Überdeckungen in Hypergraphen:

Aufgaben, Ansätze und Anwendungen

Henning Fernau

15. Mai 2006

Überblick

- $\mathcal{FPT}$: die allgemeine Methodik
- Einführung anhand von Beispielen
- Reduktionsregeln und Suchbäume
- Analyse von Algorithmen mit einem Hilfsparameter
- Anwendungen & ähnliche Probleme

Überblick

- $\mathcal{FPT}$: die allgemeine Methodik
- Einführung anhand von Beispielen
- Reduktionsregeln und Suchbäume
- Analyse von Algorithmen mit einem Hilfsparameter
- Anwendungen & ähnliche Probleme

The Curse of Combinatorics

Folklore: *"Interessante" Probleme in der Graphentheorie sind "(NP-)hart".*

Beispiele:

- Finde eine Knotenüberdeckung der Größe $\leq k$.
- Finde eine dominierende Menge der Größe $\leq k$.
- …

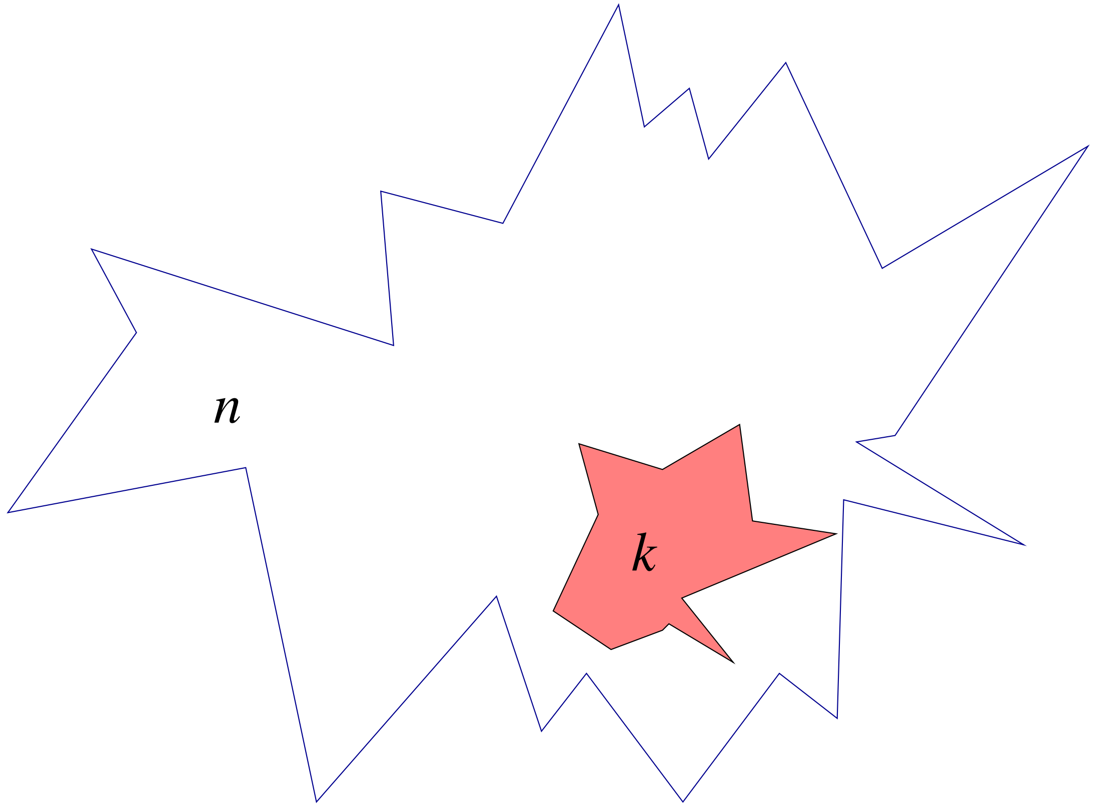
Parameterisierte Probleme und Algorithmen

Idee: Entwickle *zwei-dimensionale Sicht*. Miss Problemgröße

- in der Gesamtgröße n und
- im Parameter [der Größe] k.

Für Minimierungsprobleme ist natürlicher Parameter die Größe der Zielmenge, z.B.: die Größe der Knotenüberdeckung.

Kontrollierte Explosion funktioniert bei kleinem Parameter:



Parameterisierte Komplexität kurzgefasst: die Klasse $\mathcal{FPT}$

Laufzeit $\mathcal{O}(f(k)p(n))$

Problemkern der Größe $g(k)$, berechenbar in Polynomzeit.

Satz: Beide Ansätze liefern die selbe Komplexitätsklasse, genannt $\mathcal{FPT}$.

Frage: Wie gelangt man zu solchen Algorithmen?
Standardansätze: Suchbäume und Problemkernalgorithmen

Das Böse: Laufzeit $\mathcal{O}(n^k)$. Formal: W[1]-Härte

Alternativen?!

1. Approximation
   insbesondere klingt $\mathcal{PTAS}$ interessant ?!
   Nachteile: keine exakten Ergebnisse, Laufzeit evtl. $\mathcal{O}(n^{10000})$?!

2. Heuristiken
   Nachteil: keine Aussagen über Qualität der Ergebnisse

3. exakte Algorithmen (eindimensionale Sicht)
   Nachteil: skaliert “böse” mit Gesamtgröße

Überblick

- $\mathcal{FPT}$: die allgemeine Methodik
- Einführung anhand von Beispielen
- Reduktionsregeln und Suchbäume
- Analyse von Algorithmen mit einem Hilfsparameter
- Anwendungen & ähnliche Probleme

Dominierende Menge

DOMINIERENDE MENGE (DS)
Eingabe: ein Graph $G = (V, E)$
Parameter: k
Frage: Gibt es dominierende Menge $D \subseteq V$, d.h. $N[D] = V$, mit $|D| \leq k$?

Hierbei: $N(x)$ sind Nachbarn von x plus x selbst
$N[D] = \bigcup_{x \in D} N[x]$

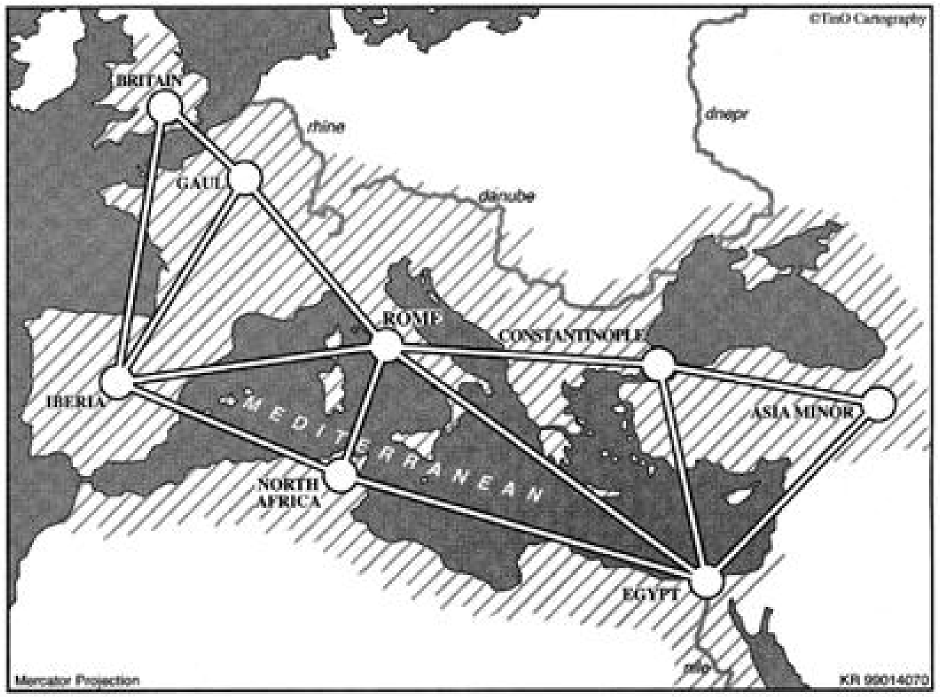
Satz: DS is W[1]-hart.

Warum Dominierende Mengen (in Trier) ?

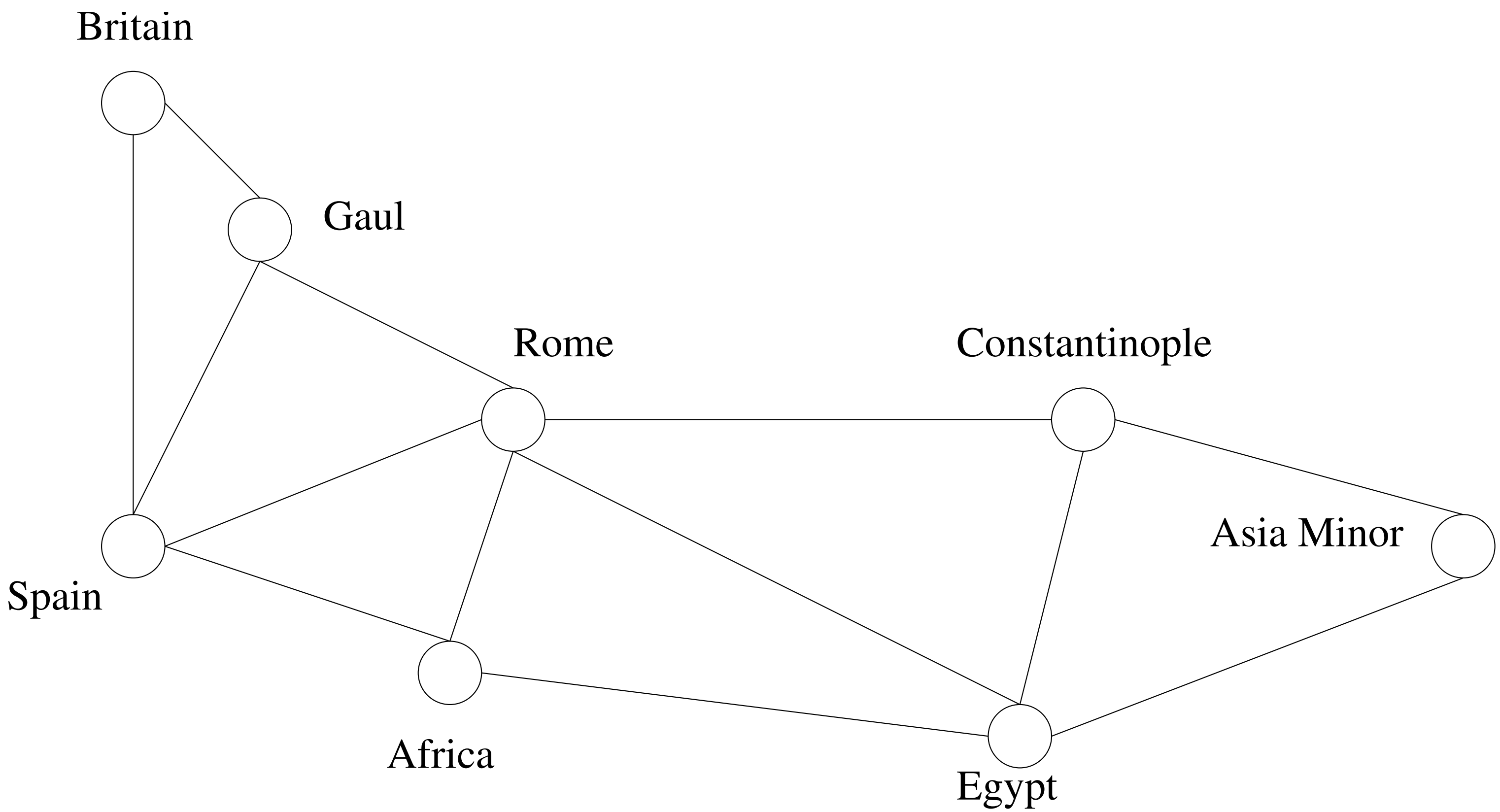
Kaiser Konstantins Thronsaal

Ein historische Anwendung

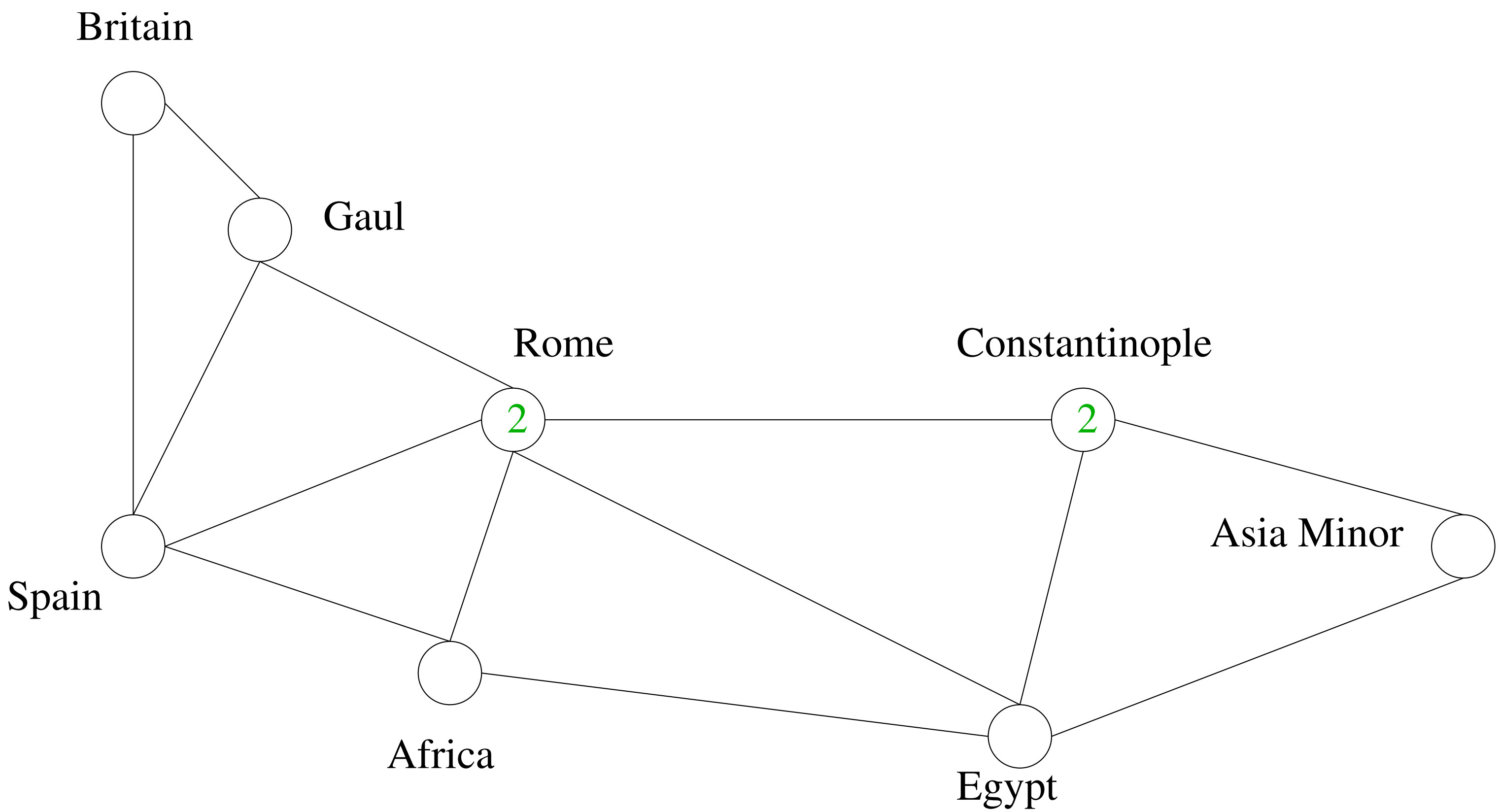
Das römische Reich zur Zeit Konstantins des Großen



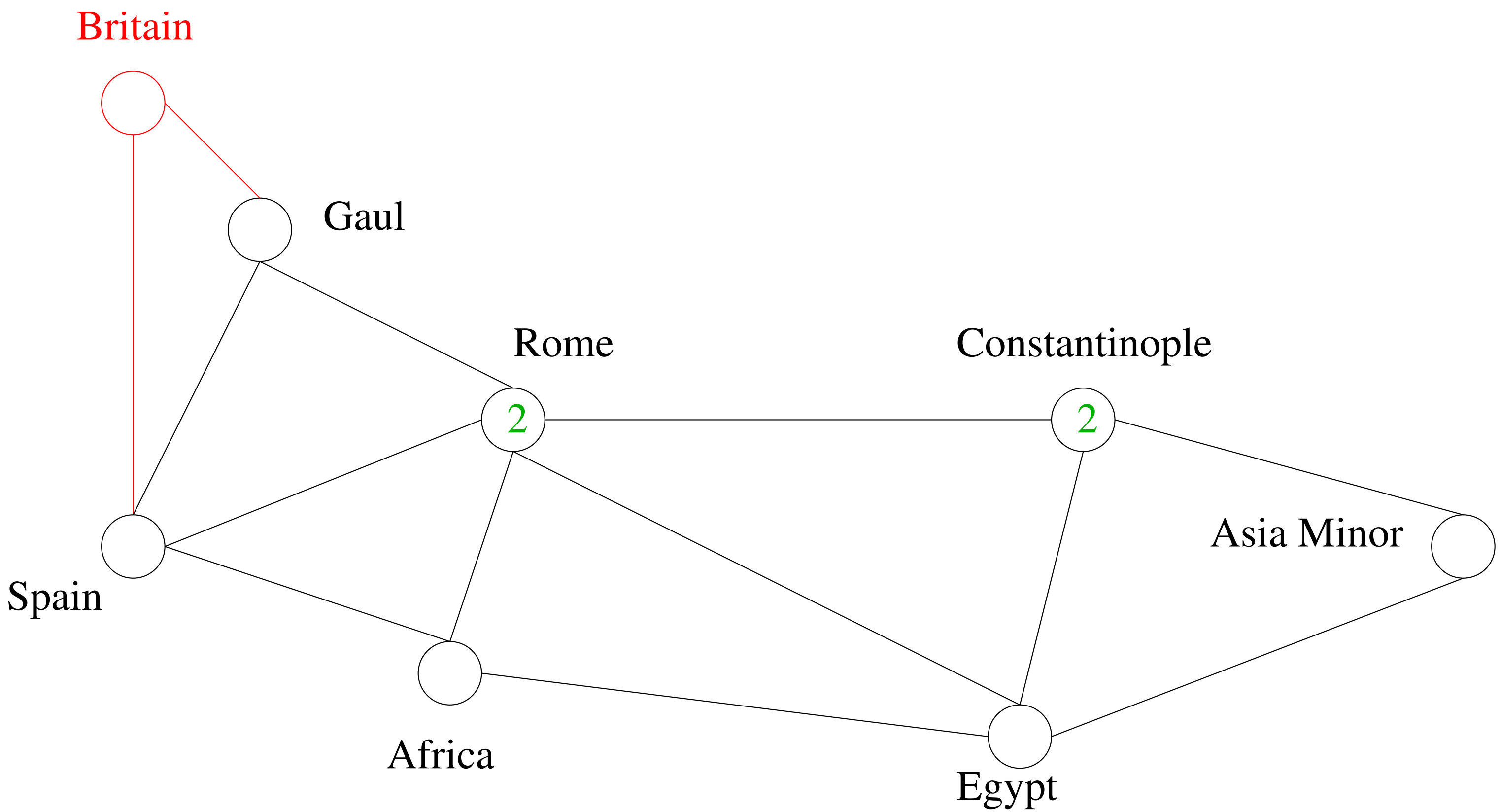
Ein reines Graphenmodell



Konstantins Lösung



Britannien in Gefahr



Zurück zu DS. . .

- $\mathcal{FPT}$? “Vermutlich” nein (W[1]-hart)

- Approximation ? “Vermutlich” nicht mit konstantem Faktor

- Exakte Algorithmik ?
  Bessser als $\mathcal{O}^*(2^{|V|})$?
  Fomin / Grandoni / Kratsch (ICALP 2005) fast $\mathcal{O}^*(1.5^{|V|})$!
  Grundidee: DS als Knotenüberdeckungsproblem in Hypergraphen

DS: Vielleicht doch parameterisiert ?

Idee: bei Graphen mit Maximalgrad drei ? (auch noch $\mathcal{NP}$-hart)

Jeder Knoten kann höchstens drei andere Knoten dominieren:
$\leadsto$ Hat Graph $> 4k$ Knoten, so NEIN-Instanz.

Diese einfache Reduktionsregel liefert Problemkern der Größe $4k$
$\leadsto$ Satz: k-DS auf Graphen mit Maximalgrad drei liegt in $\mathcal{FPT}$.
Genauer: k-DS kann dort in Zeit $\mathcal{O}^*(5.4265^k)$ gelöst werden.

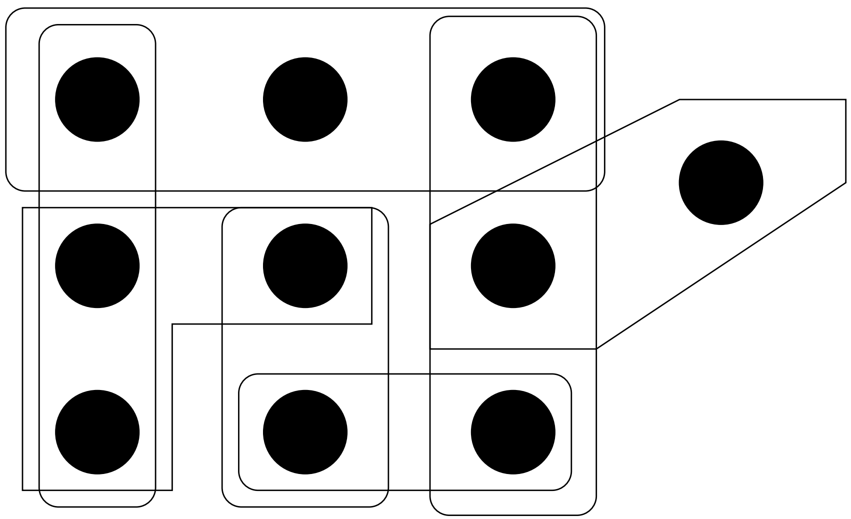
d-HITTING SET: Knotenüberdeckungsproblem auf Hypergraphen

Eingabe: Hypergraph $G = (V, E)$ mit *Kantengröße* höchstens d: $\forall e \in E(|e| \leq d)$
Parameter: k
Frage: Gibt es eine Knotenüberdeckung (hitting set) mit höchstens k Knoten:

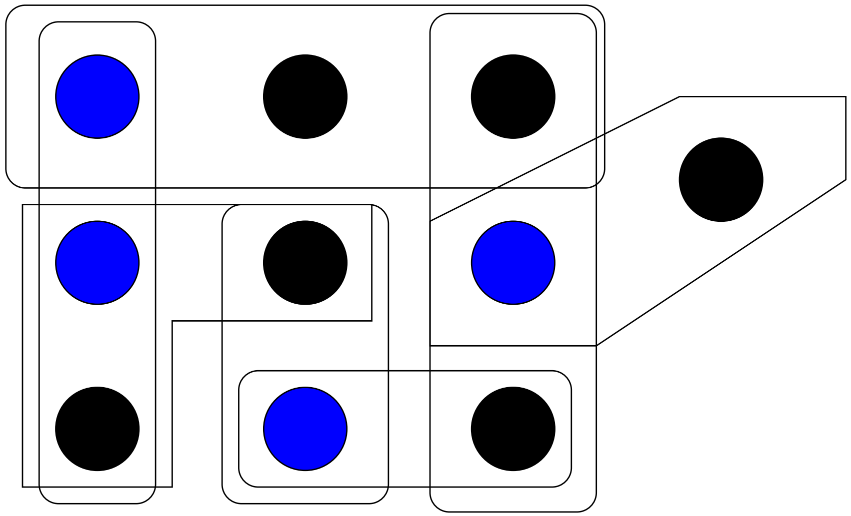
$$\exists C \subseteq V \, (|C| \leq k \wedge \forall e \in E(C \cap e \neq \emptyset)) \, ?$$

Variante: Knoten haben *Gewichte* ≥ 1, formal $w : V \to [1, \infty)$.

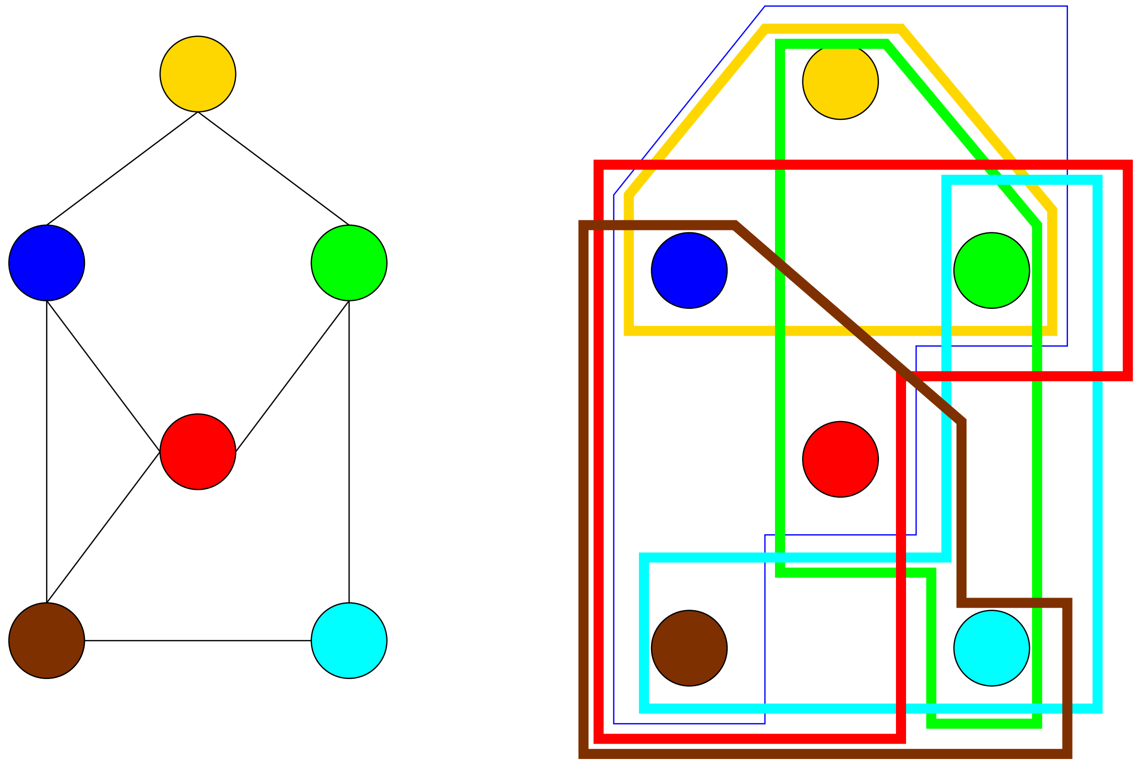
Ein abstraktes Beispiel



Eine kleinste Überdeckung



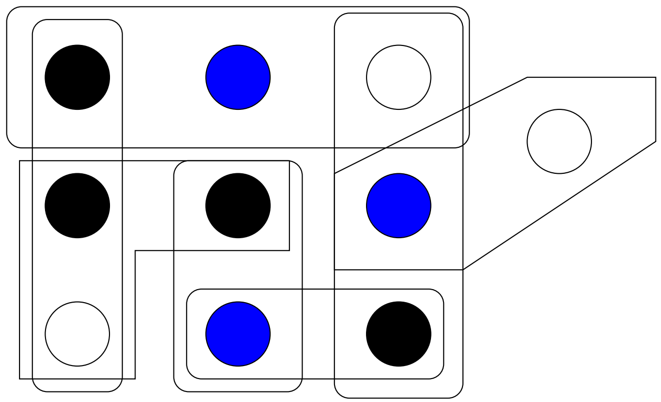
Zusammenhang mit DS:



Gewichtetes DS entspricht gewichtetem HS.

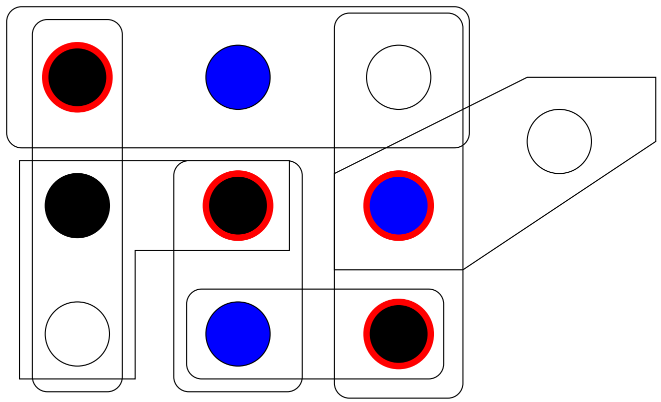
Ein einfaches gewichtetes Beispiel für $d = 3$ (d: maximale Kantengröße):

<u>Gewichtskonvention</u>: schwarz: eins, blau: zwei, weiß: drei



Geht es besser ?

Konvention: schwarz: eins, blau: zwei, weiß: drei



$\boxed{\text{Überblick}}$

- $\mathcal{FPT}$: die allgemeine Methodik
- Einführung anhand von Beispielen
- Reduktionsregeln und Suchbäume
- Analyse von Algorithmen mit einem Hilfsparameter
- Anwendungen & ähnliche Probleme

Ein einfacher Suchbaum:

```
simple-WHS(G = (V, E, w), k, S):
  IF k > 0 AND G has some edges THEN
    choose some edge e; // to be refined
    S' = ∅; // solution to be constructed
    FOREACH x ∈ e DO // recursively branch
      G' = (V \ {x}, {e ∈ E | x ∉ e});
      S' = simple-WHS(G', k − w(x), S ∪ {x})
      IF S' ≠ failure THEN break
    return S'
  ELSIF E = ∅ THEN return S ELSE return failure
```

Problem: Geht es besser als $\mathcal{O}^*(d^k)$?

Reduktionsregeln

Kantendominierung: $f \subset e$. $\leadsto$ entferne e.

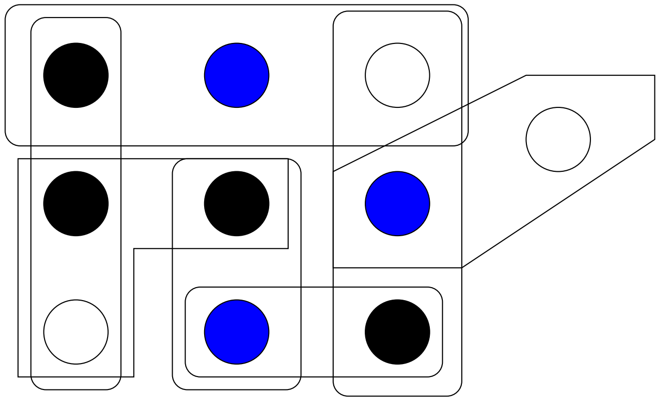
Kleine Kanten: $e = \{v\} \leadsto v$ kommt ins HS; entferne e.

Gewichtete Knotendominierung: Ein Knoten x heiße *dominiert* durch einen Knoten y, falls $w(y) \leq w(x)$ und $\{e \in E \mid x \in e\} \subseteq \{e \in E \mid y \in e\}$. $\leadsto$ entferne x.
Spezialfall: Gilt $x, y \in e$ mit $\delta(x) = 1$ und $w(y) \leq w(x)$,$\leadsto$ entferne x.

(*Kantenüberdeckung*: Enthält G eine Komponente C mit Maximalgrad zwei, dann löse C in Polynomzeit.)

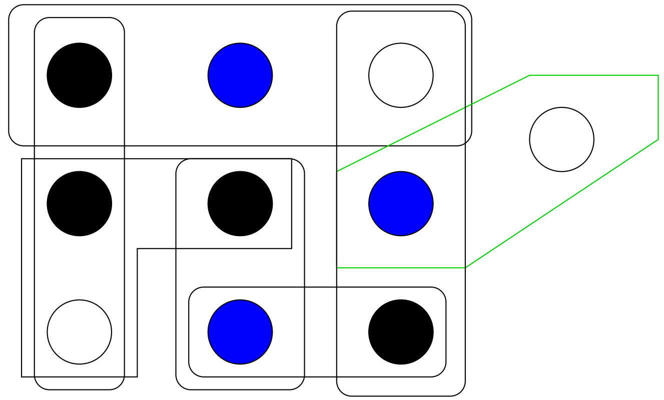
Ein einfaches Beispiel mit $\mathrm{d} = 3$: Wie arbeiten Reduktionsregeln konkret ?

Knoten-Konvention: schwarz: Gewicht eins, blau: Gewicht zwei, weiß: Gewicht drei



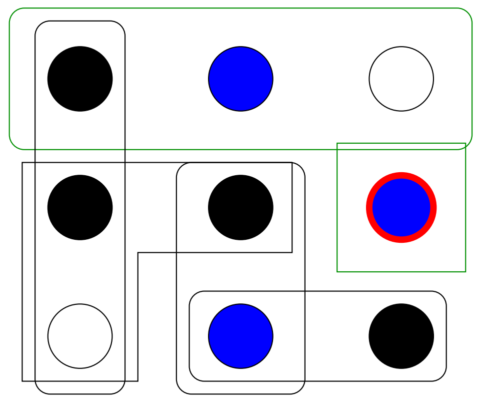
Gewichtete Knotendominierung

Knoten-Konvention: schwarz: Gewicht eins, blau: Gewicht zwei, weiß: Gewicht drei



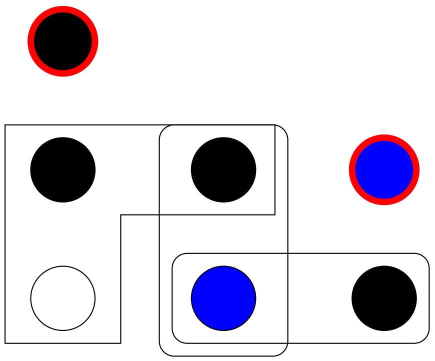
Kantendominierung & Kleine Kanten

Knoten-Konvention: schwarz: Gewicht eins, blau: Gewicht zwei, weiß: Gewicht drei



Welche Regeln braucht man hier ?

Knoten-Konvention: schwarz: Gewicht eins, blau: Gewicht zwei, weiß: Gewicht drei



Ohne die Knotenüberdeckungsregel ist die Instanz jetzt irreduzibel.

Ein besserer Suchbaum

Erste Idee: Verzweige an kleinen Kanten.

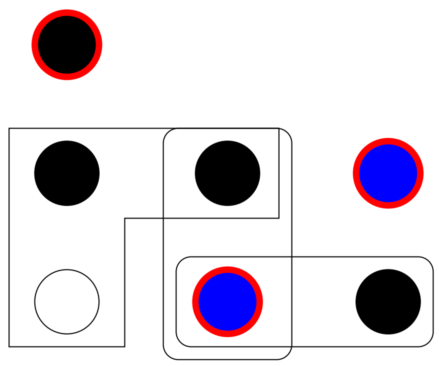
$\rightsquigarrow$ unmittelbare Verbesserungen der trivialen d-Verzweigung.

Zweite Idee: Verzweige an Knoten x von hohem Grad

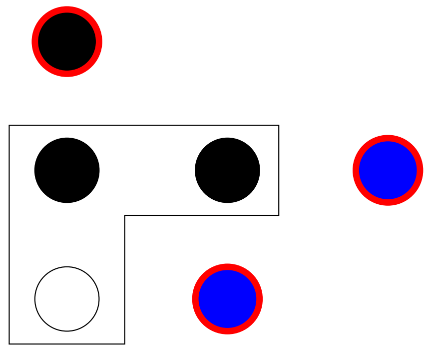
$\rightsquigarrow$ mittelbare Verbesserung im Falle, dass x nicht in die Überdeckung kommt.

$\rightsquigarrow$ *heuristische Prioritäten*

Wie arbeiten **heuristische Prioritäten** ? Der Fall $x \in C$:

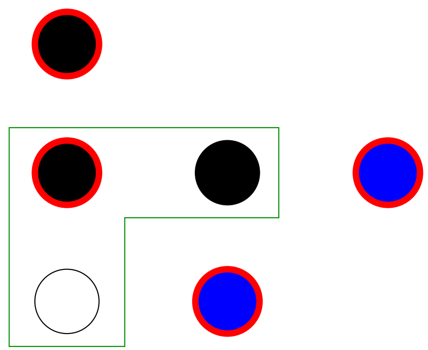


Wie arbeiten **heuristische Prioritäten** ? Der Fall $x \in C$:

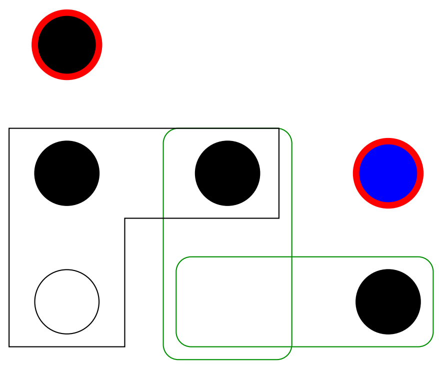


Wie arbeiten **heuristische Prioritäten** ? Der Fall $x \in C$:

Jetzt feuern wieder Reduktionsregeln ! (Welche ?)

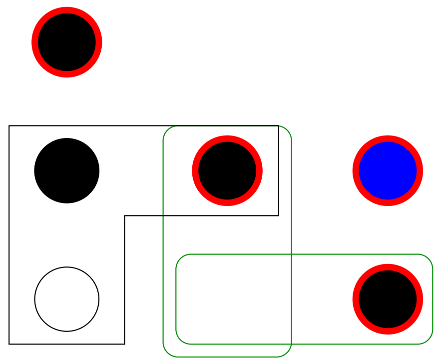


Wie arbeiten **heuristische Prioritäten** ? Der Fall $x \notin C$:



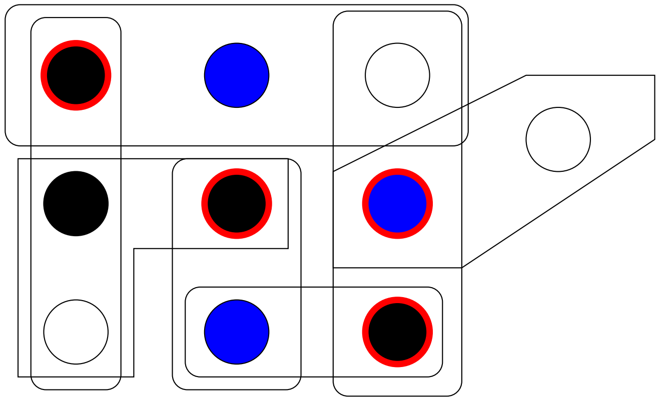
Wie arbeiten **heuristische Prioritäten** ? Der Fall $x \notin C$:

Jetzt feuern wieder Reduktionsregeln ! (Welche ?)



Die beste Lösung findet sich im zweiten Zweig.

Knoten-Konvention: schwarz: Gewicht eins, blau: Gewicht zwei, weiß: Gewicht drei



Der verbesserte Algorithmus für d-HS:

```
WHS-ST(G = (V, E, w), k, S):
  exhaustively apply reduction rules;
  IF k > 0 THEN
    IF E = ∅ THEN return S;
    choose some vertex x according to the heuristic priorities;
    S' = ∅; // solution to be constructed
    E' = {e ∈ E | x ∉ e};
    S' = WHS-ST((V \ {x}, E'), k − w(x), S ∪ {x});
    IF S' ==failure THEN
       E'' = {e \ {x} | e ∈ E};
       S' = WHS-ST((V \ {x}, E''), k, S);
    return S'
  ELSIF G contains some edges or k < 0 return failure
  ELSE return S
```

Zusammenfassung unserer Ergebnisse

für gewichtetes k-d-HS (CIAC 2006):

d	3	4	5	6	7	8	9	10	100
$c_d \leq$	2.2470	3.1479	4.1017	5.0640	6.0439	7.0320	8.0243	9.0191	99.0002

Graphentheoretische Anwendung

4HS hilft zur Lösung von DS auf Graphen mit Maximalgrad drei.

$\leadsto$ Satz. k-DS für Maximalgrad drei kann in Zeit $\mathcal{O}^*(3.1479^k)$ gelöst werden.

Alternativlösung mit folgendem Satz (Fomin / Høie (2005)):
Zu jedem $\epsilon > 0$ gibt es ein n_ϵ, sodass für $n > n_\epsilon$ gilt: Ein n-Knoten-Graph mit Maximalgrad drei hat Pfadweite von höchstens $(1/6 + \epsilon) \cdot n$.

Folgerung: Auf n-Knoten-Graphen vom Maximalgrad drei kann MDS in Zeit $O^*(3^{n/6})$ gelöst werden.

Folgerung: Auf Graphen vom Maximalgrad drei kann k-DS in Zeit $\mathcal{O}^*(3^{2k/3}) = O^*(c^k)$ mit $c \leq 2.0801$ gelöst werden.

Überblick

- $\mathcal{FPT}$: die allgemeine Methodik
- Einführung anhand von Beispielen
- Reduktionsregeln und Suchbäume
- Analyse von Algorithmen mit einem Hilfsparameter
- Anwendungen & ähnliche Probleme

Einführung eines Hilfsparameters

$T^{\ell}(k)$ (for $\ell \geq 0$): # Blätter im Suchbaum unter der Annahme, (wenigstens) ℓ "kleine Kanten", d.h., Größe von (höchstens) $d - 1$

ℓ ist der *Hilfsparameter* in unseren Rekursionen.

Monotonie: $T^{\ell+1}(k) \leq T^{\ell}(k)$

Eine einfache Analyse

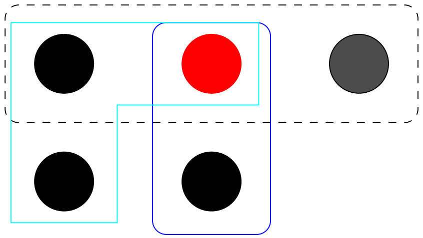
$$T^0(k) \leq T^0(k-1) + T^3(k).$$

Vor dem Verzweigen ist die Instanz irreduzibel
$\leadsto$ Es gibt Knoten x vom Mindestgrad drei (Kantenüberdeckungsregel) in Kante der Größe d.

Zweig 1: Kommt x in Lösung, so verringert sich das Budget wenigstens um eins.

Zweig 2: Andernfalls entstehen wenigstens drei “neue kleine Kanten.”

Etwas trickreicher... (nur für $d = 3$)



$$T_3^1(k) \le T_3^0(k-1) + T_3^1(k-1).$$

Betrachte Kante e mit $|e| = 2$.
Es sei x ein Knoten maximalen Grades in $e \rightsquigarrow$ verzweige bei x.
Die Instanz ist irreduzibel $\rightsquigarrow \delta(x) \ge 2$ (gewichtete Knotendominierung).
$\rightsquigarrow \exists f : x \in e \cap f$; o.B.d.A.: $|f| = 3$. (Sonst feuern Reduktionsregeln.)
1. Nimm x in Hitting Set: $\rightsquigarrow$ ein $T^0(k-1)$-Teilbaum.
2. Andernfalls:
$\rightsquigarrow$ Es gibt neue Kante $f' = f \setminus \{x\}$ der Größe zwei sowie eine Kante $e' = e \setminus \{x\}$ der Größe eins.
$\rightsquigarrow$ e' wird durch Reduktionsregel "erledigt" und der Parameter um (wenigstens) eins reduziert.

Ein Ungleichungssystem

$$
\begin{aligned}
T_3^0(k) &\leq T_3^0(k-1) + T_3^3(k) \\
T_3^1(k) &\leq T_3^0(k-1) + T_3^1(k-1) \\
T_3^2(k) &\leq \max\{T_3^1(k-1) + T_3^2(k-1), T_3^0(k-1) + T_3^0(k-2)\} \\
T_3^3(k) &\leq \max\left\{\begin{array}{l} T_3^1(k-1) + T_3^0(k-2), \\ T_3^0(k-1) + T_3^0(k-3), \\ T_3^2(k-1) + T_3^3(k-1) \end{array}\right\}
\end{aligned}
$$

Problem: Wie löst man so etwas ?

Auflösen eines Ungleichungssystems

Betrachte die größte positive Wurzel c_3 des Polynoms $x^3 - 2x^2 - x + 1$,
($c_3 \leq 2.2470$) $\rightsquigarrow$
$T_3^0(k) = c_3^k$,
$T_3^1 = c_3^k/(c_3 - 1)$,
$T_3^2(k) = c_3^k/(c_3 - 1)^2$,
$T_3^3(k) = c_3^{k-1}(c_3 - 1)$.

Satz. GEWICHTETES 3-HITTING SET kann in Zeit $\mathcal{O}^*(c_3^k)$ gelöst werden.

Wie löst man solche Ungleichungssysteme ?

1. Diskutiere die “Extremfälle” (alle Kombinationen, wie das Maximum angenommen werden kann), formuliere das entsprechende Gleichungssystem E und berechne Basis c_E, sodass: $T^0(k) = c_E^k$.

2. Wähle das Maximum c unter allen c_E.

3. Drücke $T^j(k)$ als $T^j(k) = \alpha_j c^k$ aus.

4. Überprüfe, dass das ursprüngliche Ungleichungssystem durch die gewählten Funktionen erfüllt wird.

Ein möglicher Extremfall in unserem Fall:

$$
\begin{aligned}
T_3^0(k) &= T_3^0(k-1) + T_3^3(k) \\
T_3^1(k) &= T_3^0(k-1) + T_3^1(k-1) \\
T_3^2(k) &= \max\{T_3^1(k-1) + T_3^2(k-1), T_3^0(k-1) + T_3^0(k-2)\} \\
T_3^3(k) &= \max \left\{ \begin{array}{l} T_3^1(k-1) + T_3^0(k-2), \\ T_3^0(k-1) + T_3^0(k-3), \\ T_3^2(k-1) + T_3^3(k-1) \end{array} \right\}
\end{aligned}
$$

Die rote Wahl E (T_3^2 ist egal) liefert als charakteristische Polynom:

$$0 = c_E(c_E - 1)^2 - c_E - (c_E - 1) = c_E^3 - 2c_E^2 - c_E + 1.$$

Dies ist tatsächlich der schlimmste Fall, d.h., $c_3 = c_E$.

Bestimmung der Wert für α_ℓ im schlimmsten Fall:

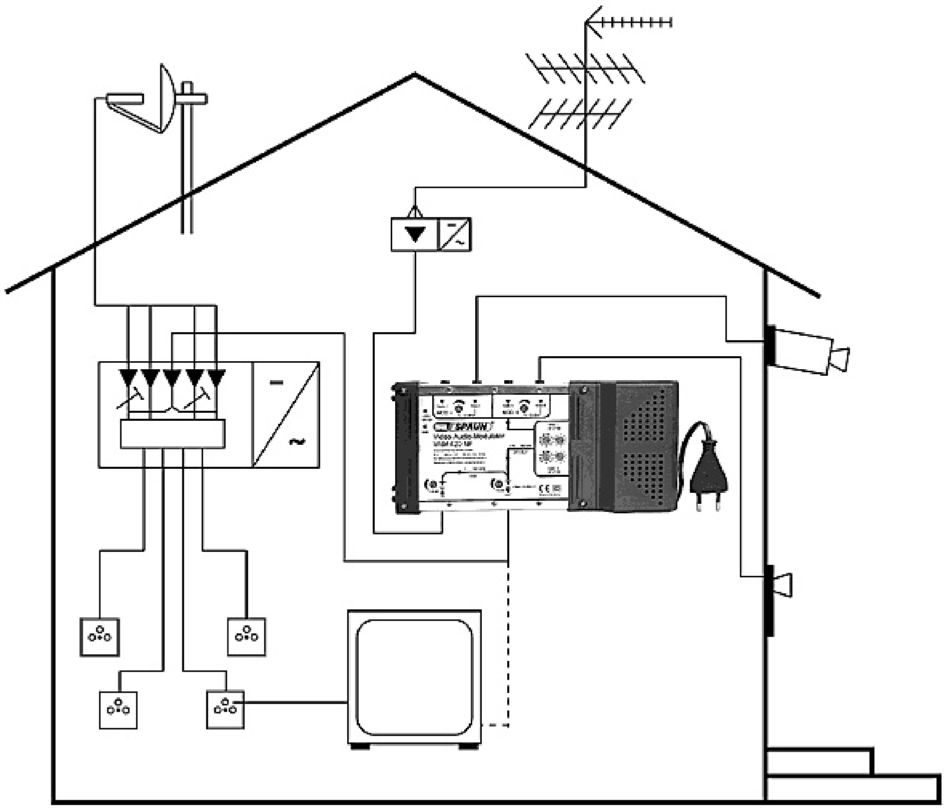
$T_3^0(k) = c_3^k$, d.h., $\alpha_1 = 1$;
$T_3^1 = c_3^k/(c_3 - 1)$, also: $\alpha_1 = 1/(c_3 - 1) \approx 0.80$;
$T_3^2(k) = c_3^k/(c_3 - 1)^2$, d.h., $\alpha_2 = 1/(c_3 - 1)^2 = (c_3 + 1)/c_3^2 \approx 0.64$
(wegen des charakteristischen Polynoms);
$T_3^3(k) = c_3^{k-1}(c_3 - 1)$, i.e., $\alpha_3 = (c_3 - 1)/c_3 \approx 0.55$.

Bem.: $\alpha_0 = 1 \geq \alpha_1 \geq \alpha_2 \geq \alpha_3 > 0$.

Überblick

- $\mathcal{FPT}$: die allgemeine Methodik
- Einführung anhand von Beispielen
- Reduktionsregeln und Suchbäume
- Analyse von Algorithmen mit einem Hilfsparameter
- Anwendungen & ähnliche Probleme

Eine Motivation für Hitting Set: Systemanalyse á la Reiter



Was ist ein System? (nach R. Reiter)

- Systembestandteile (Komponenten) $\mathcal{C}$
- Systembeschreibung (wie? $\leadsto$ Logik) $\mathcal{SD}$:
  Aussagen über erwartetes Systemverhalten,
  d.h., Beziehungen zwischen den Komponenten.
- beobachtetes Systemverhalten (Observationen) $\mathcal{OBS}$

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 Komponenten sind die Knoten,
die Konfliktmengen sind die Hyperkanten,
die Diagnose die Überdeckungsmenge.

Graphentheoretische Anwendungen

Viele Probleme lassen sich als Graphen-Editier-Probleme darstellen:

+ Lösche höchstens k Knoten, um alle Dreiecke in einem Graphen zu zerstören.

○ Lösche höchstens k Kanten, damit ein Graph biplanar wird (s.u.) [Sugiyama].

? Lösche höchstens k Kanten, damit ein Graph bipartit wird.

Zerstöre alle Dreiecke

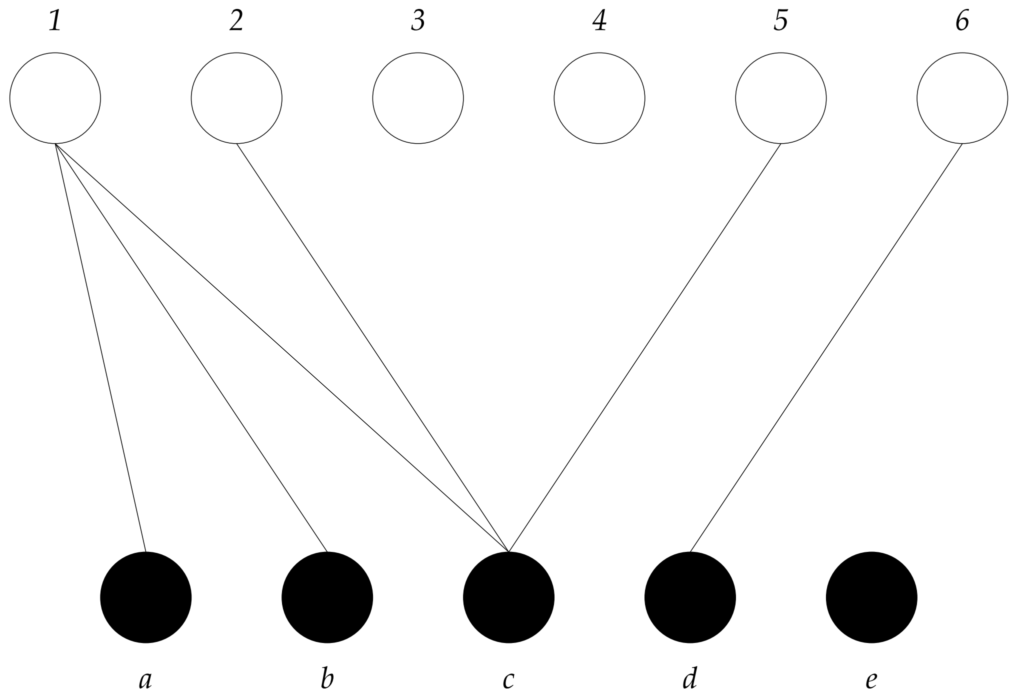
Übersetzung in äquivalente 3-HS Instanz:

Gegeben: Ein Graph $G = (V, E)$.

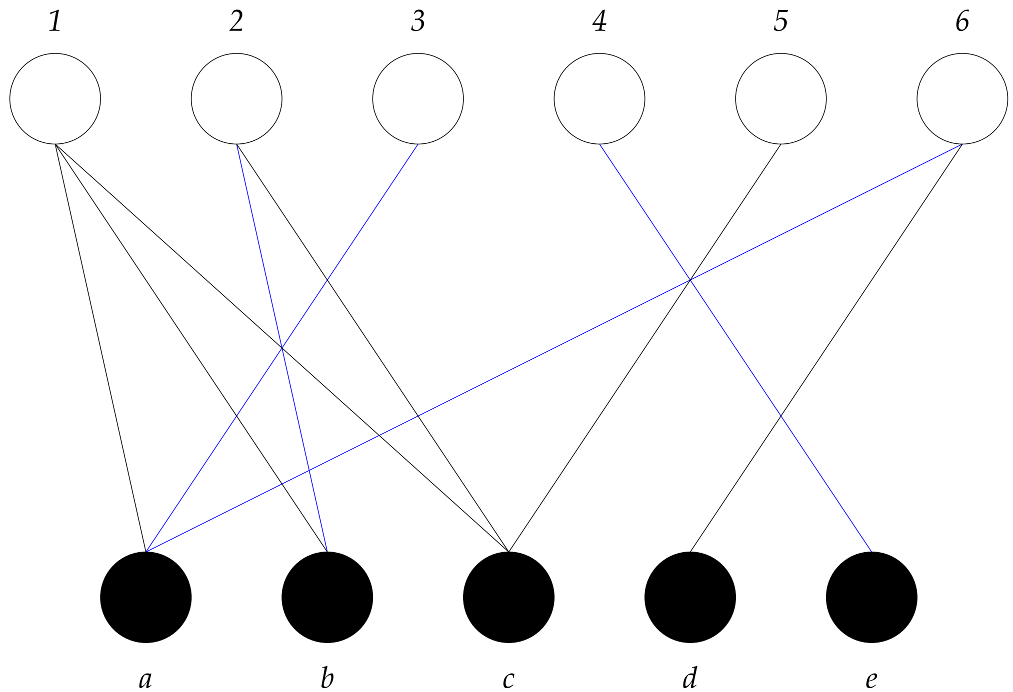
Gesucht: Ein Hypergraph $H = (V_H, E_H)$.

1. $V_H := V$.
2. Setze $E_H := \emptyset$.
3. Für jedes Tripel $(v_1, v_2, v_3) \in V^3$ tue:
4. Falls $\{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}\} \subseteq E$, so füge $\{v_1, v_2, v_3\}$ zu E_H hinzu.

Biplanarität

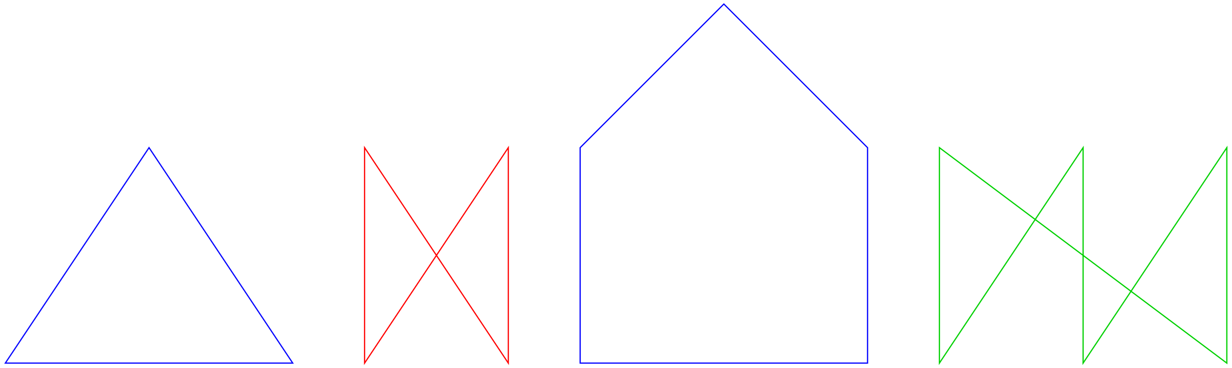


Vier Kanten mehr. . .

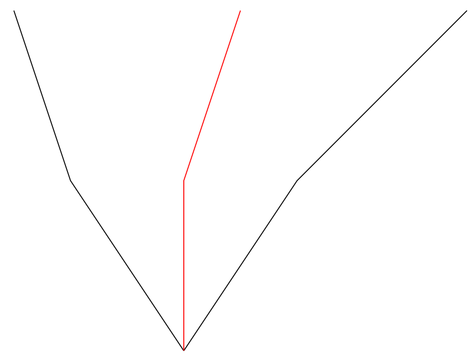


Frage: Muss man wirklich 4 Kanten entfernen zur Biplanarisierung ?

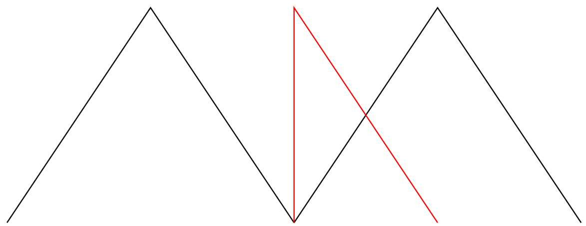
Verbotene Strukturen I: Kreise



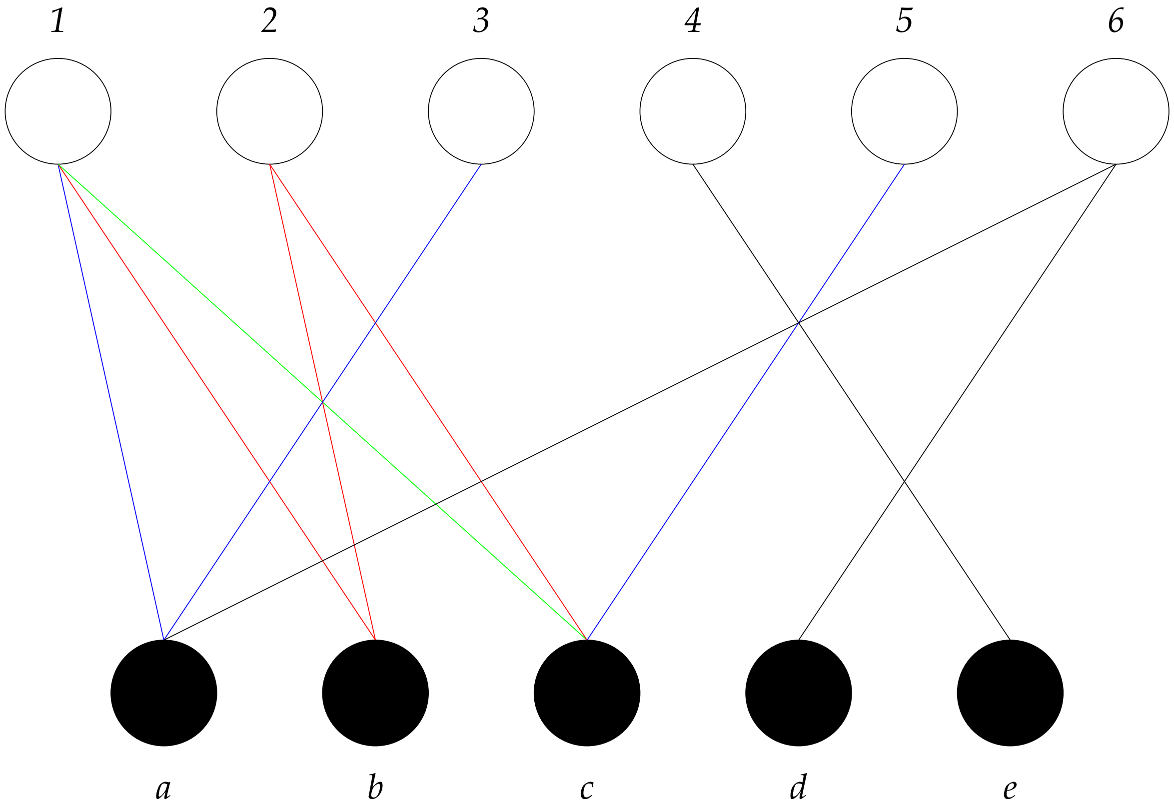
Verbotene Strukturen II: 2-claws



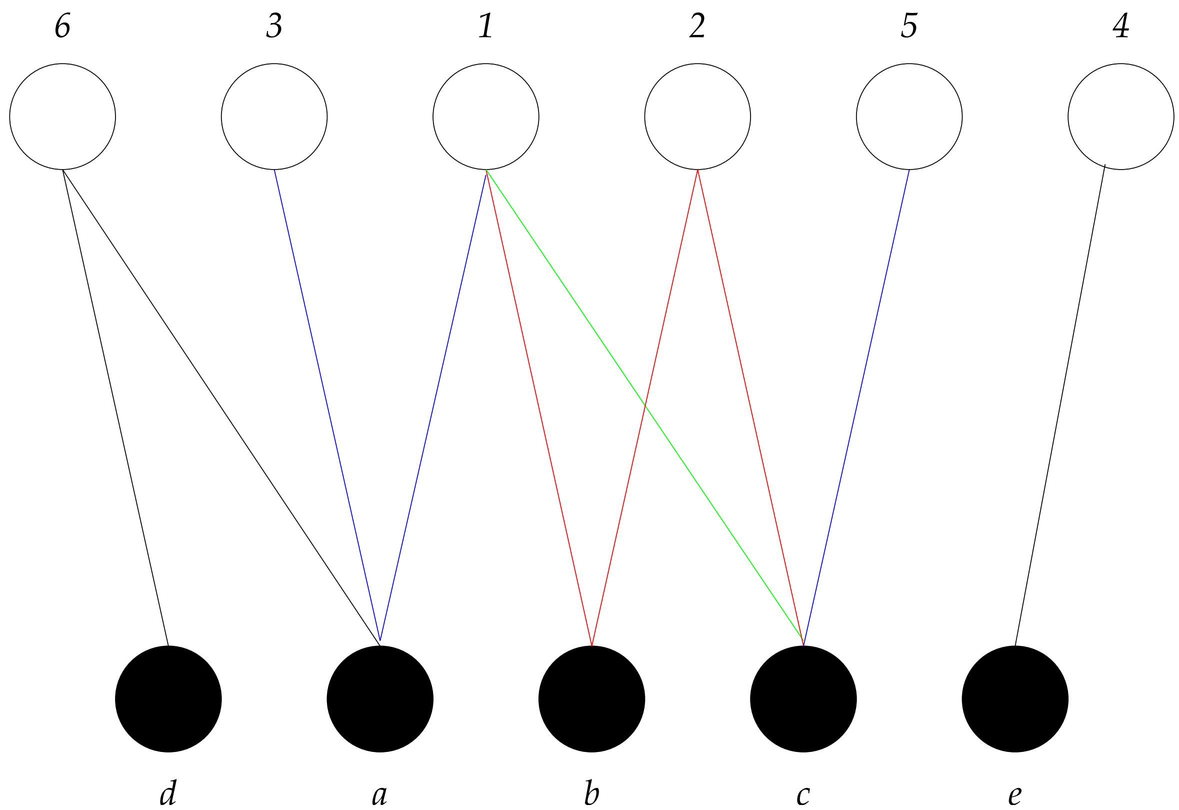
Das Problem bei 2-claws



Verbotenes in unserem Beispiel



Eine mögliche Lösung



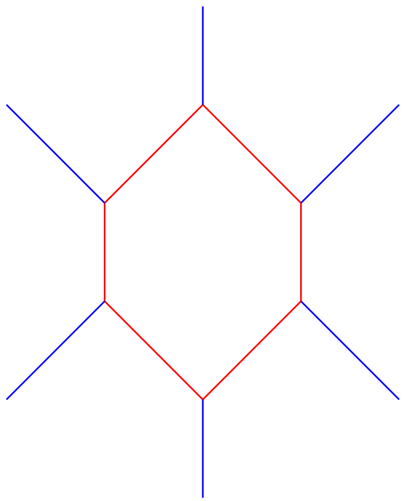
Ziel: wenige kleine verbotene Strukturen

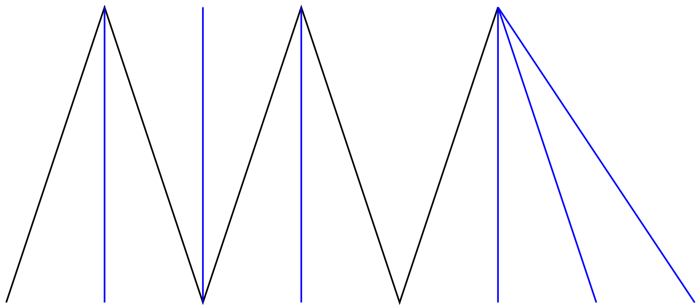
Der *Nicht-Blatt-Grad* eines Knoten v ist die Anzahl zu v inzidenter Nicht-Blatt-Kanten, kurz: $\deg'_G(v)$.

Lemma: Gibt es einen Knoten v mit Nicht-Blatt-Grad wenigstens drei, so ist v Teil einer 2-claw oder eines Kreises der Längen drei oder vier.

Lemma: Gibt es keinen Knoten mit Nicht-Blatt-Grad wenigstens drei, so besteht der Graph aus lauter Kränzen und Raupen.

Kränze und Raupen





Ein einfacher Suchbaumalgorithmus

```
TLP(G = (V, E), k)
IF(∃ v ∈ V : deg'_G(v) ≥ 3 and k ≥ 0)
  Determine 2-claw, 3-cycle or 4-cycle C containing v.
  Branch at all edges e ∈ C.
ELSE
  return k ≥ # wreath  components
```

Verbesserungen ähnlich dem zu (6-)Hitting Set Gesagten sind möglich.

siehe: JGAA 2005; noch besser: M. Suderman (PhD Thesis)

Bipartisierung

Problem: unendliche Reihe verbotener Graphen (ungerade Kreise); kein “Trick” wie bei TLP bekannt.

Lange Zeit offen, erst 2004 von B. Reed, K. Smith und A. Vetta in $\mathcal{FPT}$ gezeigt mit $\mathcal{O}^*(3^k)$ (dynamisches Programmieren auf Mengen).

Verbesserung mit hoher Wahrscheinlichkeit (CTW 2006), beruhend auf:
Satz (Erdös, Kleitman und Rothschild (1976)). Fast alle dreiecksfreien Graphen sind bipartit.
Satz (Prömel, Schickinger und Steger (2002)). Fast alle nicht-bipartiten dreiecksfreien Graphen können durch Entfernen eines Knotens bipartit gemacht werden.
$\leadsto$ “Fast immer” kann ein Graph bipartit gemacht werden, indem man alle Dreiecke zerstört und dann noch höchstens einen weiteren Knoten entfernt.

Weiterer Vorteil: Teure Nein-Instanzen werden früh gefunden.

Interessen des Arbeitsbereichs Theoretische Informatik

parameterisierte Algorithmen(analyse) & Komplexität

Lernalgorithmen

Formale Sprachen