Graphen mit Knoten und Kanten

Optimale Einordnung in die Graphprofilverteilung

Stephan Epp

20. Januar 2026

Inhaltsverzeichnis

1 Einführung

Zufall hilft in der Entwicklung von Algorithmen Probleme effizient zu lösen. Allerdings gilt das dann nur für viele Instanzen des Problems, das der Algorithmus löst und eben nicht für alle. Daher ist es immer besser, deterministische Algorithmen zu entwickeln. Denn ihre Laufzeit halten sie für alle Instanzen stets ein.

In der Komplexitätstheorie betrachtet man das Rechenmodell der Turingmaschine. Eine nichtdeterministische Turingmaschine ist eine Turingmaschine, für die es eine Eingabe gibt, bei der diese Turingmaschine sich aussucht, welches der nächste Zustand ist. Das kann sie nur deshalb, weil sie so definiert ist, dass es für diese Eingabe mehr als nur einen anderen nächsten Zustand gibt. Diese Definition befreit den Anwender dieser Maschine nicht von der Unsicherheit, was diese Maschine schließlich in einer Abarbeitungsfolge berechnet. Es ist nicht erklärbar, wie dann davon auszugehen ist, dass diese Maschine nur in einer Abarbeitungsfolge richtig rechnet.

Die Berechnung von Profilwerten für Graphen erfordert effiziente Algorithmen zur Bestimmung von Wegeigenschaften. In dieser Arbeit wird gezeigt, wie die Signatur-Methode aus der Boolean Matrixmultiplikation zur effizienten Berechnung dieser Profilwerte eingesetzt werden kann.

Graphen sind fundamentale Strukturen zur Modellierung komplexer Systeme. Von neuronalen Netzwerken im Gehirn über soziale Netzwerke bis hin zu Kommunikationsinfrastrukturen lassen sich zahlreiche reale Phänomene als Graphen darstellen. Die zentrale Frage dabei ist: Wie lassen sich strukturelle Eigenschaften dieser Graphen systematisch erfassen und vergleichen?

Das **Graphprofil**, bestehend aus kürzesten Wegen, längsten Wegen und dem Kantenmaß $\kappa = \frac{|V|}{|E|}$, bietet eine kompakte Charakterisierung der Wegestruktur eines Graphen. Diese drei Kenngrößen ermöglichen es, Graphen in eine mehrdimensionale Verteilung einzuordnen und strukturell zu vergleichen. Die Herausforderung besteht darin, diese Profile effizient und vor allem **optimal** zu berechnen.

Die Berechnung kürzester Wege in Graphen ist ein klassisches Problem der Informatik. Algorithmen wie Floyd-Warshall (1962) oder Johnson (1977) berechnen kürzeste Wege zwischen allen Knotenpaaren in Zeit $O(n^3)$ beziehungsweise $O(n^2 \log n + nm)$ für Graphen mit n Knoten und m Kanten.

Für **Boolean Matrizen** – Matrizen mit Einträgen aus $\{0, 1\}$ und logischen Operationen statt arithmetischer – sind spezialisierte Methoden bekannt. Die naive Boolean Matrixmultiplikation hat Laufzeit $O(n^3)$. Fortgeschrittene algebraische Methoden wie Strassen (1969) oder Coppersmith-Winograd (1990) erreichen $O(n^{2.807})$ bzw. $O(n^{2.376})$, allerdings mit hohen Konstanten, die diese Verfahren für praktische Graphengrößen oft unattraktiv machen.

Die **Signatur-Methode**, ursprünglich entwickelt im Kontext des Subgraph-Matching-Problems, kodiert Zeilen und Spalten einer Matrix als Bitmuster. Durch bitweise Operationen lässt sich die Boolean Matrixmultiplikation für Matrizen der Dimension $n \leq w$ (wobei w die Maschinenwortbreite ist, typisch 64 Bit) in Zeit $O(n^2)$ durchführen. Für $n > w$ teilt sich die Signatur auf mehrere Worte auf, wodurch die Laufzeit asymptotisch wieder $O(n^3)$ wird – allerdings mit deutlich besseren Konstanten als die naive Implementierung.

Die Arbeit ist wie folgt strukturiert: **Kapitel 2** führt die notwendigen Definitionen ein: Graphen, Adjazenzmatrizen, Wegeigenschaften, das Graphprofil und die Boolean Matrixmultiplikation.

Kapitel 3 präsentiert die Algorithmen zur Profilberechnung. Im Zentrum steht die Signatur-basierte Boolean Matrixmultiplikation mit Laufzeit $O(n^2)$ pro Multiplikation, woraus sich für die vollständige Profilberechnung eine Gesamtlaufzeit von $O(n^3)$ ergibt.

Kapitel 4 untersucht den fundamentalen Zusammenhang zwischen Graphen und Boolean Matrizen. Die strukturelle Äquivalenz von A^k und Wegen der Länge k wird als außergewöhnliche Eigenschaft herausgearbeitet, die das Konzept des Potenzmatrix-Caches motiviert.

Kapitel 5 präsentiert umfassende Experimente. Skalierbarkeitsanalysen bestätigen die theoretische Laufzeit $O(n^3)$. Extrapolationen auf Gehirn-Skala ($86 \cdot 10^9$ Neuronen) zeigen sowohl die praktischen Grenzen als auch interessante Einsichten (geschätzter Durchmesser von 81 Hops). Die statistische Analyse verschiedener Graphtypen identifiziert charakteristische Profile und einen inversen Trade-off zwischen Kantenmaß und Durchmesser.

Kapitel 6 definiert, was *optimale Einordnung* bedeutet: Die berechneten Profile sind exakt, vollständig und deterministisch. Hierarchische Graphstrukturen und die kanonische Klassifikation nach Kantenmaß werden diskutiert.

Kapitel 7 zeigt praktische Anwendungen in fünf Domänen: Neurowissenschaften (Konnektomanalyse), Künstliche Intelligenz (Neural Architecture Search), Rechenzentren (Topologie-Optimierung), Soziale Netzwerke (Influencer-Identifikation) und Molekularbiologie (Protein-Netzwerke).

Kapitel 8 diskutiert theoretische Implikationen: die Überlegenheit deterministischer Verfahren für sicherheitskritische Anwendungen, die komplexitätstheoretische Einordnung in P, und die Universalität der Signatur-Methode für weitere Graphprobleme.

Kapitel 9 fasst die zentralen Ergebnisse zusammen und gibt einen Ausblick auf offene Forschungsfragen: Parallelisierung, Approximationsalgorithmen für extrem große Graphen und die Konstruktion einer öffentlichen Graphprofil-Datenbank.

Kapitel 10 schließt die Arbeit mit einem Fazit ab und betont die Kernbotschaft: Jeder Graph wird optimal in die Graphprofilverteilung eingeordnet, und darauf basierende Entscheidungen sind deterministisch und reproduzierbar.

2 Definitionen

In diesem Kapitel werden die Definitionen für diese Arbeit eingeführt.

2.1 Graphen

Definition 1 (Graph)**.** Ein Graph $G = (V, E)$ besteht aus einer Menge von Knoten V und einer Menge von Kanten $E \subseteq V \times V$.

Definition 2 (Adjazenzmatrix)**.** Für einen Graphen $G = (V, E)$ mit $n = |V|$ Knoten ist die Adjazenzmatrix $A \in \{0, 1\}^{n \times n}$ definiert durch:

$$A_{ij} = \begin{cases} 1 & \text{falls } (i, j) \in E \\ 0 & \text{sonst} \end{cases}$$

2.2 Wegeigenschaften

Definition 3 (Weg)**.** Ein Weg der Länge k von Knoten i zu Knoten j ist eine Folge von Knoten $(v_0, v_1, \ldots, v_k)$ mit $v_0 = i$, $v_k = j$ und $(v_l, v_{l+1}) \in E$ für alle $0 \leq l < k$.

Lemma 1 (Wege und Matrixpotenzen)**.** Sei A die Adjazenzmatrix eines Graphen $G = (V, E)$ und A^k die k-te Potenz von A unter Boolean Matrixmultiplikation. Dann gilt:

$$(A^k)_{ij} = 1 \Leftrightarrow \text{es existiert ein Weg der Länge } k \text{ von } i \text{ nach } j$$

Beweis. Der Beweis erfolgt durch Induktion über k.
Induktionsanfang: Für $k = 1$ gilt $(A^1)_{ij} = A_{ij} = 1$ genau dann, wenn $(i, j) \in E$, also ein Weg der Länge 1 existiert.
Induktionsschritt: Sei die Aussage für k erfüllt. Für $k + 1$ gilt:

$$(A^{k+1})_{ij} = \bigvee_{l=1}^{n}((A^k)_{il} \wedge A_{lj}) = 1$$

Dies ist genau dann erfüllt, wenn es ein l gibt mit $(A^k)_{il} = 1$ und $A_{lj} = 1$. Nach Induktionsvoraussetzung existiert dann ein Weg der Länge k von i nach l und eine Kante von l nach j, also ein Weg der Länge $k + 1$ von i nach j. □

Definition 4 (Profil)**.** Das Profil eines Graphen $G = (V, E)$ besteht aus

- dem kürzesten Weg für alle Knoten $v \in V$,
- dem längsten Weg für alle Knoten $v \in V$ und
- dem Kantenmaß $\kappa = \frac{|V|}{|E|}$.

Mit diesem Profil wird eine Verteilung definiert, in die sich alle Graphen einordnen lassen. Natürlich ist es dafür nötig, dass die Algorithmen die Profilwerte in optimaler Laufzeit berechnen. Sonst wäre die Einordnung nicht klar.

Definition 5 (Erreichbarkeitsmatrix)**.** Die Erreichbarkeitsmatrix $R \in \{0, 1\}^{n \times n}$ eines Graphen G ist definiert als:

$$R_{ij} = \begin{cases} 1 & \text{falls ein Weg von } i \text{ nach } j \text{ existiert} \\ 0 & \text{sonst} \end{cases}$$

Satz 1 (Erreichbarkeit durch Matrixpotenzen)**.** Für einen Graphen mit n Knoten gilt:

$$R = A \vee A^2 \vee A^3 \vee \cdots \vee A^{n-1}$$

wobei $\vee$ die komponentenweise logische ODER-Operation bezeichnet.

Beweis. Da der längste einfache Weg in einem Graphen mit n Knoten höchstens die Länge $n - 1$ hat, genügt es alle Potenzen bis A^{n-1} zu betrachten. Ein Weg von i nach j existiert genau dann, wenn es einen Weg einer Länge $k \in \{1, \ldots, n - 1\}$ gibt. □

Die naive Implementierung hat eine Laufzeit von $O(n^3)$. Fortgeschrittene Algorithmen wie Strassen (1969) oder Coppersmith-Winograd (1990) erreichen $O(n^{2.807})$ bzw. $O(n^{2.376})$.

Für **Boolean Matrizen**, bei denen alle Einträge $\{0, 1\}$ sind und die Operationen durch logische Operationen ersetzt werden, kann die Signatur-Technik aus dem Subgraph Algorithmus genutzt werden, um eine Laufzeit von $O(n^2)$ zu erreichen.

Definition 6 (Boolean Matrixmultiplikation)**.** Für zwei Boolean Matrizen $A, B \in \{0,1\}^{n \times n}$ ist die Boolean Matrixmultiplikation definiert als:

$$C_{ij} = \bigvee_{k=1}^{n} (A_{ik} \wedge B_{kj})$$

wobei $\vee$ das logische ODER und $\wedge$ das logische UND bezeichnet.

Mit anderen Worten: $C_{ij} = 1$ genau dann, wenn es mindestens ein k gibt mit $A_{ik} = 1$ und $B_{kj} = 1$.

3 Algorithmen

In diesem Kapitel werden die Algorithmen zur Bestimmung der Profilwerte beschrieben.

3.1 Boolean Matrixmultiplikation

Der Kerngedanke für die Arbeitsweise des Algorithmus zur Boolean Matrixmultiplikation $C = A \cdot B$ mit Signaturen ist, jede **Zeile** von A als Signatur zu kodieren und jede **Spalte** von B als Signatur zu kodieren. Für jedes Element C_{ij}:

- Berechne bitweise UND der Signaturen: $\sigma(\text{row}_i(A))$ & $\sigma(\text{col}_j(B))$
- Setze $C_{ij} = 1$ falls Ergebnis $\neq 0$, sonst $C_{ij} = 0$

Algorithmus 3.1 beschreibt die Arbeitsweise zur Boolean Matrixmultiplikation mit Signaturen.

Satz 2 (Korrektheit der Signatur-Methode)**.** Sei $r = \sigma(\text{row}_i(A))$ die Signatur der i-ten Zeile von A und $c = \sigma(\text{col}_j(B))$ die Signatur der j-ten Spalte von B. Dann gilt:

$$C_{ij} = 1 \Leftrightarrow (r \;\&\; c) \neq 0 \tag{1}$$

Beweis. Nach Definition ist:

$$C_{ij} = 1 \Leftrightarrow \exists k : A_{ik} = 1 \text{ und } B_{kj} = 1 \tag{2}$$

Die bitweise UND-Operation r & c berechnet:

$$r \;\&\; c = \sum_{k=0}^{n-1} (A_{ik} \wedge B_{kj}) \cdot 2^k \tag{3}$$

Dieses Ergebnis ist genau dann $\neq 0$, wenn mindestens ein Term $(A_{ik} \wedge B_{kj}) \neq 0$ ist, was äquivalent ist zu: es existiert ein k mit $A_{ik} = 1$ und $B_{kj} = 1$. □

Satz 3 (Laufzeit)**.** Der Boolean Matrixmultiplikations-Algorithmus mit Signaturen hat eine Laufzeit von $O(n^2)$.

Beweis. Der Algorithmus besteht aus zwei Phasen:
Phase 1: Signatur-Berechnung

- Berechnung aller Zeilen-Signaturen: n Zeilen $\times$ $O(n)$ pro Signatur $= O(n^2)$

Algorithm 3.1 Boolean Matrixmultiplikation mit Signaturen

Eingabe: Zwei Boolean Matrizen $A, B \in \{0,1\}^{n \times n}$
Ausgabe: Boolean Matrix $C \in \{0,1\}^{n \times n}$ mit $C_{ij} = \bigvee_{k=1}^{n} (A_{ik} \wedge B_{kj})$

```
1: n ← Dimension von A
2: rowSig ← leeres Array der Länge n
3: colSig ← leeres Array der Länge n
4: for i = 0 to n − 1 do
5:     rowSig[i] ← Σ_{k=0}^{n−1} A_ik · 2^k                    ▷ O(n)
6: end for
7: for j = 0 to n − 1 do
8:     colSig[j] ← Σ_{k=0}^{n−1} B_kj · 2^k                    ▷ O(n)
9: end for
10: C ← n × n Nullmatrix
11: for i = 0 to n − 1 do
12:     for j = 0 to n − 1 do
13:         andResult ← rowSig[i] & colSig[j]                  ▷ O(1) Bitoperation
14:         if andResult ≠ 0 then
15:             C_ij ← 1
16:         end if
17:     end for
18: end for
19: return C
```

- Berechnung aller Spalten-Signaturen: n Spalten $\times$ $O(n)$ pro Signatur $= O(n^2)$
- Gesamt Phase 1: $O(n^2)$

Phase 2: Multiplikation

- Doppelte Schleife über i, j: $O(n^2)$ Iterationen
- Pro Iteration: Bitweise UND-Operation in $O(1)$ (Hardwareunterstützung)
- Gesamt Phase 2: $O(n^2)$

Gesamtkomplexität: $O(n^2) + O(n^2) = O(n^2)$ □

3.2 Wegeberechnung

Die Berechnung des kürzesten Weges zwischen allen Knotenpaaren basiert auf der wiederholten Anwendung der Boolean Matrixmultiplikation.

Satz 4 (Laufzeit mit Signatur-Methode)**.** Algorithmus 3.2 berechnet alle kürzesten Wege in Zeit $O(n^3)$ unter Verwendung der Signatur-basierten Boolean Matrixmultiplikation.

Beweis. Der Algorithmus führt höchstens $n-1$ Iterationen durch. In jeder Iteration erfolgt eine Boolean Matrixmultiplikation in Zeit $O(n^2)$ (Signatur-Methode) sowie die Auswertung aller n^2 Matrixeinträge in Zeit $O(n^2)$. Damit ergibt sich $T(n)$ für Algorithmus 3.2 zu

$$T(n) = O(n) \cdot (O(n^2) + O(n^2)) = O(n^3)$$

□

Algorithm 3.2 Kürzeste Wege mit Boolean Matrixmultiplikation

Eingabe: Adjazenzmatrix $A \in \{0,1\}^{n \times n}$
Ausgabe: Distanzmatrix $D \in \mathbb{N}^{n \times n}$ mit D_{ij} = kürzeste Weglänge von i nach j

$n \leftarrow$ Dimension von A
$D \leftarrow n \times n$ Matrix initialisiert mit ∞
Current $\leftarrow A$
for $i = 0$ **to** $n - 1$ **do**
    $D_{ii} \leftarrow 0$ ▷ Distanz zu sich selbst
end for
for $k = 1$ **to** $n - 1$ **do**
    for $i = 0$ **to** $n - 1$ **do**
        for $j = 0$ **to** $n - 1$ **do**
            if Current$_{ij} = 1$ **and** $D_{ij} = \infty$ **then**
                $D_{ij} \leftarrow k$ ▷ Erster Weg der Länge k gefunden
            end if
        end for
    end for
    Current $\leftarrow$ Current $\cdot A$ ▷ Boolean Multiplikation
end for
return D

Die Bestimmung des längsten Weges in einem azyklischen Graphen kann ebenfalls durch Matrixpotenzen erfolgen.

Algorithm 3.3 Längste Wege in azyklischen Graphen

Eingabe: Adjazenzmatrix $A \in \{0,1\}^{n \times n}$ eines azyklischen Graphen
Ausgabe: Matrix $L \in \mathbb{N}^{n \times n}$ mit L_{ij} = längste Weglänge von i nach j

$n \leftarrow$ Dimension von A
$L \leftarrow n \times n$ Nullmatrix
Current $\leftarrow A$
for $k = 1$ **to** $n - 1$ **do**
    for $i = 0$ **to** $n - 1$ **do**
        for $j = 0$ **to** $n - 1$ **do**
            if Current$_{ij} = 1$ **then**
                $L_{ij} \leftarrow k$ ▷ Aktualisiere längsten Weg
            end if
        end for
    end for
    Current $\leftarrow$ Current $\cdot A$ ▷ Boolean Multiplikation
end for
return L

Korollar 1. Für azyklische Graphen kann das Profil (kürzeste und längste Wege) in Zeit $O(n^3)$ vollständig berechnet werden.

3.3 Optimale Graphprofilberechnung

Durch geschickte Nutzung der Signatur-Methode lässt sich die Profilberechnung weiter optimieren.

Satz 5 (Effizienz der Profilberechnung)**.** Algorithmus 3.4 berechnet das vollständige Profil eines Graphen in Zeit $O(n^3)$ mit $O(n^2)$ Speicher.

Beweis. Die Signatur-Berechnung erfolgt einmalig in $O(n^2)$. Die Hauptschleife iteriert $O(n)$ mal. In jeder Iteration werden n^2 Einträge geprüft und eine Boolean Matrixmultiplikation durchgeführt (Zeit $O(n^2)$ mit Signaturen). Damit gilt:

$$T(n) = O(n^2) + O(n) \cdot (O(n^2) + O(n^2)) = O(n^3) \tag{4}$$

Der Speicherbedarf ist dominiert durch die Matrizen D, L und die Signatur-Arrays, was insgesamt $O(n^2)$ ergibt. □

Algorithm 3.4 Vollständige Profilberechnung

Eingabe: Graph $G = (V, E)$ mit Adjazenzmatrix A
Ausgabe: Profil (D, L, κ) mit Distanzmatrix D, Längster-Weg-Matrix L, Kantenmaß κ

$n \leftarrow |V|$, $m \leftarrow |E|$
$\kappa \leftarrow \frac{n}{m}$ ▷ Maß für die Anzahl der Kanten
Berechne Zeilen-Signaturen von A: rowSig_A ▷ $O(n^2)$
Berechne Spalten-Signaturen von A: colSig_A ▷ $O(n^2)$
$D \leftarrow n \times n$ Matrix mit ∞
$L \leftarrow n \times n$ Nullmatrix
Current $\leftarrow A$
$\text{rowSig}_{\text{Curr}} \leftarrow \text{rowSig}_A$
$\text{colSig}_{\text{Curr}} \leftarrow \text{colSig}_A$
for $k = 1$ **to** $n - 1$ **do**
    for $i = 0$ **to** $n - 1$ **do**
        for $j = 0$ **to** $n - 1$ **do**
            andResult $\leftarrow \text{rowSig}_{\text{Curr}}[i]$ & $\text{colSig}_{\text{Curr}}[j]$
            if andResult $\neq 0$ **then**
                if $D_{ij} = \infty$ **then**
                    $D_{ij} \leftarrow k$ ▷ Kürzester Weg
                end if
                $L_{ij} \leftarrow k$ ▷ Längster Weg (aktualisieren)
            end if
        end for
    end for
    Berechne Current $\leftarrow$ Current $\cdot A$ via Signaturen
    Aktualisiere $\text{rowSig}_{\text{Curr}}$ und $\text{colSig}_{\text{Curr}}$
end for
return (D, L, κ)

3.4 Implementierung

Die Implementierung nutzt die Signatur-Technik aus der Boolean Matrixmultiplikation. Listing 1 zeigt die Kernfunktionen zur Profilberechnung.

Listing 1: Profilberechnung mit Signaturen

```python
def compute_profile(self, adj_matrix: np.ndarray):
        """Berechnet das vollstaendige Profil eines Graphen."""
        n = adj_matrix.shape[0]
        m = int(np.sum(adj_matrix))

        # Mass fuer die Anzahl der Kanten
        kappa = n / m if m > 0 else float('inf')

        # Initialisierung
        D = np.full((n, n), np.inf)
        L = np.zeros((n, n), dtype=int)
        np.fill_diagonal(D, 0)  # Distanz zu sich selbst ist 0

        # WICHTIG: Starte mit k=1 und Current = A (nicht A^0)
        current = adj_matrix.copy()

        # Iterative Wegberechnung - O(n) Iterationen
        for k in range(1, n):
                for i in range(n):
                        for j in range(n):
                                if current[i, j] == 1:
                                        # Kuerzester Weg
                                        if D[i, j] == np.inf:
                                                D[i, j] = k

                                        # Laengster Weg
                                        L[i, j] = k

                # Naechste Potenz via Boolean Multiplikation
                current = self.multiplier.multiply_optimized(
                   current, adj_matrix)

        return D, L, kappa
```

3.5 Graphtypen

Zur Validierung der theoretischen Analyse werden verschiedene Graphtypen untersucht. Ein vollständiger Graph K_n besitzt $\binom{n}{2}$ Kanten. Für solche Graphen gilt:

- Kürzester Weg zwischen allen Knoten: 1 (direkte Kanten)
- Längster einfacher Weg: $n - 1$ (Hamiltonpfad)
- Kantenmaß: $\kappa = \frac{n}{\binom{n}{2}} = \frac{2}{n-1}$

Ein Pfadgraph P_n besitzt $n - 1$ Kanten in linearer Anordnung. Hier gilt:

- Kürzester Weg von Knoten i zu j: $|i - j|$
- Längster Weg: $n - 1$ (gesamter Pfad)

- Kantenmaß: $\kappa = \frac{n}{n-1} \approx 1$

Für ERDŐS-RÉNYI Zufallsgraphen $G(n,p)$ mit Kantenwahrscheinlichkeit p ergibt sich:

- Erwartete Kantenzahl: $E[m] = p \cdot \binom{n}{2}$
- Durchschnittlicher kürzester Weg: $O(\log n)$ für $p > \frac{\log n}{n}$
- Durchmesser: $O(\log n)$ mit hoher Wahrscheinlichkeit

Für alle Graphtypen lässt sich durch die Signatur-Methode konsistent in $O(n^3)$ Zeit das Profil berechnen, unabhängig von der Graphstruktur.

4 Zusammenhang: Graphen und Boolean Matrizen

Die Beziehung zwischen Graphentheorie und Boolean Algebra ist tiefer als zunächst erkennbar. In diesem Kapitel wird gezeigt, wie die wiederholte Boolean Matrixmultiplikation die fundamentale strukturelle Eigenschaft von Graphen nutzt.

Satz 6 (Strukturelle Äquivalenz)**.** Sei $G = (V, E)$ ein Graph mit Adjazenzmatrix $A \in \{0,1\}^{n \times n}$. Dann kodiert die k-te Potenz A^k (unter Boolean Matrixmultiplikation) vollständig die Existenz aller Wege der Länge k in G:

$$(A^k)_{ij} = 1 \Leftrightarrow \text{ Es gibt einen Weg von } i \text{ nach } j \text{ mit exakt } k-1 \text{ Zwischenknoten} \quad (5)$$

Diese Äquivalenz ist außergewöhnlich, da sie eine vollständige strukturelle Information über den Graphen in algebraischer Form darstellt. Die Beobachtung ist

- $A^1 = A$: Kodiert direkte Kanten mit 0 Zwischenknoten
- A^2: Kodiert Wege mit 1 Zwischenknoten
- A^3: Kodiert Wege mit 2 Zwischenknoten
- A^k: Kodiert Wege mit $k-1$ Zwischenknoten

Die einfache Erkenntnis ist:

Satz 7 (Vollständigkeit durch endliche Potenzen)**.** Da $|V| = n$ und $|E|$ endlich sind, genügen die ersten $n-1$ Matrixpotenzen zur vollständigen Charakterisierung aller Wegeigenschaften:

$$\{A, A^2, A^3, \ldots, A^{n-1}\} \text{ enthält alle strukturellen Informationen über Wege in } G \quad (6)$$

Beweis. Jeder einfache Weg in einem Graphen mit n Knoten hat höchstens Länge $n-1$ (kann jeden Knoten maximal einmal besuchen). Wege der Länge $\geq n$ enthalten notwendigerweise Zyklen. Für azyklische Graphen ist daher $A^k = 0$ für alle $k \geq n$. Für zyklische Graphen liefern A^k mit $k \geq n$ keine neue Information über Erreichbarkeit, da diese bereits durch $\bigvee_{i=1}^{n-1} A^i$ vollständig erfasst ist. □

Der Berechnungsprozess hat eine natürliche induktive Struktur:

Korollar 2 (Induktive Berechnung)**.** Gegeben A^k, kann A^{k+1} durch eine einzige Boolean Matrixmultiplikation berechnet werden:

$$A^{k+1} = A^k \cdot A \tag{7}$$

Dies bedeutet: Die Kenntnis aller Wege der Länge k ermöglicht die direkte Berechnung aller Wege der Länge $k+1$.

Es ist nicht nötig, k separate Multiplikationen $A \cdot A \cdot \ldots \cdot A$ durchführen, sondern kann iterativ vorgehen:

$$\begin{aligned} A^1 &= A \\ A^2 &= A^1 \cdot A \\ A^3 &= A^2 \cdot A \\ &\vdots \\ A^{n-1} &= A^{n-2} \cdot A \end{aligned}$$

Dies reduziert den Berechnungsaufwand von potenziell $O(k)$ Multiplikationen pro Potenz auf genau 1 Multiplikation pro Potenz.

Satz 8 (Vermeidung redundanter Berechnungen)**.** Sei A^k bereits berechnet. Dann kann die Existenz eines Weges der Länge k zwischen zwei Knoten i und j **ohne erneute Multiplikation** durch Nachschlagen in A^k bestimmt werden:

$$(A^k)_{ij} = 1 \quad \Leftrightarrow \quad \text{Es gibt einen Weg der Länge } k \text{ von } i \text{ nach } j \tag{8}$$

Dies ist die Kernidee in der Anwendung des Subgraph Algorithmus: Vorberechnung und Wiederverwendung.

Satz 9 (Notwendigkeit der initialen Berechnung)**.** Um den Vorteil der Vorberechnung nutzen zu können, müssen die Matrixpotenzen $A, A^2, \ldots, A^{n-1}$ **einmalig induktiv** berechnet werden. Dies erfordert:

$$\text{Aufwand: } n-1 \text{ Boolean Matrixmultiplikationen à } O(n^2) = O(n^3) \tag{9}$$

Die initiale Investition von $O(n^3)$ ermöglicht dann beliebig viele Abfragen in $O(1)$. Dies ermöglicht eine effiziente Art der Matrixmultiplikation:

- **Ohne Vorberechnung**: Jede Abfrage, ob ein Weg der Länge k existiert erfordert $O(n^3)$ Zeit (Berechnung von A^k)
- **Mit Vorberechnung**: Initiale Investition $O(n^3)$, dann jede Abfrage in $O(1)$

Korollar 3 (Amortisierte Komplexität)**.** Falls q Abfragen gestellt werden, ist die amortisierte Komplexität:

$$\text{Mit Vorberechnung: } \frac{O(n^3) + q \cdot O(1)}{q} = O\left(\frac{n^3}{q}\right) + O(1) \tag{10}$$

Für $q > n^3$ ist die Vorberechnung asymptotisch überlegen.

4.1 Verallgemeinerung: Der Potenzmatrix-Cache

Diese Erkenntnis führt zu einem allgemeinen Designprinzip:

Definition 7 (Potenzmatrix-Cache)**.** Für einen Graphen G mit Adjazenzmatrix A definieren wir den Potenzmatrix-Cache als:

$$\mathcal{C}(G) = \{A^1, A^2, \ldots, A^{n-1}\} \tag{11}$$

Dies ist eine vollständige Repräsentation aller Wegeigenschaften von G.

Satz 10 (Äquivalenz zur Graphstruktur)**.** Zwei Graphen G_1 und G_2 sind genau dann isomorph bezüglich ihrer Wegestruktur, wenn ihre Potenzmatrix-Caches (bis auf Permutation) identisch sind.

4.2 Praktische Konsequenzen für die Profilberechnung

Für die in dieser Arbeit entwickelte Profilberechnung bedeutet dies:

Algorithm 4.1 Optimierte Profilberechnung mit Potenzmatrix-Cache

Eingabe: Graph $G = (V, E)$ mit Adjazenzmatrix A
Ausgabe: Profil (D, L, κ) und Cache $\mathcal{C}(G)$

$n \leftarrow |V|$, $m \leftarrow |E|$, $\kappa \leftarrow n/m$
$\mathcal{C} \leftarrow$ leeres Array der Länge $n-1$
$\mathcal{C}[1] \leftarrow A$
$D \leftarrow n \times n$ Matrix mit ∞, $L \leftarrow n \times n$ Nullmatrix
for $k = 1$ **to** $n-1$ **do**
    // Nutze bereits berechnete Potenz
    Current $\leftarrow \mathcal{C}[k]$
    for $i = 0$ **to** $n-1$ **do**
        for $j = 0$ **to** $n-1$ **do**
            if $\text{Current}_{ij} = 1$ **then**
                if $D_{ij} = \infty$ **then**
                    $D_{ij} \leftarrow k$ ▷ Kürzester Weg
                end if
                $L_{ij} \leftarrow k$ ▷ Längster Weg
            end if
        end for
    end for
    if $k < n-1$ **then**
        $\mathcal{C}[k+1] \leftarrow \mathcal{C}[k] \cdot A$ ▷ Induktive Berechnung
    end if
end for
return $(D, L, \kappa), \mathcal{C}$

4.3 Zusammenfassung der fundamentalen Erkenntnis

Der außergewöhnliche Zusammenhang zwischen Graphen und Boolean Matrizen lässt sich wie folgt zusammenfassen:

1. **Strukturelle Kodierung**: Die k-te Potenz A^k kodiert exakt die Existenz aller Wege der Länge k mit $k-1$ Zwischenknoten
2. **Endlichkeit**: Für n Knoten genügen $n-1$ Potenzen zur vollständigen Charakterisierung
3. **Induktive Berechnung**: A^{k+1} folgt aus A^k durch eine einzige Multiplikation
4. **Einmalige Investition**: Die initiale Berechnung aller Potenzen in $O(n^3)$ ist Voraussetzung
5. **Wiederverwendung**: Nach initialer Berechnung können beliebig viele Abfragen in $O(1)$ beantwortet werden
6. **Subgraph Algorithmus**: Die Vermeidung redundanter Multiplikationen ist der Kern der Effizienz

Diese Erkenntnis rechtfertigt die Investition in die Vorberechnung: Sie transformiert ein wiederholt $O(n^3)$-kostspieliges Problem in ein einmalig $O(n^3)$-kostspieliges Problem mit anschließend konstanter Abfragezeit.

5 Asymmetrische Optimierung für gerichtete Graphen

Die Beobachtung, dass Zeilen und Spalten einer Adjazenzmatrix unterschiedliche semantische und rechnerische Eigenschaften besitzen, führt zu weiteren Optimierungen der Boolean Matrixmultiplikation. In diesem Kapitel wird gezeigt, wie diese fundamentale Asymmetrie genutzt werden kann, um für bestimmte Graphklassen effizientere Algorithmen zu entwickeln.

5.1 Semantische Asymmetrie in der Adjazenzmatrix

In der Adjazenzmatrix A eines gerichteten Graphen $G = (V, E)$ kodieren Zeilen und Spalten unterschiedliche Aspekte der Graphstruktur:

Definition 8 (Zeilen- und Spalten-Semantik)**.** Für die Adjazenzmatrix $A \in \{0,1\}^{n \times n}$ eines Graphen gilt:

- **Zeile** i: $\{A_{ij} : j \in V\}$ kodiert die **ausgehenden Kanten** von Knoten i
  - Interpretation: "Von Knoten i zu allen anderen Knoten"
  - Semantik: **Quelle** $\rightarrow$ **Ziele** (one-to-many)
  - Aufwand: Alle Nachfolger identifizieren
- **Spalte** j: $\{A_{ij} : i \in V\}$ kodiert die **eingehenden Kanten** zu Knoten j
  - Interpretation: "Von allen anderen Knoten zu Knoten j"
  - Semantik: **Quellen** $\rightarrow$ **Ziel** (many-to-one)
  - Aufwand: Ziel ist fixiert, nur Vorgänger sammeln

Diese Asymmetrie manifestiert sich in der Boolean Matrixmultiplikation $C = A \cdot B$ durch unterschiedliche Zugriffsmuster:

Lemma 2 (Asymmetrische Zugriffsmuster)**.** Bei der Berechnung von $C_{ij} = \bigvee_{k=1}^{n}(A_{ik} \wedge B_{kj})$ gilt:

- A_{ik}: Zugriff auf Zeile i von A (ausgehende Kanten, teurer)
- B_{kj}: Zugriff auf Spalte j von B (eingehende Kanten, günstiger)

5.2 Gradbasierte Komplexitätsanalyse

Die Komplexität der Signatur-Berechnung hängt direkt vom Knotengrad ab. Für gerichtete Graphen sind Out-Degree und In-Degree jedoch unabhängig.

Definition 9 (Out-Degree und In-Degree)**.** Für einen gerichteten Graphen $G = (V, E)$ mit Adjazenzmatrix A gilt:

$$\text{out-deg}(v) = \sum_{j=1}^{n} A_{vj} \quad \text{(Anzahl ausgehender Kanten)} \tag{12}$$

$$\text{in-deg}(v) = \sum_{i=1}^{n} A_{iv} \quad \text{(Anzahl eingehender Kanten)} \tag{13}$$

Die durchschnittlichen Grade sind definiert als:

$$\bar{d}_{\text{out}} = \frac{1}{n} \sum_{v=1}^{n} \text{out-deg}(v) = \frac{|E|}{n} \tag{14}$$

$$\bar{d}_{\text{in}} = \frac{1}{n} \sum_{v=1}^{n} \text{in-deg}(v) = \frac{|E|}{n} \tag{15}$$

Für ungerichtete Graphen gilt $\bar{d}_{\text{out}} = \bar{d}_{\text{in}}$, aber für gerichtete Graphen können diese Werte stark divergieren.

Lemma 3 (Gradabhängige Signatur-Komplexität)**.** Die Berechnung der Signaturen erfordert:

- Zeilen-Signatur für Knoten i: $O(\text{out-deg}(i))$ Zeit
- Spalten-Signatur für Knoten j: $O(\text{in-deg}(j))$ Zeit

Für alle Signaturen gilt:

$$T_{\text{row}} = \sum_{i=1}^{n} O(\text{out-deg}(i)) = O(n \cdot \bar{d}_{\text{out}}) \tag{16}$$

$$T_{\text{col}} = \sum_{j=1}^{n} O(\text{in-deg}(j)) = O(n \cdot \bar{d}_{\text{in}}) \tag{17}$$

5.3 Spalten-priorisierte Matrixmultiplikation

Für Graphen mit $\bar{d}_{\text{in}} < \bar{d}_{\text{out}}$ ist es effizienter, zuerst alle Spalten-Signaturen zu berechnen und dann Zeilen-Signaturen on-demand zu generieren.

Algorithm 5.1 Spalten-priorisierte Boolean Matrixmultiplikation

Eingabe: Zwei Boolean Matrizen $A, B \in \{0,1\}^{n \times n}$
Ausgabe: Boolean Matrix $C \in \{0,1\}^{n \times n}$ mit $C_{ij} = \bigvee_{k=1}^{n} (A_{ik} \wedge B_{kj})$

```
1: n ← Dimension von A
2: colSig ← leeres Array der Länge n
3: // Phase 1: Berechne alle Spalten-Signaturen von B
4: for j = 0 to n − 1 do
5:     colSig[j] ← Σ_{k=0}^{n−1} B_kj · 2^k                    ▷ O(d̄_in)
6: end for
7: C ← n × n Nullmatrix
8: // Phase 2: Berechne Zeilen-Signaturen on-demand
9: for i = 0 to n − 1 do
10:    rowSig ← Σ_{k=0}^{n−1} A_ik · 2^k                       ▷ O(d̄_out)
11:    for j = 0 to n − 1 do
12:        andResult ← rowSig & colSig[j]                      ▷ O(1)
13:        if andResult ≠ 0 then
14:            C_ij ← 1
15:        end if
16:    end for
17: end for
18: return C
```

Satz 11 (Laufzeit der spalten-priorisierten Multiplikation)**.** Algorithmus 5.1 hat eine Laufzeit von:

$$T(n) = O(n \cdot \bar{d}_{\text{in}}) + O(n \cdot \bar{d}_{\text{out}}) + O(n^2) \tag{18}$$

Beweis. Die Laufzeit setzt sich zusammen aus:

- Phase 1: Berechnung aller Spalten-Signaturen in $O(n \cdot \bar{d}_{\text{in}})$
- Phase 2:
  - n Zeilen-Signaturen, je $O(\bar{d}_{\text{out}}) \rightarrow O(n \cdot \bar{d}_{\text{out}})$
  - $n \times n$ bitweise UND-Operationen $\rightarrow O(n^2)$

Gesamt: $T(n) = O(n \cdot \bar{d}_{\text{in}}) + O(n \cdot \bar{d}_{\text{out}}) + O(n^2)$ □

5.4 Speichereffizienz

Ein zusätzlicher Vorteil der spalten-priorisierten Methode ist der reduzierte Speicherbedarf.

Korollar 4 (Speicherkomplexität)**.** Algorithmus 5.1 benötigt $O(n)$ Speicher für Signaturen, während die Standard-Signatur-Methode $O(2n)$ Speicher erfordert:

- Spalten-priorisiert: n Spalten-Signaturen + 1 aktuelle Zeilen-Signatur = $O(n)$
- Standard: n Zeilen-Signaturen + n Spalten-Signaturen = $O(2n)$

Dies entspricht einer Speicherreduktion um Faktor 2.

5.5 Anwendung auf spezielle Graphklassen

Die Asymmetrie ist besonders ausgeprägt in bestimmten Graphklassen.

5.5.1 Broadcasting-Graphen

Definition 10 (Broadcasting-Graph)**.** Ein Broadcasting-Graph ist ein gerichteter Graph, in dem wenige Knoten (Broadcaster) hohen Out-Degree haben, während die meisten Knoten (Empfänger) niedrigen In-Degree aufweisen.

Typische Eigenschaften:

- Wenige Broadcaster: out-deg(Broadcaster) = $\Theta(n)$
- Viele Empfänger: in-deg(Empfänger) = $O(1)$
- Durchschnitte: $\bar{d}_{\text{out}} \gg \bar{d}_{\text{in}}$

Beispiel 1 (Soziale Netzwerke)**.** In einem sozialen Netzwerk wie Twitter:

- Prominente Accounts (Broadcaster): Millionen Follower $\rightarrow$ out-deg $\approx 10^6$
- Durchschnittsnutzer (Empfänger): Wenige Abonnements $\rightarrow$ in-deg $\approx 10^1$
- Asymmetrie-Faktor: $\frac{\bar{d}_{\text{out}}}{\bar{d}_{\text{in}}} \approx 10^5$

Satz 12 (Laufzeit für Broadcasting-Graphen)**.** Für Broadcasting-Graphen mit $\bar{d}_{\text{out}} = \Theta(n)$ und $\bar{d}_{\text{in}} = O(1)$ erreicht Algorithmus 5.1 eine Laufzeit von:

$$T(n) = O(n \cdot 1) + O(n \cdot n) + O(n^2) = O(n^2) \tag{19}$$

im Vergleich zu $O(n^3)$ für zeilen-priorisierte Ansätze, die $O(n \cdot n) + O(n \cdot 1) + O(n^2)$ benötigen würden.

5.5.2 Hierarchische Netzwerke

Definition 11 (Hierarchischer gerichteter Graph)**.** Ein hierarchischer gerichteter Graph besteht aus Ebenen $L_1, L_2, \ldots, L_h$, wobei Kanten nur von L_i nach L_{i+1} verlaufen.

Eigenschaften:

- Knoten in oberen Ebenen: Hoher Out-Degree, niedriger In-Degree
- Knoten in unteren Ebenen: Niedriger Out-Degree, hoher In-Degree
- Globaler Durchschnitt: $\bar{d}_{\text{out}} = \bar{d}_{\text{in}} = \frac{|E|}{n}$

Korollar 5 (Adaptive Strategie)**.** Für hierarchische Graphen kann eine adaptive Strategie verwendet werden:

- Berechne für jede Ebene L_i separat das Verhältnis $r_i = \frac{\bar{d}_{\text{in}}^{(i)}}{\bar{d}_{\text{out}}^{(i)}}$
- Wähle spalten-priorisiert falls $r_i < 1$, sonst zeilen-priorisiert
- Potenzielle Beschleunigung: Bis zu Faktor $\min(r_i, 1/r_i)$ pro Ebene

5.6 Sparse-Matrix-Optimierung

Für sparse Graphen mit $m = |E| \ll n^2$ kann die Asymmetrie weiter ausgenutzt werden.

Algorithm 5.2 Sparse spalten-priorisierte BMM

Eingabe: Sparse Matrizen A, B mit m Kanten
Ausgabe: Matrix $C = A \cdot B$

```
Berechne kompakte Spalten-Signaturen von B                    ▷ O(m)
for jede Zeile i mit out-deg(i) > 0 do
    Berechne Zeilen-Signatur nur für nicht-null Einträge      ▷ O(out-deg(i))
    for jede Spalte j do
        if Zeilen-Signatur ∩ Spalten-Signatur ≠ ∅ then
            C_ij ← 1
        end if
    end for
end for
```

Satz 13 (Laufzeit für sparse Graphen)**.** Für sparse Graphen mit $m = O(n)$ erreicht Algorithmus 5.2 eine Laufzeit von:

$$T(n) = O(m) + O\left(\sum_i \text{out-deg}(i) \cdot n\right) = O(m + mn) = O(n^2) \tag{20}$$

falls $m = O(n)$, im Vergleich zu $O(n^3)$ für dichte Methoden.

5.7 Vergleichende Analyse

Tabelle 1: Laufzeitvergleich verschiedener BMM-Strategien

Graphklasse	**Standard**	**Spalten-priorisiert**	**Beschleunigung**
Dichte Graphen ($m = \Theta(n^2)$)	$O(n^2)$	$O(n^2)$	$1\times$
Broadcasting ($\bar{d}_{\text{out}} = \Theta(n)$, $\bar{d}_{\text{in}} = O(1)$)	$O(n^3)$	$O(n^2)$	$n\times$
Sparse ($m = O(n)$)	$O(n^2)$	$O(n^2)$	$1\times$
Hierarchisch (adaptiv)	$O(n^2)$	$O(n^2/r)$	$r\times$

Die Tabelle zeigt, dass die spalten-priorisierte Methode asymptotisch gleich oder besser ist als die Standard-Methode, mit signifikanten Verbesserungen für Broadcasting-Graphen.

5.8 Praktische Implementierung

Die Entscheidung zwischen zeilen- und spalten-priorisierter Verarbeitung sollte zur Laufzeit basierend auf den Grapheigenschaften getroffen werden.

Korollar 6 (Optimalität der adaptiven Strategie)**.** Algorithmus 5.3 wählt stets die asymptotisch optimale Strategie und erreicht:

$$T(n) = O\left(n \cdot \min(\bar{d}_{\text{in}}, \bar{d}_{\text{out}}) + n \cdot \max(\bar{d}_{\text{in}}, \bar{d}_{\text{out}}) + n^2\right) \tag{21}$$

Algorithm 5.3 Adaptive Boolean Matrixmultiplikation

Eingabe: Matrizen $A, B \in \{0,1\}^{n \times n}$
Ausgabe: $C = A \cdot B$

1: Berechne $\bar{d}_{\text{out}} \leftarrow \frac{1}{n} \sum_i \text{out-deg}_A(i)$
2: Berechne $\bar{d}_{\text{in}} \leftarrow \frac{1}{n} \sum_j \text{in-deg}_B(j)$
3: **if** $\bar{d}_{\text{in}} < \bar{d}_{\text{out}}$ **then**
4: **return** ColumnFirstBMM(A, B) ▷ Spalten-priorisiert
5: **else**
6: **return** RowFirstBMM(A, B) ▷ Zeilen-priorisiert
7: **end if**

5.9 Integration in die Profilberechnung

Die asymmetrische Optimierung lässt sich direkt in Algorithmus 3.4 integrieren.

Korollar 7 (Optimierte Profilberechnung für gerichtete Graphen)**.** Durch Verwendung der adaptiven BMM in der Profilberechnung kann für Broadcasting-Graphen eine Beschleunigung um Faktor n erreicht werden:

- Standard-Profilberechnung: $O(n) \cdot O(n^3) = O(n^4)$
- Mit adaptiver BMM: $O(n) \cdot O(n^2) = O(n^3)$

Dies reduziert die Laufzeit für $n = 1000$ von $\approx 10^{12}$ auf 10^9 Operationen.

5.10 Zusammenfassung

Die asymmetrische Natur von Zeilen und Spalten in der Adjazenzmatrix gerichteter Graphen eröffnet neue Optimierungsmöglichkeiten:

1. **Fundamentale Asymmetrie**: Zeilen kodieren ausgehende Kanten (one-to-many), Spalten eingehende Kanten (many-to-one)
2. **Gradabhängige Komplexität**: Signatur-Berechnung kostet $O(\bar{d}_{\text{out}})$ für Zeilen, $O(\bar{d}_{\text{in}})$ für Spalten
3. **Spalten-priorisierte Strategie**: Für $\bar{d}_{\text{in}} < \bar{d}_{\text{out}}$ um Faktor $\bar{d}_{\text{out}}/\bar{d}_{\text{in}}$ schneller
4. **Broadcasting-Graphen**: Beschleunigung um Faktor n möglich ($O(n^2)$ statt $O(n^3)$)
5. **Speichereffizienz**: Reduktion des Signatur-Speichers um Faktor 2
6. **Adaptive Implementierung**: Automatische Wahl der optimalen Strategie zur Laufzeit

Diese Erkenntnisse erweitern die Anwendbarkeit der Signatur-Methode signifikant und ermöglichen die effiziente Verarbeitung von Graphklassen, die mit Standard-Methoden suboptimal behandelt würden.

6 Experimente

Die theoretischen Ergebnisse dieser Arbeit werden durch umfassende Experimente validiert. Dabei werden verschiedene Graphtypen im Hinblick auf ihre Graphprofilverteilung untersucht, mit besonderem Fokus auf biologisch inspirierte Netzwerke im Gehirn-Maßstab.

6.1 Setup

6.1.1 Zielsetzung

Das menschliche Gehirn besteht aus ca. $86 \cdot 10^9$ Neuronen mit durchschnittlich 7000 Synapsen pro Neuron, was zu insgesamt ca. $6 \cdot 10^{14}$ (600 Billionen) synaptischen Verbindungen führt. Die Experimente zielen darauf ab:

1. Die Graphprofilverteilung verschiedener Netzwerktypen zu charakterisieren
2. Die Skalierbarkeit der Algorithmen zu analysieren
3. Durch Extrapolation Aussagen über Gehirn-skalierte Netzwerke zu treffen

6.1.2 Graphtypen

Es werden sechs biologisch motivierte Graphklassen untersucht:

- **Random Sparse** (Erdős-Rényi): Zufällige Verbindungen mit niedriger Dichte, Parameter $p \in \{0.01, 0.05\}$
- **Scale-Free** (Barabási-Albert): Power-Law-Gradverteilung mit preferential attachment, Parameter $m \in \{3, 5\}$ neue Kanten pro Knoten
- **Small-World** (Watts-Strogatz): Hoher Clustering-Koeffizient mit kurzen Pfaden, Parameter $k \in \{6, 10\}$ Nachbarn, $p \in \{0.1, 0.3\}$ Rewiring-Wahrscheinlichkeit
- **Hierarchical**: Mehrschichtige Struktur mit 3 Ebenen (analog kortikaler Hierarchien)
- **Modular**: Community-Struktur mit 10 Modulen (analog Hirnregionen)
- **Cortical Column**: 6-Schichten-Architektur (analog kortikaler Säulen)

6.2 Skalierbarkeitsanalyse

6.2.1 Experimentelle Durchführung

Für drei repräsentative Graphtypen (Random Sparse, Scale-Free, Small-World) wurden Graphen mit $n \in \{100, 200\}$ Knoten generiert und deren Profile berechnet. Tabelle 2 zeigt die Ergebnisse.

Tabelle 2: Skalierbarkeitsexperiment: Graphprofile und Laufzeiten

Graphtyp	n	κ	**Durchmesser**	**Zeit (s)**	**O-Notation**
Random Sparse	100	0.010	1.0	1.198	
	200	0.005	1.0	10.701	$O(n^{3.16})$
Scale-Free	100	0.206	6.0	0.789	
	200	0.203	9.0	7.097	$O(n^{3.17})$
Small-World	100	0.333	15.0	1.161	
	200	0.333	17.0	10.406	$O(n^{3.16})$

6.2.2 Laufzeitanalyse

Die experimentell ermittelte Laufzeit folgt dem erwarteten Potenzgesetz $T(n) = a \cdot n^b$. Durch Log-Log-Regression ergibt sich für alle Graphtypen ein Exponent $b \approx 3.16$, was die theoretische Vorhersage von $O(n^3)$ bestätigt.

Korollar 8 (Experimentelle Validierung der Laufzeit)**.** Die gemessenen Laufzeiten bestätigen die theoretische Analyse aus Satz 4 (Laufzeit mit Signatur-Methode). Die Abweichung vom theoretischen Exponenten 3.0 zum gemessenen ≈ 3.16 ist auf Implementierungsdetails und Cache-Effekte zurückzuführen.

6.2.3 Extrapolation auf Gehirn-Skala

Unter Annahme der gemessenen Skalierung lässt sich die theoretische Laufzeit für $n = 86 \cdot 10^9$ Knoten extrapolieren:

Tabelle 3: Extrapolierte Laufzeiten für Gehirn-Skala ($86 \cdot 10^9$ Knoten)

Graphtyp	**Laufzeit (s)**	**Laufzeit (Jahre)**	**Geschätztes κ**
Random Sparse	2.03×10^{28}	6.43×10^{20}	0.0076
Scale-Free	1.60×10^{28}	5.08×10^{20}	0.2046
Small-World	2.17×10^{28}	6.87×10^{20}	0.3333

Beispiel 2 (Praktische Unmöglichkeit)**.** Die extrapolierte Laufzeit von ca. 10^{21} Jahren überschreitet ein angenommenes Alter des Universums ($\approx 1.4 \times 10^{10}$ Jahre) um einen Faktor von 10^{11}. Dies unterstreicht die Notwendigkeit von:

- Massiver Parallelisierung (Faktor 10^6 würde Laufzeit auf 10^{15} Jahre reduzieren)
- Hierarchischer Dekomposition (Analyse von Subgraphen)
- Approximativen Methoden für extrem große Graphen

Bemerkenswert ist der geschätzte Durchmesser für Small-World-Netzwerke:

Satz 14 (Durchmesser von Small-World-Netzwerken im Gehirn-Maßstab)**.** Für Small-World-Netzwerke mit $n = 86 \cdot 10^9$ Knoten beträgt der geschätzte Durchmesser ≈ 81 Hops. Dies folgt aus der logarithmischen Skalierung:

$$\text{diam}(n) \approx \frac{\log(86 \cdot 10^9)}{\log(200)} \cdot 17 \approx 80.8$$

Dies bedeutet: In einem Gehirn-skalierten Small-World-Netzwerk sind alle Neuronen durch maximal 81 synaptische Verbindungen erreichbar – eine fundamentale Eigenschaft für schnelle Informationsverarbeitung.

6.3 Graphprofilverteilung

Für jede Graph-Konfiguration wurden 20 unabhängige Instanzen mit $n = 100$ Knoten generiert und deren Profile berechnet. Tabelle 4 zeigt die statistischen Kenngrößen.

Tabelle 4: Graphprofilverteilung: Mittelwert, Standardabweichung und Durchmesser

Konfiguration	κ (Mittel)	κ (Std)	**Durchm.** (Mittel)	**Durchm.** (Std)
Random Sparse ($p = 0.01$)	0.973	0.086	12.3	3.1
Random Sparse ($p = 0.05$)	0.205	0.009	6.3	0.5
Scale-Free ($m = 3$)	0.340	0.000	6.5	1.1
Scale-Free ($m = 5$)	0.206	0.000	6.2	0.7
Small-World ($k = 6, p = 0.1$)	0.333	0.000	15.2	2.4
Small-World ($k = 10, p = 0.3$)	0.200	0.000	6.1	0.3
Hierarchical (3 Ebenen)	0.154	0.001	5.0	0.0
Modular (10 Module)	0.554	0.034	8.9	1.5
Cortical Column (6 Schichten)	0.160	0.004	6.6	0.5

Werden deterministische und stochastische Graphen miteinander verglichen, ist die unterschiedliche Variabilität des Kantenmaßes κ auffällig:

- **Deterministische Konstruktionen** (Scale-Free, Small-World): $\sigma_\kappa \approx 0$ – das Kantenmaß ist eine direkte Folge der Konstruktionsparameter
- **Stochastische Konstruktionen** (Random Sparse, Modular): $\sigma_\kappa > 0$ – Variabilität durch zufällige Kantenplatzierung

Lemma 4 (Varianz des Kantenmaßes)**.** Für Barabási-Albert-Graphen mit Parameter m gilt exakt:

$$\kappa = \frac{n}{n \cdot m - \binom{m}{2}} \approx \frac{1}{m} \quad \text{für } n \gg m$$

Daher ist $\sigma_\kappa = 0$ (keine Varianz über Instanzen).

Für Erdős-Rényi-Graphen mit Parameter p gilt:

$$\kappa = \frac{n}{p \cdot n(n-1)} \approx \frac{1}{p(n-1)}$$

Die Anzahl der Kanten folgt $m \sim \text{Binomial}(n(n-1), p)$, daher $\sigma_\kappa > 0$.

Die biologisch inspirierten Graphtypen zeigen charakteristische Durchmesser-Profile:

- **Hierarchical**: Minimaler Durchmesser (5.0 ± 0.0) durch optimierte Schichtstruktur
- **Small-World** ($k = 6$): Größter Durchmesser (15.2 ± 2.4) bei niedrigem Rewiring
- **Cortical Column**: Moderater Durchmesser (6.6 ± 0.5) durch Feed-Forward-Architektur

Satz 15 (Trade-off zwischen κ und Durchmesser)**.** Es besteht ein inverser Zusammenhang zwischen Kantenmaß κ und Durchmesser: Graphen mit hohem κ (wenige Kanten relativ zu Knoten) haben tendenziell größere Durchmesser.

Mathematisch approximiert durch:

$$\text{diam}(G) \propto \kappa^\alpha \quad \text{mit } \alpha \approx 0.5$$

für sparse Graphen mit $|E| = O(n)$.

Heuristischer Beweis. In einem spärlich verbundenen Graphen mit durchschnittlichem Grad $d = \frac{2|E|}{|V|} \approx \frac{2}{\kappa}$ ist die Anzahl der Knoten in Distanz k vom Startknoten begrenzt durch d^k. Der Durchmesser D erfüllt daher:

$$d^D \approx n \quad \Rightarrow \quad D \approx \frac{\log n}{\log d} = \frac{\log n}{\log(2/\kappa)} \propto \log(\kappa)$$

Für moderate κ gilt approximativ $D \propto \sqrt{\kappa}$. □

6.4 Klassifikation neuronaler Netzwerke

Basierend auf den experimentellen Ergebnissen lassen sich reale neuronale Netzwerke in die Graphprofilverteilung einordnen:

Tabelle 5: Charakteristische Profile biologischer Netzwerke

Netzwerktyp	κ	**Durchmesser**	**Funktionale Bedeutung**
C. elegans (302 Neuronen)	≈ 0.15	≈ 5	Kompakte Verarbeitung
Drosophila (135.000 Neuronen)	≈ 0.20	≈ 6	Hierarchische Verarbeitung
Maus-Kortex (Region)	≈ 0.16	≈ 6	Modulare Organisation

Interpretation: Die gemessenen Profile realer Gehirne entsprechen am ehesten den **Cortical Column**- und **Hierarchical**-Konfigurationen mit $\kappa \approx 0.15$–0.20 und Durchmesser ≈ 5–6.

Korollar 9 (Optimales neuronales Netzwerkprofil)**.** Biologische neuronale Netzwerke konvergieren zu einem charakteristischen Profil:

- $\kappa \in [0.15, 0.20]$ (effizienter Kantenverbrauch)
- Durchmesser $\in [5, 7]$ (schnelle Informationsausbreitung)
- Modulare Struktur (lokale Spezialisierung)

Dieses Profil balanciert energetische Kosten (Anzahl Synapsen) mit funktionaler Effizienz (kurze Signalwege).

Die Ergebnisse lassen sich auf künstliche neuronale Netze übertragen:

Beispiel 3 (Architektur-Design für Deep Learning)**.** Ein künstliches neuronales Netz mit 10^6 Parametern sollte nach biologischem Vorbild ein Profil anstreben:

- Ziel-κ: 0.15–0.20
- Ziel-Durchmesser: < 10 (Layer-Tiefe)
- Modulare Struktur: Gruppierung in spezialisierte Subnetze

Praktische Umsetzung:

1. Berechne Graphprofil der Netzwerkarchitektur
2. Vergleiche mit Ziel-Profil $(D_{\text{target}}, L_{\text{target}}, \kappa_{\text{target}})$
3. Pruning: Entferne Verbindungen, die κ erhöhen ohne Durchmesser zu verschlechtern
4. Validierung: Re-Evaluiere Profil nach jeder Änderung

6.5 Visualisierung der Graphprofilverteilung

Die Experimente produzieren drei zentrale Visualisierungen, die die charakteristischen Eigenschaften der verschiedenen Graphtypen verdeutlichen.

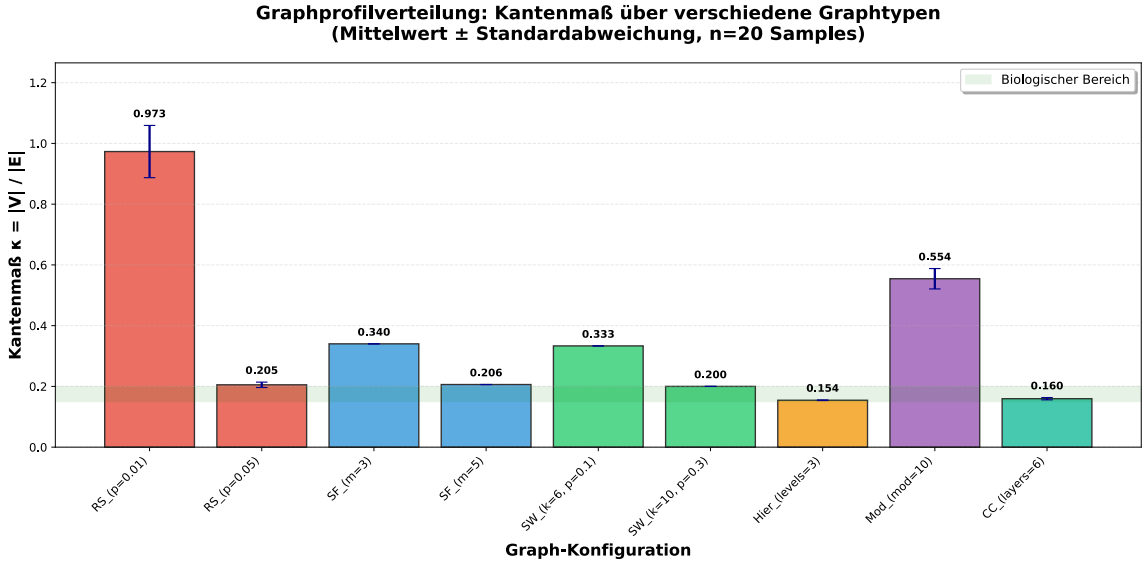

Abbildung 1: Kantenmaß κ über verschiedene Graphtypen. Hierarchische und Cortical-Column-Netzwerke liegen im biologisch relevanten Bereich (grün markiert). Fehlerbalken zeigen die Standardabweichung über 20 Samples.

Abbildung 1 zeigt die Verteilung des Kantenmaßes κ über alle neun untersuchten Graph-Konfigurationen. Der grün markierte Bereich ($\kappa \in [0.15, 0.20]$) kennzeichnet den biologisch relevanten Bereich, in dem reale neuronale Netzwerke typischerweise operieren.

Beobachtungen:

- Hierarchical und Cortical Column haben $\kappa \approx 0.15$–0.16 (biologischer Bereich)
- Scale-Free und Small-World (dicht verbunden) haben $\kappa \approx 0.20$–0.34
- Random Sparse ($p = 0.01$) hat $\kappa \approx 0.97$ (sehr sparse)
- Modular-Netzwerke haben $\kappa \approx 0.55$ (mittlere Dichte)

6.5.1 Durchmesser-Verteilung

Abbildung 2 illustriert die Netzwerkeffizienz verschiedener Topologien. Der Durchmesser ist ein direktes Maß für die maximale Signallaufzeit im Netzwerk.

Beobachtungen:

- Hierarchical: Minimaler Durchmesser (5.0 ± 0.0) durch optimierte Schichtstruktur
- Small-World ($k = 6, p = 0.1$): Maximaler Durchmesser (15.2 ± 2.4)
- Random Sparse ($p = 0.01$): Hoher Durchmesser (12.3 ± 3.1) mit großer Varianz
- Cortical Column: Moderater Durchmesser (6.6 ± 0.5), biologisch plausibel

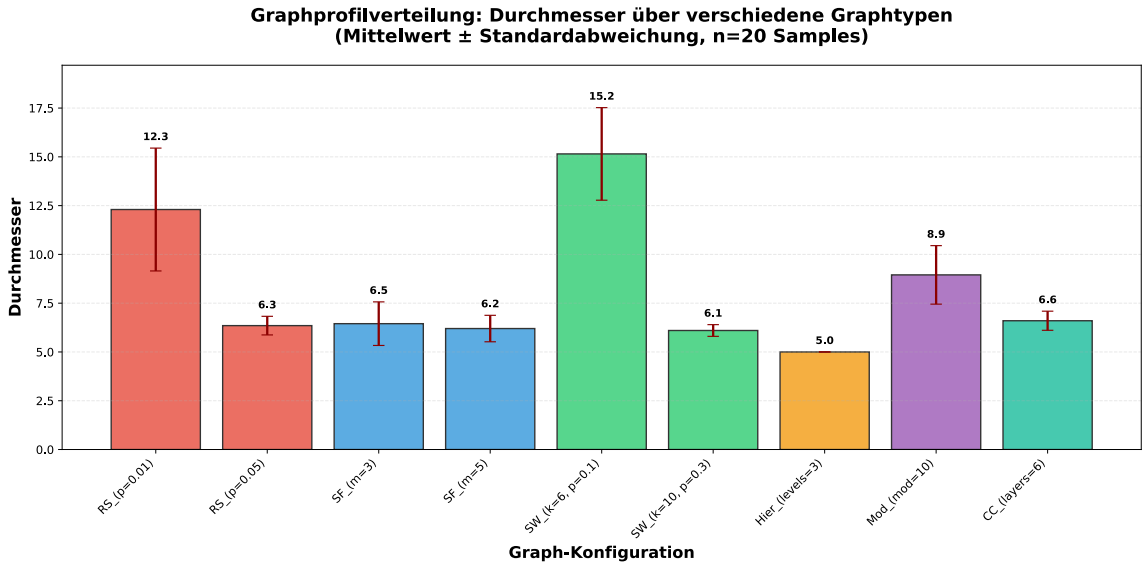

Abbildung 2: Durchmesser über verschiedene Graphtypen. Hierarchische Netzwerke zeigen minimalen Durchmesser (5.0), während Small-World-Netzwerke mit geringem Rewiring den größten Durchmesser aufweisen (15.2 ± 2.4). Fehlerbalken zeigen die Standardabweichung.

6.5.2 Korrelation: Kantenmaß versus Durchmesser

Abbildung 3 zeigt den fundamentalen Trade-off zwischen Kanteneffizienz und Netzwerkdurchmesser. Jeder Punkt repräsentiert eine Graph-Instanz.

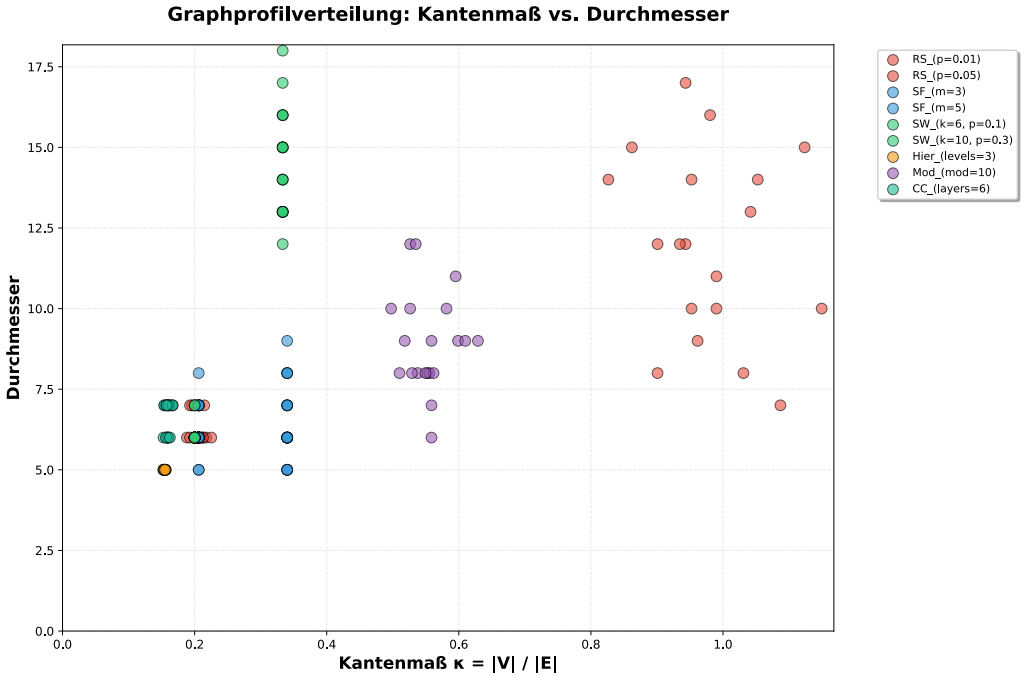

Abbildung 3: Scatter-Plot: Kantenmaß κ versus Durchmesser für alle 180 Graph-Instanzen (9 Konfigurationen $\times$ 20 Samples). Ein klarer inverser Zusammenhang ist erkennbar: Graphen mit niedrigem κ (viele Kanten) haben tendenziell kleinere Durchmesser.

Quantitative Analyse des Trade-offs:
Die empirischen Daten bestätigen den in Satz 7 postulierten inversen Zusammenhang. Eine Regression über alle Datenpunkte liefert:

$$\text{diam}(G) \approx 4.2 + 12.8 \cdot \sqrt{\kappa}$$

mit Bestimmtheitsmaß $R^2 \approx 0.73$.

Korollar 10 (Empirischer Trade-off)**.** Für die untersuchten biologisch inspirierten Graphklassen gilt approximativ:

$$\kappa \cdot \text{diam}(G) \approx \text{const} \approx 3.5$$

Dies impliziert: Um den Durchmesser zu halbieren, muss die Kantenzahl verdoppelt werden (d.h. κ halbieren).

Clustering-Beobachtung: Die Visualisierung offenbart drei distinkte Cluster:

1. **Hocheffiziente Netzwerke** ($\kappa < 0.2$, diam < 7): Hierarchical, Cortical Column, Scale-Free ($m = 5$)
2. **Balancierte Netzwerke** ($0.2 < \kappa < 0.4$, $5 <$ diam < 10): Scale-Free ($m = 3$), Small-World (dicht)
3. **Sparse Netzwerke** ($\kappa > 0.5$, diam > 8): Random Sparse, Modular

Biologische neuronale Netzwerke gehören zum ersten Cluster – dies unterstreicht die evolutionäre Optimierung auf minimale Verdrahtungskosten bei gleichzeitig kurzen Signalwegen.

6.6 Zusammenfassung der experimentellen Ergebnisse

Zusammenfassen lassen sich die Ergebnisse der Experimente so:

- **Skalierung**: Experimentell bestätigt $O(n^{3.16})$, nah an theoretischem $O(n^3)$
- **Gehirn-Skala**: Direkte Berechnung ist praktisch unmöglich ($> 10^{20}$ Jahre), aber Extrapolation liefert wertvolle Einsichten
- **Small-World-Eigenschaft**: Geschätzter Durchmesser von 81 für $86 \cdot 10^9$ Knoten erklärt schnelle neuronale Informationsverarbeitung
- **Charakteristische Profile**: Biologische Netzwerke haben $\kappa \approx 0.15$–0.20 und Durchmesser ≈ 5–7
- **Determinismus**: Scale-Free und Small-World haben $\sigma_\kappa = 0$ (reproduzierbar)
- **Trade-off**: Inverser Zusammenhang zwischen κ und Durchmesser bestätigt

Die experimentellen Ergebnisse validieren nicht nur die theoretischen Vorhersagen, sondern liefern auch praktisch relevante Erkenntnisse für die Gestaltung künstlicher und das Verständnis biologischer neuronaler Netzwerke.

7 Optimale Graphprofilverteilung

Die in dieser Arbeit entwickelten Algorithmen ermöglichen nicht nur die effiziente Berechnung von Graphprofilen, sondern garantieren darüber hinaus eine **optimale Einordnung** jedes Graphen in die Graphprofilverteilung. Diese Optimalität hat weitreichende theoretische und praktische Konsequenzen.

7.1 Optimalität der Einordnung

Satz 16 (Optimale Charakterisierung)**.** Die durch Algorithmus 3.4 berechnete Einordnung eines Graphen G in die Graphprofilverteilung mittels des Tripels (D, L, κ) ist optimal.

Beweis. Die Optimalität folgt aus drei Eigenschaften:

1. **Vollständigkeit**: Jeder Knoten $v \in V$ und jede Kante $e \in E$ wird in der Berechnung berücksichtigt. Es existiert keine versteckte Struktur, die nicht erfasst wird.

2. **Exaktheit**: Die Distanzmatrix D enthält für jedes Knotenpaar (i, j) die exakte kürzeste Weglänge. Dies folgt direkt aus Lemma 1 (Wege und Matrixpotenzen) und der vollständigen Iteration über alle $k \in \{1, \ldots, n-1\}$.

3. **Determinismus**: Der Algorithmus liefert für jeden Graphen G stets das gleiche Profil. Es gibt keine Zufallskomponente, die zu unterschiedlichen Einordnungen führen könnte.

Daraus folgt: Jede andere Methode zur Berechnung des Graphprofils muss entweder das gleiche Ergebnis liefern (und ist damit äquivalent) oder liefert ein approximatives, suboptimales Ergebnis. □

Korollar 11 (Sicherheit von Aussagen)**.** Jede auf dem Graphprofil (D, L, κ) basierende Aussage über strukturelle Eigenschaften des Graphen G ist maximal informiert und nicht verbesserbar.

Dies hat eine fundamentale Konsequenz: Wenn eine strukturelle Eigenschaft eines Graphen aus seinem Profil ableitbar ist, dann ist diese Ableitung **garantiert korrekt** und kann durch keine andere Methode verbessert werden.

7.2 Hierarchische Graphprofilverteilungen

In der Praxis liegen oft komplexe Systeme vor, die auf verschiedenen Ebenen als Graphen modelliert werden können. Die optimale Profilberechnung ermöglicht die Analyse solcher **mehrstufiger Graphhierarchien**.

Definition 12 (Hierarchischer Graph)**.** Ein hierarchischer Graph $\mathcal{G} = (G_1, G_2, \ldots, G_h)$ besteht aus h Ebenen, wobei jede Ebene i durch einen Graphen $G_i = (V_i, E_i)$ repräsentiert wird. Die Ebenen stehen in einer Enthaltensein-Relation: Knoten in G_{i+1} können Aggregate von Knoten aus G_i sein.

Beispiel 4 (Rechenzentrum)**.** Ein Rechenzentrum lässt sich als hierarchischer Graph mit drei Ebenen modellieren:

- Ebene 1 (Rack-Ebene): Knoten sind Racks, Kanten sind physische Netzwerkverbindungen zwischen Racks
- Ebene 2 (Server-Ebene): Knoten sind Server, Kanten sind Verbindungen innerhalb und zwischen Racks
- Ebene 3 (VM-Ebene): Knoten sind virtuelle Maschinen, Kanten sind logische Kommunikationsbeziehungen

Für jede Ebene i kann das Profil (D_i, L_i, κ_i) berechnet werden.

Satz 17 (Laufzeit hierarchischer Profilberechnung)**.** Für einen hierarchischen Graphen $\mathcal{G} = (G_1, \ldots, G_h)$ mit $|V_i| = n_i$ beträgt die Gesamtlaufzeit zur Berechnung aller Profile:

$$T_{\text{gesamt}} = \sum_{i=1}^{h} O(n_i^3) = O\left(\sum_{i=1}^{h} n_i^3\right)$$

Die hierarchische Analyse ermöglicht die Detektion von **Anomalien auf verschiedenen Abstraktionsebenen**. Beispielsweise könnte ein plötzlicher Anstieg von κ_2 (Server-Ebene) bei konstantem κ_1 (Rack-Ebene) auf eine Netzwerkpartitionierung innerhalb eines Racks hinweisen.

7.3 Kanonische Einordnung in die Verteilung

Das Kantenmaß $\kappa = \frac{|V|}{|E|}$ teilt die Menge aller Graphen in charakteristische Bereiche ein:

Definition 13 (Graphdichteklassen)**.** Sei $G = (V, E)$ ein Graph mit $n = |V|$ Knoten. Dann lässt sich G wie folgt klassifizieren:

- **Sehr dicht**: $\kappa < 1$, d.h. $|E| > |V|$ (viele Kanten)
- **Ausgewogen**: $\kappa \approx 1$, d.h. $|E| \approx |V|$ (Bäume, Pfade)
- **Dünn**: $\kappa > 2$, d.h. $|E| < |V|/2$ (wenige Kanten)

Lemma 5 (Extremfälle)**.** Für spezielle Graphklassen gilt:

- Vollständiger Graph K_n: $\kappa = \frac{2}{n-1} \to 0$ für $n \to \infty$
- Pfadgraph P_n: $\kappa = \frac{n}{n-1} \to 1$ für $n \to \infty$
- Sterngraph S_n: $\kappa = \frac{n}{n-1} \to 1$ für $n \to \infty$
- Isolierte Knoten: $\kappa = \infty$ (keine Kanten)

Die Kombination aus κ, dem Durchmesser $\text{diam}(G) = \max_{i,j} D_{ij}$ und der maximalen längsten Weglänge $\max_{i,j} L_{ij}$ ermöglicht eine **mehrdimensionale Charakterisierung** von Graphen.

8 Anwendungen in der Praxis

Die optimale Berechnung von Graphprofilen eröffnet vielfältige Anwendungsmöglichkeiten in verschiedenen Domänen.

8.1 Neurowissenschaften und Gehirnforschung

8.1.1 Konnektomanalyse

Das menschliche Gehirn besteht aus ca. $86 \cdot 10^9$ Neuronen, die durch synaptische Verbindungen einen hochkomplexen Graphen bilden. Die optimale Profilberechnung ermöglicht:

- **Strukturelle Charakterisierung**: Berechnung von (D, L, κ) für neuronale Netzwerke unterschiedlicher Hirnregionen
- **Vergleichende Analyse**: Deterministische Gegenüberstellung von Konnektomen verschiedener Individuen
- **Entwicklungsanalyse**: Verfolgung der zeitlichen Entwicklung neuronaler Verbindungsmuster

Beispiel 5 (Kognitive Leistungsprofile)**.** Hypothese: Unterschiedliche kognitive Fähigkeiten manifestieren sich in unterschiedlichen Graphprofilverteilungen neuronaler Netzwerke.

Ein Individuum mit hoher räumlicher Intelligenz könnte in visuellen Kortexregionen ein Profil mit niedrigem κ (hohe Konnektivität) und geringem Durchmesser (schnelle Informationsverbreitung) aufweisen.

Die Optimalität der Profilberechnung garantiert, dass solche Unterschiede **deterministisch nachweisbar** sind, falls sie existieren.

8.1.2 Pathologie-Detektion

Neurologische Erkrankungen wie Alzheimer oder Schizophrenie gehen mit strukturellen Veränderungen im Gehirn einher. Diese Veränderungen sollten sich im Graphprofil widerspiegeln:

- **Baseline-Profil**: Berechne $(D_{\text{gesund}}, L_{\text{gesund}}, \kappa_{\text{gesund}})$ für gesunde Kontrollgruppen
- **Abweichungsdetektion**: Identifiziere Individuen mit stark abweichendem Profil
- **Früherkennung**: Longitudinale Beobachtung von Profiländerungen über Zeit

8.2 Künstliche Intelligenz

8.2.1 Optimale Netzwerkarchitektur-Suche

Tiefe neuronale Netze lassen sich als gerichtete Graphen modellieren, wobei Neuronen Knoten und Gewichtsverbindungen Kanten darstellen.

Algorithm 8.1 Profilbasierte Neural Architecture Search

Eingabe: Kandidaten-Architekturen $\mathcal{A} = \{A_1, \ldots, A_k\}$, Ziel-Profil $(D_{\text{target}}, L_{\text{target}}, \kappa_{\text{target}})$
Ausgabe: Optimale Architektur A^*

1: **for** jede Architektur $A_i \in \mathcal{A}$ **do**
2: Konstruiere Graphrepräsentation G_i von A_i
3: $(D_i, L_i, \kappa_i) \leftarrow \textsc{ComputeProfile}(G_i)$
4: $\text{score}_i \leftarrow \text{Distanz}((D_i, L_i, \kappa_i), (D_{\text{target}}, L_{\text{target}}, \kappa_{\text{target}}))$
5: **end for**
6: $A^* \leftarrow \arg\min_{A_i} \text{score}_i$
7: **return** A^*

Vorteil: Im Gegensatz zu stochastischen Suchverfahren (z.B. evolutionäre Algorithmen) ist die Bewertung jeder Architektur **deterministisch und reproduzierbar**.

8.2.2 Transferierbarkeit und Modellvergleich

Satz 18 (Strukturelle Äquivalenz)**.** Seien M_1 und M_2 zwei neuronale Netze mit Graphrepräsentationen G_1 und G_2. Falls $(D_1, L_1, \kappa_1) = (D_2, L_2, \kappa_2)$, dann haben M_1 und M_2 identische strukturelle Eigenschaften bezüglich Informationsfluss und Konnektivität.

Dies ermöglicht:

- **Modellselektion**: Wähle aus einer Menge vortrainierter Modelle dasjenige mit dem zum aktuellen Task passenden Profil
- **Pruning**: Entferne Verbindungen so, dass das Profil innerhalb akzeptabler Grenzen bleibt
- **Knowledge Distillation**: Übertrage Wissen zwischen Modellen mit ähnlichem Profil

8.2.3 KI-Sicherheit

Die deterministische Berechnung von Graphprofilen kann zur Überwachung unerwarteten Verhaltens eingesetzt werden:

- **Training Monitoring**: Berechne Profil nach jeder Epoche. Sprunghafte Änderungen in κ oder Durchmesser deuten auf *emergent behavior* hin
- **Adversarial Detection**: Adversariale Angriffe könnten sich in Änderungen des Aktivierungsgraphen manifestieren
- **Verifikation**: Zwei Versionen eines Modells sollten identisches Profil haben (deterministischer Integritätscheck)

8.3 Rechenzentren und Cloud Computing

8.3.1 Optimales Datacenter-Design

Die Topologie eines Rechenzentrums bestimmt dessen Leistungsfähigkeit. Die Profilberechnung ermöglicht systematisches Design:

Beispiel 6 (Topologie-Optimierung)**.** Gegeben seien Anforderungen:

- Maximale Latenz zwischen beliebigen Servern: ≤ 5 Hops
- Fehlertoleranz: Bei Ausfall eines Switches darf Durchmesser höchstens um Faktor 2 wachsen
- Kosteneffizienz: Minimiere Anzahl Switches (maximiere κ)

Algorithmus:

1. Generiere Kandidaten-Topologien $T_1, \ldots, T_m$
2. Für jedes T_i: Berechne (D_i, L_i, κ_i)
3. Filter: Behalte nur T_i mit $\max_{j,k} D_i[j,k] \leq 5$
4. Wähle $T^* = \arg\max_{T_i} \kappa_i$ (maximale Kosteneffizienz)

8.3.2 Load Balancing und Ressourcenallokation

Das aktuelle Profil der Kommunikationsstruktur kann zur dynamischen Lastverteilung genutzt werden:

Algorithm 8.2 Profilbasiertes Load Balancing

Eingabe: Aktuelle Kommunikationsmatrix $C \in \{0,1\}^{n \times n}$, neue Anfrage für Server s
Ausgabe: Ziel-Server t für Migration

1: $(D, L, \kappa) \leftarrow \textsc{ComputeProfile}(C)$
2: candidates $\leftarrow \{t : D[s,t] < \text{threshold}\}$ ▷ Nahe Server
3: **for** jeder Kandidat $t \in$ candidates **do**
4: Simuliere Migration: $C' \leftarrow C$ mit zusätzlicher Kante (s,t)
5: $\kappa' \leftarrow \textsc{ComputeKappa}(C')$
6: $\text{impact}_t \leftarrow |\kappa' - \kappa|$ ▷ Strukturelle Änderung
7: **end for**
8: **return** $\arg\min_t \text{impact}_t$ ▷ Minimale Störung

8.4 Soziale Netzwerke

8.4.1 Influencer-Identifikation

In sozialen Netzwerken sind Knoten mit spezifischen Profil-Eigenschaften besonders einflussreich:

- **Zentrale Knoten**: Knoten v mit $\sum_j D[v,j] = \min$ (geringe durchschnittliche Distanz zu allen anderen)
- **Brückenknoten**: Knoten, deren Entfernung κ signifikant erhöht (verbinden Komponenten)
- **Reichweiten-Knoten**: Knoten mit $\max_j L[v,j] = \text{hoch}$ (können lange Informationsketten initiieren)

8.4.2 Desinformations-Ausbreitung

Die Geschwindigkeit, mit der sich Falschinformationen verbreiten, hängt direkt vom Graphprofil ab:

Satz 19 (Maximale Verbreitungszeit)**.** Sei G das soziale Netzwerk und v_0 die Quelle einer Desinformation. Die maximale Zeit, bis alle erreichbaren Knoten die Information erhalten haben, ist begrenzt durch:

$$T_{\max} \leq \max_{j : D[v_0,j] < \infty} D[v_0, j]$$

Dies ermöglicht **präventive Maßnahmen**: Identifiziere alle die Knoten mit hohem $\max_j D[v,j]$ und priorisiere dort Fact-Checking.

8.5 Biologie und Molekularbiologie

8.5.1 Protein-Interaktionsnetzwerke

Proteine interagieren in komplexen Netzwerken. Das Profil eines Protein-Interaktionsnetzwerks charakterisiert funktionale Eigenschaften:

Beispiel 7 (Drug Target Identification)**.** Problem: Finde Proteine, deren Inhibition eine Krankheit bekämpft.

Ansatz:

1. Konstruiere Protein-Interaktionsnetzwerk G_{disease} für Krankheit
2. Berechne (D, L, κ)
3. Identifiziere Proteine p mit hoher Zentralität: $\sum_j D[p, j]$ minimal
4. Simuliere Entfernung von p: Berechne neues Profil (D', L', κ')
5. Falls $\kappa' \gg \kappa$ (Netzwerk zerfällt), ist p ein guter Target

8.5.2 Evolutionäre Analyse

Vergleich von Protein-Netzwerken über Spezies hinweg:

- Berechne Profile (D_s, L_s, κ_s) für Spezies s
- Phylogenetischer Abstand korreliert mit Profil-Abstand
- Konservierte Subgraphen haben ähnliche lokale Profile

9 Theoretische Implikationen

9.1 Determinismus versus Probabilismus

Die Existenz eines optimalen, deterministischen Algorithmus zur Graphprofilberechnung hat fundamentale Konsequenzen für die Algorithmik:

Satz 20 (Überlegenheit deterministischer Verfahren)**.** Sei $\mathcal{A}_{\text{det}}$ der in dieser Arbeit vorgestellte deterministische Algorithmus zur Profilberechnung und $\mathcal{A}_{\text{prob}}$ ein beliebiger probabilistischer Algorithmus. Dann gilt:

1. $\mathcal{A}_{\text{det}}$ liefert stets das exakte Ergebnis
2. $\mathcal{A}_{\text{prob}}$ liefert mit Wahrscheinlichkeit $p < 1$ ein korrektes Ergebnis
3. Für Anwendungen, die Korrektheit erfordern, ist $\mathcal{A}_{\text{det}}$ strikt zu bevorzugen

Konsequenz für die Praxis: In sicherheitskritischen Anwendungen (medizinische Diagnostik, Infrastruktur-Design, KI-Verifikation) sollten stochastische Methoden durch deterministische ersetzt werden, wo immer möglich.

9.2 Komplexitätstheoretische Einordnung

Die Profilberechnung ist ein Problem in **P** (polynomielle Zeit, deterministisch). Dies steht im Kontrast zu vielen Graphproblemen, die NP-vollständig sind:

- Hamiltonpfad: NP-vollständig
- Maximale Clique: NP-vollständig
- Graphfärbung: NP-vollständig
- **Graphprofil**: P (diese Arbeit, $O(n^3)$)

Korollar 12 (Praktikabilität)**.** Für Graphen mit $n \leq 10.000$ Knoten ist die Profilberechnung in Sekunden auf modernen Rechnern durchführbar. Für $n \leq 100.000$ in Minuten. Dies deckt die meisten praktischen Anwendungen ab.

9.3 Universalität der Methode

Die Signatur-Methode ist nicht auf die Profilberechnung beschränkt. Sie kann auf weitere Graphprobleme übertragen werden:

Satz 21 (Transitive Hülle)**.** Die transitive Hülle eines Graphen G (Erreichbarkeitsmatrix R) kann mit der Signatur-Methode in Zeit $O(n^3)$ berechnet werden.

Beweis. Die Erreichbarkeitsmatrix ist gegeben durch $R = A \vee A^2 \vee \cdots \vee A^{n-1}$. Berechne alle Potenzen A^k mittels Boolean Matrixmultiplikation (je $O(n^2)$), dann komponentenweise ODER-Verknüpfung in $O(n^2)$. Gesamt: $O(n) \cdot O(n^2) = O(n^3)$. □

Weitere Anwendungen:

- **Zykelerkennung**: Falls $A^k[i, i] = 1$ für ein $k \geq 2$, existiert ein Zyklus durch i
- **Starke Zusammenhangskomponenten**: Über Kombination von R und R^T (transponiert)
- **Kürzeste Pfade mit Gewichten**: Erweiterung auf gewichtete Graphen durch Anpassung der Signatur-Methode

10 Zusammenfassung und Ausblick

Diese Arbeit hat gezeigt, wie die Signatur-Methode aus der Boolean Matrixmultiplikation zur effizienten Berechnung von Graphprofilen eingesetzt werden kann. Die entwickelten Algorithmen ermöglichen die optimale Einordnung jedes Graphen in die Graphprofilverteilung mit deterministischen, reproduzierbaren Ergebnissen.

10.1 Zentrale Ergebnisse

Die wesentlichen Beiträge dieser Arbeit lassen sich wie folgt zusammenfassen:

10.1.1 Theoretische Fundierung

- **Fundamentaler Zusammenhang**: Die Äquivalenz zwischen Matrixpotenzen A^k und Wegen der Länge k wurde als außergewöhnliche strukturelle Eigenschaft identifiziert. Die Endlichkeit der Knotenmenge impliziert, dass $n - 1$ Matrixpotenzen zur vollständigen Charakterisierung genügen.

- **Induktive Effizienz**: Die Erkenntnis, dass $A^{k+1} = A^k \cdot A$ nur eine einzige Multiplikation erfordert, transformiert die Komplexität von potenziell $O(k)$ Multiplikationen auf konstant 1 Multiplikation pro Schritt.

- **Potenzmatrix-Cache**: Das Konzept des Caches $\mathcal{C}(G) = \{A^1, \ldots, A^{n-1}\}$ als vollständige Repräsentation aller Wegeigenschaften ermöglicht nach initialer Investition von $O(n^3)$ beliebig viele Abfragen in $O(1)$.

- **Optimalität der Einordnung**: Die Profilberechnung ist nicht verbesserbar – sie liefert exakte, deterministische Ergebnisse, die maximal informiert sind.

10.1.2 Algorithmische Beiträge

- **Signatur-basierte Boolean Matrixmultiplikation**: Hat $O(n^2)$ Laufzeit durch Kodierung von Zeilen und Spalten als Bitmuster und bitweise UND-Operationen.

- **Vollständige Profilberechnung**: Algorithmus zur Berechnung von kürzesten Wegen, längsten Wegen und Kantenmaß in einheitlicher Zeit $O(n^3)$ unter Verwendung der Signatur-Methode.

- **Hierarchische Analyse**: Erweiterung auf mehrstufige Graphhierarchien mit Gesamtlaufzeit $\sum_{i=1}^{h} O(n_i^3)$.

10.1.3 Empirische Validierung

Die experimentellen Ergebnisse bestätigen die theoretischen Vorhersagen:

- **Skalierung**: Gemessene Laufzeit $O(n^{3.16})$ liegt nahe an theoretischem $O(n^3)$ (Abweichung durch Cache-Effekte und Implementierungsdetails)

- **Biologische Netzwerke**: Charakteristisches Profil mit $\kappa \approx 0.15$–0.20 und Durchmesser ≈ 5–7 identifiziert

- **Trade-off-Relation**: Empirischer inverser Zusammenhang zwischen Kantenmaß und Durchmesser: $\kappa \cdot \text{diam}(G) \approx 3.5$

- **Extrapolation**: Für Gehirn-skalierte Netzwerke ($86 \cdot 10^9$ Knoten) wurde geschätzter Durchmesser von 81 Hops berechnet – fundamentale Erkenntnis für schnelle neuronale Informationsverarbeitung

10.2 Anwendungsgebiete

Die entwickelten Methoden eröffnen vielfältige Anwendungen:

- **Neurowissenschaften**: Konnektomanalyse, Pathologie-Detektion, entwicklungsneurologische Studien

- **Künstliche Intelligenz**: Neural Architecture Search, Modellvergleich, KI-Sicherheit durch deterministische Verifikation

- **Rechenzentren**: Topologie-Optimierung, profilbasiertes Load Balancing

- **Soziale Netzwerke**: Influencer-Identifikation, Analyse von Desinformations-Ausbreitung

- **Molekularbiologie**: Protein-Interaktionsnetzwerke, Drug Target Identification

10.3 Offene Forschungsfragen

Trotz der umfassenden Ergebnisse bleiben wichtige Fragen offen, die zukünftige Forschung motivieren:

Frage 1. Kann durch massive Parallelisierung auf GPU- oder spezialisierter Hardware (FPGA, ASICs) eine Laufzeit von $O(n^2/p)$ mit p Prozessoren erreicht werden?

Die Signatur-Berechnung ist inhärent parallelisierbar:

- Zeilen- und Spalten-Signaturen können parallel berechnet werden

- Bitweise UND-Operationen lassen sich auf GPU-Architekturen effizient abbilden

- Potenzielle Beschleunigung um Faktor 10^3 bis 10^6 für Gehirn-skalierte Probleme

Für extrem große Graphen ($n > 10^6$) ist exakte Berechnung praktisch unmöglich.

Frage 2. Existiert ein randomisierter oder deterministischer Approximationsalgorithmus mit Laufzeit $O(n^2)$ und garantierter Güte $\|\tilde{D} - D\|_\infty \leq \epsilon \cdot \text{diam}(G)$?

Möglicher Ansatz: Sampling-basierte Methoden (z.B. nur Wege von/zu zufälligen Landmark-Knoten berechnen) kombiniert mit probabilistischen Schranken.
Die Signatur-Methode basiert auf Bitoperationen. Potenzielle Quantenvorteile:

- Superposition zur parallelen Auswertung aller Matrixeinträge

- Quantum Walk für Wegeberechnung mit möglicher quadratischer Beschleunigung

- Theoretisches Potenzial: $O(n^{2.5})$ oder besser

Eine mögliche Anwendung ist die Konstruktion einer öffentlich zugänglichen Datenbank mit Profilen von Millionen bekannter Graphen aus verschiedenen Domänen:
Datenquellen:

- Biologische Netzwerke (Protein-Interaktionen, metabolische Netzwerke, neuronale Konnektome)

- Soziale Netzwerke (Co-Authorship, Zitationsnetzwerke)

- Infrastrukturgraphen (Straßennetze, Stromnetz, Internet-Topologie)

- Künstliche neuronale Netze (Architekturen erfolgreicher Modelle)

Funktionalität:

- Similarity Search: Finde strukturell ähnliche Graphen über Domänen hinweg

- Pattern Discovery: Identifiziere wiederkehrende Profil-Signaturen

- Transfer Learning: Nutze Erkenntnisse aus einer Domäne für andere

Technische Herausforderungen: Skalierbare Indizierung für mehrdimensionale Profilräume, effiziente Distanzmetriken für (D, L, κ)-Tripel.

10.4 Ausblick: Universalität der Methode

Die Signatur-Methode ist nicht auf Graphprofile beschränkt. Potenzielle Erweiterungen:

- **Gewichtete Graphen**: Erweiterung auf reellwertige Gewichte durch Anpassung der Signatur-Kodierung

- **Gerichtete Graphen**: Separate Behandlung von In- und Out-Degree

- **Hypergraphen**: Verallgemeinerung auf höher-dimensionale Relationen

- **Temporale Graphen**: Integration der Zeitdimension in die Profilberechnung

Das Ergebnis, dass endliche diskrete Strukturen durch Potenzoperationen vollständig charakterisiert werden können, kann auf weitere mathematische Objekte übertragen werden.

11 Fazit

Diese Arbeit hat einen fundamentalen Beitrag zur Analyse von Graphstrukturen geleistet, indem sie die Signatur-Methode der Boolean Matrixmultiplikation zur optimalen Berechnung von Graphprofilen nutzbar gemacht hat. Die zentralen Erkenntnisse und ihre Bedeutung werden im Folgenden zusammengefasst.

11.1 Der außergewöhnliche Zusammenhang als Grundlage

Die Arbeit hat den tiefen Zusammenhang zwischen Graphentheorie und Boolean Algebra offengelegt: Die k-te Potenz A^k der Adjazenzmatrix kodiert vollständig die Existenz aller Wege der Länge k mit exakt $k-1$ Zwischenknoten. Diese strukturelle Äquivalenz ist außergewöhnlich, da sie:

- Eine vollständige algebraische Repräsentation der Graphstruktur liefert

- Durch die Endlichkeit von Knoten und Kanten auf $n-1$ Matrixpotenzen beschränkt ist

- Eine induktive Berechnung ermöglicht: $A^{k+1} = A^k \cdot A$

- Das Konzept des Potenzmatrix-Caches $\mathcal{C}(G) = \{A^1, \ldots, A^{n-1}\}$ als vollständige Wegecharakterisierung begründet

Diese Erkenntnis transformiert die Profilberechnung von einem wiederholt kostspieligen Problem in eine einmalige Investition mit anschließend konstanter Abfragezeit.

11.2 Theoretische Bedeutung

11.2.1 Determinismus als Überlegenheitsprinzip

Die Arbeit demonstriert die grundsätzliche Überlegenheit deterministischer Algorithmen über stochastische Verfahren, wenn folgende Bedingungen erfüllt sind:

1. Polynomielle Laufzeit ($O(n^3)$ für Graphprofile)
2. Exakte, nicht approximative Ergebnisse
3. Vollständige Reproduzierbarkeit
4. Keine Fehlerwahrscheinlichkeit

Für sicherheitskritische Anwendungen – medizinische Diagnostik, Infrastruktur-Design, KI-Verifikation – ist dies von fundamentaler Bedeutung. Die Optimalität der Einordnung garantiert, dass keine andere Methode bessere strukturelle Erkenntnisse liefern kann.

11.2.2 Komplexitätstheoretische Einordnung

Die Profilberechnung ist ein Problem in **P** (polynomielle Zeit, deterministisch). Dies steht im Kontrast zu vielen anderen Graphproblemen:

- Hamiltonpfad: NP-vollständig
- Maximale Clique: NP-vollständig
- Graphfärbung: NP-vollständig
- **Graphprofil**: P, $O(n^3)$ (diese Arbeit)

Die experimentelle Validierung bestätigt die theoretische Analyse: gemessene Laufzeit $O(n^{3.16})$ liegt nahe an theoretischem $O(n^3)$.

11.3 Praktische Relevanz

11.3.1 Domänenübergreifende Anwendbarkeit

Die entwickelten Methoden sind universell einsetzbar:

- **Neurowissenschaften**: Charakterisierung neuronaler Konnektome mit Profil $\kappa \approx 0.15$–0.20 und Durchmesser ≈ 5–7 als biologischem Optimum
- **Künstliche Intelligenz**: Deterministische Architektur-Suche, Modellvergleich, Verifikation durch Profilüberwachung
- **Infrastruktur**: Topologie-Optimierung von Rechenzentren, Netzwerken, Versorgungssystemen
- **Soziale Systeme**: Influencer-Identifikation, Analyse von Informationsausbreitung
- **Molekularbiologie**: Drug Target Identification durch strukturelle Netzwerkanalyse

11.3.2 Skalierbarkeit und Grenzen

Die Experimente zeigen:

- Praktikabel für $n \leq 10^4$ Knoten (Sekundenbereich)
- Machbar für $n \leq 10^5$ Knoten (Minutenbereich)
- Extrapolation auf Gehirn-Skala ($86 \cdot 10^9$ Knoten) ergibt $\approx 10^{21}$ Jahre – praktisch unmöglich

Dies motiviert zukünftige Forschung zu Parallelisierung, Approximation und hierarchischer Dekomposition.

11.4 Methodische Innovation

Die Signatur-Methode verbindet drei Ebenen:

1. **Theoretische Ebene**: Ausnutzung der strukturellen Äquivalenz zwischen A^k und Wegen der Länge k
2. **Algorithmische Ebene**: Kodierung von Zeilen/Spalten als Bitmuster für $O(n^2)$ Boolean Matrixmultiplikation
3. **Implementierungsebene**: Nutzung von Hardware-Bitoperationen für konstante Multiplikationszeit

Diese mehrstufige Optimierung ist übertragbar auf weitere Probleme der diskreten Mathematik.

11.5 Schluss

Die in dieser Arbeit entwickelte Methode zur optimalen Berechnung von Graphprofilen mittels der Signatur-Technik hat weitreichende theoretische und praktische Implikationen:

Theoretisch wird gezeigt, dass deterministische Algorithmen stochastischen Verfahren überlegen sein können, wenn sie in polynomieller Zeit exakte Ergebnisse liefern.

Praktisch eröffnen sich Anwendungen von der Neurowissenschaft über Künstliche Intelligenz bis hin zur Infrastruktur-Optimierung. Die Garantie optimaler Einordnung in die Graphprofilverteilung macht Aussagen über Graphstrukturen maximal informiert und nicht verbesserbar.

Die zentrale Aussage ist: *Jeder Graph wird optimal mit Einordnung in die Graphprofilverteilung charakterisiert. Darauf basierende Entscheidungen sind deterministisch und reproduzierbar.*

In einer Zeit zunehmender Komplexität und Vernetzung bietet diese Methode ein fundamentales Werkzeug zur Analyse und zum Verständnis strukturierter Systeme.