Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Algorithmen & Komplexität

Institut für Informatik

Algorithmische Graphentheorie

Sommersemester 2020

10. Vorlesung

Festparameter-Berechenbarkeit

Alexander Wolff

Lehrstuhl für Informatik I

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

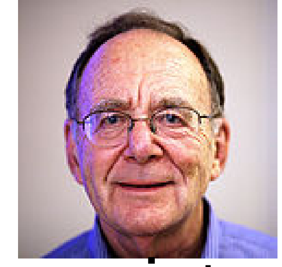
Ein Beispiel: Vertex Cover

Def. (Zur Erinnerung)
Sei $G = (V, E)$ ein ungerichteter Graph.
$C \subseteq V$ heißt *Knotenüberdeckung* (engl. *vertex cover*) von G, falls für alle $uv \in E$ gilt $\{u, v\} \cap C \neq \emptyset$.

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)

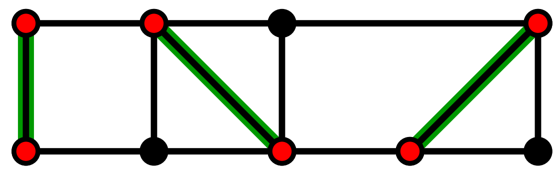
Previous Work



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

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

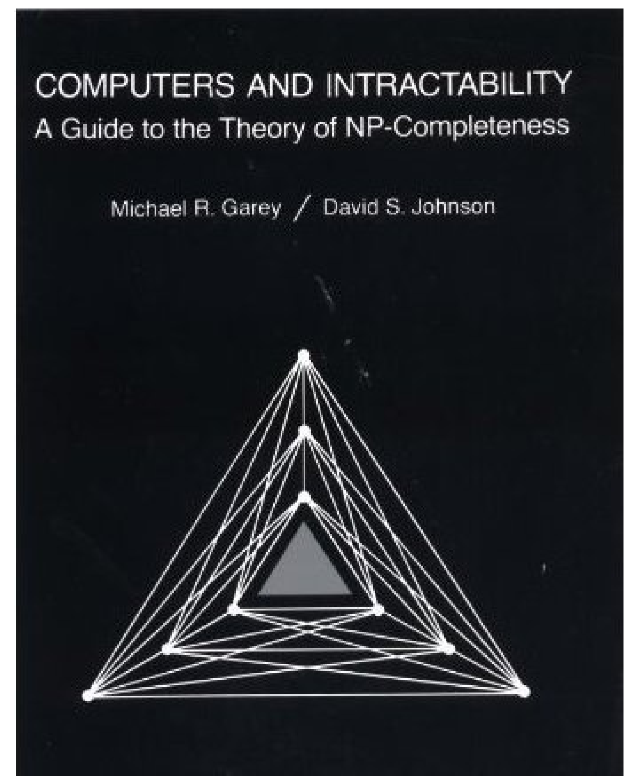
- approximierbar:



  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 ∈ (V choose k) do
      // teste, ob C VC
      vc = true
      foreach uv ∈ E do
          if {u, v} ∩ C = ∅ then
              vc = false
      if vc then
          return ("yes", C)
  return ("no", ∅)
```

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

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

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

$$O(f(k) + |I|^c) =: O^\star(f(k))$$

ignoriert polynomielle Faktoren!

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

D.h. die Laufzeit soll abhängen
– beliebig vom Parameter k, *Schwierigkeit der Instanz*
– 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 $N(v) \subseteq C$.

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

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

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

Beob. 3. Falls $|E| > k^2$ 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

$C = \{v \in V \mid \deg v > k\}$
if $|C| > k$ **then return** (“no”, $\emptyset$)
$G' = (V', E') := G[V \setminus C]$ (ohne isolierte Knoten)
$k' = k - |C|$
if $|E'| > k^2$ **then return** (“no”, $\emptyset$)

} $O(n+m)$ Zeit

II) Lösung der Restinstanz mit roher Gewalt

(vc, C') = BruteForceVC(G', k')
return (vc, $C \cup C'$)

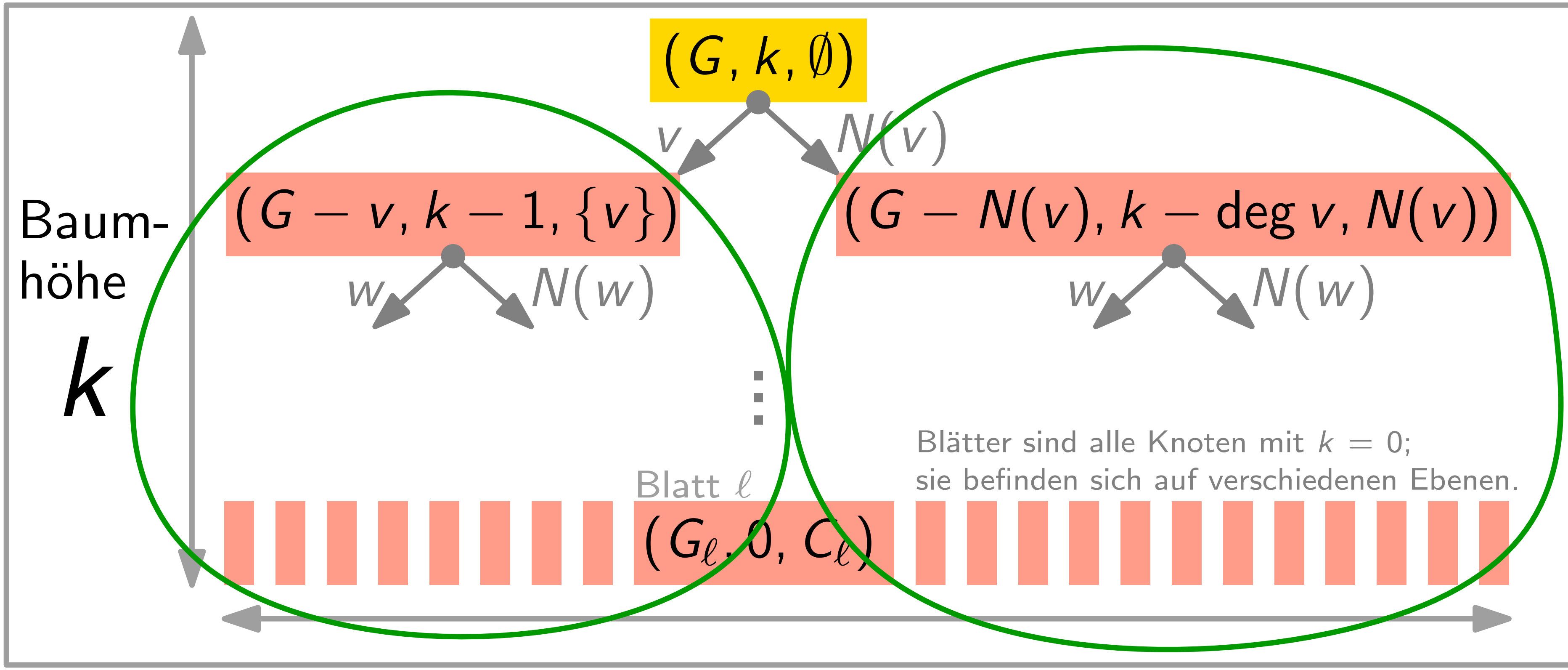
} $O(m' \cdot (n')^{k'})$ Zeit
wobei $m' := |E'| \leq k^2$
$\Rightarrow n' := |V'| \leq 2k^2$

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

Also: k-VC $\in \mathcal{FPT}$!

Suchbaum-Algorithmus

Idee. Verbessere Phase II durch Aufbau eines Suchbaums.



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

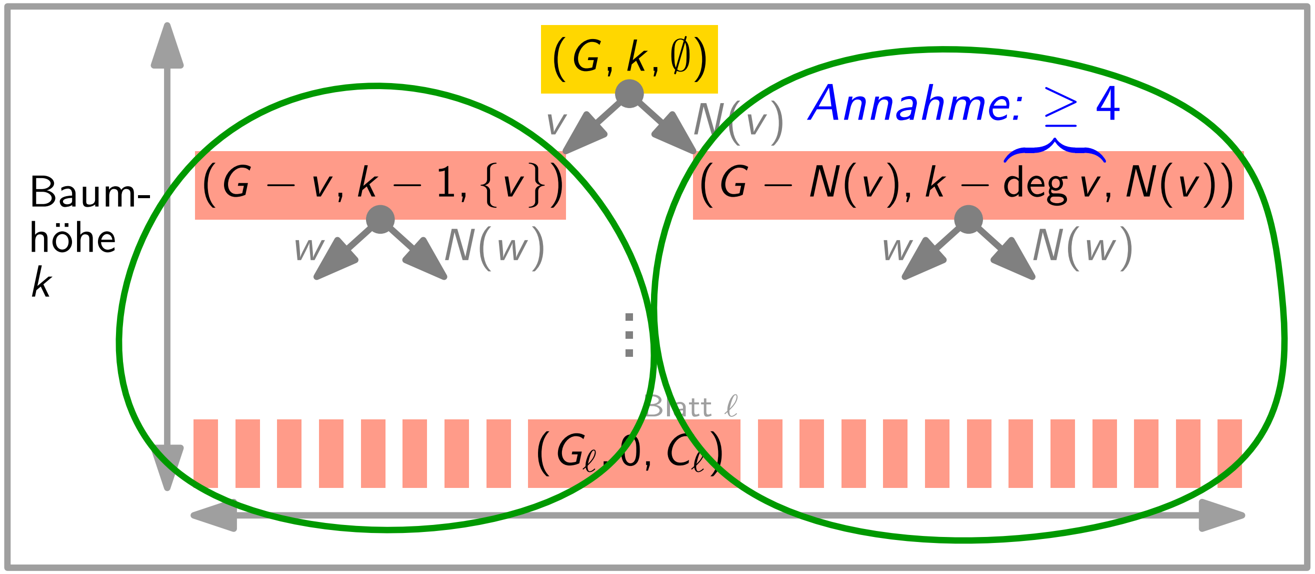
$\Rightarrow$ **Laufzeit:** $O^\star(2^k)$

JA: Gibt es ein Blatt ℓ mit $E_\ell = \emptyset$, so ist C_ℓ ein k-VC von G.

NEIN: Gibt es kein solches Blatt, so hat G kein k-VC. (Warum?)

Der Grad-4-Algorithmus

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



$\Rightarrow \quad T(k) = T(k - \boxed{4}) + T(k - \boxed{1}) + 1, \quad T(\leq 4) = const.$

Verzweigungsvektor $(4, 1)$

Teste $T(k) = z^k - 1 \quad \Rightarrow \quad z^k = z^{k-4} + z^{k-1}$ $\quad \cdot \frac{1}{z^{k-4}}$

$\Rightarrow$ Charakteristisches Polynom: $z^4 = 1 + z^3$

$\Rightarrow$ Größte positive Lösung: $z \approx 1{,}38$ (*Verzweigungszahl*)

$\Rightarrow \quad T(k) \in O(1{,}38^k)$. Aber wie stellen wir $\deg v \geq 4$ sicher?

Kernbildung II

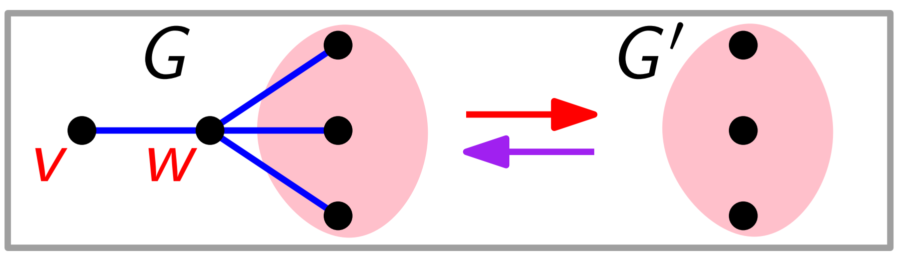
Bisherige Kernbildung:

Regel K: Eliminiere Knoten mit Grad $> k$

Regel 0: Eliminiere Knoten mit Grad 0

Verbesserte Kernbildung:

Regel 1: Eliminiere Knoten mit Grad 1

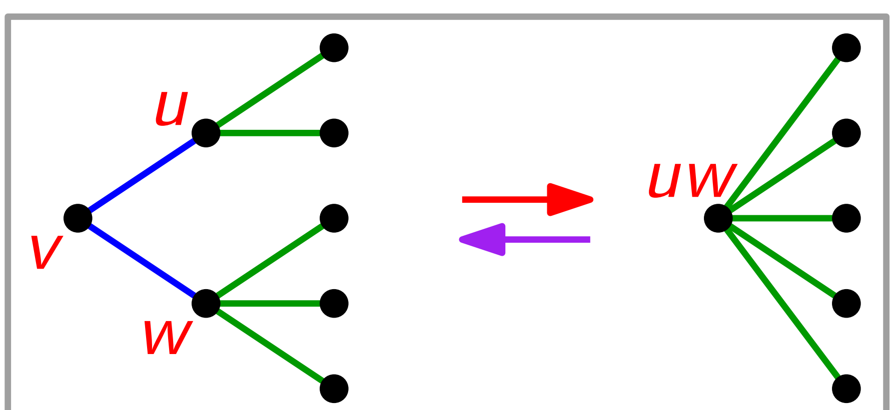


Setze $k' = k - 1$.

Berechne C'.

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

Regel 2: Eliminiere Knoten mit Grad 2



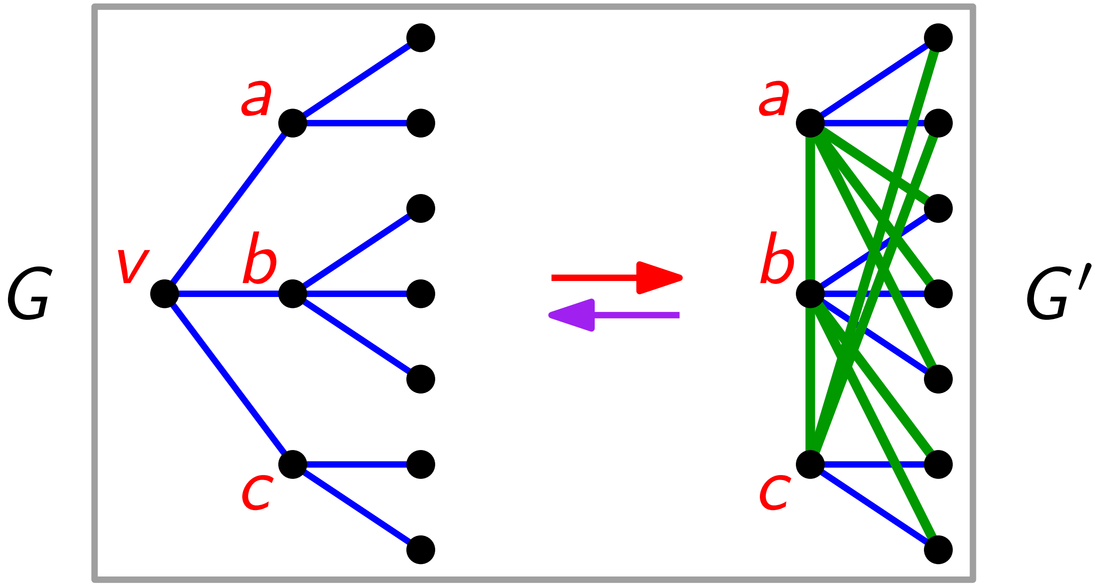
Setze $k' = k - 1$.

Berechne C'.

Falls $uw \in C'$, nimm u und w in C, sonst v.

Regel 3: Eliminiere Knoten mit Grad 3

Regel 3.1: $G[N(v)]$ enthält keine Kanten.



Beh. Es gibt ein k-VC in G $\Leftrightarrow$ Es gibt ein k-VC in G'.

[Beweis ausgelassen]

Regel 3.2: Es gibt Kanten in $G[N(v)]$.

$\dots$

Der Grad-4-Algorithmus

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

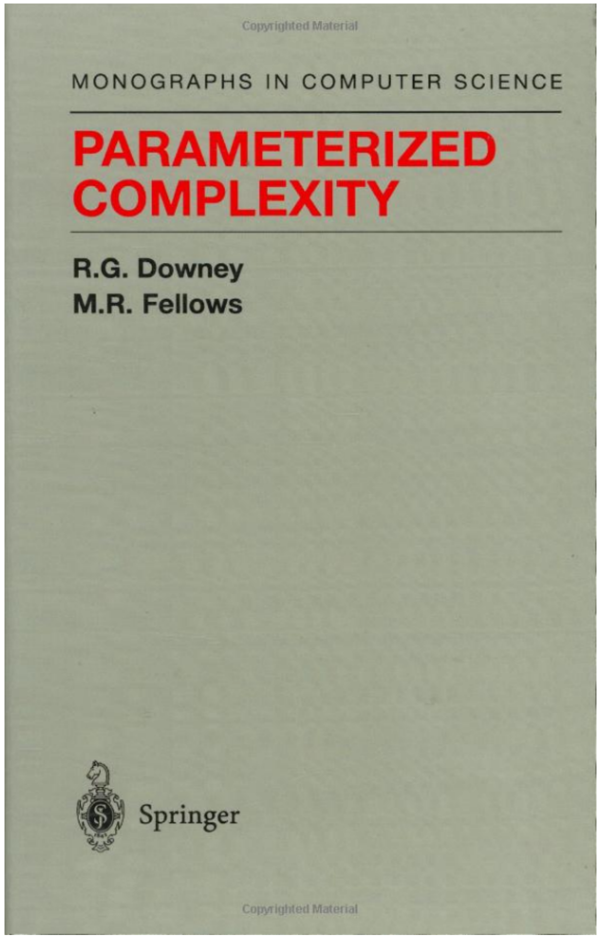
$\Rightarrow$ **Laufzeit**: $O(\boxed{nk} + \boxed{k^2} \cdot 1{,}38^k) \quad \subseteq \quad O^\star(1{,}38^k)$

Vorverarbeitung Kernbildung in jedem Knoten

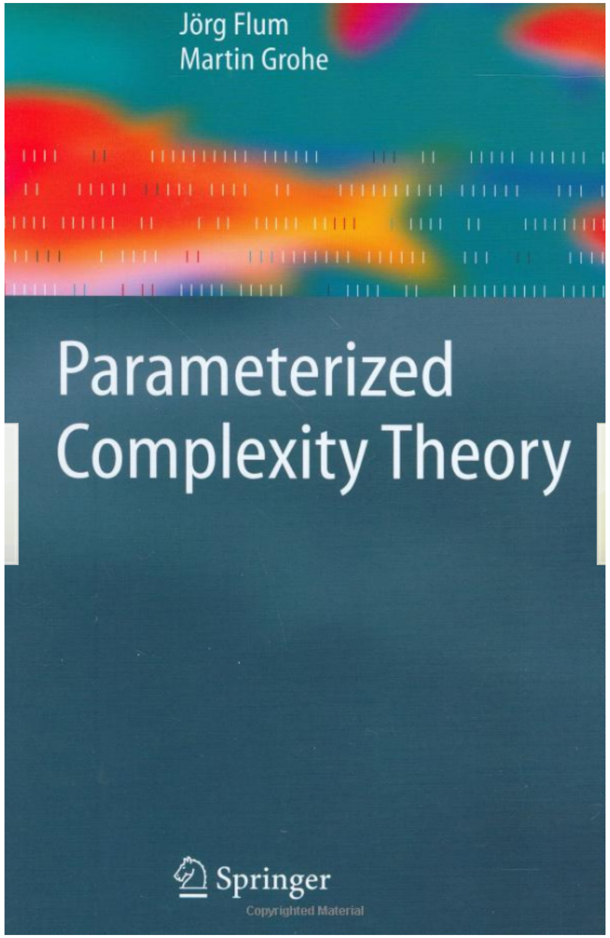
Fazit

- k-VC kann in $O(nk + 1{,}38^k k^2)$ Zeit gelöst werden.
- Parametrisierte Komplexität $=$
  neuer Werkzeugkasten für schwere Probleme:
  Kernbildung, Tabellen, Suchbäume, ...
- Es ist immer sinnvoll, beschränkte Parameter zu identifizieren – FPT nutzt sie!
- Hoffnung:
  „natürliches“ Problem $P \in \mathcal{FPT} \;\Rightarrow\; f(k)$ erträglich.

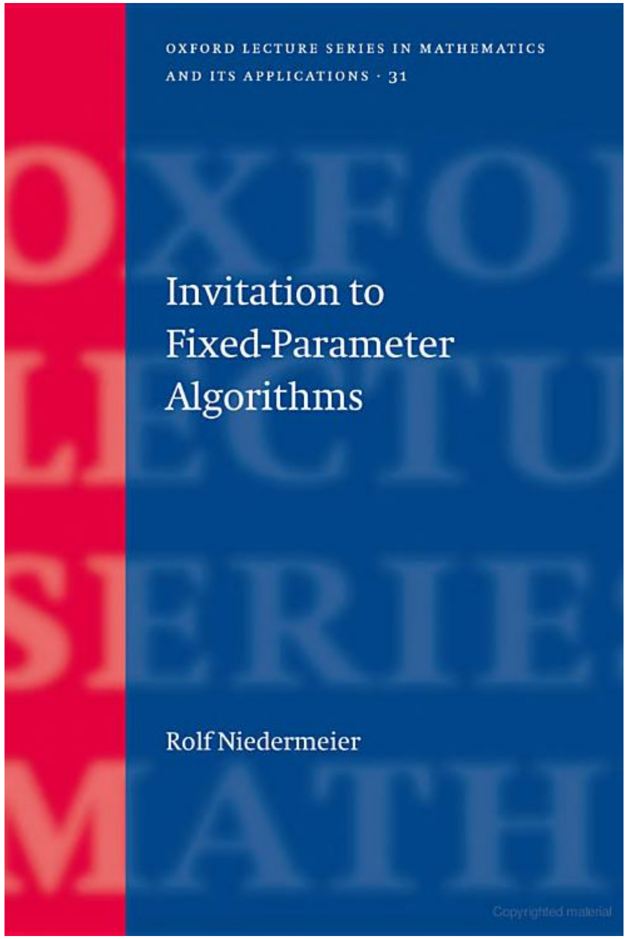
Bücher zum Thema



1999



2006



2006