

Parameterisierte Algorithmen

WS 2007/08 in Trier

Henning Fernau

fernau@uni-trier.de

Parameterisierte Algorithmen

Gesamtübersicht

- Einführung
- Grundbegriffe
- Problemkerne
- Suchbäume
- Graphparameter
- Weitere Methoden
- Komplexitätstheorie—parameterisiert

Organisatorisches

Vorlesung: Montag 16-18 Uhr, H7

Übungen (Daniel Raible): noch zu vereinbaren
Vorschlag: vierzehntägig, Beginn 2. Semesterwoche

Meine Sprechstunde: DO, 9-10 Uhr

Kontakt: fernau@uni-trier.de

Hausaufgaben / Schein ?! n.V.

Einführung 1: Motivation

Das Gute: P

Erinnerung: P ist die Klasse von (Entscheidungs-)Problemen, die von *deterministischen* Turing-Maschinen in Polynomzeit entschieden werden können.

Beispiele:

- Sortieren von n Gegenständen: $\mathcal{O}(n \log(n))$.
- Wortproblem für kontextfreie Sprachen: $\mathcal{O}(n^3)$.
- Matrixmultiplikation: $\mathcal{O}(n^3)$.

Das Böse: *NP*-Härte

Erinnerung: *NP* ist die Klasse von (Entscheidungs-)Problemen, die von *nichtdeterministischen* Turing-Maschinen in Polynomzeit entschieden werden können.

Meist wird von solch einem *NP*-Entscheidungsalgorithmus eine Lösung mitgeliefert. M.a.W.: Es gibt (dann) einen *deterministischen* Algorithmus, der die Gültigkeit der gefundenen Lösung *überprüft*.

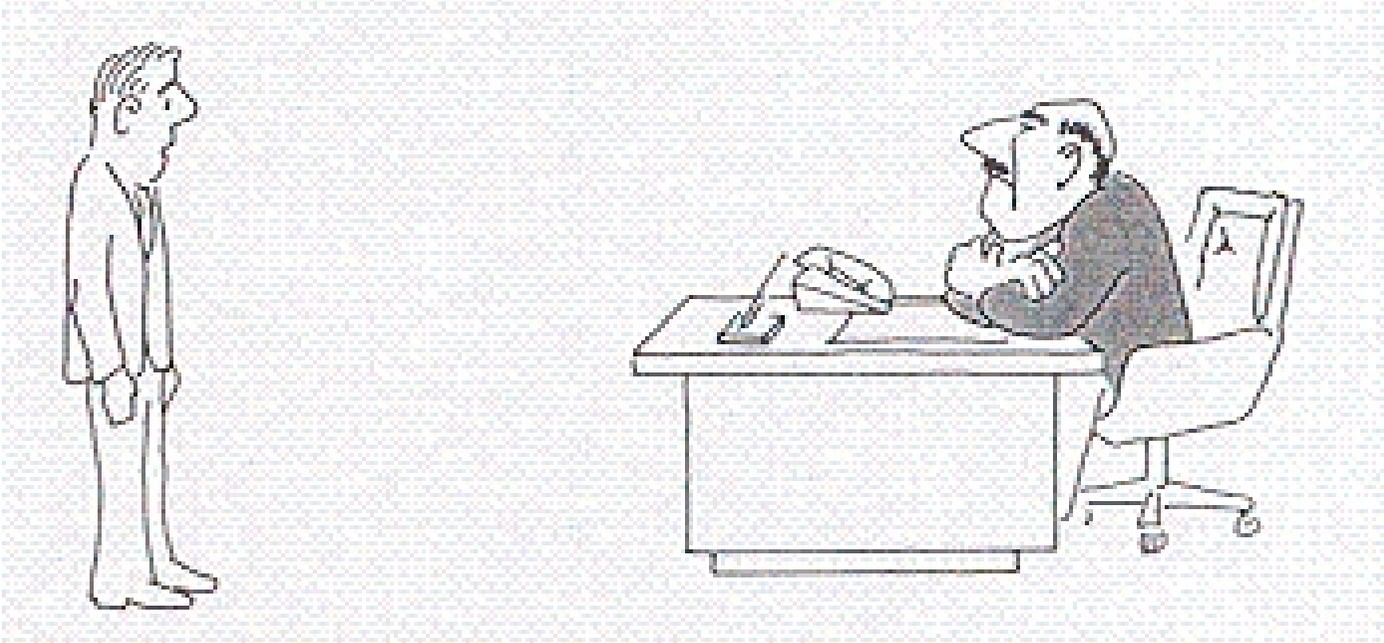
NP-vollständige Probleme sind die “schwierigsten” Probleme in *NP*. *NP-harte* Probleme sind keineswegs einfacher, liegen aber nicht unbedingt in *NP*.

Motivation

Viele interessante Probleme (aus der Praxis!) sind NP-hart
⇒ wohl keine Polynomialzeitalgorithmen sind zu erwarten.

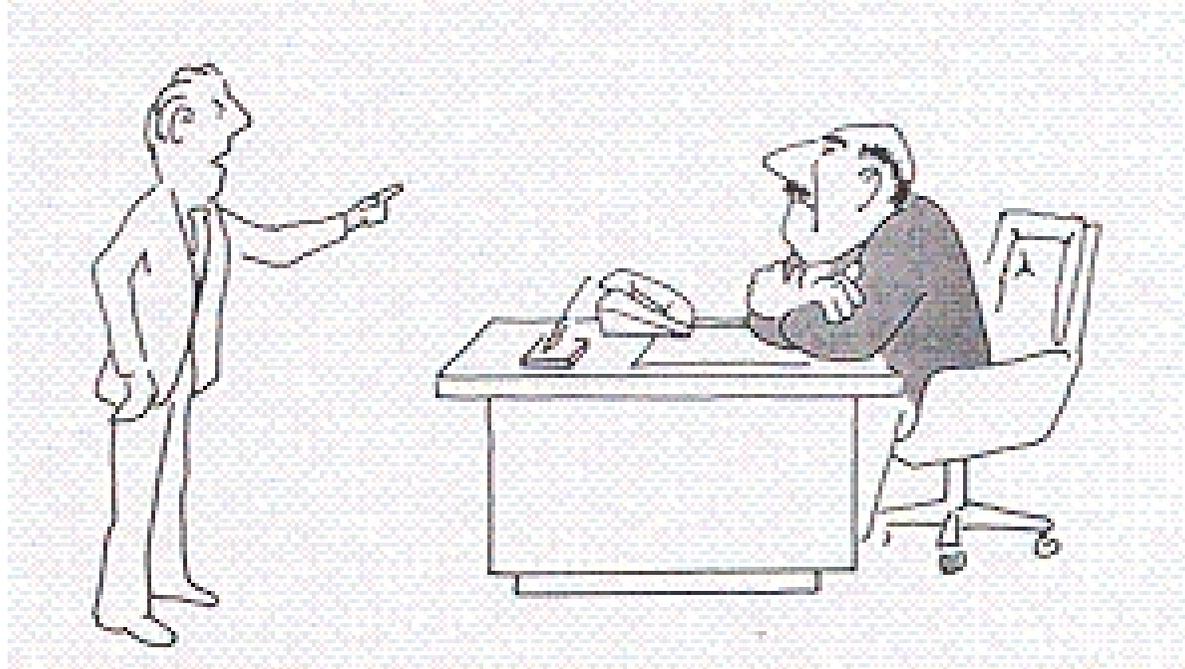
Motivation

siehe <http://max.cs.kzoo.edu/~kschultz/CS510/ClassPresentations/NPCartoons.html> wiederum aus Garey / Johnson



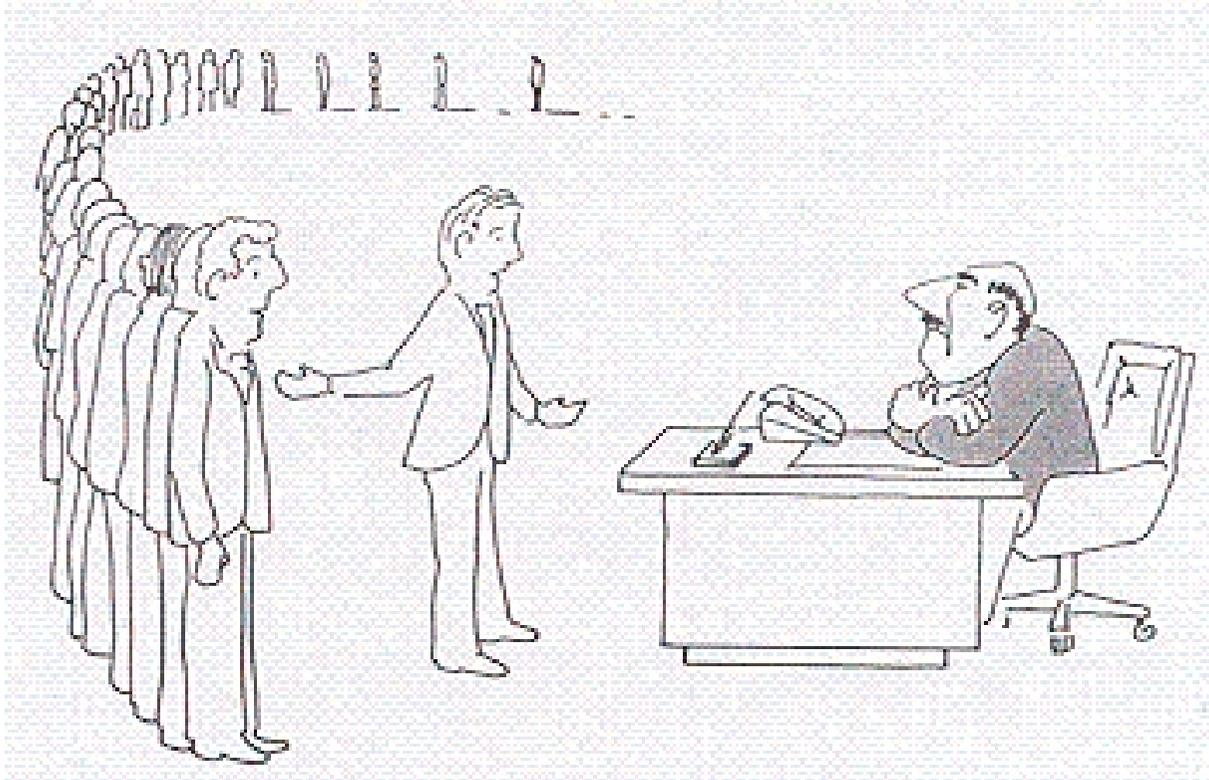
Sorry Chef, aber ich kann für das Problem keinen guten Algorithmus finden. . .

Die beste Antwort wäre hier aber...



... Ich kann aber beweisen, dass es für das Problem keinen guten Algorithmus geben kann !

Was die Komplexitätstheorie statt dessen liefert...



... Ich kann aber beweisen, dass das alle anderen auch nicht können !

Das Credo: $P \subsetneq NP$

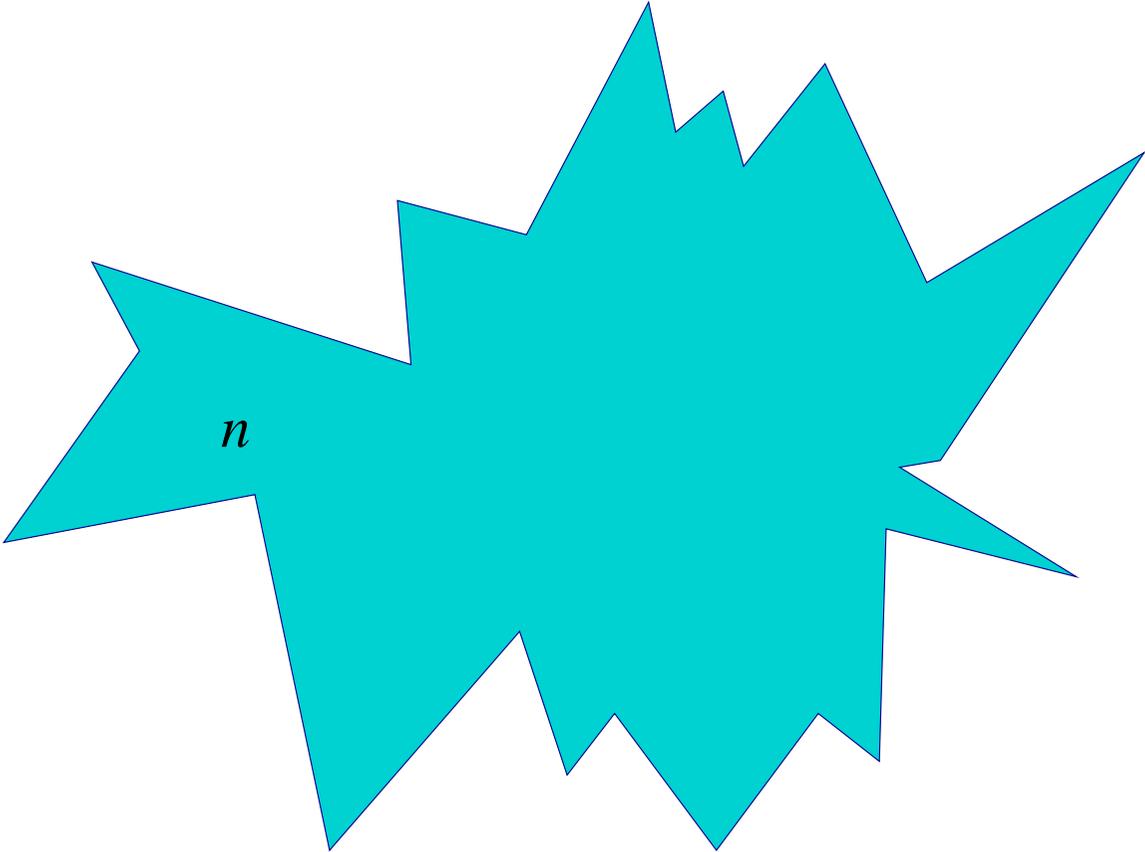
Folgerung: Für *NP*-harte Probleme “glaubt man” nicht an Polynomzeitalgorithmen zu ihrer Lösung.

“Ergo” (?) Exponentialzeitalgorithmen sind unvermeidlich für exakte Lösungen *NP*-harter Probleme.

Beispiele für *NP*-vollständige Probleme

- SATisfiability (Erfüllbarkeit logischer Formeln: Satz von Cook)
- Viele Graphprobleme:
 - Gibt es eine Knotenüberdeckung der Größe $\leq k$? (Vertex Cover VC)
 - Gibt es eine unabhängige Menge der Größe $\geq k$? (Independent Set IS)
 - Gibt es eine dominierende Menge der Größe $\leq k$? (Dominating Set DS)

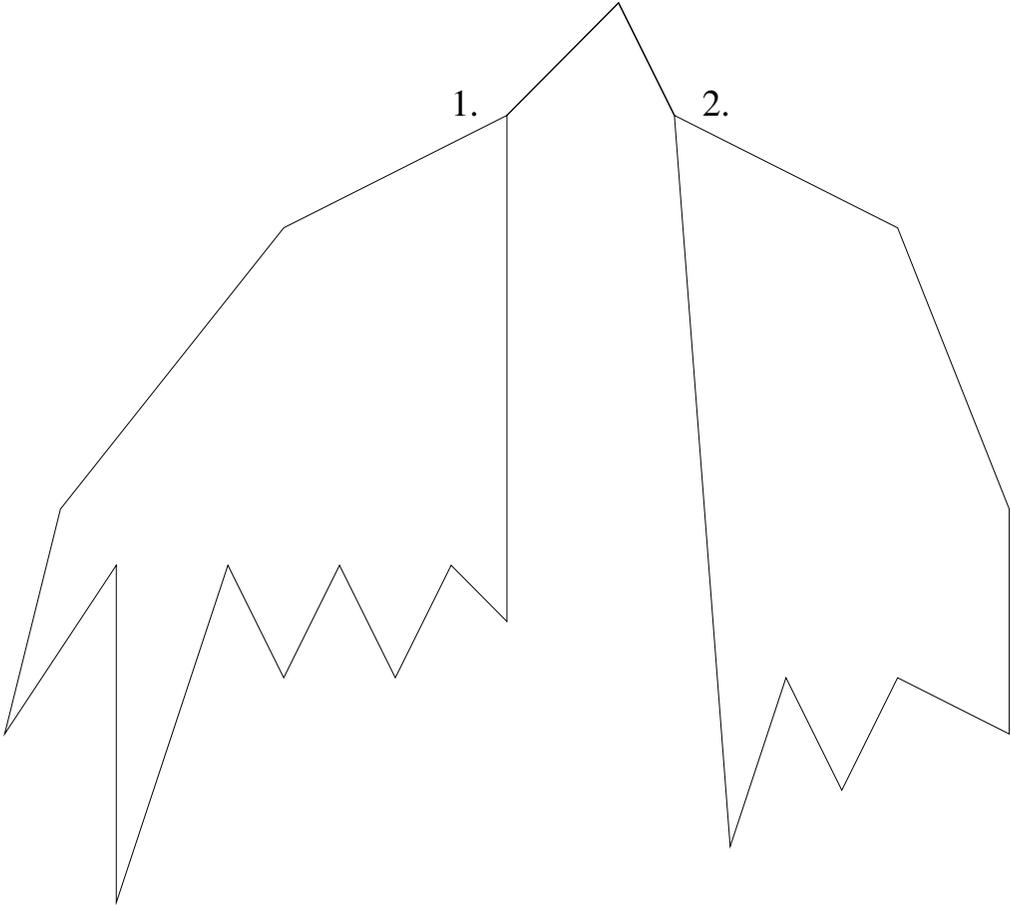
The Curse of Combinatorics (Folklore)



Auswege:

- **Suchbäume** (branch & bound):
Exponentialzeit; Laufzeitgarantien?
- Näherungsalgorithmen:
Polynomzeit; Güteschranken; nicht optimal
- Heuristiken, u.a. **Reduktionsregeln**
- Randomisierung

Suchbäume:



Wie “schlimm” ist Exponentialzeit?

Vergleichen wir exponentielle mit polynomiellen Laufzeiten für $n = 50$ auf einem Rechner, der 10^8 Operationen je Sekunde bearbeitet.

Polynomiell		Exponentiell	
Komplexität	Laufzeit	Komplexität	Laufzeit
n^2	25 μs	1.2^n	91 μs
n^3	1 ms	1.5^n	6 s
n^5	3 s	2^n	130 Tage
n^{100}	$9.13 \cdot 10^{156}$ Jahre	3^n	$228 \cdot 10^6$ Jahre

Sehr wichtig: Komplexitätsabschätzung

Thm. 3-HITTING SET ist lösbar in Zeit $\mathcal{O}(2.17 \dots^k + kn)$.

$k = \dots$	10	15	20	25	30
3^k	60 ms	15 s	3487 s \approx 1 h	847289 s \approx 10 d	57192 h \approx 6.5 y
2.18^k	3 ms	0.12 s	6 s	290 s \approx 4 min	4 h

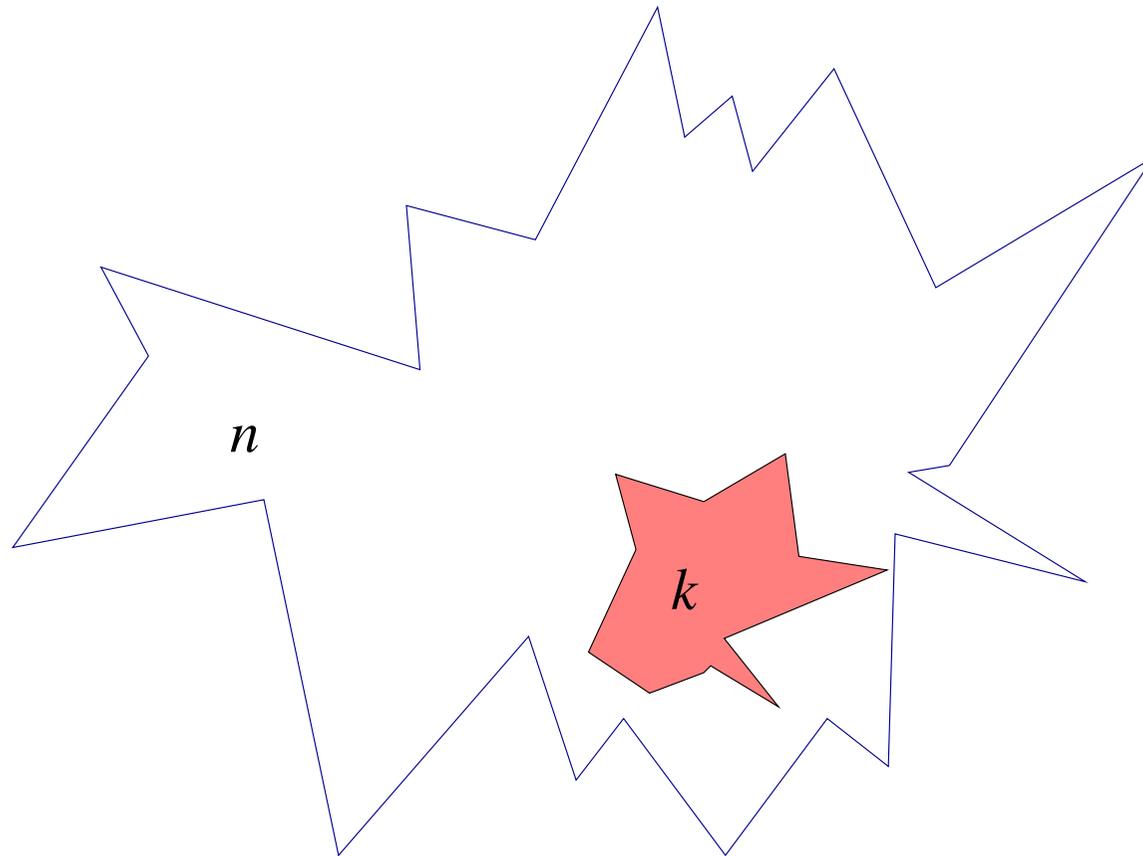
(Annahme: 10^6 Suchbaumknoten pro Sek.)

Einführung 1: Parameterisierte Algorithmen im Überblick

Parameterisierte Algorithmen

- Die parameterisierte Sicht
- Beispiel HITTING SET

The Curse of Combinatorics Eine zweidimensionale Sicht



Ziel:

- exakte Algorithmen mit
- Laufzeitgarantien

Methoden: (u.a.)

- “Problemkerne” (Datenreduktion)
- Suchbäume

Parameterisierte Algorithmik

Die Klasse **FPT (fixed parameter tractable)** enthält Sprachen $L \subseteq \Sigma^* \times \mathbb{N}$, für die es einen Algorithmus gibt, der die Frage “ $(x, k) \in L?$ ” in Zeit

$$f(k) \cdot p(|x|)$$

entscheidet (f bel., p Polynom).

Satz. Gleichwertig hiermit ist: Es gibt einen **Problemkern** (x', k') zu Instanz (x, k) mit $k', |x'| \in \mathcal{O}(g(k))$ für g beliebig.

Die zugehörige Problemkernreduktion ist in Polynomzeit berechenbar.

Spezialfall linearer Kern: $|x'| \leq \alpha k, k' \leq k$.

Literatur

R. Downey / M. Fellows: Parameterized Complexity. Springer, 1999.

H. Fernau: Parameterized Algorithmics: A Graph-Theoretic Approach (auf meiner Homepage "insgeheim" unter habil.pdf zu erhalten)

J. Flum / M. Grohe: Parameterized Complexity Theory. Springer, 2006.

R. Niedermeier: Invitation to Fixed-Parameter Algorithms. Oxford University Press, 2006.

Skript: Parametrisierte Algorithmen von R. Niedermeier / J. Alber (leicht verändert von J. Gramm) in Tübingen.

Foliensatz von P. Rossmanith in Aachen.

Datenreduktion

- Die parameterisierte Sicht
- Beispiel HITTING SET

Exkurs: Hypergraphen

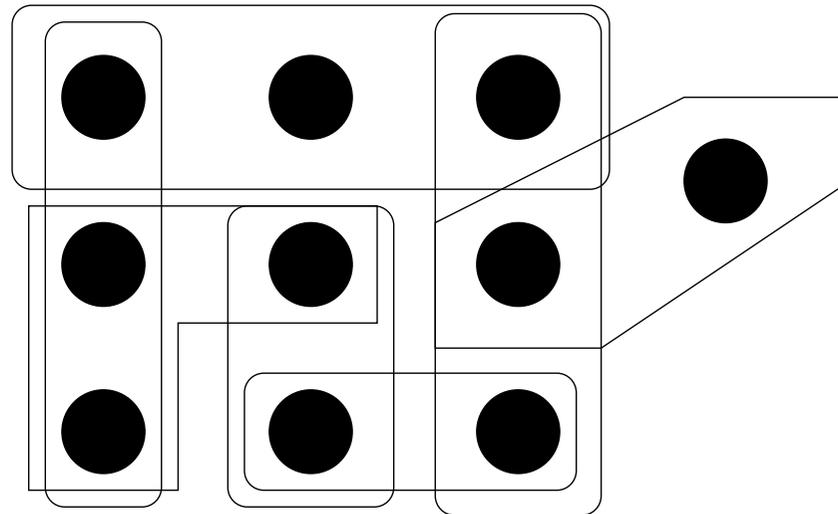
Hypergraph $G = (V, E)$:

V : Knotenmenge

$E \subseteq 2^V$: Kantenmenge

Spezialfall: $\forall e \in E : |e| \leq 2$ (ungerichteter) Graph

Ein Beispiel



3-HITTING SET: Knotenüberdeckungsproblem auf Hypergraphen

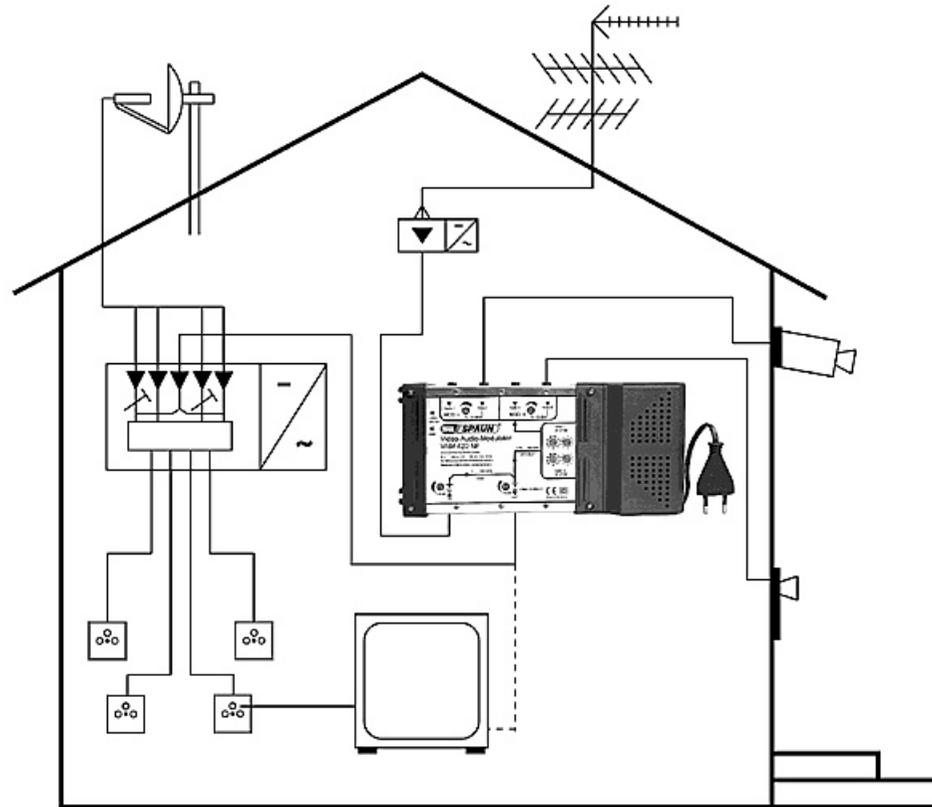
Eingabe: Hypergraph $G = (V, E)$ mit *Kantengröße* höchstens 3: $\forall e \in E (|e| \leq 3)$

Parameter: k

Frage: Gibt es eine **Knotenüberdeckung** (**hitting set**) mit höchstens k Knoten:

$$\exists C \subseteq V (|C| \leq k \wedge \forall e \in E (C \cap e \neq \emptyset)) ?$$

Motivation: Systemanalyse á la Reiter



Was ist ein System? (nach R. Reiter)

- **Systembestandteile** (Komponenten) **C**
- **Systembeschreibung** (wie? \rightsquigarrow Logik) **SD**:
Aussagen über erwartetes Systemverhalten,
d.h., Beziehungen zwischen den Komponenten.
- **beobachtetes Systemverhalten** (Observationen) **OBS**

Was ist ein fehlerbehaftetes System?

- spezielles Prädikat $ab(c)$ für jede Komponente $c \in C$:
kennzeichnet **abnormes Verhalten** (Fehler)
SD enthält auch Aussagen der Form:
“Wenn $ab(c)$, dann gilt:...” bzw.
“Wenn $\neg ab(c)$, dann gilt:...”
- ein System (C, SD, OBS) ist **fehlerbehaftet**, wenn in
$$SD \cup OBS \cup \{\neg ab(c) \mid c \in C\}$$
ein Widerspruch zu erkennen ist.

Konfliktmengen und Diagnosen

Eine *Konfliktmenge* ist eine Menge C' von Komponenten, so dass in

$$SD \cup OBS \cup \{\neg ab(c) \mid c \in C'\}$$

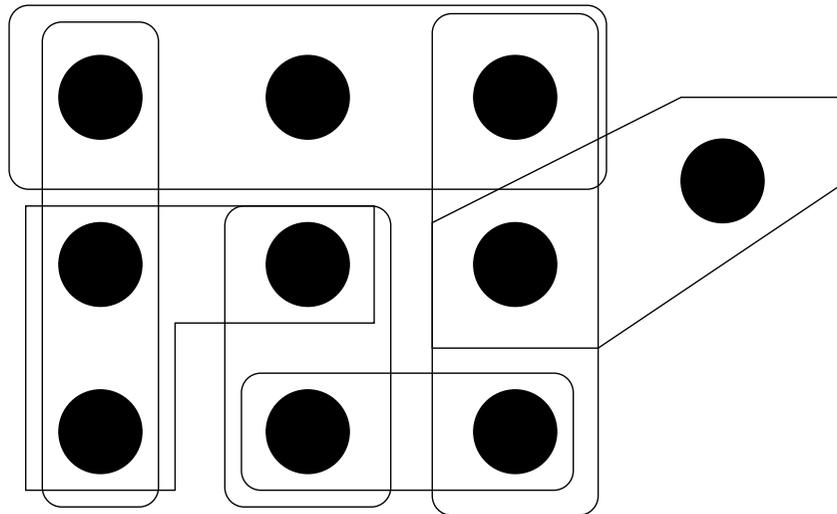
ein Widerspruch zu erkennen ist.

Eine *Diagnose* ist eine möglichst kleine Menge C' von Komponenten, so dass $C \setminus C'$ keine Konfliktmenge ist.

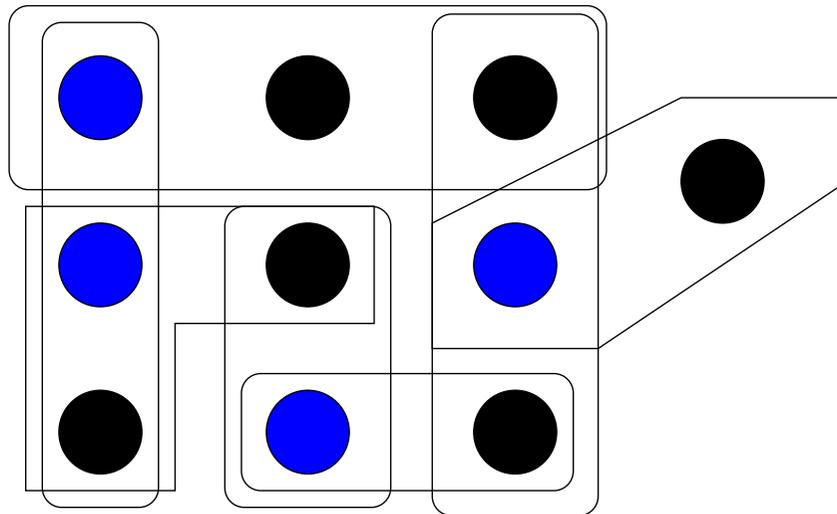
Übersetzung in Hitting Set:

Die *Hypergraphknoten* sind die *Komponenten*,
die *Konfliktmengen* sind die *Kanten*,
die *Diagnose* die *Überdeckungsmenge*.

Ein abstrakteres Beispiel



Eine kleinste Überdeckung

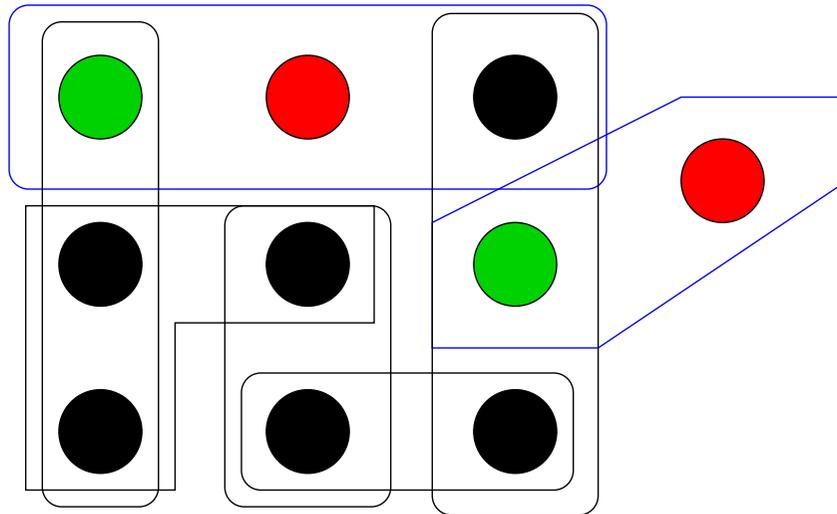


Datenreduktionsregeln

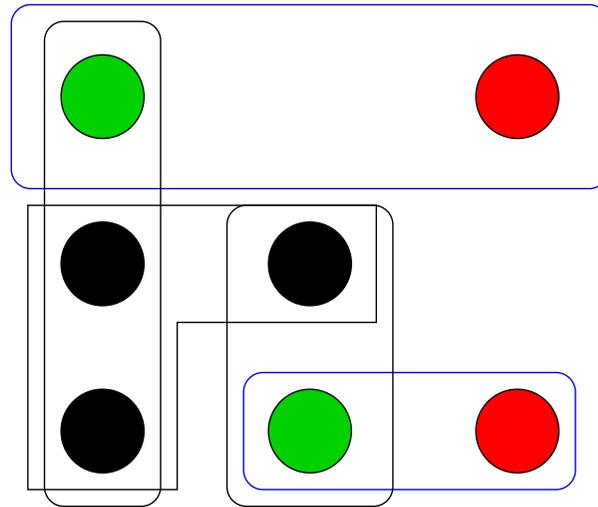
1. Kantendominierung: $f \subset e. \rightsquigarrow$ entferne e
2. Kleine Kanten: $e = \{v\} \rightsquigarrow v$ kommt ins HS; entferne e
3. Knotendominierung: Ein Knoten x heiÙe *dominiert* durch einen Knoten y , falls $\{e \in E \mid x \in e\} \subseteq \{e \in E \mid y \in e\} \rightsquigarrow$ entferne x

R. S. Garfinkel and G. L. Nemhauser. *Integer Programming*. John Wiley & Sons, 1972.
Oft wiederentdeckt: K. Weihe (Zugnetzoptimierung), R. Niedermeier & P. Rossmanith (param. HS, 2003)
Also: R. Reiter (Theory of Diagnosis \rightsquigarrow HS Bäume, 1987)

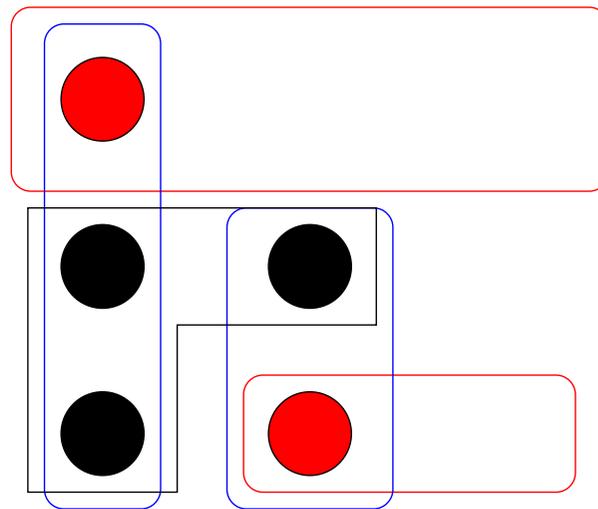
Knotendomination



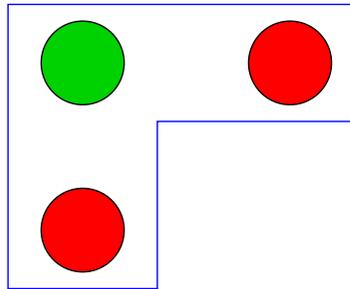
Knotendomination



Kantenregeln



Knotendomination



Verzweigungsregeln (heuristic priorities)

Bevorzuge Verzweigungen bezüglich

1. kleiner Kanten
2. Knoten hohen Grades
3. ...

Der Algorithmus HS(G, k)

1. Wende erschöpfend alle Reduktionsregeln an; das liefert (G', k')
2. **if** $k' \leq 0$ **then** return $(k' = 0 \& E(G') = \emptyset)$
3. **if possible:** Wähle $x \in V(G')$ gemäß Verzweigungsregeln
4. **if** HS($G' - E[x], k' - 1$) **then** return YES
5. **else** return HS($G' - x, k'$)