Algorithmen für schwierige Probleme

Britta Dorn

Wintersemester 2011/12

7. Dezember 2011

Parametrisierte Komplexitätstheorie

Gegeben: Zwei Probleme: P (schwer) und U. Will Schwierigkeit von P und U vergleichen.

- Starte mit einer Instanz von P: (I_P, k_P).
- Baue diese Instanz so um, dass sie aussieht wie eine Instanz (I_U, k_U) von U.
  (z.B.: Mache aus einem Graphen eine Formel, oder aus einem Graphen, auf dem man P lösen möchte, einen neuen Graphen, auf dem man dann U lösen möchte.)
- Dieser Umbauvorgang darf nur FPT Zeit kosten $(f(k_P) \cdot |I_P, k_P|)$ und das neue k_U darf in seiner Größe nur vom alten k_P abhängen.
- Zeige: Wenn man (I_U, k_U) löst, dann hat man gleichzeitig (I_P, k_P) gelöst — d.h., U ist mindestens so schwer wie P, also $P \leq U$.

Weighted 2-KNF-SAT

Gegeben: Formel in 2-KNF, k
Gefragt: Gibt es eine erfüllende Belegung der Formel, in der genau k Variablen auf 'wahr' gesetzt sind?

Beispiel:
$F = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_4} \vee x_2)$, $k = 1$? $k = 2$? $k = 3$?

Parametrisierte Komplexitätstheorie

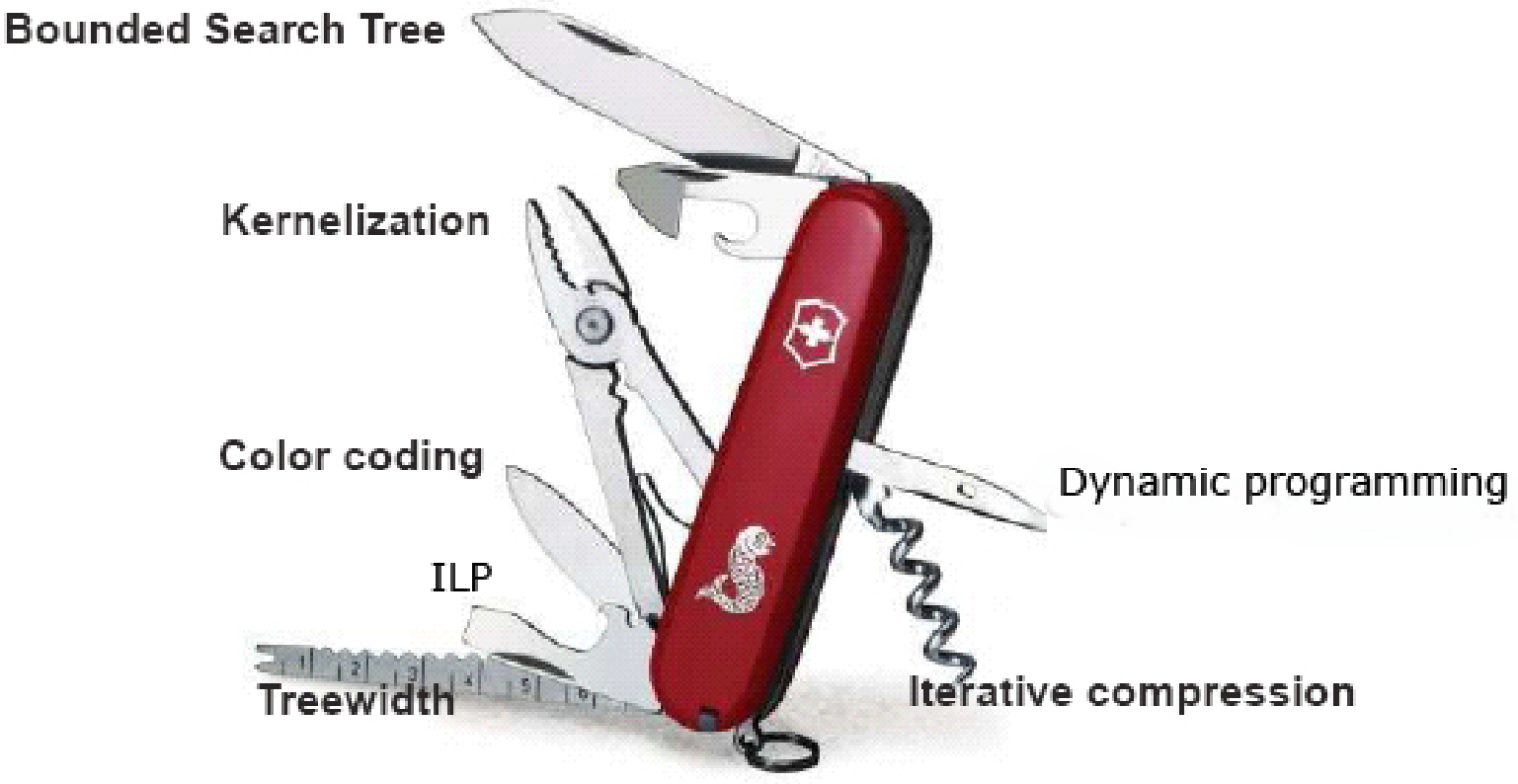
9.4.1 Definition

1. Die Klasse $W[1]$ enthält alle Probleme, die mittels einer parametrisierten Reduktion auf WEIGHTED 2-KNF-SAT reduzierbar sind.
   (Alle Probleme U, für die gilt: $U \leq$ W-2-KNF-SAT)
2. Ein parametrisiertes Problem heisst $W[1]$-**schwer**, wenn WEIGHTED 2-KNF-SAT mittels einer par. Red. auf dieses Problem reduziert werden kann.
   (W-2-KNF-SAT $\leq U$)
3. Erfüllt ein Problem beide Eigenschaften 1) und 2), so heisst es $W[1]$-**vollständig**.

Bild

Die Klasse $W[2]$ wird analog definiert, ersetze dabei WEIGHTED 2-KNF-SAT durch WEIGHTED KNF-SAT (Klauseln dürfen beliebige Länge haben).

Wiederholung/Übersicht: Block II – Methoden/Techniken



Wiederholung/Übersicht

Themen der Vorlesung

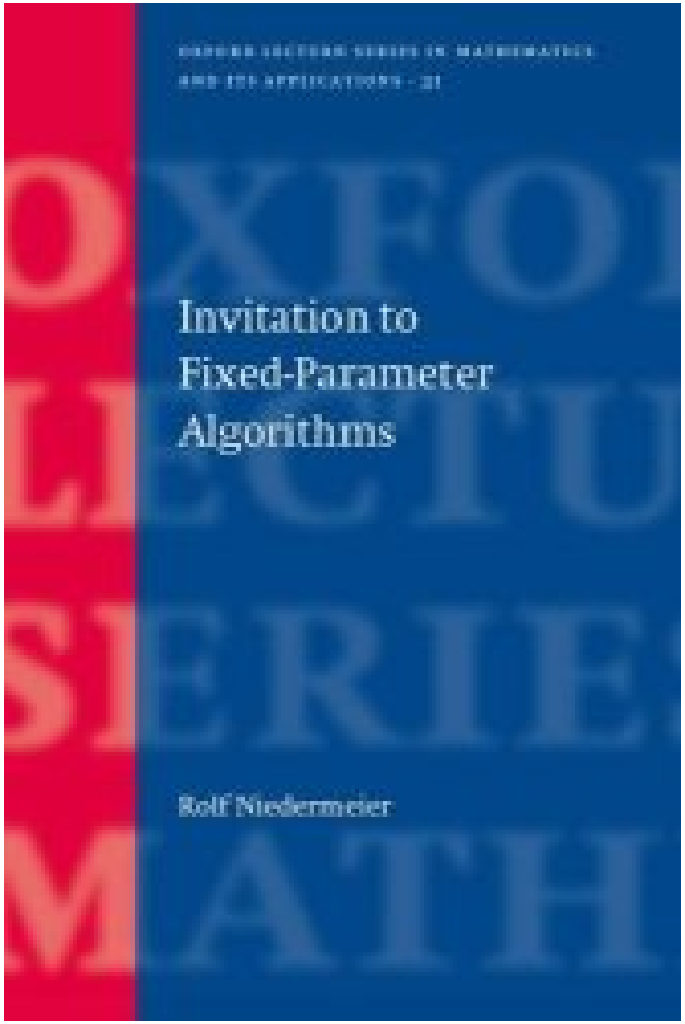
Einführung, Basics, Komplexität, Probleme (Graphen, SAT) ...

1. Grundlagen
   - Definitionen Parametrisiertes Problem, FPT, Hauptbeispiel Vertex Cover
2. Algorithmische Methoden
   - Datenreduktion und Problemkerne
   - Suchbäume
   - ILP
   - Iterative Kompression
   - Dynamisches Programmieren
   - Farbkodierung
   - Baumzerlegung von Graphen
3. Komplexitätstheorie dazu
   - Parametrisierte Reduktion
   - Parametrisierte Komplexitätsklassen, vollständige Probleme

Beispiel: Übersicht

- Welche Probleme haben wir angeschaut?
- Für welche Parameter sind sie in FPT? (was heisst das, was gilt für andere Parameter...?)
- Mit welchen Methoden konnte man das zeigen? Geht es vielleicht auch mit anderen?
- Wie nützlich sind die Algorithmen?
- Sätze, Definitionen... wie einordnen?
- Welche Probleme sind in W[1] oder höheren anderen Klassen? Was bedeutet das?
- Zusammenhang zu “normaler” Komplexitätstheorie

Buch


Invitation to
Fixed-Parameter
Algorithms
Rolf Niedermeier

Selbst Probleme lösen? Masterarbeit?

- Methoden aus der Vorlesung genauer
- Andere Methoden
- Konkrete Probleme untersuchen,
  z.B. Graphprobleme, Wahlprobleme, Pancakes, ...

Aktuelle Forschung

Wahlsysteme!

Interessante Probleme:

- Ermittlung eines Gewinners, Konsens, Rankings...
- Manipulation
- Kontrolle
- Bestechung
- Lobbyismus

Hier ist NP-Schwere eine wünschenswerte Eigenschaft!

Aktuelle Forschung: Wahlsysteme

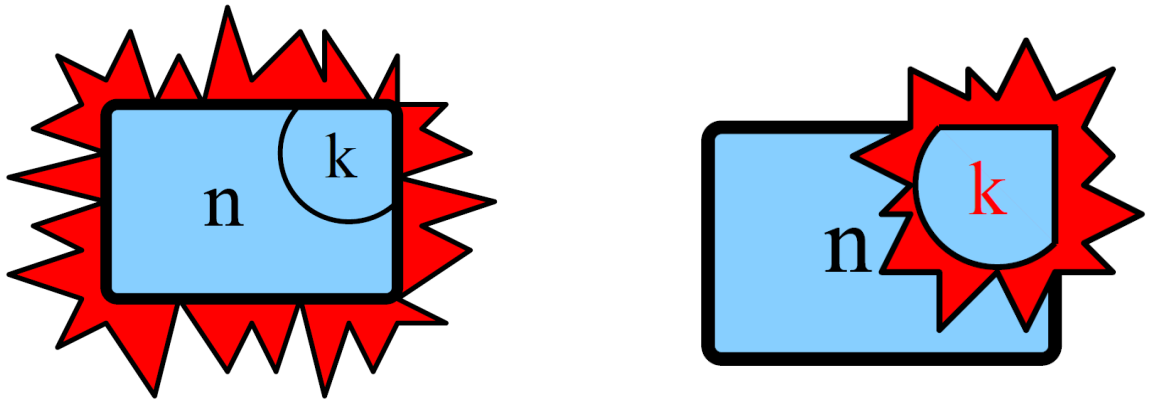
Beispiel: Wahlkontrolle

Kann der Ausgang einer Wahl durch das gezielte Zulassen/Disqualifizieren von Wählern oder Kandidaten beeinflusst werden?

gegeben: Menge von Kandidaten, Menge von Wählern, Präferenzlisten der Wähler über alle Kandidaten, $k \in \mathbb{N}$.
Frage: Kann ein bestimmter Kandidat gewinnen, wenn k Wähler disqualifiziert werden?

Dieses Problem ist für bestimme Wahlsysteme NP-vollständig. (schön, denn das heisst, das Wahlsystem ist "sicher" gegen Wahlkontrolle)

Ich habe fertig.



Viel Spaß beim Parametrisieren :)