

Lernziele – Kompetenzen

- ▶ **eigenständiges Einarbeiten** in einen Themenbereich der theoretischen Informatik
- ▶ eine Einführung in das Thema in einem 5-minütigen **Kurzvortrag** geben
- ▶ das Thema anschaulich und gut aufbereitet in einem 35-minütigen **wissenschaftlichen Vortrag** vermitteln
- ▶ Themen der anderen Teilnehmer **aktiv diskutieren**
 - Grundfähigkeiten des wissenschaftlichen Arbeitens
 - Vorbereitung auf Präsentation der Bachelorarbeit

Anforderungen

- ▷ **eigenständiges** Einarbeiten
- ▷ Präsentation des Themas in **Kurzvortrag** (5 min) und **Hauptvortrag** (35 min)
- ▷ **Anwesenheit** an allen Terminen und Diskussionsbeteiligung
- ▷ Einhalten der gesetzten **Fristen**
- ▷ **schriftliche Ausarbeitung** von ca. 10 Seiten
 - **Benotung:** Qualität der Vorträge (Inhalt und Form),
schriftliche Ausarbeitung
Diskussionsbeteiligung

Betreuung

- ▷ Ihr Betreuer ist Ihr **Ansprechpartner** bei allen Fragen, sowohl inhaltlich als auch zum Vortrag.
- ▷ Es liegt in **Ihrer Verantwortung** auf ihn/sie zuzugehen.
- ▷ Sie erhalten **Feedback und Hilfestellungen** auf Vorabversionen des Vortrags und der Ausarbeitung.

→ **Verbindliche Treffen:**

mind. 1 Woche vor dem Kurzvortrag

mind. 2 Wochen vor dem Hauptvortrag

Folien mind. 1 Woche vor dem Hauptvortrag

Ausarbeitung 11.7. für Feedback, **22.7. finale Abgabe**

Quelle

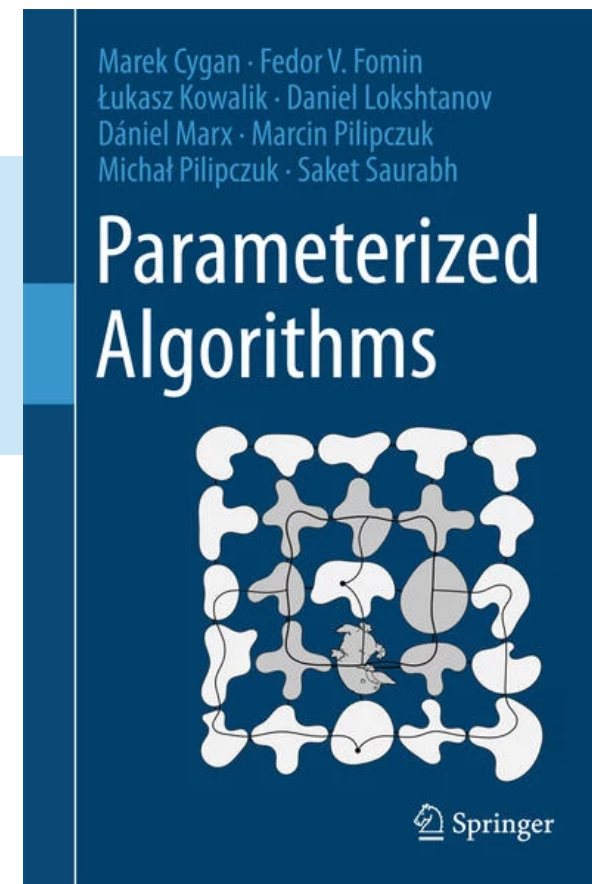
- ▶ Wir behandeln das Buch

“Parameterized Algorithms”
von *Cygan, Fomin, Kowalik, Lokshtanov, Marx, Pilipczuk, Pilipczuk und Saurabh.*

- ▶ Einführung in FPT-Algorithmen
- ▶ viele Beispiele, ausführliche Erklärungen
- ▶ ca. 10 Seiten pro Thema
- ▶ Kapitel 1,2,3,4,5 und 7 (in Teilen)

→ verfügbar als E-Ressource:

<https://services.bibliothek.kit.edu/primo/start.php?recordid=KITSRC455177058>



Themen

(1) Introduction and Definitions

- ▷ Motivation, Einführung
- ▷ formale Definitionen
- ▷ FPT Algorithmen, XP Algorithmen, W[1]-schwer
- ▷ Beispiele und Anschauung!

Problem	Good news
BAR FIGHT PREVENTION	$\mathcal{O}(2^k \cdot k \cdot n)$ -time algorithm
CLIQUE WITH Δ	$\mathcal{O}(2^\Delta \cdot \Delta^2 \cdot n)$ -time algorithm
CLIQUE WITH k	$n^{\mathcal{O}(k)}$ -time algorithm
VERTEX COLORING	

Material: Kapitel 1, Seiten 3–14

Betreuer: Tim Zeitz

Themen

(2) Kernelization

- ▷ Preprocessing
- ▷ Reduktion der Instanz
- ▷ formale Definitionen: Kernel
- ▷ 3 erste Beispiele

While usually in Computer Science we measure the efficiency of an algorithm by estimating its running time, the central measure of the efficiency of a kernelization algorithm is a bound on its output size.

Lemma 2.2 implies that a decidable problem admits a kernel if and only if it is fixed-parameter tractable. Thus, in a sense, kernelization can be another way of defining fixed-parameter tractability.

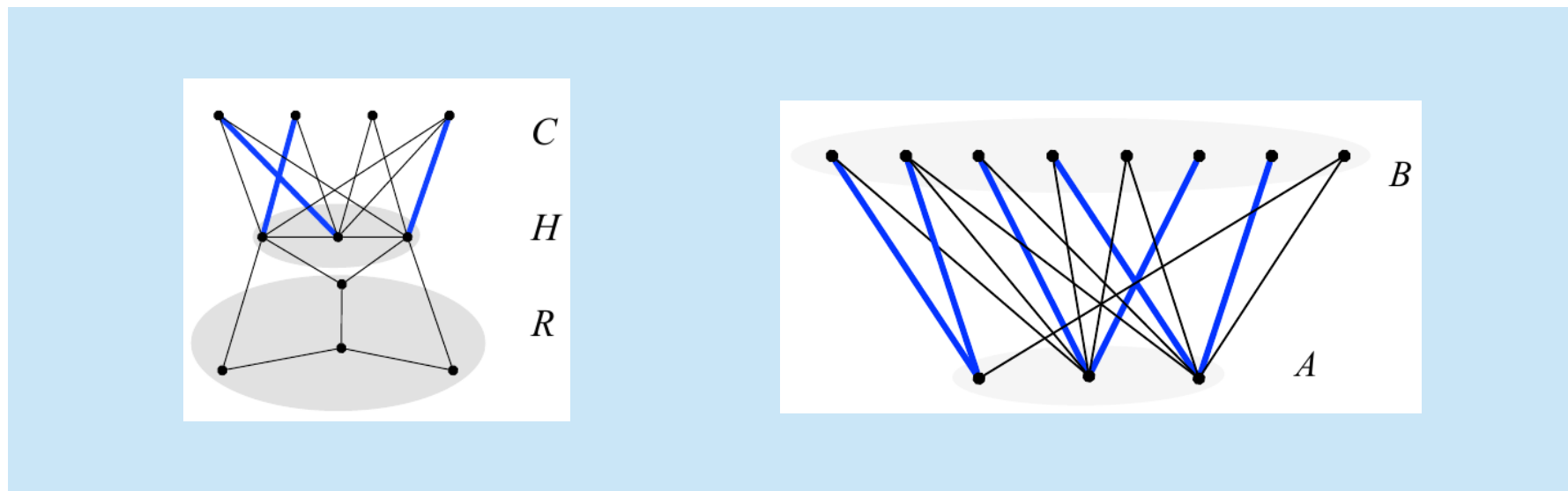
Material: Kapitel 2.1 und 2.2, Seiten 17–26

Betreuer: Matthias Wolf

Themen

(3) Other Kernelization Techniques

- ▷ Crown decomposition
- ▷ Expansion Lemma
- ▷ Sunflower Lemma
- ▷ Matchings in bipartiten Graphen
- ▷ Vertex Cover, Max SAT, Hitting Set



Material: Kapitel 2.3, 2.4 und 2.6, Seiten 26–33 und 38–39

Betreuer: Laura Merker

Themen

(4) Bounded Search Trees

- ▷ branching, backtracking
- ▷ Rekursion und Rekursionsgleichungen
- ▷ Vertex Cover
- ▷ Feedback Arc Set
- ▷ Closest String

$$T(i) = \begin{cases} T(i-1) + T(i-2) & \text{if } i \geq 2, \\ 1 & \text{otherwise.} \end{cases}$$

$$\implies T(k) \leq \left(\frac{1+\sqrt{5}}{2}\right)^k$$

Material: Kapitel 3 außer 3.4, Seiten 51–60 und 67–69

Betreuer: Christopher Weyand

Themen

(5) Parametrized Algorithms from LP-Relaxations

- ▷ Lineare Programme (LP)
- ▷ Ganzzahlige Lin. Prog. (ILP)
- ▷ Vertex Cover
- ▷ Kernelization, Branching Algorithm

ILP	min	$\sum_{v \in V(G)} x_v$	
	subject to	$x_u + x_v \geq 1$	for every $uv \in E(G)$,
		$x_v \in \{0, 1\}$	for every $v \in V(G)$.
LP	min	$\sum_{v \in V(G)} x_v$	
	subject to	$x_u + x_v \geq 1$	for every $uv \in E(G)$,
		$0 \leq x_v \leq 1$	for every $v \in V(G)$.

Material: Kapitel 2.5 und 3.4, Seiten 33–38 und 60–67

Betreuer: Max Göttlicher

Themen

(6) Iterative Compression

- ▷ gegeben:
Lösung mit Wert $k + 1$
- ▷ finde:
Lösung mit Wert k
- ▷ Feedback Vertex Set
in Tournaments
- ▷ Feedback Vertex Set

Solution compression: First, apply some simple trick so that you can assume that a slightly too large solution is available. Then exploit the structure it imposes on the input graph to construct an optimal solution.

Material: Kapitel 4 bis inklusive 4.3.1, Seiten 77–88

Betreuer: Maximilian Katzmann

Themen

(7) Randomized Methods

- ▷ Monte Carlo Algorithmus
- ▷ zufälliges Finden einer Lösung
- ▷ Feedback Vertex Set
- ▷ Color Coding Methode
- ▷ Random Separation Variante

If a one-sided error Monte Carlo algorithm has success probability at least p , then repeating it independently $\lceil \frac{1}{p} \rceil$ times gives constant success probability. In particular, if $p = 1/f(k)$ for some computable function f , then we get an FPT one-sided error Monte Carlo algorithm with additional $f(k)$ overhead in the running time bound.

Material: Kapitel 5 bis inklusive 5.3, Seiten 99–108

Betreuer: Sascha Gritzbach

Themen

(8) Chromatic Coding and Derandomization

- ▷ zufällige Kantenfärbung
- ▷ Lösung zwischen den Farben
- ▷ subexponential runtime
- ▷ simuliere den Zufall deterministisch

The basic idea of derandomization is as follows: instead of picking a random coloring $\chi: [n] \rightarrow [k]$, we deterministically construct a family \mathcal{F} of functions $f: [n] \rightarrow [k]$ such that it is guaranteed that one of the functions from \mathcal{F} has the property that we hope to attain by choosing a random coloring χ .

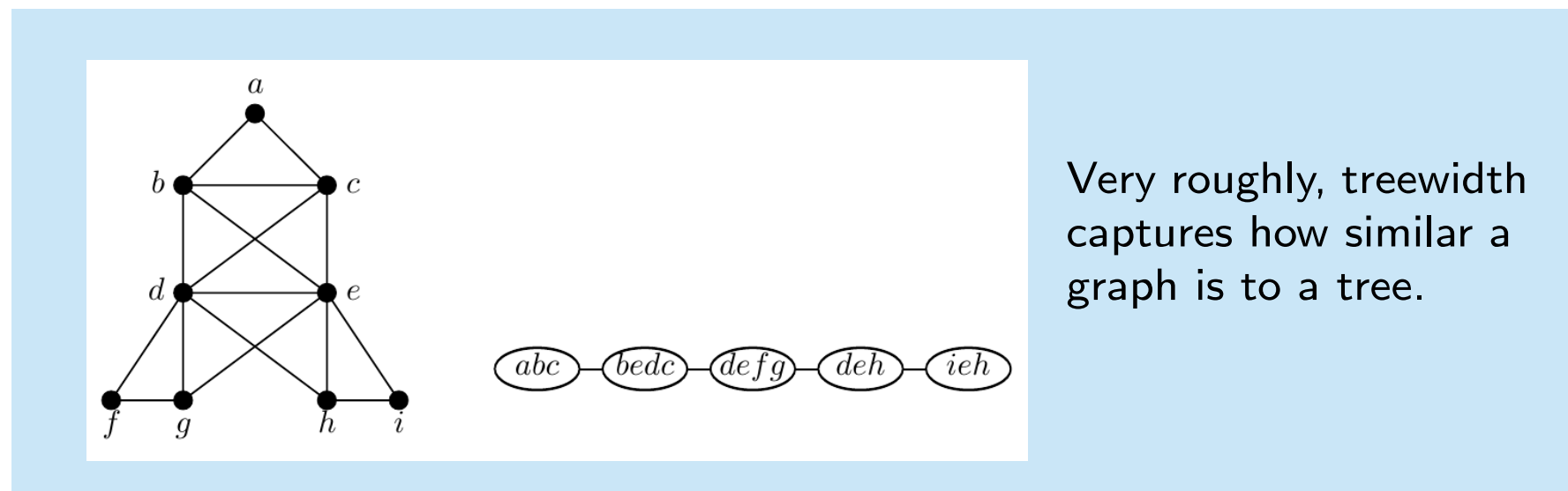
Material: Kapitel 5.5 und 5.6, Seiten 113–122 außer Seite 120

Betreuer: Adrian Feilhauer

Themen

(9) Introduction to Treewidth

- ▷ Motivation, Anschauung
- ▷ Einstiegsbeispiel: Grids
- ▷ Formale Definitionen
- ▷ Separatoren
- ▷ nice tree decompositions



Material: Kapitel 7 bis inklusive 7.2, Seiten 151–162

Betreuer: Torsten Ueckerdt

Themen

(10) Treewidth and Dynamic Programming

- ▷ Berechnung entlang nice tree decompositions
- ▷ 4 Typen von Knoten
- ▷ Weighted Independent Set
- ▷ Dominating Set

Most often, proving correctness boils down to showing two inequalities for each type of node: one relating an optimum solution for the node to some solutions for its children, and the second showing the reverse correspondence. It is usual that a precise definition of a state of the dynamic program together with function c denoting its value already suggests natural recursive formulas for c .

Material: Kapitel 7.3 außer 7.3.3, Seiten 162–171

Betreuer: Paul Jungeblut

Themen

(11) Treewidth and Monadic Second-Order Logic

- ▷ Definition MSO_2
- ▷ Beispiele, kleine Toolbox
- ▷ Satz von Courcelle
- ▷ Bezug zur Baumweite

$$\begin{aligned}\text{conn}(X) &= \forall Y \subseteq V [(\exists u \in X u \in Y \wedge \exists v \in X v \notin Y) \\ &\Rightarrow (\exists e \in E \exists u \in X \exists v \in X \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y)]\end{aligned}$$

Now, we rewrite this formula in English.

For every subset of vertices Y , if X contains both a vertex from Y and a vertex outside of Y , then there exists an edge e whose endpoints u, v both belong to X , but one of them is in Y and the other is outside of Y .

Material: Kapitel 7.4, Seiten 177–184

Betreuer: Jonas Sauer

Themen

(12) Computing Treewidth

- ▷ Baumweite ist FPT
- ▷ balanced separators
- ▷ knoten-disjunkte Wege
- ▷ Satz von Menger
- ▷ rekursive Zerlegung

Like trees, graphs of “small” treewidth have “small” balanced separators. This property is heavily exploited in numerous algorithmic applications of treewidth.

Material: Kapitel 7.6, Seiten 190–199

Betreuer: Marcus Wilhelm

(1)
**Introduction
and
Definitions**
Tim

(2)
Kernelization
Matthias

(3)
**Other
Kernelization
Techniques**
Laura

(4)
**Bounded
Search Trees**
Christopher

(5)
**Parametrized
Algorithms
from LP-
Relaxations**
Max

(6)
**Iterative
Compression**
Maximilian

(7)
**Randomized
Methods**
Sascha

(8)
**Chromatic
Coding and
Derandomiza-
tion**
Adrian

(9)
**Introduction
to Treewidth**
Torsten

(10)
**Treewidth and
Dynamic
Programming**
Paul

(11)
**Treewidth and
Monadic
Second-Order
Logic**
Jonas

(12)
**Computing
Treewidth**
Marcus

