zeigt Ulrik Brandes zudem einen Algorithmus, der die Metrik mit einer Zeitkomplexität von $O(|V| \cdot |E|)$ berechnen kann. Es ist zudem erwähnenswert, dass diese Metrik auch explizit für gewichtete Graphen berechenbar ist. [Bra01] Zudem ist sie in Wolfram, SageMath und auch Matlab nativ vertreten und verfügbar. [Wol20a; Sag; Mat20c]

### Closeness Centrality

Eine weitere Zentralitätsmetrik ist die „Closeness Centrality“. Auch diese Metrik beschreibt die Wichtigkeit eines Knotens. Dabei kann mit der Kennzahl vor allem ausgesagt werden, inwieweit ein Knoten effizient Informationen innerhalb eines Graphen verteilen kann, vorausgesetzt der Graph stellt eine Struktur dar, die diese Interpretation zulässt. [neo20b]

Die „Closeness Centrality“ misst die durchschnittliche invertierte Distanz zu allen anderen Knoten. Erzielt ein Knoten dabei einen hohen Wert, so hat dieser im Schnitt die kleinste Distanz zu allen anderen Knoten. Bei einer Interpretation der Metrik geht man also meist davon aus, dass ein Knoten der eine kurze Distanz zu allen anderen Knoten hat, auch wichtig sein muss. Für die Berechnung der Metrik ist es vor allem wichtig zu wissen, wie hoch die Distanz ist von dem zu untersuchenden Knoten $v$ zu allen anderen. Auf Basis dessen lässt sich die Kennzahl folgendermaßen berechnen: [Coh+14]

$$C_C(v) = \frac{|V| - 1}{\sum_{u \in V} d_G(v, u)}$$

Die Berechnung dieser Metrik lässt sich in polynomialer Zeit unternehmen. Grundsätzlich sind die Distanzen für den Knoten $v$ auszurechnen und anschließend die obige Funktion anzuwenden. Zur Berechnung der Distanzen kann man z.B., wie schon in 2.2 erwähnt, die Breitensuche verwenden. Als Pseudocode könne die Berechnung der „Closeness Centrality“ dann folgendermaßen aussehen:

```
1  Closeness_Centrality(G(V, E), v)
2    distanz_gesamt = 0
3    for each a in V\v
4      distanz_gesamt += d(v, a)
5    return (G.order - 1)/distanz_gesamt
```

Bedenkt man, dass die Breitensuche eine Zeit-Komplexität von $O(|V| + |E|)$ hat und die $|V|$-mal gemacht wird lässt sich leicht die Gesamtkomplexität der „Closeness Centrality“ für einen Knoten ermitteln: $O(|V|) \cdot O(|V| + |E|) = O(|V| \cdot (|V| + |E|))$. [Sar+13]

Diese Zentralitätsmetrik ist weit verbreitet und lässt sich in den MatLab-, SageMath- und Wolfram-Bibliotheken finden. [Mat20c; Sag; Wol20a]

### Eigenvektor Centrality

Die „Eigenvektor Centrality“ ist auch eine Metrik für einen Graph-Knoten. Hierbei soll nicht nur darauf geachtet werden, inwieweit ein Knoten direkten Einfluss auf eine Netzstruktur hat, sondern auch seine transitive Wichtigkeit betrachtet werden. Anwendungen können z.B. Ranking-Systeme sein, die einem Daten-Knoten eine bestimmte Wichtigkeit zuordnen. [neo20c] Die grundlegende Idee der Metrik ist es durch die Transitivität nicht nur zu betrachten, wie wichtig der betrachtete Knoten selbst ist, sondern auch mit einzubeziehen, wie wichtig seine Benachbarten Knoten sind. So ist ein Knoten, der wichtig ist und dazu noch wichtige Nachbarn hat, unter Umständen wichtiger als ein Knoten, der zwar selbst als wichtig eingeschätzt wird, aber keine wichtigen Nachbarn hat.

Die Berechnung der „Eigenvektor Centrality“ basiert auf der Adjazenzmatrix des Graphen. Hierbei ist beim jeweiligen Eintrag $a_{v,t}$ eine Eins eingetragen, falls eine Kante zwischen $v$ und $t$ vorhanden ist, anderen Falls ist eine Null eingetragen. Die „Eigenvektor Centrality“ $x$ von Knoten $v$ ist nun folgendermaßen definiert ($M(v)$ ist die Menge an adjazenten Knoten von $v$): [BP15]

$$x_v = \frac{1}{\lambda} \sum_{t \in M(v)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t$$

Übersetzt ist die „Eigenvektor Centrality“ also die Aufsummierung der „Eigenvektor Centrality“ aller Nachbarknoten geteilt durch $\lambda$. Formuliert man die Formel um und betrachtet sie im Kontext der gesamten Adjazenzmatrix, so kann man auch schreiben: $Ax = \lambda x$, wobei $A$ die quadratische Adjazenzmatrix ist und $x$ der Vektor mit den jeweiligen „Eigenvektor Centrality“-Werten. Hierbei fällt auf, dass dies gleichzeitig auch die Formel für den Eigenvektor und Eigenwert einer Martix ist. Aufgrund dessen erklärt sich auch der Name der Metrik. Stellt man nämlich die Eigenvektor-Formel um, so ergibt sich $(Ax/\lambda) = x$. Betrachtet man in dieser Rechnung nur einen Knoten so ergibt sich durch die Konsequenz der Matrix-Multiplikation die erstgenannte Funktion für $x_v$. Laut dieser Ausführungen ist $\lambda$ ein Eigenwert der Adjazenzmatrix. Allerdings wird $\lambda$ ausdrücklich als Konstante innerhalb dieser Metrik aufgefasst. Warum ist $\lambda$ konstant? Es ist definiert, dass der dann Eigenvektor $x$ nicht negativ ist. Nach dem Perron-Frobenius-Theorem kann mit dieser Einschränkung $\lambda$ nur der größtmögliche Eigenwert für $A$ und damit auch konstant sein.

Zur Berechnung der Kennzahl für jeden Knoten kann man sich beispielsweise der sogenannten „Power-iteration method“ bedienen. Diese Methode dient dazu, für eine gegebene Matrix möglichst genau einen Eigenvektor und einen Eigenwert zu finden.

D.h. über Iterationen wird sich einem Ergebnis angenähert. Für das Verfahren im Falle der „Eigenvektor Centrality“ wird im ersten Schritt die Adjazenzmatrix mit $n$ Zeilen und $n$ Spalten mit einem $n$ großen Spalten-Vektor multipliziert, die vollständig mit Einsen gefüllt ist. Der entstehende Spalten-Vektor wird normiert und für die nächste Iteration zur Multiplikation mit $A$ verwendet. Dies macht man so lange bis der Spaltenvektor konvergiert, bzw. eine feste Anzahl an Iterationen durchlaufen wurde. Dieser Vektor ist dann idealerweise ein Eigenvektor von $A$ und der berechnete Normalisierungswert ist der korrespondierende Eigenwert $\lambda$. Als Beispiel und zum besseren Verständnis sei die Abbildung 2.1 gegeben, bei der ein Beispiel durchgerechnet wurde. [Meg15] Der limitierende Faktor dieser Methode zur Berechnung der „Eigenvektor Centrality“ ist

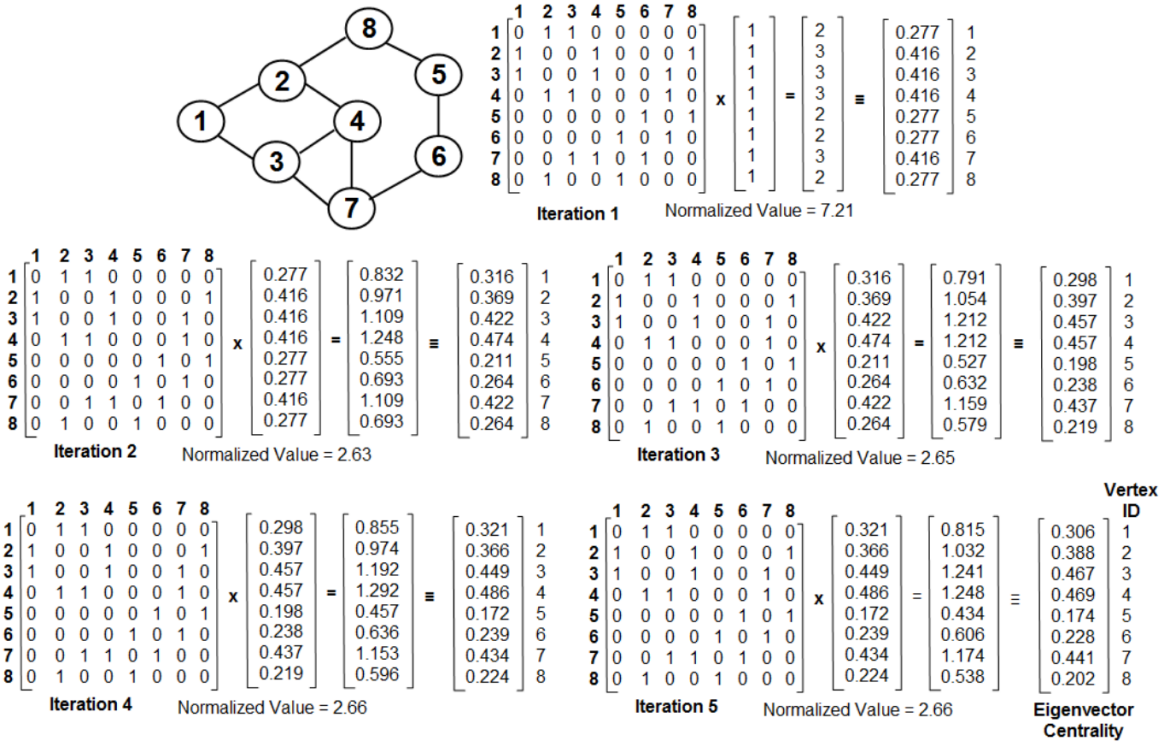


Abbildung 2.1: Eigenvektor Centrality: Power-iteration Method [Meg15]

die Matrix-Multiplikation. Implementiert man diese naiv so ergibt sich eine Komplexität von $O(n^3)$. Die Normierung lässt sich in linearer Laufzeit realisieren. Somit ist die Berechnung mit der „Power-iteration method“ in polynominaler Laufzeit möglich. Die Metrik selbst lässt sich nativ in Wolfram und in Matlab finden. [Wol20a; Mat20c]

### Page Rank

Zum Abschluss der Zentralitätsmetriken soll eine Metrik dieser Kategorie vorgestellt werden, die einen hohen praktischen Nutzen findet. Der „Page Rank“ oder die „Page Rank Centrality“ beschreibt die Wichtigkeit eines Knotens auf Basis seines Ausgangsgrads, bzw. seiner Verbindungen zu anderen Knoten und des „Page Ranks“ seiner benachbarten Knoten. Hierbei ähneln sich „Page Rank“ und „Eigenvektor Centrality“.

Die Metrik wurde erstmals in einem Google-Paper von Page und Brin veröffentlicht und soll Webseiten im World Wide Web bewerten. [BP98]

Es sei $A$ eine Seite, bzw. ein Knoten und dieser Knoten hat Einwegkanten (Links) zu den Seiten $T_1 \dots T_n$. $d$ sei ein frei wählbarer „Dämpfungs"-Faktor, der meist auf 0,85 gesetzt wird. Zudem ist $C(A)$ als der Ausgangsgrad für eine Seite (Knoten) definiert. Der „Page Rank" ist dann folgendermaßen definiert.

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)})$$

Die Berechnung des „Page Ranks" erfolgt durch einen iterativen Algorithmus, bie dem zunächst jedem Knoten der „Page Rank" $1/|V|$ gegeben wird. Darauf aufbauend kann dann die obige Funktion zur weiteren Berechnung und die weiteren Iterationen herangenommen werden, bis die Metrik konvergiert. Der Algorithmus weißt dabei eine Zeit-Komplexität von $O(k \cdot |E|)$ auf, wobei $k$ für die Anazahl der Iterationen steht. [Ora17] „Page Rank" ist in allen angeführten Mathematikbibliotheken vertreten. [Sag; Wol20a; Mat20c]

## 2.6 Chromatische Zahl und chromatischer Index

Ein bekanntes Problem der Informatik ist das Färbeproblem. Es geht dabei darum einen Graphen so zu färben, sodass zwei benachbarte Knoten nicht die selbe Farbe haben. Anwendungen dafür kann man in vielen Problemen finden. Klassischerweise nimmt man das Beispiel der Karteneinfärbung bei der jedes Land ein Knoten ist und die Karte so eingefärbt werden soll, dass zwei benachbarte Länder nicht die gleiche Farbe haben. Allerdings lässt sich das Färbeproblem auch auf andere relevantere Probleme anwenden. Man kann z.B. so auch einen konfliktfreien Stundenplan erstellen oder andere ähnliche Konflikt-Probleme lösen. Die Eckenfärbung eines Graphen selbst ist eine Abbildung $c : V \mapsto S$, wobei S die Menge an möglichen Färbungen ist. [Die00; Aig15]

### Chromatische Zahl

Auf Basis der Färbeproblems können nun eine Reihe an Metriken definiert werden. Betrachtet man das Problem mithilfe der Abbildungsfunktion $c$, so ist der Graph korrekt gefärbt, wenn gilt: $\forall v, w(v \in V \land w \in V \land adjazent(v, w) \Rightarrow c(v) \neq c(w))$. Die „chromatische Zahl" beschreibt nun die Mächtigkeit der minimalen Menge $S$ für die diese Bedingung zutrifft. Man bezeichnet die chromatische Zahl auch als $\chi(G)$. Für einen leeren Graphen ist die chromatische Zahl 1. Liegt ein bipartiter Graph vor, so ist die chromatische Zahl offensichtlicherweise 2. [Die00] Die chromatische Zahl kann sowohl von SageMath als auch von Wolfram berechnet werden [Sag20a; Wol20b]

Das Herausfinden der chromatischen Zahl ist ein NP-vollständiges Problem und kann deshalb vermutlich auch nicht effizient berechnet werden. [Weib; Kar96] So ist es möglich mittels eines Brute-Force-Algorithmus sämtliche Färbemöglichkeiten durchzuarbeiten und dann die Färbung herausnehmen, die gültig ist und die niedrigste Anzahl an Farben hat. Dieser Ansatz hätte eine exponentielle Laufzeit, die schon bei kleinen Graphen zu extrem hohen Rechenzeiten führen würde. Neben dieser naiven Variante werden u.a. auch heuristische Algorithmen genutzt, die versuchen die chromatische Zahl effizient zu approximieren. Ein bekannter Algorithmus hierfür ist folgendermaßen gegeben:

```
1  Chromatic-Number(G(V, E))
2    vertices <- sortiere_nach_Grad_absteigend(V)
3    highestColor <- 1
4    for (vertex in vertices)
5      color <- lowestPossibleColor(G, vertex)
6      vertex.color = color
7      highestColor <- Max{highestColor, color}
8    return highestColor
```

Dieser Algorithmus hat allerdings den Nachteil, dass es nur eine Annäherung an die chromatische Zahl ist und nicht gesichert ist, dass das Ergebnis korrekt ist. Seine Strategie besteht darin, den Graphen mit möglichst wenig Farben zu färben, indem zuerst der Knoten gefärbt wird, der die meisten Nachbarn hat.

### Chromatischer Index

Neben der Markierung bzw. Färbung von Knoten, ist es auch möglich in einem Graphen Kanten zu färben. Hierbei ist ähnlich wie bei der Knotenfärbung gemeint, dass eine Kante aus der Menge $E(G)$ auf eine Zahl, bzw. Farbe abgebildet wird: $k : E \mapsto S$. Im Rahmen der Kantenfärbung ist es nun wichtig, dass zwei indizente Kanten nicht die selbe Farbe zugewiesen bekommen. Das bedeutet, dass die Farben aller Kanten, die mit dem gleichen Knoten verbunden sind, alle eine unterschiedliche Farbe haben müssen. Formal lässt sich das so ausdrücken: $\forall v, w(v \in E \land w \in E \land inzident(v, w) \Rightarrow k(v) \neq k(w))$. Findet man eine Abbildung, für die diese Bedingung zutrifft, so nennt man den Graphen $k$-Kanten-färbbar. $k$ ist dabei die Mächtigkeit der Menge S, der gefundenen Abbildung. Das niedrigst mögliche $k$ für einen Graphen ist dann dessen „chromatischer Index“. Man schreibt für dafür auch $X'(G)$ Der „chromatischer Index“ ist sowohl in Wolfram, als auch in SageMath verfügbar. [And77; Sag20a; Wol20b]

Jeder Graph fällt bei Betrachtung seines chromatisches Indexes in eine von zwei Klassen. Bei der ersten Klasse ist der chromatische Index $X'(G) = \Delta(G)$, wobei $\Delta(G)$ der Maximalgrad des Graphen ist. Fällt der Graph nicht in diese Klasse, so ist sein chromatischer Index $X'(G) = \Delta(G) + 1$. Obwohl dabei der chromatische Index sich pro Graph in einem extrem engen Korridor aufhält, ist die exakte Ermittlung des Indexes

ein NP-vollständiges Problem und somit vermutlich nicht effizient, also in polynomialer Laufzeit, zu berechnen. Allerdings gibt es Spezialfälle, bei denen die Zuordnung des Graphen in einer der beiden Klassen einfach ist. So ist $X'(G) = \Delta(G)$, wenn der Graph bipartit ist. Ist der Graph vollständig und die Anzahl der Knoten ist gerade, ist dies ebenfalls der Fall. Ein kompletter Graph mit ungerader Knotenanzahl hat dann logischerweise den chromatischen Index $X'(G) = \Delta(G) + 1$. [Pla83]

Anwendung findet die Kantenfärbung in verschiedenen praktischen Problemen. Beispielsweise kann man es nutzen, um ein sogenanntes „Round-Robin"-Turnier mit möglichst wenig Runden zu planen. Ein „Round-Robin"-Turnier ist ein Turnier, bei dem jeder Teilnehmer auf jeden Teilnehmer einmal trifft. Dieses Problem ist zu lösen, indem jeder Teilnehmer durch einen Knoten repräsentiert wird und jede Begegnung durch eine Kante. Eine Farbe repräsentiert eine Runde. Werden die Kanten nun so gefärbt, dass der Turnier-Graph $k$-Kanten-gefärbt ist, hat man einen Turnier-Plan gefunden, der keine Konflikte hat. Ist der Graph zudem $X'(G)$-Kanten-gefärbt, so hat man den Plan gefunden, der möglichst wenig Runden erfordert. Der chromatische Index gibt in diesem Zusammenhang also an, wie viele Runden man minimal benötigt, sodass sich alle Teilnehmer einmal begegnen. Da der Graph bei dem sich alle einmal begegnen ein vollständiger Graph sein muss, ist auch schnell ersichtlich, wie viele Runden man braucht, betrachtet man die Ergebnisse des letzten Abschnitts. Neben „Round-Robin"-Turnieren lassen sich auch individuelle Turniere so planen. [GY04]

### Fraktionierte chromatische Zahl

Neben der klassischen Art und Weise einen Graphen zu färben gibt es auch weitere Methoden. In der fraktionierten Graphentheorie ist es auch möglich in einem Graphen seinen Knoten mehrere Farben zuzuweisen. Auf Basis dessen lassen wich weitere Metriken definieren, die die Graphenfärbung als Grundlage haben. Hierzu zählt die „fraktionierte chromatische Zahl". Um diese Zahl zu definierten, ist zunächst zu klären, wie fraktionierte Färbung formal definiert werden kann. Eine $b$-fache Färbung eines Graphen weist jedem Knoten eine Menge an $b$ Farben zu. Auch dies kann durch eine Abbildung dargestellt werden, indem ein Knoten auf eine Menge an Färbungen abgebildet wird. Bei der Färbung muss nun darauf geachtet werden, dass zwei adjazente Knoten mit ihren zugewiesenen Farben keine Schnittmenge bilden. Besser kann dies durch eine Abbildung dargestellt werden. Sei $A$ die Menge an verfügbaren Farben, so kann die Färbung mit folgender Abbildung dargestellt werden: $c : V \mapsto A^b$. Für die Graphen-Färbung muss nun gelten: $\forall v, w(v \in V \wedge w \in V \wedge adjazent(v, w) \Rightarrow c(v) \cap c(w) = \emptyset)$. Für eine $b$-Färbung wird eine bestimmte Anzahl $a$ an Farben benötigt. Zu einer solchen Färbung wird dann auch gesagt, dass es eine $a : b$-Färbung ist. Das niedrigste $a$ für das ein Graph eine $b$-Färbung hat, nennt man die $b$-fache chromatische Zahl ($\chi_b(G)$). Dabei ist logischerweise $\chi_1(G) = \chi(G)$. Die fraktionierte chromatische

Zahl ist dabei im Gegensatz folgendermaßen definiert [SU11]:

$$\chi_f(G) = \lim_{b\to\infty} \frac{\chi_b(G)}{b}$$

Die fraktionierte chromatische Zahl ist also der Grenzwert für die $b$-fache chromatische Zahl geteilt durch $b$, wenn $b$ gegen unendlich geht. Im Gegensatz zur „normalen“ chromatischen Zahl ist die fraktionierte chromatische Zahl eine rationale und keine natürliche Zahl. Die Berechnung der Metrik ist ein NP-vollständiges Problem. Allerdings gibt es für einige Graphen vorgefertigte Werte, die genutzt werden können. So ist die fraktionierte chromatische Zahl eines zyklischen Graphen $C_{2n+1}$ z.B. $2 + (\frac{1}{n})$. Der Algorithmus zur Berechnung der Kennzahl bedient sich einem linearem Programm und ist äußerst schwer zu berechnen. Die Komplexität steigt exponentiell zur Ordnung des Graphen. Eine native Implementierung dazu findet sich in SageMath. Wie bei der normalen Färbung kann auch die fraktionierte Färbung für diverse konfliktfreie Planungen und dergleichen verwendet werden. [Weib; Sag20a; SU11]

## 2.7 Arborizität

Ein wichtiger Teil der Graphentheorie ist die Betrachtung von Bäumen. Ein Baum ist dabei eine Zusammenhangskomponente eines Waldes. Ein Wald wiederum ist ein Graph, der keine Zyklen aufweist. Mithilfe dieses Wissens kann eine weitere Metrik definiert werden. Betrachtet man einen Graphen, zusammenhängend oder nicht, kann man sich die Frage stellen, wie es möglich ist, diesen Graphen aus einer Menge an Wäldern zu erstellen. Wichtiger noch ist es herauszufinden, aus welcher minimalen Anzahl an Wäldern es möglich ist, den vorliegenden Graphen aufzubauen. Diese minimale Anzahl ist die „Arborizität“. Präziser ausgedrückt, ist die „Arborizität“ von $G$ ($Y(G)$) die minimale Anzahl an azyklischen Subgraphen (Wäldern), dessen Vereinigung $G$ ergibt. Hierbei teilen sich die Subgraphen keine gemeinsamen Kanten. [Weia]

Zur Berechnung der Arborizität lassen sich sowohl Spezialfälle ausmachen, aber auch eine allgemeine Berechnungsvorschrift finden. So ist es zunächst ersichtlich, dass ein bereits azyklischer Graph eine Arborizität von $Y(G) = 1$ hat. Ein vollständiger Graph $K_n$ hat wiederum eine Arborizität von $Y(K_n) = \lceil n : 2 \rceil$ und ein vollständiger bipartiter Graph $K_{m,n}$ weißt den Wert $Y(K_{m,n}) = \lceil (mn) : (m + n - 1) \rceil$ auf. Möchte man die Arborizität allgemein berechnen, ist ein bestimmter Parameter des Graphen zu berechnen. $m_p$ gibt die maximale Anzahl an Kanten eines Subgraphen von $G$ an, der $p$ Knoten hat. Dieser Wert $m_p$ muss nun für alle $|G| > p > 1$ berechnet werden. Mit dieser Berechnung lässt sich die Arborizität mit folgender Formel für jeden $G$ ermitteln. [Weia; Nas61]

$$Y(G) = \max_{p>1} \left\lceil \frac{m_p}{p-1} \right\rceil$$

Die Berechnung der Arborizität ist in polynomialer Laufzeit möglich und kann in verschiedesten Szenarien angewandt werden. Einerseits kann es als Graph-Dichte-Maß herangezogen werden, da ein dichter, mit vielen Kanten durchzogener Graph, logischerweise eine höhere Arborizität aufweisen muss. Wie bereits in vorigen Kapiteln erwähnt, kann die Dichte für die Zuverlässigkeit bzw. Angriffssicherheit eines Netzwerks herangezogen werden. Weiterhin kann die Arborizität aber auch z.B. Aussagen über die Steifigkeit von Strukturen machen oder bei der Analyse von elektrischen Netzwerken helfen. Die Metrik ist in SageMath fest eingebaut. [GW92; Sag20b]

Die Spezialisierung der Arborizität ist die „lineare Arborizität“. Um diese Metrik zu definieren, muss zunächst der Begriff des „linearen Waldes“ geklärt werden. Wie ein normaler Wald besteht ein linearer Wald ausschließlich aus azyklischen Graphen. Allerdings ist jede Zusammenhangskomponente des linearen Waldes ein Pfad. Man kann sich also einen linearen Wald graphisch als eine Ansammlung von vollständig unverzweigten Graphen vorstellen. Die „lineare Arborizität“ $la(G)$ ist nun analog zur normalen Arborizität die minimale Anzahl an linearen Wäldern, dessen Vereinigung $G$ ergibt. [Alo88]

Auch für die Berechnung dieser Metrik können Spezialfälle ausgemacht werden. Beispielsweise hat jeder $d$-reguläre Graph eine lineare Arborizität von $\lceil (d+1)/2 \rceil$. Außerdem konnte mithilfe des „Linear Arboricity Conjecture“ bewiesen werden, dass für alle planaren Graphen gilt, dass die lineare Arborizität $\lceil \Delta : 2 \rceil$ oder $\lceil (\Delta + 1) : 2 \rceil$ ist. $\Delta$ ist dabei der Maximalgrad von $G$. Im Gegensatz zur normalen Arborizität ist die Ermittlung der linearen Arborizität nicht in polynomialer Laufzeit berechnenbar. Die Ermittlung dieser Metrik ist ein NP-vollständiges Problem. Die Kennzahl ist zudem in keiner er genannten Bibliotheken vorhanden. [Pér84; CKL10]

## 2.8 Weitere Metriken

## 2.9 Übersicht der vorgestellten Graphmetriken

Nachdem nun eine weite Reihe an Graph-Metriken vorgestellt wurden, sollen diese mit Hilfe einer Aufzählung zusammengefasst werden. Aufgelistet werden jeweils der Name, bzw die Bezeichnung, der Metrik, ihre Definition und die Komplexität zur Berechnung der Kennzahl. Bei der Definition wird dabei eine Mathematische oder eine wörtliche Beschreibung angegeben. Gegebenenfalls auch beides, falls es für das Verständnis förderlich ist.

**Ordnung**

| | |
|---|---|
| Definition: | Anzahl der Knoten eines Graphen |
| Komplexität: | $O(1)\ oder\ O(n)$ |

**Größe**

| | |
|---|---|
| Definition: | Anzahl der Kanten eines Graphen |
| Komplexität: | Adjazenzmatrix/liste: $O(n^2)$; Inzidenzmatrix/liste: $O(1)\ oder\ O(n)$ |

# 3 Ähnlichkeit von Graphen

# 4 Implementierung und Umsetzung der Metriken

## 4.1 Implementierung in verschiedenen Graphdatenbanken

## 4.2 Vergleich der Implementierungen

# 5 Graphmetriken und Ähnlichkeit in Anwendung

# 6 Fazit und Zusammenfassung

## 6.1 Zusammenfassung der Ergebnisse

## 6.2 Fazit

# Glossar

# Literatur


[Aig15] Martin Aigner. *Graphentheorie: eine Einführung aus dem 4-Farben Problem*. 2., überarbeitete Auflage. Springer Studium Mathematik Bachelor. OCLC: 927721160. Wiesbaden: Springer Spektrum, 2015. 196 S. ISBN: 978-3-658-10322-4 978-3-658-10323-1.

[Alo88] N. Alon. "The linear arboricity of graphs". In: *Israel Journal of Mathematics* 62.3 (Okt. 1988), S. 311–325. ISSN: 0021-2172, 1565-8511. DOI: 10.1007/BF02783300. URL: http://link.springer.com/10.1007/BF02783300.

[And77] Lars Dovling Andersen. "On edge-colorings of graphs." In: *MATHEMATICA SCANDINAVICA* 40 (1. Dez. 1977), S. 161. ISSN: 1903-1807, 0025-5521. DOI: 10.7146/math.scand.a-11685. URL: http://www.mscand.dk/article/view/11685 (besucht am 24. 10. 2020).

[Bal97] V. K. Balakrishnan. *Schaum's outline of theory and problems of graph theory*. Schaum's outline series. New York: McGraw-Hill, 1997. 293 S. ISBN: 978-0-07-005489-9.

[BP15] Anand Bihari und Manoj Pandia. "Eigenvector centrality and its application in research professionals' relationship network". In: *2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, ABLAZE 2015*. 2015. DOI: 10.1109/ABLAZE.2015.7154915.

[BP98] Sergey Brin und Lawrence Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. 1998. URL: http://infolab.stanford.edu/~backrub/google.html (besucht am 28. 01. 2021).

[Bra01] Ulrik Brandes. "A faster algorithm for betweenness centrality*". In: *The Journal of Mathematical Sociology* 25.2 (Juni 2001), S. 163–177. ISSN: 0022-250X, 1545-5874. DOI: 10.1080/0022250X.2001.9990249. URL: http://www.tandfonline.com/doi/abs/10.1080/0022250X.2001.9990249 (besucht am 26. 01. 2021).

[Chv06] V. Chvátal. "Tough graphs and hamiltonian circuits". In: *Discrete Mathematics* 306.10 (Mai 2006), S. 910–917. ISSN: 0012365X. DOI: 10.1016/j.disc.2006.03.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0012365X06001397 (besucht am 09. 12. 2020).

[CKL10] Marek Cygan, Łukasz Kowalik und Borut Lužar. “A Planar Linear Arboricity Conjecture”. In: *Algorithms and Complexity*. Hrsg. von Tiziana Calamoneri und Josep Diaz. Bd. 6078. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 204–216. ISBN: 978-3-642-13072-4 978-3-642-13073-1. DOI: 10.1007/978-3-642-13073-1_19. URL: http://link.springer.com/10.1007/978-3-642-13073-1_19 (besucht am 22. 01. 2021).

[Coh+14] Edith Cohen u. a. “Computing Classic Closeness Centrality, at Scale”. In: *Proceedings of the Second ACM Conference on Online Social Networks*. COSN ’14. event-place: Dublin, Ireland. New York, NY, USA: Association for Computing Machinery, 2014, S. 37–50. ISBN: 978-1-4503-3198-2. DOI: 10.1145/2660460.2660465. URL: https://doi.org/10.1145/2660460.2660465.

[Cun85] William H. Cunningham. “Optimal Attack and Reinforcement of a Network”. In: *J. ACM* 32.3 (Juli 1985). Place: New York, NY, USA Publisher: Association for Computing Machinery, S. 549–561. ISSN: 0004-5411. DOI: 10.1145/3828.3829. URL: https://doi.org/10.1145/3828.3829.

[Die00] Reinhard Diestel. *Graphentheorie*. 2., neu bearb. und erw. Aufl. Springer-Lehrbuch. OCLC: 247312585. Berlin: Springer, 2000. 314 S. ISBN: 978-3-540-67656-0.

[EK13] W. Ellens und R. E. Kooij. *Graph measures and network robustness*. _eprint: 1311.5064. 2013.

[Eve12] Shimon Even. *Graph algorithms*. Unter Mitarb. von Guy Even. 2nd ed. Cambridge, NY: Cambridge University Press, 2012. 189 S. ISBN: 978-0-521-51718-8 978-0-521-73653-4.

[Fre78] Linton C. Freeman. “Centrality in social networks conceptual clarification”. In: *Social Networks* 1.3 (1978), S. 215–239. ISSN: 0378-8733. DOI: https://doi.org/10.1016/0378-8733(78)90021-7. URL: http://www.sciencedirect.com/science/article/pii/0378873378900217.

[GIT14] GITTA. *Durchmesser eines Graphen*. Durchmesser eines Graphen. 2014. URL: https://www.gitta.info/Accessibiliti/de/html/StructPropNetw_learningObject2.html (besucht am 18. 11. 2020).

[GW92] Harold N. Gabow und Herbert H. Westermann. “Forests, frames, and games: Algorithms for matroid sums and applications”. In: *Algorithmica* 7.1 (Juni 1992), S. 465–497. ISSN: 0178-4617, 1432-0541. DOI: 10.1007/BF01758774. URL: http://link.springer.com/10.1007/BF01758774 (besucht am 21. 01. 2021).

[GY04] Jonathan L. Gross und Jay Yellen, Hrsg. *Handbook of graph theory*. Discrete mathematics and its applications. Boca Raton: CRC Press, 2004. 1167 S. ISBN: 978-1-58488-090-5.

[Har01] Frank Harary. *Graph theory*. 15. print. OCLC: 248770458. Cambridge, Mass: Perseus Books, 2001. 274 S. ISBN: 978-0-201-41033-4.

[Jun13] Dieter Jungnickel. *Graphs, networks and algorithms*. 4th ed. Algorithms and computation in mathematics 5. OCLC: 821566132. Berlin: Springer, 2013. 675 S. ISBN: 978-3-642-32278-5 978-3-642-32277-8.

[Kar96] Richard M. Karp. "Reducibility among combinatorial problems". In: *INFORMS Jounal on Computing* 8.4 (1996), S. 344–354.

[Kne19] Helmut Knebl. *Algorithmen und Datenstrukturen: Grundlagen und probabilistische Methoden für den Entwurf und die Analyse*. OCLC: 1123167896. 2019. ISBN: 978-3-658-26511-3.

[Lov12] László Lovász. *Large networks and graph limits*. American Mathematical Society colloquium publications volume 60. Providence, Rhode Island: American Mathematical Society, 2012. 475 S. ISBN: 978-0-8218-9085-1.

[Mat20a] Matlab. *Directed and Undirected Graphs - MATLAB & Simulink - MathWorks Deutschland*. Directed and Undirected Graphs. 2020. URL: https://de.mathworks.com/help/matlab/math/directed-and-undirected-graphs.html (besucht am 10. 11. 2020).

[Mat20b] Matlab. *Graph and Network Algorithms*. Graph and Network Algorithms. 2020. URL: https://de.mathworks.com/help/matlab/graph-and-network-algorithms.html?searchHighlight=graph&s_tid=srchtitle (besucht am 12. 01. 2021).

[Mat20c] Matlab. *Measure node importance*. centrality. 2020. URL: https://de.mathworks.com/help/matlab/ref/graph.centrality.html#bu86660 (besucht am 26. 01. 2021).

[Mat20d] Matlab. *Shortest path distances of all node pairs*. Distances. 2020. URL: https://de.mathworks.com/help/matlab/ref/graph.distances.html (besucht am 18. 11. 2020).

[Meg15] Natarajan Meghanathan. "Use of Eigenvector Centrality to Detect Graph Isomorphism". In: *Computer Science & Information Technology* 5 (2015). DOI: 10.5121/csit.2015.51501.

[Nas61] C. St.J. A. Nash-Williams. "Edge-Disjoint Spanning Trees of Finite Graphs". In: *Journal of the London Mathematical Society* s1-36.1 (1961), S. 445–450. ISSN: 00246107. DOI: 10.1112/jlms/s1-36.1.445. URL: http://doi.wiley.com/10.1112/jlms/s1-36.1.445 (besucht am 21. 01. 2021).

[neo20a] neo4j. *Betweenness Centrality - Centrality algorithms*. Graph Data Science. 2020. URL: https://neo4j.com/docs/graph-data-science/current/algorithms/betweenness-centrality/ (besucht am 27. 01. 2021).

[neo20b] neo4j. *Closeness Centrality - Centrality algorithms*. Graph Data Science. 2020. URL: https://neo4j.com/docs/graph-data-science/current/algorithms/closeness-centrality/ (besucht am 27. 01. 2021).

[neo20c] neo4j. *Eigenvector Centrality - Centrality algorithms*. Graph Data Science. 2020. URL: https://neo4j.com/docs/graph-data-science/current/algorithms/eigenvector-centrality/ (besucht am 27. 01. 2021).

[Ora17] Oracle. *PageRank and variants*. Oracle PGX 2.4.0 Documentation. 2017. URL: https://docs.oracle.com/cd/E56133_01/2.4.0/reference/algorithms/pagerank.html (besucht am 28. 01. 2021).

[Pér84] B. Péroche. "NP-completeness of some problems of partitioning and covering in graphs". In: *Discrete Applied Mathematics* 8.2 (Mai 1984), S. 195–208. ISSN: 0166218X. DOI: 10.1016/0166-218X(84)90101-X. URL: https://linkinghub.elsevier.com/retrieve/pii/0166218X8490101X (besucht am 22. 01. 2021).

[Pla83] Michael J. Plantholt. "The chromatic index of graphs with large maximum degree". In: *Discrete Mathematics* 47 (1983), S. 91–96. ISSN: 0012365X. DOI: 10.1016/0012-365X(83)90074-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0012365X83900742 (besucht am 17. 01. 2021).

[Sag] SageMath. *Generic graphs (common to directed/undirected) — Sage 9.2 Reference Manual: Graph Theory*. Sage Math Reference Manual. URL: https://doc.sagemath.org/html/en/reference/graphs/sage/graphs/generic_graph.html (besucht am 10. 11. 2020).

[Sag20a] SageMath. *Graph coloring*. 2020. URL: https://doc.sagemath.org/html/en/reference/graphs/sage/graphs/graph_coloring.html (besucht am 14. 01. 2021).

[Sag20b] SageMath. *Graph Theory*. Sage Math Reference Manual. 2020. URL: https://doc.sagemath.org/html/en/reference/graphs/index.html (besucht am 25. 10. 2020).

[Sar+13] Ahmet Erdem Sariyuce u. a. "Incremental algorithms for closeness centrality". In: 2013, S. 487–492. DOI: 10.1109/BigData.2013.6691611.

[SU11] Edward R. Scheinerman und Daniel H. Ullman. *Fractional graph theory: a rational approach to the theory of graphs*. Dover books on matehmatics. OCLC: ocn721885660. Minola, N.Y: Dover Publications, 2011. 211 S. ISBN: 978-0-486-48593-5.

[Tit19] Peter Tittmann. *Graphentheorie: Eine anwendungsorientierte Einführung*. 3., aktualisierte Auflage. München: Hanser, Carl, 2019. 168 S. ISBN: 978-3-446-46052-2 978-3-446-46503-9.

[Tru93] V. A. Trubin. “Strenght of a graph and packing of trees and branchings”. In: *Cybernetics and Systems Analysis* (1993).

[Tur04] Volker Turau. *Algorithmische Graphentheorie*. Oldenbourg Wissenschaftsverlag, 1. Jan. 2004. ISBN: 978-3-486-59377-8. DOI: 10.1524/9783486593778. URL: https://www.degruyter.com/view/title/310250 (besucht am 24. 10. 2020).

[Vöc+08] Berthold Vöcking u. a., Hrsg. *Taschenbuch der Algorithmen*. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN: 978-3-540-76393-2 978-3-540-76394-9. DOI: 10.1007/978-3-540-76394-9. URL: http://link.springer.com/10.1007/978-3-540-76394-9 (besucht am 06. 12. 2020).

[Weia] Eric W. Weisstein. *Arboricity*. Publisher: Wolfram Research, Inc. URL: https://mathworld.wolfram.com/Arboricity.html (besucht am 20. 01. 2021).

[Weib] Eric W. Weisstein. *Chromatic Number*. Publisher: Wolfram Research, Inc. URL: https://mathworld.wolfram.com/ChromaticNumber.html (besucht am 15. 01. 2021).

[WM19] Peter Wills und Francois G. Meyer. *Metrics for Graph Comparison: A Practitioner's Guide*. _eprint: 1904.07414. 2019.

[Wol20a] Wolfram. *Graph Measures & Metrics*. Wolfram Language Documentation. 2020. URL: https://reference.wolfram.com/language/guide/GraphMeasures.html (besucht am 25. 10. 2020).

[Wol20b] Wolfram. *Wolfram Function Repository*. 2020. URL: https://resources.wolframcloud.com/FunctionRepository/ (besucht am 14. 01. 2021).

[Wol20c] Wolfram. *Wolfram Language & System Documentation Center*. Documentation Center. 2020. URL: https://reference.wolfram.com/language/ (besucht am 06. 12. 2020).