Baumsuche: Theorie und Anwendungen

Rainer Sinn

27.06.2008

Teil I

Bäume und Wälder

Wiederholung: Graph

- Menge der Knoten N
- Menge der Kanten E
- Graph $G := (N, E)$

Gerichteter Graph: E Menge von geordneten Tupeln $\subset N \times N$

Wald

- Weg: Folge $(v_i)_{1 \leq i \leq n}$ mit $(v_i, v_j) \in E$
- Zyklus: Weg $(v_i)_{1 \leq i \leq n}$ mit $v_1 = v_n$

Wald: zyklusfreier Graph

Baum

- Zwei Knoten v, w verbunden: Weg von v nach w
- Zusammenhangskomponente eines Knotens v: alle mit v verbundenen Knoten

Baum: Zusammenhangskomponente eines Waldes
(Wald disjunkte Vereinigung von Bäumen)

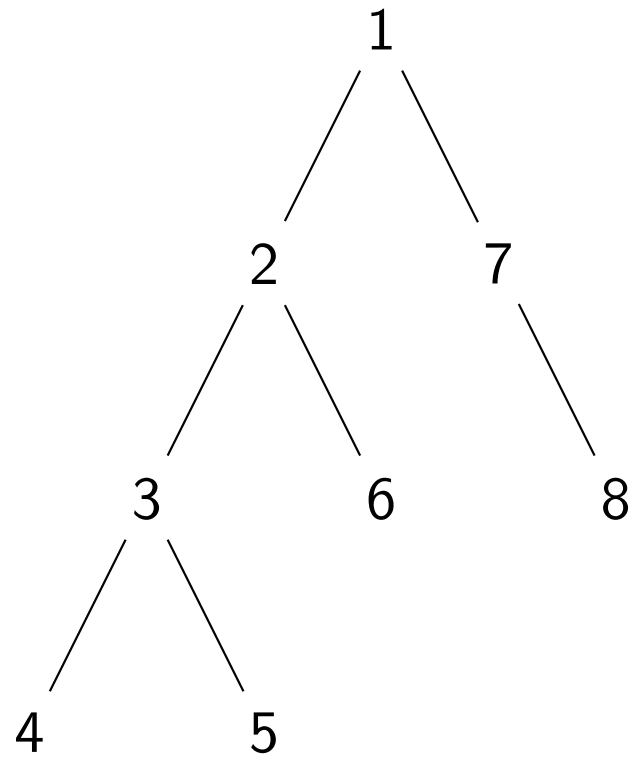
Kindknoten und Blätter

$G = (N, E)$ gerichteter Baum

- Wurzel: Knoten, von dem aus alle anderen erreichbar sind
- w Kindknoten von v: $(v, w) \in E$, v näher an der Wurzel
- Blatt: Knoten ohne Kindknoten

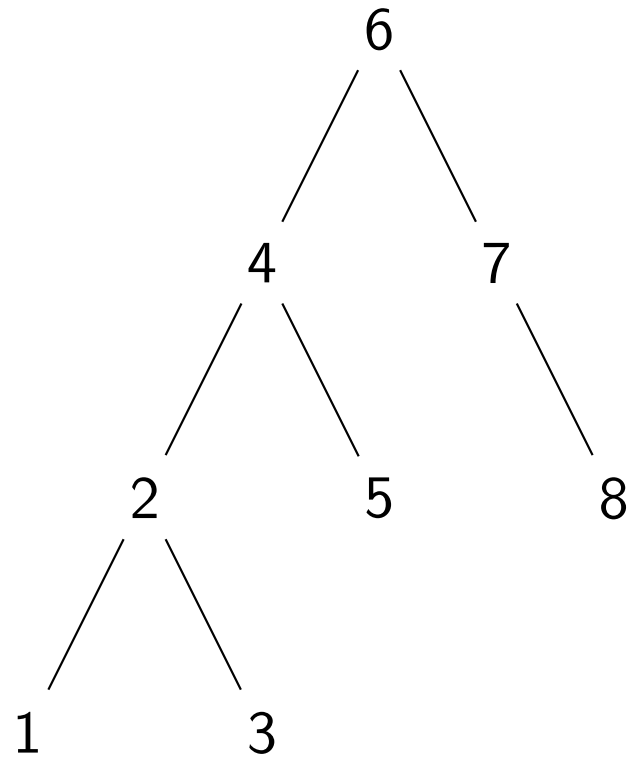
Pre-order Traversierung von Binärbäumen

Wurzel - Linker Teilbaum - Rechter Teilbaum



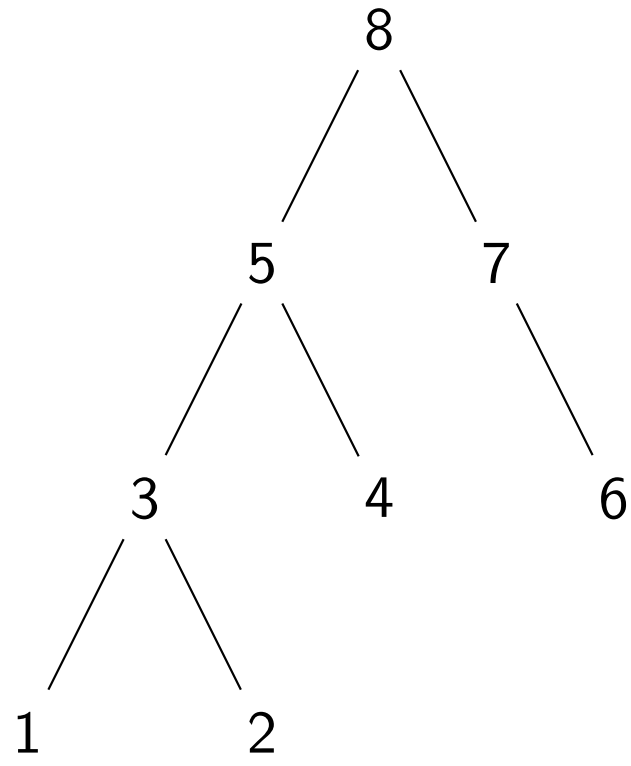
In-order Traversierung von Binärbäumen

Linker Teilbaum - Wurzel - Rechter Teilbaum

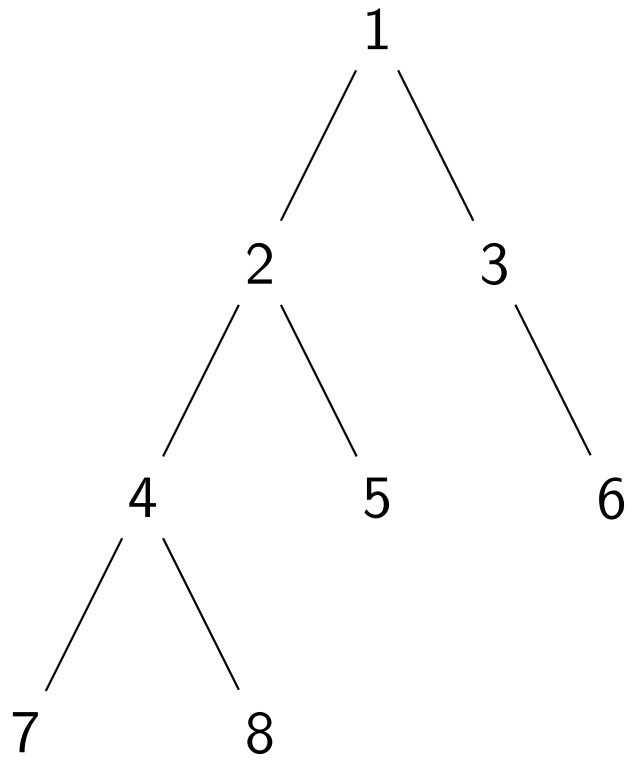


Post-order Traversierung von Binärbäumen

Linker Teilbaum - Rechter Teilbaum - Wurzel



Level-order Traversierung von Binärbäumen



Teil II

Baum-Such-Probleme

Tourenplanung

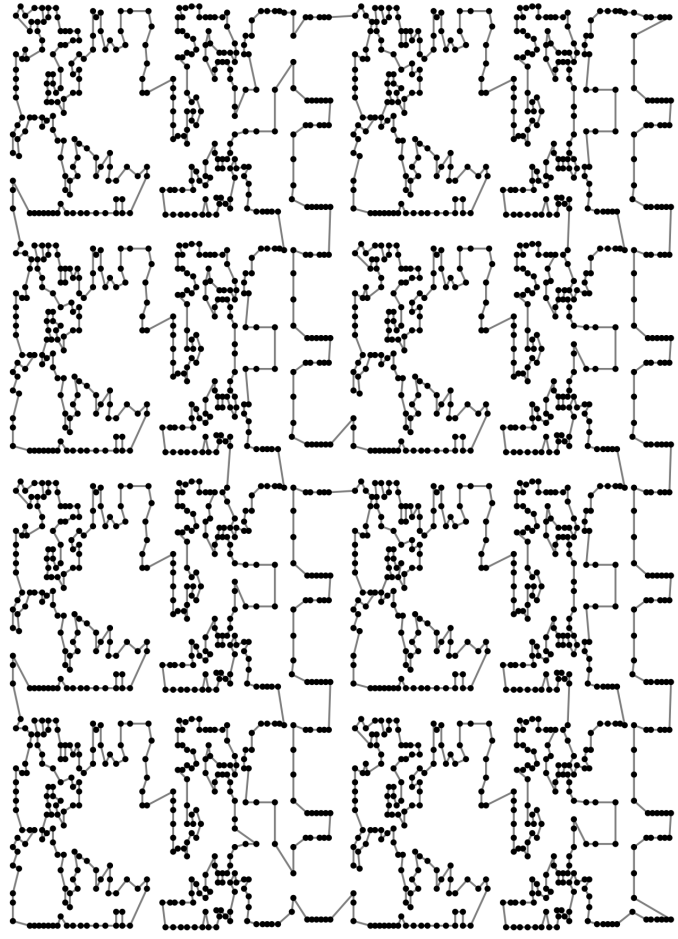
Problem: Kürzester Zyklus durch n Städte - travelling salesman problem (TSP)
Gerichteter Baum: Knoten entsprechen Folgen von Städten

- Richtung: nächstes Reiseziel
- zusammenhängend: jede Stadt besuchen
- zyklusfrei: nach Modellierung

  (NP-vollständig!)

Anwendungen des TSP

- Fahrtroutenoptimierung
- Leiterplattenfertigung
- Verdrahtungsprobleme (Chipdesign)
- Produktionsplanung



von: http://www.iwr.uni-heidelberg.de/groups/comopt

Künstliche Intelligenz

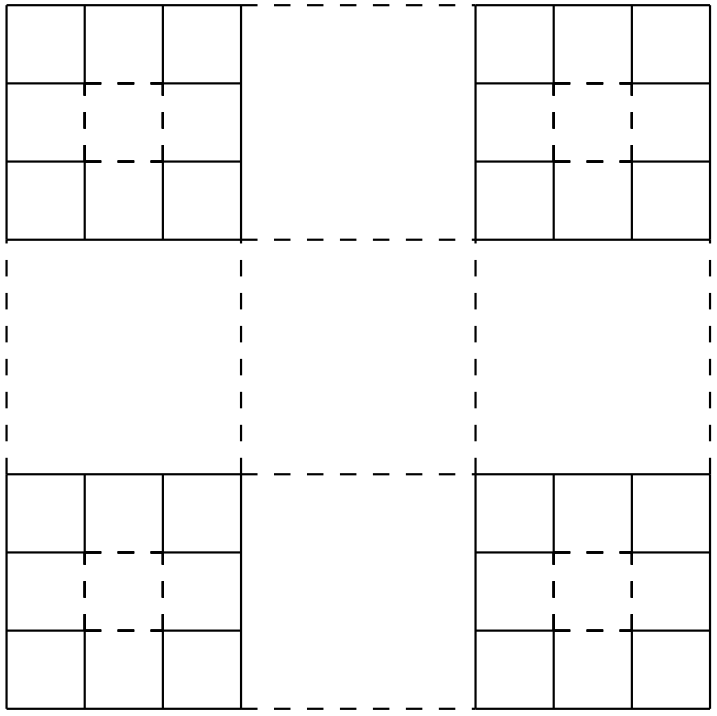
Entscheidungsalgorithmen, Konsequenzenabschätzungen

- Spiele-KI (Schach, GO)
- Medizin (Diagnostik, Notfallmedizin)
- Expertensysteme bei Klassifizierungsaufgaben

Arbeitspferd Sudoku - Regeln

Sudoku der Größe $n \times n$, $n = m^2$:
Unterteile in n viele $m \times m$ Blöcke
Trage Zahlen $1, \ldots, n$ in jede Zeile ein,
unter Beachtung von:

- keine Ziffer in einer Reihe doppelt
- keine Ziffer in einer Spalte doppelt
- keine Ziffer in einem Block doppelt



Arbeitspferd Sudoku - Codierung in Baumsuche

Codiere freie Felder in Zahlenfolge der Länge (Anzahl freier Felder)

- Knoten: partielle Ausfüllungen der Zahlenfolge (nicht ausgefüllt $= 0$)
- Blätter: vollständige Ausfüllungen

Erhalte: Sudoku entspricht gerichtetem Baum mit Wurzel $(0, \ldots, 0)$

Arbeitspferd Sudoku - Backtracking

Lösungsalgorithmus Backtracking-Algorithmus:

1. Problem nicht gelöst, dann weiter
2. prüfe, ob aktuelles Feld gültig
   - falls ja: gehe ein Feld weiter (falls möglich), schreibe 1 hinein
   - falls nein und aktuelles Feld kleiner als n: erhöhe um eins
   - falls nein und aktuelles Feld gleich n: schreibe 0 in aktuelles Feld, gehe ein Feld zurück
3. weiter mit 1

Teil III

Load-Balancing bei paralleler Baumsuche

Naive Arbeitsteilung

- Teile Baum in Prozessanzahl viele Unterbäume
- Verteile Unterbäume auf Prozesse
- Sammle die Ergebnisse

Problem: Keine a-priori Abschätzung über Bearbeitungszeit der Unterbäume

Erste Weiterentwicklung: Master-Slave-Modell

Masterprozess verteilt Arbeit an Slave-Prozesse
Probleme:

- keine a-priori Abschätzung der Bearbeitungszeit
- deshalb: Vorzerlegung des Baumes (möglichst viele Teile)
- ungünstig bei großer Prozesszahl

Receiver induced tree splitting I

- Prozesse arbeiten an einem Unterbaum
- falls Unterbaum abgearbeitet, Anfrage bei Nachbar-Prozess (Netzwerktopologie)
- Baumteilung auf Anfrage

Receiver induced tree splitting II

Phasen: feste Zeit T_{Phase}

- falls Unterproblem leer: schicke Anfrage an Nachbarprozess
- sonst: empfange Anfragen
- bearbeite Anfragen
- setze Arbeit fort

Problem: Prozesse „clustern“

Receiver induced tree splitting III

Zufallspermutationen nach fester Anzahl Phasen

- bestimme zufälligen Prozessrang
- verschicke eigenes Unterproblem an diesen
- empfange Unterproblem
- beginne Arbeitsphase

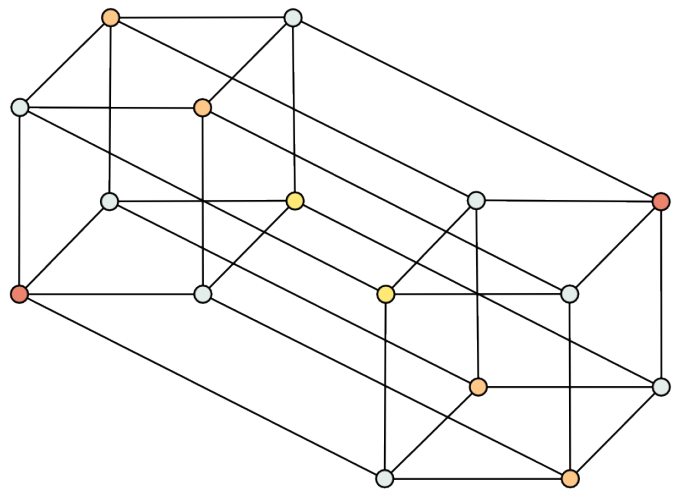
Teil IV

Performance analysis: Sudoku

Netzwerktopologie

Hypercube-Netzwerke

- Einheitshypercubus Dimension n: $[0, 1]^n$
- NT: Jeder Knoten hat n Nachbarn
- Beispiel: 4-dimensionaler Hypercubus



Hypercubus in MPI

Initialisierung Hypercubus-Topologie (kartesische Netzwerktopologie):
MPI_Cart_Create(oldco,dim,PperD,per?,renumpid?,newco)
Operationen

- MPI_Cart_rank(comm,coords,rank): IN(Koordinaten), OUT(Rang)
- MPI_Cart_coords(comm,rank,maxdims,coords): IN(Rang), OUT(Koordinatenarray)
- MPI_Cart_shift(comm,dir,disp,rank_source,rank_dest): OUT(Rang des Prozesses disp weiter in Richtung dir)

Kollektive Kommunikation

- von allen Prozessen aufzurufen
- immer blockierende Kommunikation
- Datentypengleichheit
- keine Tags

Getrennt von Direktkommunikation

Kollektive Operationen

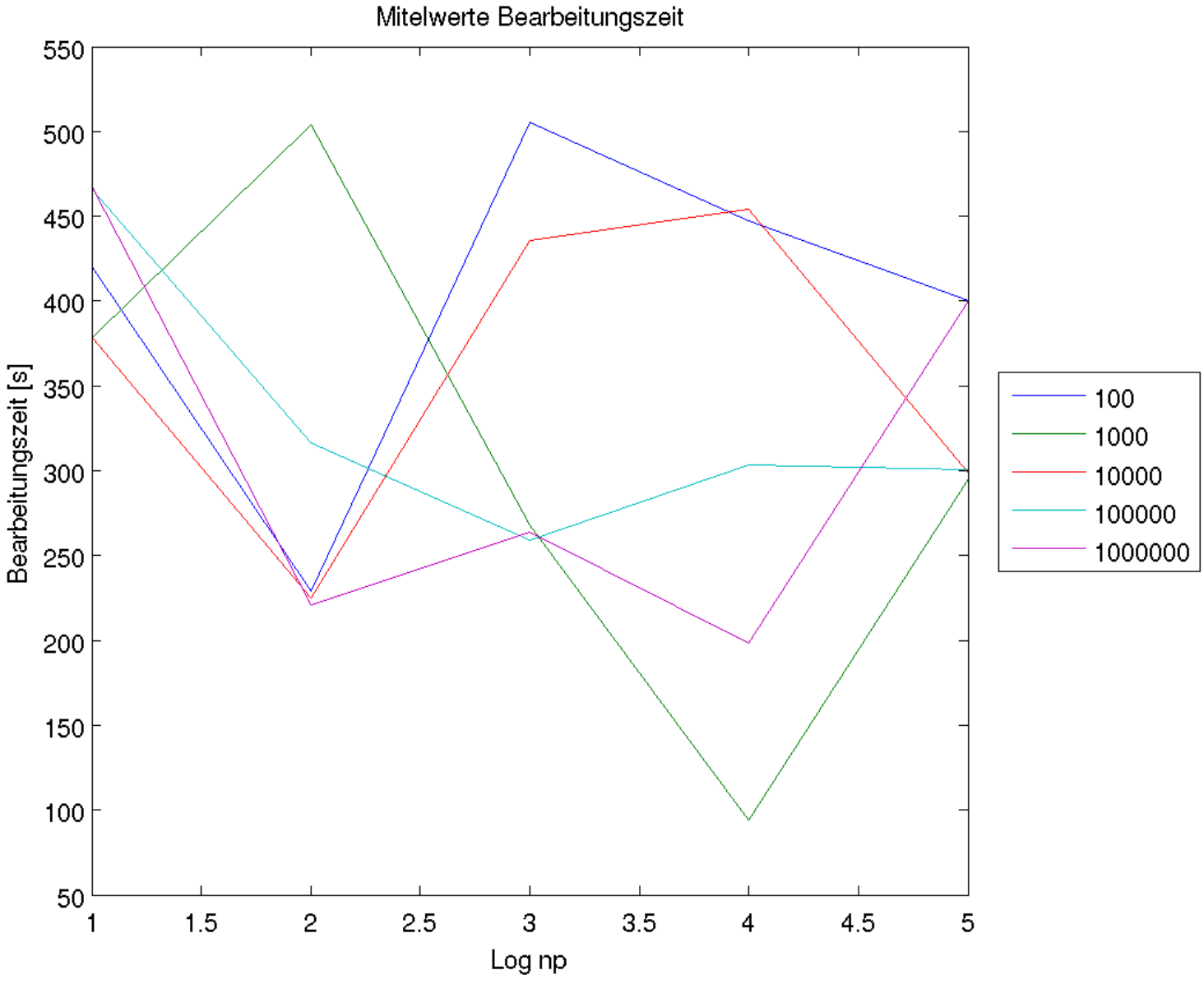
- MPI_Bcast(buffer,count,datatype,root,comm)
- MPI_Scatter/_Gather(sendbuf,sendcount,sendtype, recvbuf,recvcount,recvtype,root,comm)
- MPI_Reduce(Sbuf,Rbuf,count,datatype,op,root,comm):
  1. MPI_MAX, MPI_MIN
  2. MPI_SUM, MPI_PROD
  3. logische Operationen

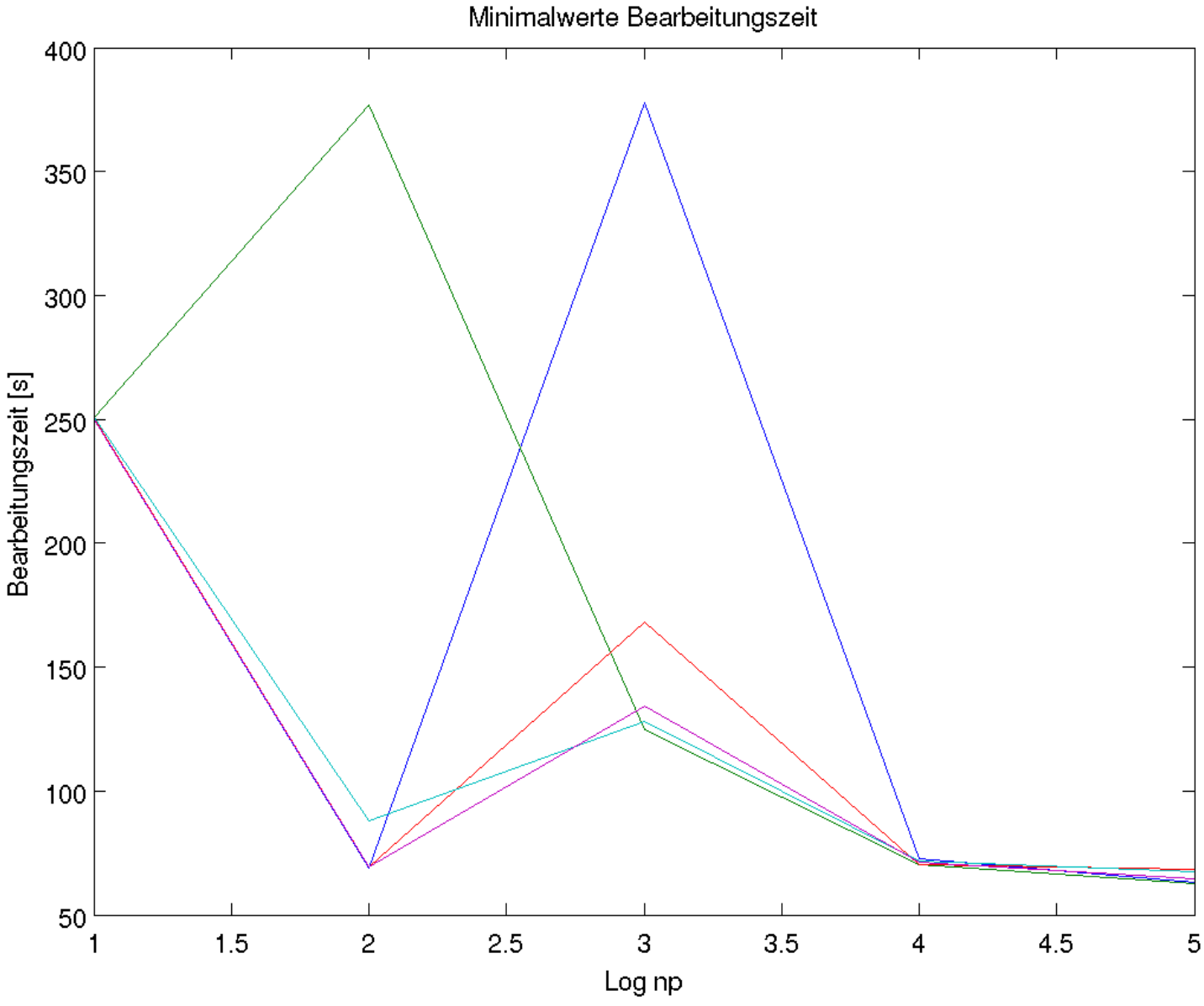
Theoretischer Wert für Hypercube-Netzwerke

T_{seq} sequentielle Bearbeitungszeit, T_{par} parallele Bearbeitungszeit, n Anzahl Prozesse, h Baumtiefe
Auf einem $log_2(n)$-dimensionalen Hypercube-Netzwerk:
Für alle $\epsilon > 0$ gibt es eine Phasenzeit mit

$$T_{par} \in (1+\epsilon)\frac{T_{seq}}{n} + \tilde{O}(h)$$

(Für kleiner werdendes ϵ ist längere Phasenzeit notwendig)


Mitelwerte Bearbeitungszeit
Bearbeitungszeit [s]
Log np
550
500
450
400
350
300
250
200
150
100
50
1
1.5
2
2.5
3
3.5
4
4.5
5
100
1000
10000
100000
1000000


Minimalwerte Bearbeitungszeit
Bearbeitungszeit [s]
400
350
300
250
200
150
100
50
1
1.5
2
2.5
3
3.5
4
4.5
5
Log np