

# Algorithmische Graphentheorie

Sommersemester 2022

10. Vorlesung

## Festparameter-Berechenbarkeit

# Herangehensweisen an NP-schwere Probleme

- Exponentialalgorithmen, z.B. Backtracking
- Approximationsalgorithmen:  
Tausche Qualität gegen Laufzeit
- Heuristiken: empirische Untersuchung auf Benchmarks
- Randomisierung: Suche nach der Nadel im Heuhaufen
- Entwurf von parametrisierten Algorithmen



NEU

# Ein Beispiel: Vertex Cover

**Def.** (Zur Erinnerung)

Sei  $G$  ein ungerichteter Graph mit Knotenmenge  $V(G)$ .  
 $C \subseteq V(G)$  heißt *Knotenüberdeckung* (eng. *vertex cover*)  
von  $G$ , falls für alle  $uv \in E(G)$  gilt, dass  $\{u, v\} \cap C \neq \emptyset$ .

**Prob.** *Kleinste Knotenüberdeckung* – Optimierungsproblem

Gegeben: Graph  $G$

Gesucht: eine kleinste Knotenüberdeckung von  $G$

**Prob.** *k-Knotenüberdeckung (k-VC)* – Entscheidungsproblem

Gegeben: Graph  $G$ , natürliche Zahl  $k$

Gesucht: Knotenüberdeckung der Größe  $\leq k$  von  $G$  –  
falls eine solche existiert  
(sonst gib „nein“ zurück)

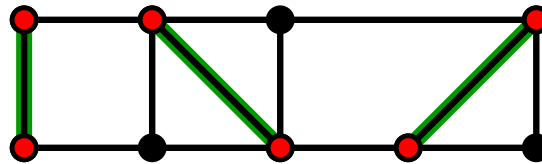
# Previous Work



- eines der ersten Probleme, dessen NP-Schwere gezeigt wurde ( $\text{SAT} \preceq_p \text{CLIQUE} \preceq_p \text{VC} \preceq_p \dots$ ) [Karp, 1972]

- eines der sechs grundlegenden NP-vollständigen Probleme in dem Klassiker: [Garey & Johnson, 1979]

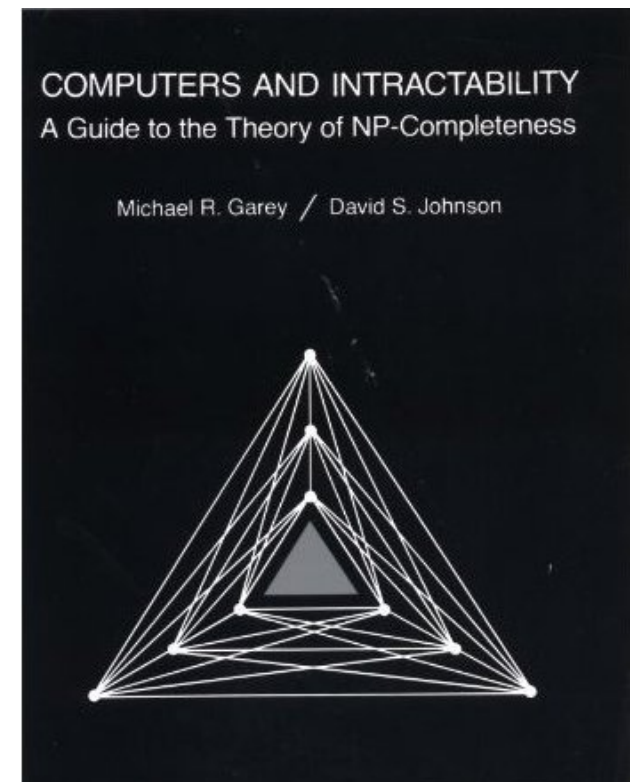
- approximierbar:



Nicht-erweiterbares **Matching** „liefert“ Faktor-2-Approximation für **VC**.

- ..., aber nicht beliebig gut:

Es gibt keine Faktor-1,3606-Approximation für VC, falls  $\mathcal{P} \neq \mathcal{NP}$ .  
[Dinur & Safra, 2004]



# Ein exakter Algorithmus für $k$ -VC

```
BruteForceVC(Graph  $G$ , Integer  $k$ )
```

```
  foreach  $C \in \binom{V(G)}{k}$  do
```

```
    // teste, ob  $C$  VC
```

```
     $vc = true$ 
```

```
    foreach  $uv \in E(G)$  do
```

```
      if  $\{u, v\} \cap C = \emptyset$  then
```

```
         $vc = false$ 
```

```
    if  $vc$  then
```

```
      return ("yes",  $C$ )
```

```
  return ("no",  $\emptyset$ )
```

$$\left| \binom{V}{k} \right| = \binom{|V|}{k} = \binom{n}{k} = O(n^k)$$

Warum?

$$O(E(G)) = O(m)$$

**Laufzeit.**  $O(n^k m)$  – Dies ist *nicht* polynomiell in der Größe der Eingabe ( $= n + m$ ), da  $k$  keine Konstante, sondern Teil der Eingabe.

## Ein neues Ziel

**Bemerkung.**

Die Klasse  $\mathcal{FPT}$  ändert sich nicht, wenn statt  $+$  hier  $\cdot$  steht.

Finde einen Algorithmus für  $k$ -VC mit Laufzeit

$$O(f(k) + |I|^c) =: O_{\zeta}^*(f(k))$$

*ignoriert polynomielle Faktoren!*

wobei  $f: \mathbb{N} \rightarrow \mathbb{N}$  berechenbare Funktion (unabh. von  $I$ ),  
 $I$  gegebene Instanz,  $c$  Konstante (unabh. von  $I$ )

D.h. die Laufzeit soll abhängen

- beliebig vom Parameter  $k$ ,
- polynomiell von der Größe  $|I|$  der Instanz  $I$ .

Ein Problem, das in dieser Zeit gelöst werden kann, heißt *festparameterberechenbar* (*fixed-parameter tractable*) bzgl.  $k$ .

$\mathcal{FPT}$  = Klasse der festparameterberechenbaren Probleme.

**Bem.** BruteForceVC hat *nicht* die gewünschte Laufzeit.

# Ein paar einfache Beobachtungen...

Sei  $G$  Graph,  $C$  ein VC für  $G$ ,  $v$  Knoten, der nicht in  $C$  liegt.  
Welche Knoten liegen dann sicher in  $C$ ?

**Beob. 1.** Sei  $G$  ein Graph,  $C$  ein VC für  $G$ ,  $v$  ein Knoten.  
Dann gilt:  $v \in C$  oder  $N_G(v) \subseteq C$ .

Betrachte Entscheidungsproblem  $k$ -VC.  
Was gilt für Knoten mit Grad  $> k$ ?

**Beob. 2.** Jeder Knoten mit Grad  $> k$  ist in jedem  $k$ -VC von  $G$  enthalten.

Was gilt, falls  $|E(G)| > k^2$  und alle Knoten Grad  $\leq k$  haben?

**Beob. 3.** Falls  $|E(G)| > k^2$  und  $\Delta(G) := \max_{v \in V} \deg v \leq k$ ,  
so hat  $G$  kein  $k$ -VC.

# Algorithmus von Buss

BussVC(Graph  $G$ , Integer  $k$ )

I) *Reduktion der Instanz auf ihren harten Kern*

$C = \{v \in V(G) \mid \deg_G(v) > k\}$

**if**  $|C| > k$  **then return** ("no",  $\emptyset$ )

$G' := G[V \setminus C]$  (entferne auch isolierte Knoten)

**if**  $|E(G')| > k^2$  **then return** ("no",  $\emptyset$ )

$O(n + m)$   
Zeit

II) *Lösung der Restinstanz mit roher Gewalt*

$k' = k - |C|$

(yesorno,  $C'$ ) = BruteForceVC( $G'$ ,  $k'$ )

**return** (yesorno,  $C \cup C'$ )

$O(m' \cdot (n')^{k'})$  Zeit  
wobei  $m' := |E'| \leq k^2$   
 $\Rightarrow n' := |V'| \leq 2k^2$

**Laufzeit.**  $O(n + m + k^2 \cdot (2k^2)^k) = O(\underbrace{n + m}_{|I|^1} + \underbrace{k^2 2^k k^{2k}}_{f(k)})$

**Also:**  $k$ -VC  $\in$  FPT!



# Vorlesungsumfrage AGT

Sehr geehrter Herr Wolff,

die Rücklaufquote der Umfrage *Algorithmische Graphentheorie* liegt aktuell bei 19% (Teilnehmerzahl insgesamt: 154).

Dieser Wert liegt unter einer definierten Schwelle von 50%.

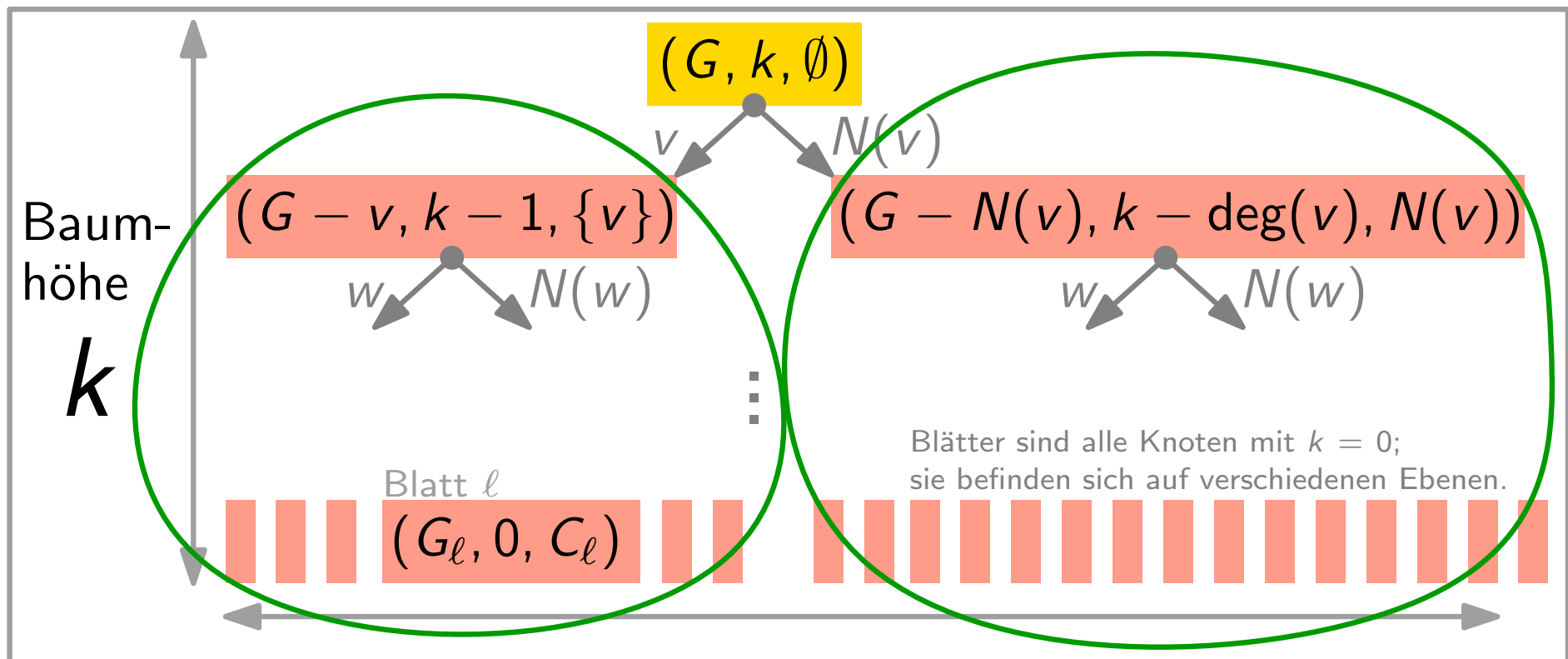
Wir möchten Sie daher bitten, Ihre Teilnehmer nochmals zu ermuntern, sich an der Umfrage zu beteiligen.

Bitte beachten Sie, dass die Umfrage am 10.07.2022 23:59:00 geschlossen wird.

Ihr EvaSys Administrator

# Suchbaum-Algorithmus

**Idee.** Verbessere Phase II durch Aufbau eines Suchbaums.

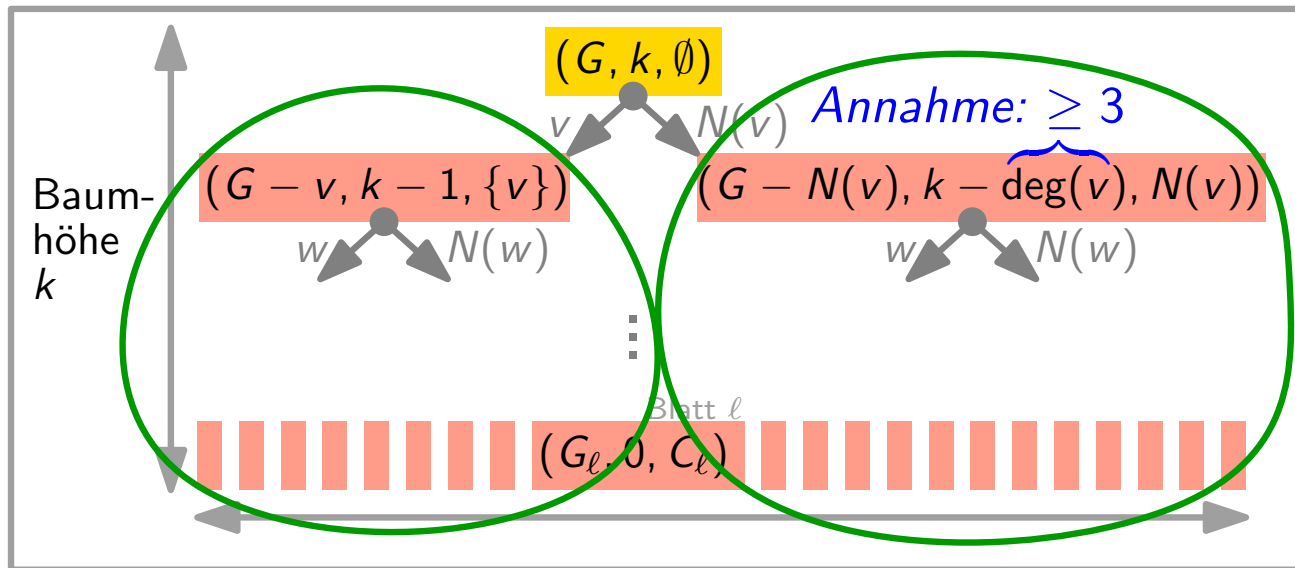


#Knoten:  $T(k) \leq 2T(k-1) + 1$ ,  $T(0) = 1 \Rightarrow T(k) \leq 2^{k+1} - 1 \in O(2^k)$   
 $\Rightarrow$  **Laufzeit:**  $O^*(2^k)$

JA: Gibt es ein Blatt  $\ell$  mit  $E_\ell = \emptyset$ , so ist  $C_\ell$  ein  $k$ -VC von  $G$ .  
 NEIN: Gibt es kein solches Blatt, so hat  $G$  kein  $k$ -VC. (Warum?)

# Der Grad-3-Algorithmus

**Idee.** Verbessere Abschätzung von  $|N(v)|$ .



$$\Rightarrow T(k) = T(k - 3) + T(k - 1) + 1, \quad T(\leq 3) = \text{const.}$$

Verzweigungsvektor  $(3, 1)$

$$\text{Teste } T(k) = z^k - 1 \quad \Rightarrow \quad z^k = z^{k-3} + z^{k-1} \quad \cdot \frac{1}{z^{k-3}}$$

$\Rightarrow$  Charakteristisches Polynom:  $z^3 = 1 + z^2$

$\Rightarrow$  Größte positive Lösung:  $z \approx 1,47$  (Verzweigungszahl)

$\Rightarrow T(k) \in O(1,47^k)$ . **Aber wie stellen wir  $\deg(v) \geq 3$  sicher?**

# Kernbildung II

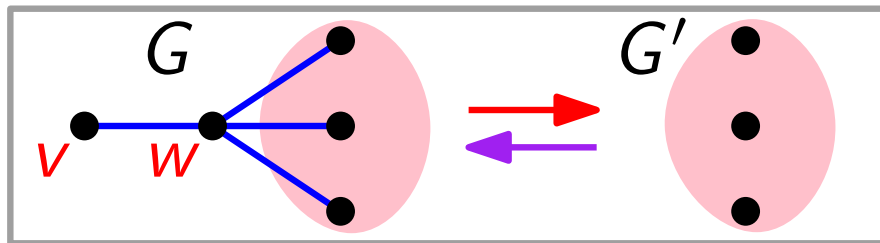
## Bisherige Kernbildung:

Regel K: Eliminiere Knoten mit Grad  $> k$

Regel 0: Eliminiere Knoten mit Grad 0

## Verbesserte Kernbildung:

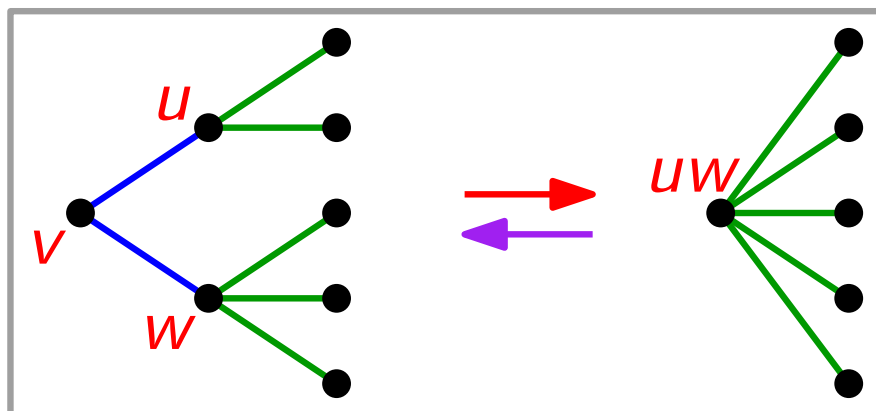
Regel 1: Eliminiere Knoten mit Grad 1



Setze  $k' = k - 1$ .  
Berechne  $C'$ .

$$C = C' \cup \{w\}$$

Regel 2: Eliminiere Knoten mit Grad 2



Setze  $k' = k - 1$ .  
Berechne  $C'$ .

Falls  $uw \in C'$ ,  
nimm  $u$  und  $w$  in  $C$ ,  
sonst  $v$  ( $\Rightarrow k = k' + 1$ )

# Der Grad-3-Algorithmus

**Idee:** Wende die verbesserte Kernbildung *in jedem Knoten* des Suchbaums *erschöpfend* an!

$$\Rightarrow \text{Laufzeit: } O(\boxed{nk} + \boxed{k^2} \cdot 1,47^k) \subseteq O^*(1,47^k)$$

↑  
Vorverarbeitung

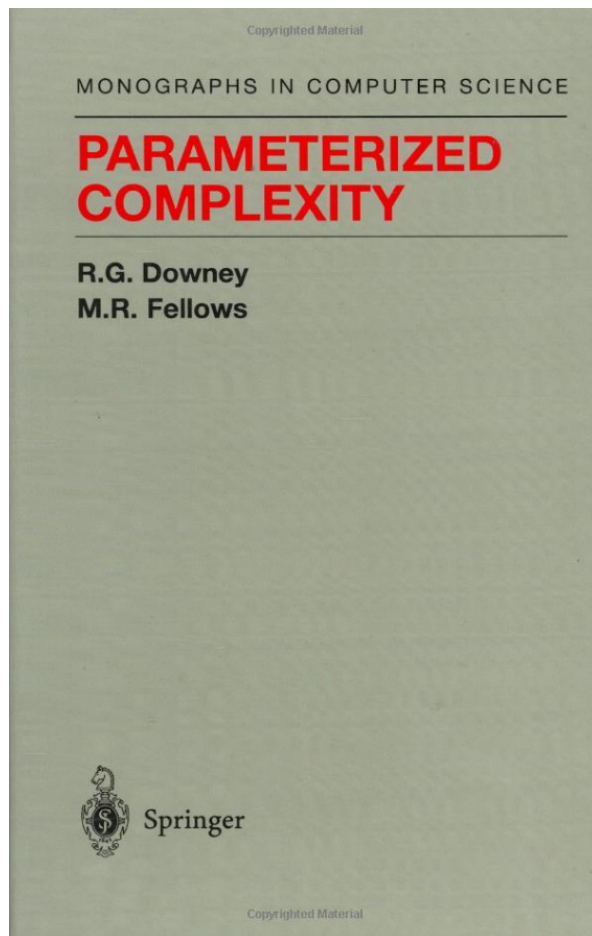
↑  
Kernbildung in jedem Knoten

Der Grad-3-Algorithmus ist also ein Fest-Parameter-Algorithmus.

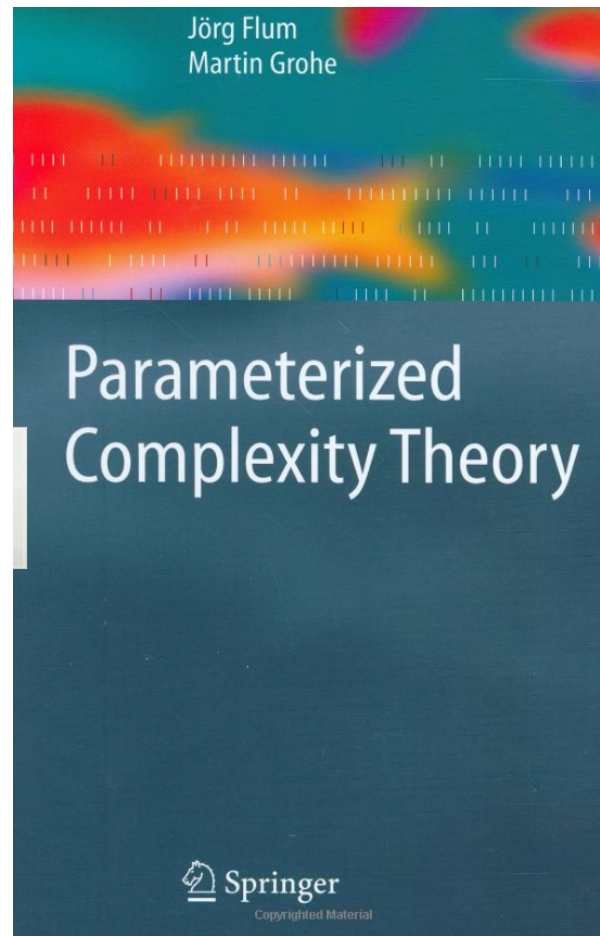
# Fazit

- $k$ -VC kann in  $O(nk + 1,47^k k^2)$  Zeit gelöst werden.  
Der momentan beste FPT-Algo läuft in  $O^*(1,27^k)$  Zeit.  
[Chen, Kanj, Jia, 2001]
- Parametrisierte Komplexität =  
neuer Werkzeugkasten für schwere Probleme:  
Kernbildung, Suchbäume, dynamische Programmierung, ...
- Es ist immer sinnvoll, beschränkte Parameter zu identifizieren – FPT nutzt sie!
- Hoffnung:  
„natürliches“ Problem  $P \in \mathcal{FPT} \Rightarrow f(k)$  erträglich.

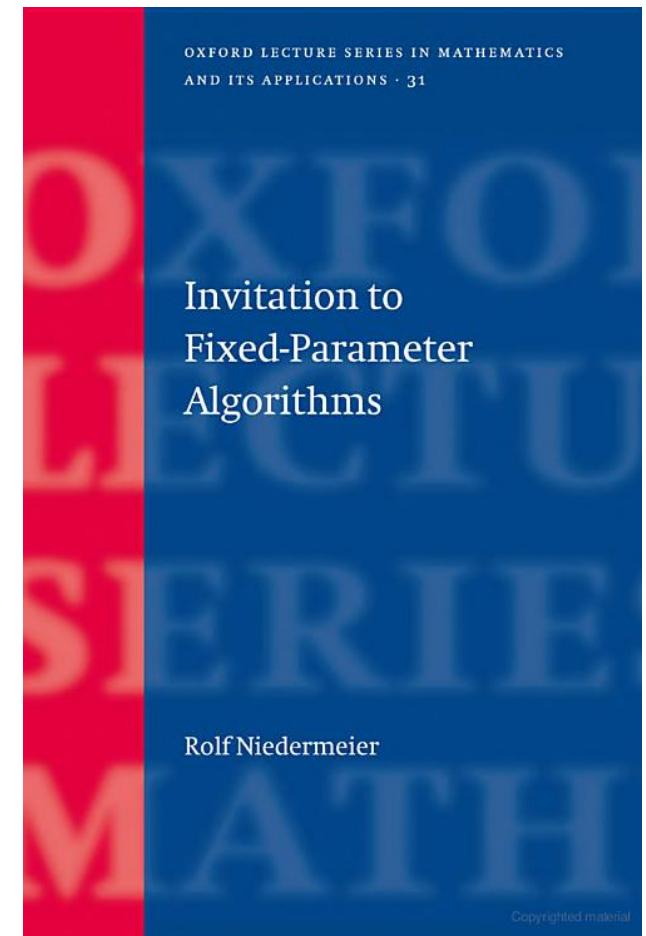
# Bücher zum Thema



1999



2006



2006