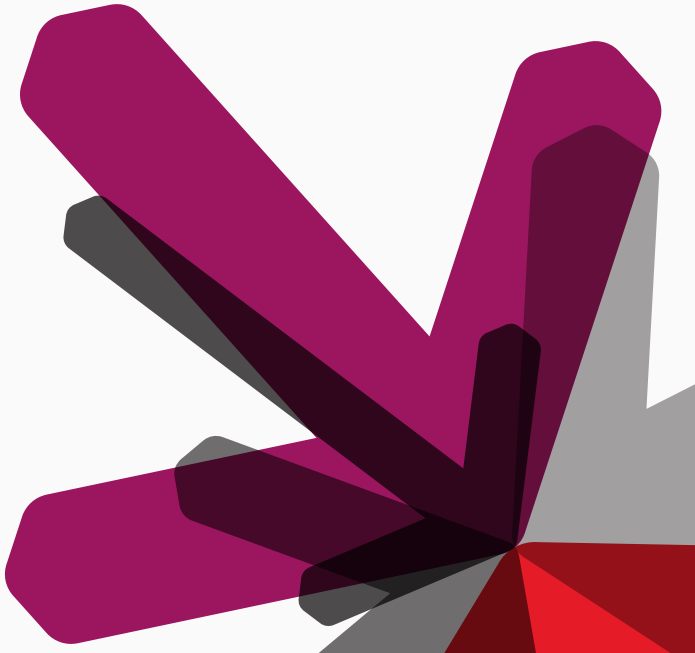
TEILE UND HERRSCHE

Parallele Suchalgorithmen für integrale Bäume

Viola Campos

19. Februar 2019

Master Projekt im WS 2018/19
Hochschule RheinMain



Übersicht

Einleitung

Algorithmen

Parallelisierung

Ergebnisse

Einleitung

Projektidee

Suche nach 'Integralen Bäumen'

Ausgangssituation

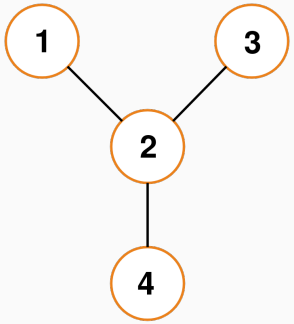
- Paper 'Small integral trees' von A.E. Brouwer mit Tabelle aller integralen Bäume mit maximal 50 Knoten
- Einfache Überprüfung auf Korrektheit

Ziel

- Minimalziel: Brouwers Ergebnisse verifizieren
- Suche nach Optimierungsmöglichkeiten
- Untersuchung der Parallelisierbarkeit des Problems

Adjazenzmatrix

Für einen Graphen $G = (V, E)$ ist die Adjazenzmatrix $A = [a_{ij}]$ definiert als



$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

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

Eigenwerte und Eigenvektoren

λ ist Eigenwert der Matrix A und v ist Eigenvektor, wenn gilt:

$$A \cdot v = \lambda v$$

$n \times n$-Matrix besitzt n Eigenwerte und Eigenvektoren

Spektrum eines Graphen

Nach Größe geordnete Folge der Eigenwerte der Adjazenzmatrix

Integraler Graph

Graph, dessen Spektrum nur ganzzahlige Werte enthält

Integraler Baum

Zusammenhängender, ungerichteter, azyklischer integraler Graph

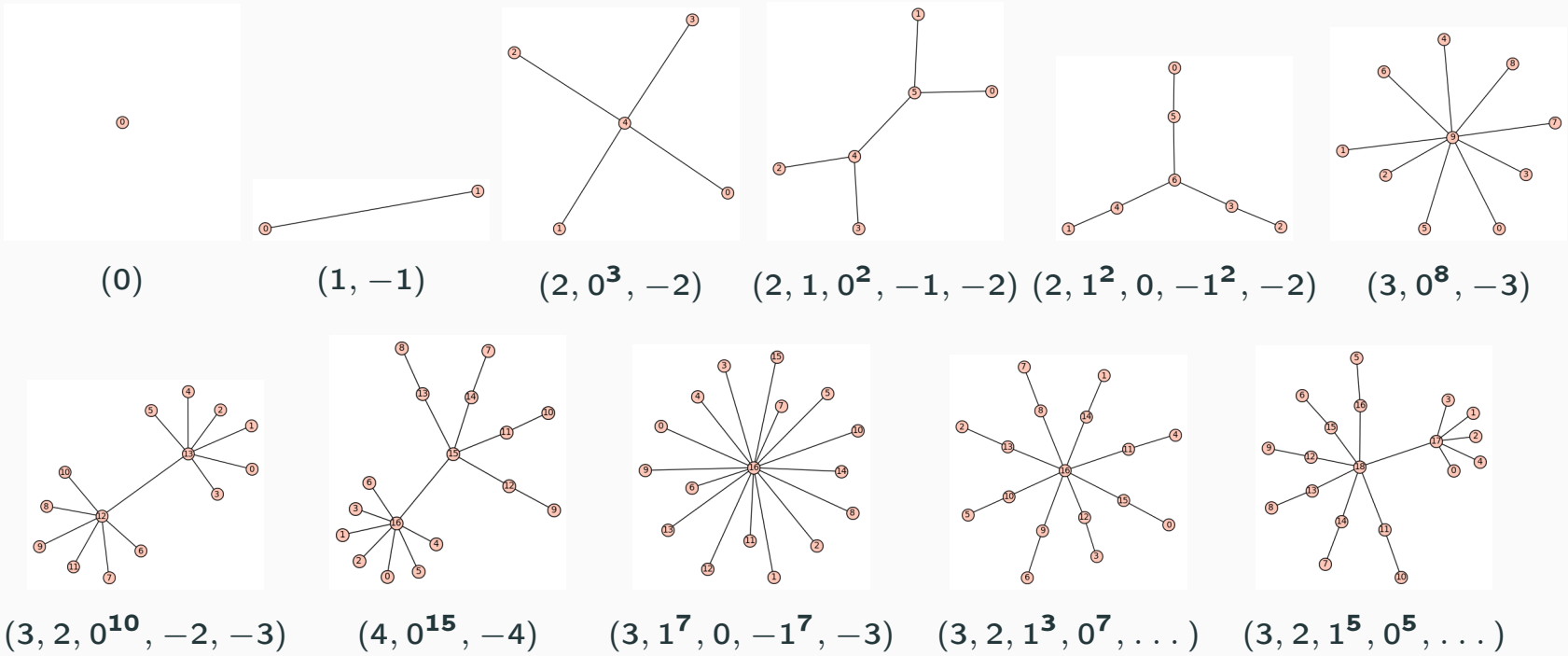
Eigenschaften integraler Bäume

- Spektren von Bäumen sind symmetrisch bzgl. 0
- Ungerichtete Graphen haben symmetrische Adjazenzmatrix und deshalb reelle Eigenwerte

- Es gibt Familien von integralen Bäumen, die nach bestimmten Regeln konstruiert werden können
- Die meisten der gefundenen Bäume lassen sich solchen Familien zuordnen

Beispiele für integrale Bäume

Integrale Bäume sind selten. Von 1.346.025 nicht-isomorphen Bäumen mit maximal 20 Knoten sind 11 integral:



(0) $(1, -1)$ $(2, 0^{3}, -2)$ $(2, 1, 0^{2}, -1, -2)$ $(2, 1^{2}, 0, -1^{2}, -2)$ $(3, 0^{8}, -3)$

$(3, 2, 0^{10}, -2, -3)$ $(4, 0^{15}, -4)$ $(3, 1^{7}, 0, -1^{7}, -3)$ $(3, 2, 1^{3}, 0^{7}, \dots)$ $(3, 2, 1^{5}, 0^{5}, \dots)$

Abbildung 1: Integrale Bäume bis 20 Knoten und ihre Spektren[1]

[1] Multiplizitäten sind als Exponenten dargestellt

Algorithmen

Ausgangslage

Naiver Brute-Force-Algorithmus zur Einschätzung der Komplexität:

Algorithm 1: Brute-Force Implementierung

Input : n: number of nodes

Output: *resultSet*: set of integral trees with n nodes

```
candidateSet ← createAllNonisomorphicTrees(n)
for each tree ∈ candidateSet do
    s ← computeSpectrum(tree)
    if isIntegral(s) then
        resultSet ← resultSet ∪ {tree}
```

# Knoten	10	20	30	40	50
Rechenzeit	0.64 s	6,9 Std	30 Jahre	10^6 Jahre	10^{10} Jahre

Tabelle 1: Hochrechnung des Zeitaufwands

Einschränkung des Suchraums (1/3)

Es gibt mehr als 10^{19} Bäume mit 50 Knoten. Alle zu überprüfen wird nicht funktionieren.

Ansatz: Interlacing

Seien θ und η Folgen reeller Zahlen $\theta_1 \geq \cdots \geq \theta_n$ und $\eta_1 \geq \cdots \geq \eta_m$ mit $m < n$. Die zweite Folge '*interlaced*' die erste, wenn gilt:

$$\theta_i \geq \eta_i \geq \theta_{n-m+i}, \text{ für } i = 1, \ldots, m.$$

Gilt $m = n - 1$, folgt:

$$\theta_1 \geq \eta_1 \geq \theta_2 \geq \eta_2 \geq \cdots \geq \eta_m \geq \theta_n$$

Interlacing von Graphenspektren[2]

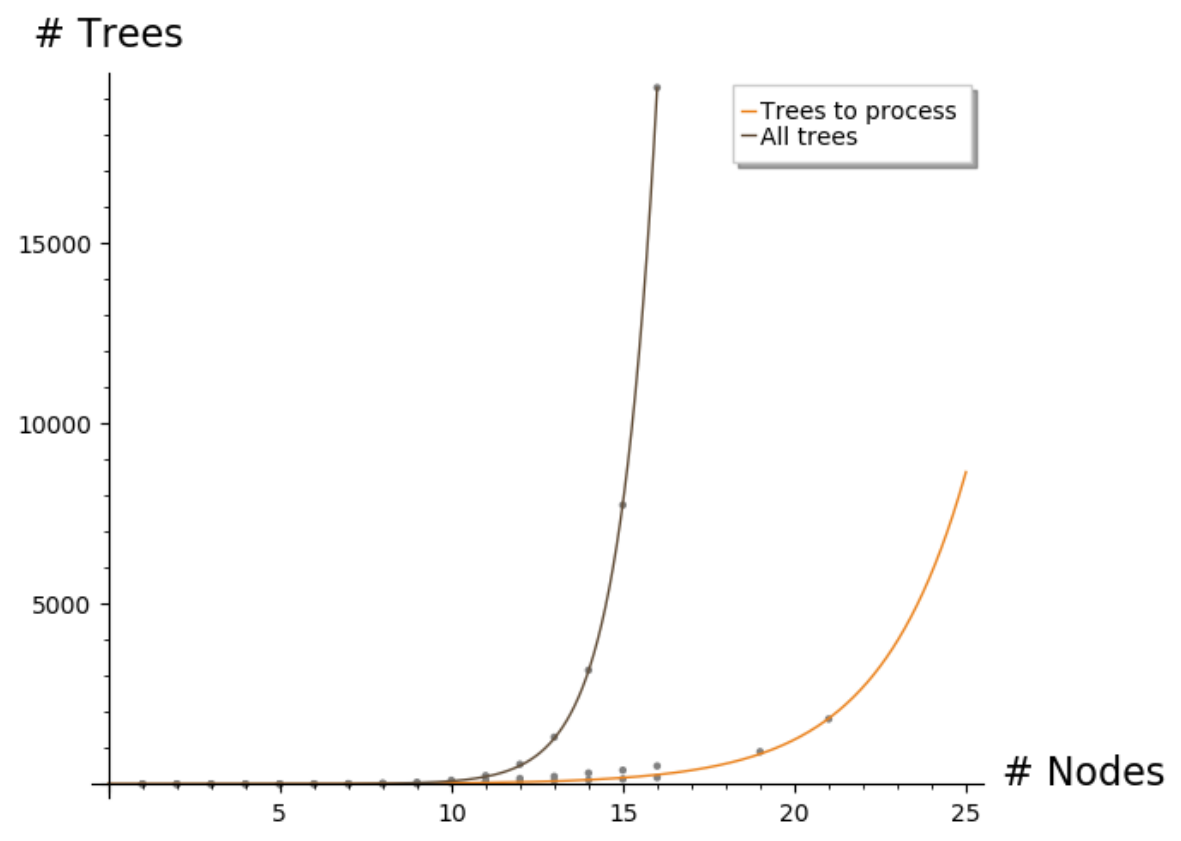
Sei x ein Knoten eines Graphen $G = (V, E)$ und $G' = G \setminus x$ der Teilgraph, der aus G durch Entfernen von x entsteht. Die Eigenwerte von G' 'interlacen' die von G.

Hat G' also 2 Eigenwerte zwischen aufeinander folgenden Ganzzahlen a und $a + 1$, dann hat auch G einen Eigenwert zwischen a und $a + 1$ und kann folglich nicht integral sein.

[2] aus A.E.Brouwer, 'Small Integral Trees'

Einschränkung des Suchraums (3/3)

- Lässt sich als Abbruchbedingung für Konstruktion neuer Bäume nutzen
- Hat ein Baum 2 Eigenwerte zwischen a und $a+1$, kann durch Hinzufügen eines Knotens kein integraler Baum entstehen



Isomorphie

Iterative Erzeugung neuer Bäume erfordert Test auf Isomorphie

- Variante 1:
  - Halte Liste der bearbeiteten Bäume
  - Bestimme für jeden neu erzeugten Baum, ob Isomorphie eines bereits bearbeiteten Baumes
- Variante 2:
  - Definiere kanonische Darstellung der Bäume
  - Liste bearbeiteter Bäume in kanonischer Darstellung
  - Komplexität von Kanonisierung entspricht Isomorphie zwischen Graphen, aber weniger Vergleiche nötig
- Problem: Speicherbedarf für Liste der bearbeiteten Bäume

Laufzeit

# Knoten	10	20	30	40	50
Brute Force	0.64 s	6,9 Std.	30 Jahre	10^6 Jahre	10^{10} Jahre
Mit Interlacing	0.7 s	1.8 min	4.4 Std	27 Tage	11 Jahre

Tabelle 2: Vergleich Zeitaufwand

Berechnung der Spektren (1/2)

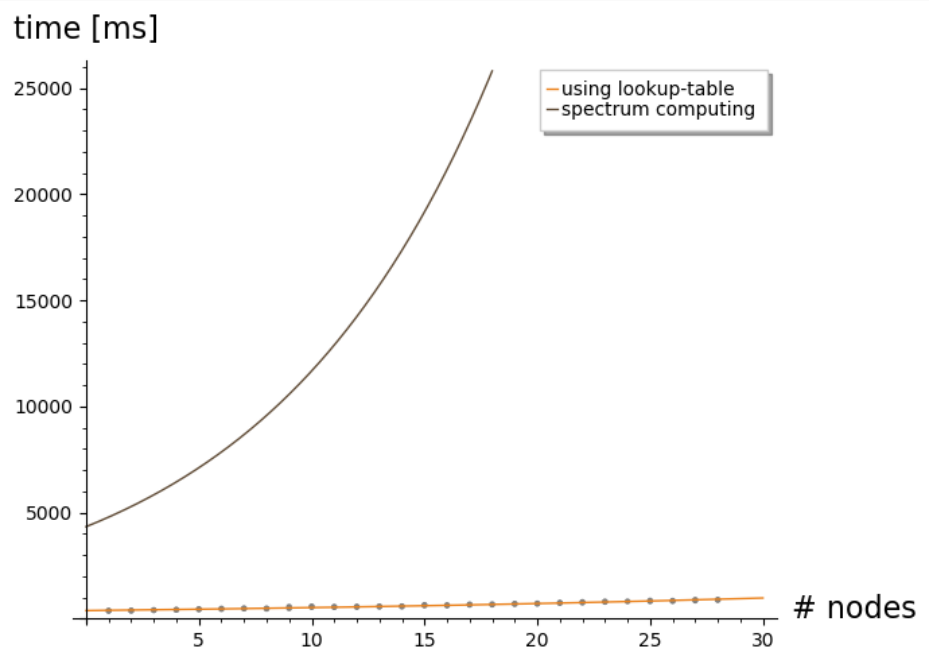
93% der Rechenzeit entfallen auf Berechnung der Graphenspektren

Ansatz: Divide-and-Conquer

- Spektrum eines nichtverbundenen Graphen ist die Vereinigung der Spektren seiner Komponenten
- Hat bereits eine Komponente von $G \setminus x$ (oder die Vereinigung einiger Komponenten) 2 Eigenwerte zwischen aufeinander folgenden Ganzzahlen, dann auch G
- Zur Überprüfung auf die Abbruchbedingung: Entferne zentralen Knoten & überprüfe Komponenten

Ansatz: Dynamische Programmierung

- Bäume werden rekursiv in Komponenten geteilt & deren Spektren bestimmt
- Lookup-Table bereits berechneter Spektren verhindert Mehrfach-berechnungen



Laufzeiten

# Knoten	10	20	30	40	50
Brute Force	0.64 s	6,9 Std	30 Jahre	10^6 Jahre	10^{10} Jahre
Interlacing	0.7 s	1.8 min	4.4 Std	27 Tage	11 Jahre
Dyn. Spektren[3]	1.9 s	51 s	23 min	10.5 Std	12 Tage

Tabelle 3: Vergleich der Laufzeiten

[3] rekursive Spektrenberechnung & Lookup-Table

Parallelisierung

Parallelisierung (1/2)

Algorithmus ist nur begrenzt parallelisierbar

- hoher Kommunikationsaufwand (Liste bereits erzeugter Bäume, Lookup-Table für Spektren)
- 2 Möglichkeiten: Gemeinsam genutzte Objekte mit Lockingmechanismen (langsamer) oder Redundanz (speicherintensiver)
- Prozesse erzeugen eigene globale Datenstrukturen

Ansatz: Aufbau einer Pipeline

- Schrittweise Verarbeitung, Prozesse verarbeiten die Ergebnisse der Vorgänger
- ermöglicht Trennung von rechenintensiven und weniger rechenintensiven Schritten
- Anzahl paralleler Prozesse kann für jeden Schritt dynamisch zugewiesen werden, vermeidet Engpässe

Übersicht

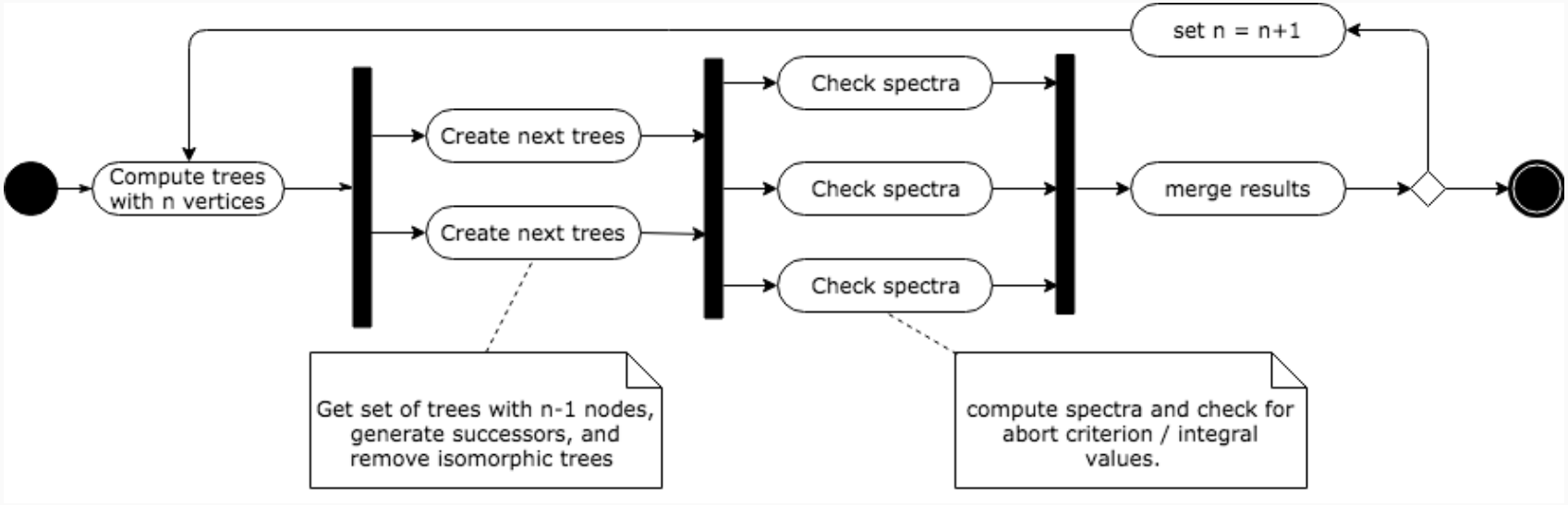


Abbildung 3: Parallele Verarbeitung

Laufzeiten

# Knoten	10	20	30	40	50	60
Brute Force	0.6 s	7 Std	30 J.	10^6 J.	10^{10} J.	
Interlacing	0.7 s	2 min	4.4 Std	27 Tage	11 J.	
Dyn. Spektren	1.9 s	51 s	23 min	10.5 Std	12 Tage	
Parallelisiert[4]	3.5 s	43 s	9 min	106 min	21.6 Std	9 Tage

Tabelle 4: Vergleich der Laufzeiten

[4] 6 Cores mit 2.2GHz und 20GB RAM

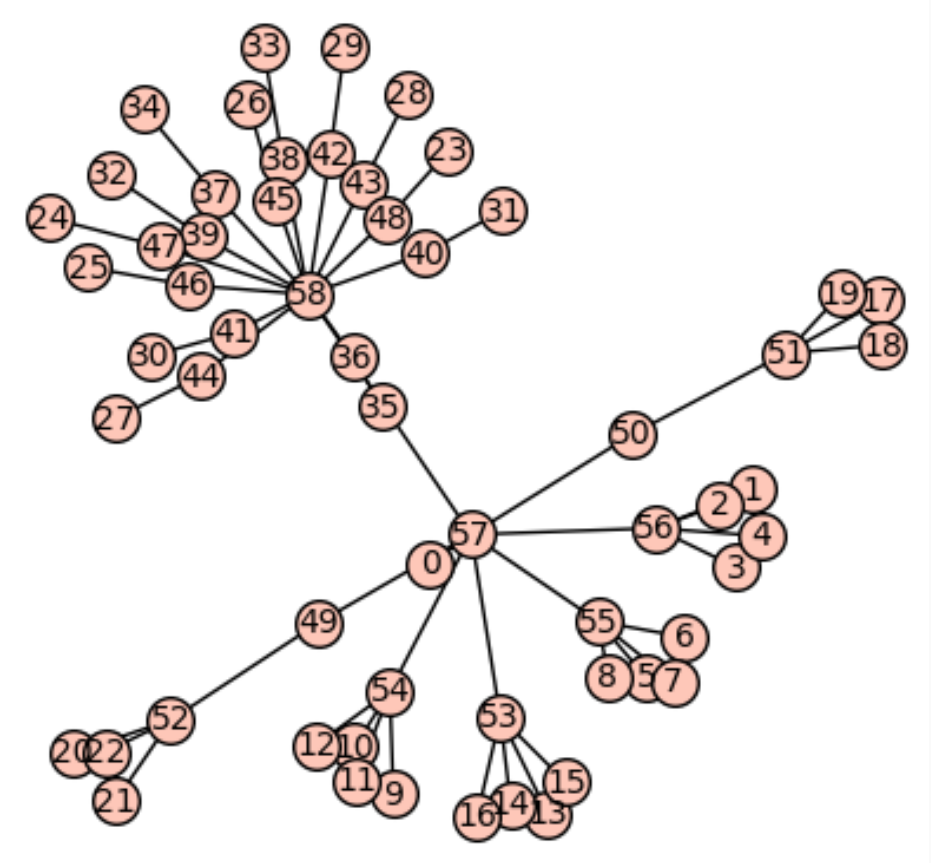
Ergebnisse

Ergebnisse

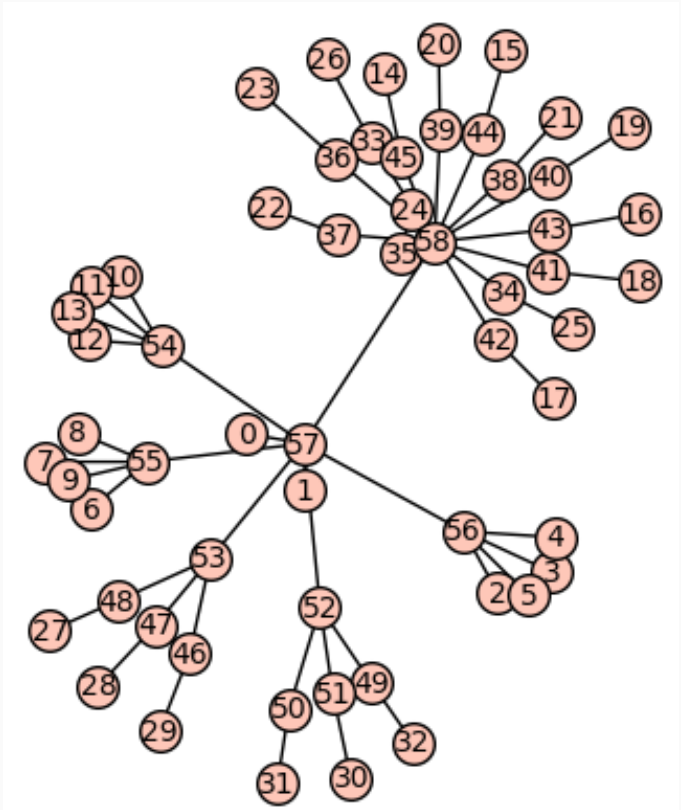
- Optimierung von Laufzeit und Speicherbedarf
- Tabelle aller integralen Bäume bis zu einer Größe von 60 Knoten berechnet
- bis 50 Knoten mit denselben Ergebnissen wie Brouwer[5]
- zusätzlich 2 neue Bäume mit 59 Knoten gefunden

[5] Sammlung aller bekannten integralen Bäume

Neue integrale Bäume



$(4, 3, 2^{\mathbf{5}}, 1^{\mathbf{13}}, 0^{\mathbf{19}}, -1^{\mathbf{13}}, -2^{\mathbf{5}}, -3, -4)$



$(4, 3, 2^{\mathbf{4}}, 1^{\mathbf{17}}, 0^{\mathbf{1}}3, -1^{\mathbf{17}}, -2^{\mathbf{4}}, -3, -4)$

Abbildung 4: Neue integrale Bäume mit 59 Knoten und ihre Spektren

Ausblick & offene Fragen

- Verteiltes Rechnen zur weiteren Suche (BOINC)
  - Problem: Paketierung
  - auf Basis des vorhandenen parallelisierten Algorithmus möglich
- Häufigkeitsverteilung der Cache-Hitraten

Offene Fragen

- Werden mit der genutzten Abbruchbedingung tatsächlich alle Bäume bis 50 Knoten gefunden?
- Die meisten der gefundenen Bäume lassen sich bekannten Familien integraler Bäume zuordnen. Gibt es Muster für die übrigen Fälle?

Vielen Dank für die Aufmerksamkeit!