Technische Universität Braunschweig

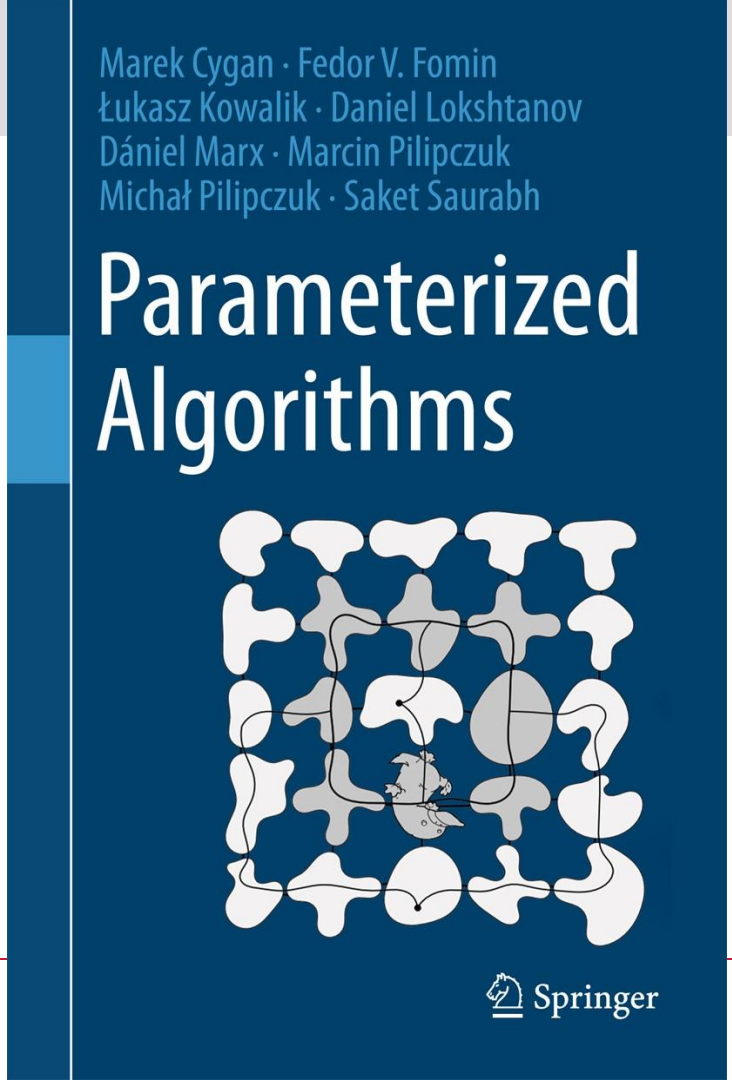
Parametrisierte Algorithmen

Arne Schmidt

Literatur

Buchempfehlung

Download-Link über Mailingliste



Kapitel 1 – Einführung in FPT

Kapitel 1.1 – Bar Fight Problem

Bar Fight Problem

Ein Eichhörnchen, ein Schnabeltier, eine Maus und ein Vogel wollen in eine Bar; den Algo Pub!

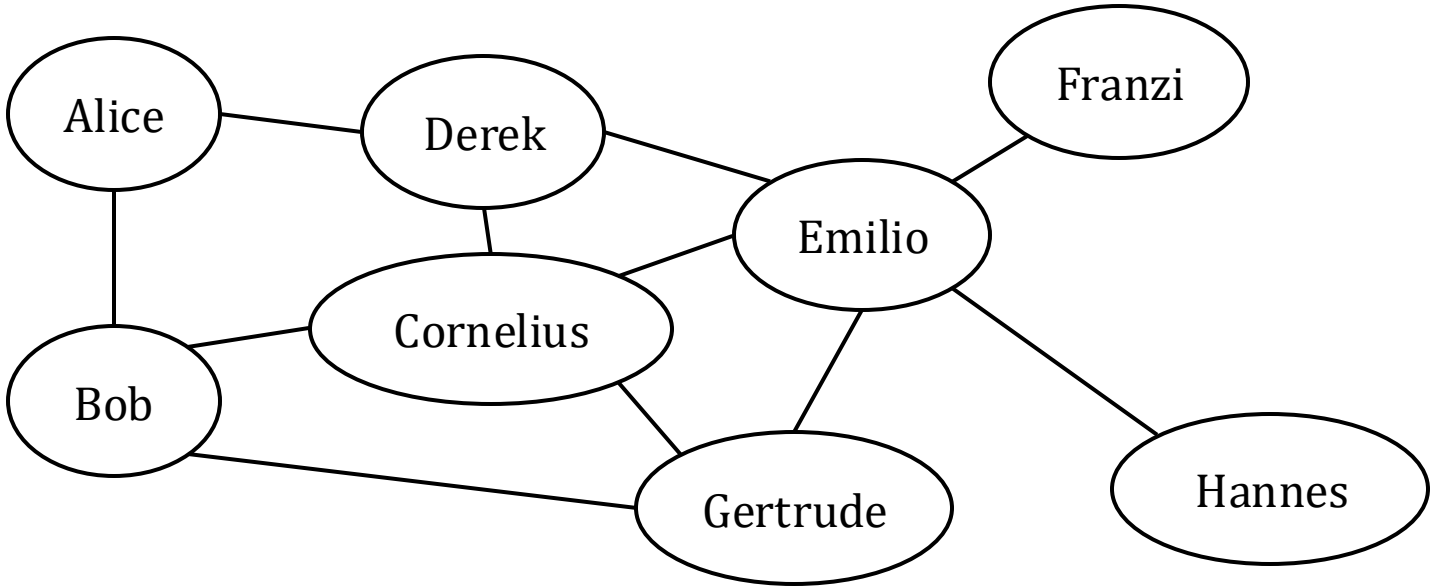
Problem:
Nicht jeder versteht sich mit jedem. Es gibt Konflikte!

Lösung:
Lass nur Tiere herein, die sich untereinander verstehen!

Frage:
Kann der Türsteher maximal k Tiere abweisen, sodass keine Konflikte in der Bar entstehen?

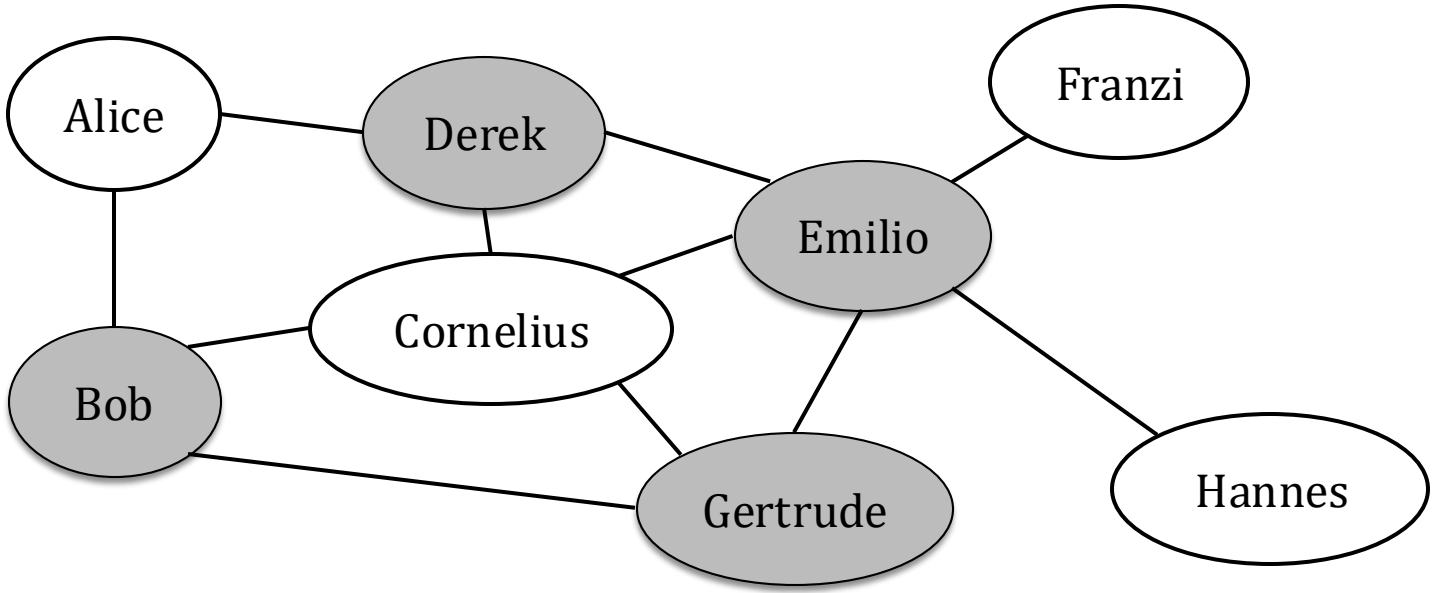
Ein Graph

Genügt es, 4 Knoten zu entfernen, um alle Konflikte aufzulösen?



Ein Graph

Genügt es, 4 Knoten zu entfernen, um alle Konflikte aufzulösen?

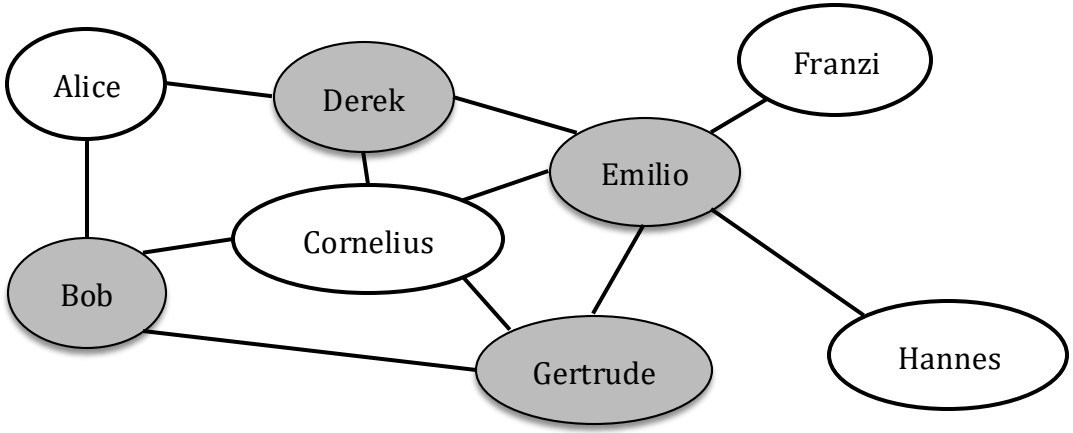


Welche Ideen gibt es, um das Problem algorithmisch (für beliebige Graphen) zu lösen?

Das Bar-Fight-Problem

Problem 1.1 (Bar Fight Problem, aka Vertex Cover)
Gegeben: Ein Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$.
Frage: Existiert $C \subseteq V$ mit

- $|C| = k$
- $\forall \{u, v\} \in E: u \in C \vee v \in C$



Kapitel 1.2 – Brute Force bis Bounded Search Tree

Ansatz 1: Brute Force

Lemma 1.2:
Brute Force löst das Bar-Fight-Problem in Zeit $O\left(n^{k+1} \cdot k\right)$.

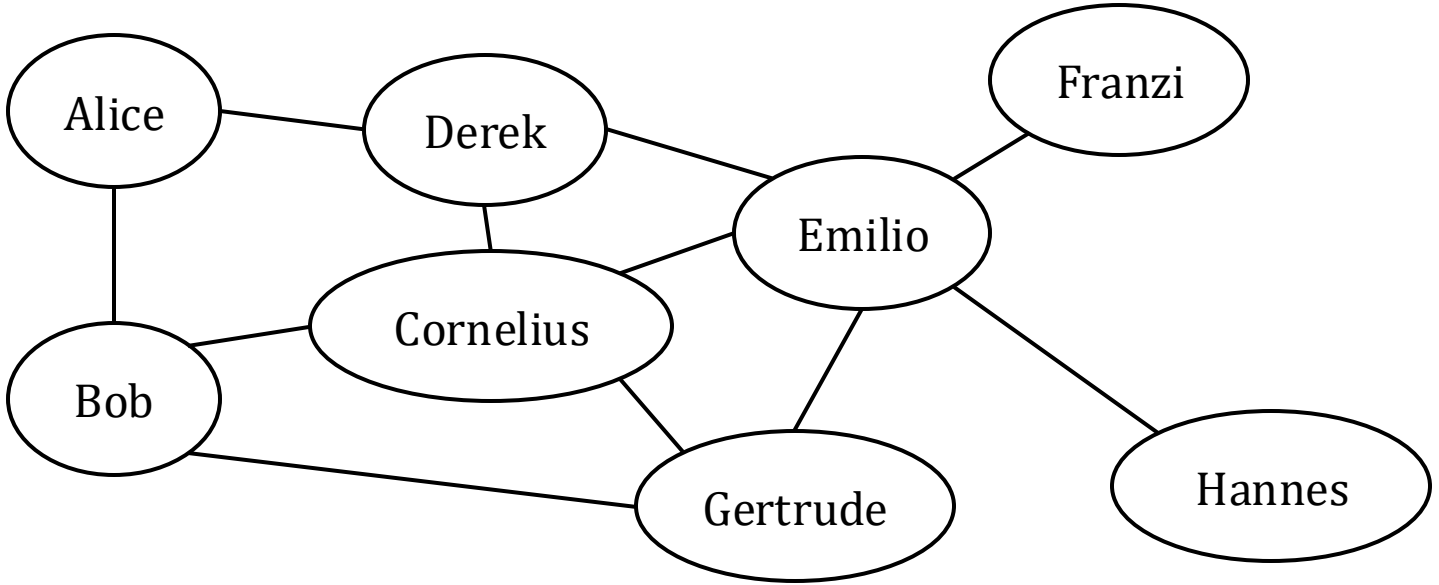
Idee:
Betrachte jede Auswahl von k Knoten und prüfe, ob es ein Vertex Cover ist.

Dazu:
n^k Möglichkeiten von Knotenmengen.
Jede Überprüfung dauert $O(m+n)$ Zeit.

Claim:
Wir können davon ausgehen, dass $m \leq kn$ gilt.

Ansatz 2: Bounded Tree Search

Lemma 1.3:
Das Bar-Fight-Problem lässt sich in Zeit $O(2^k \cdot k \cdot n)$ lösen.



Ansatz 2: Bounded Tree Search

Lemma 1.3:
Das Bar-Fight-Problem lässt sich in Zeit $O\left(2^k \cdot k \cdot n\right)$ lösen.

Idee:

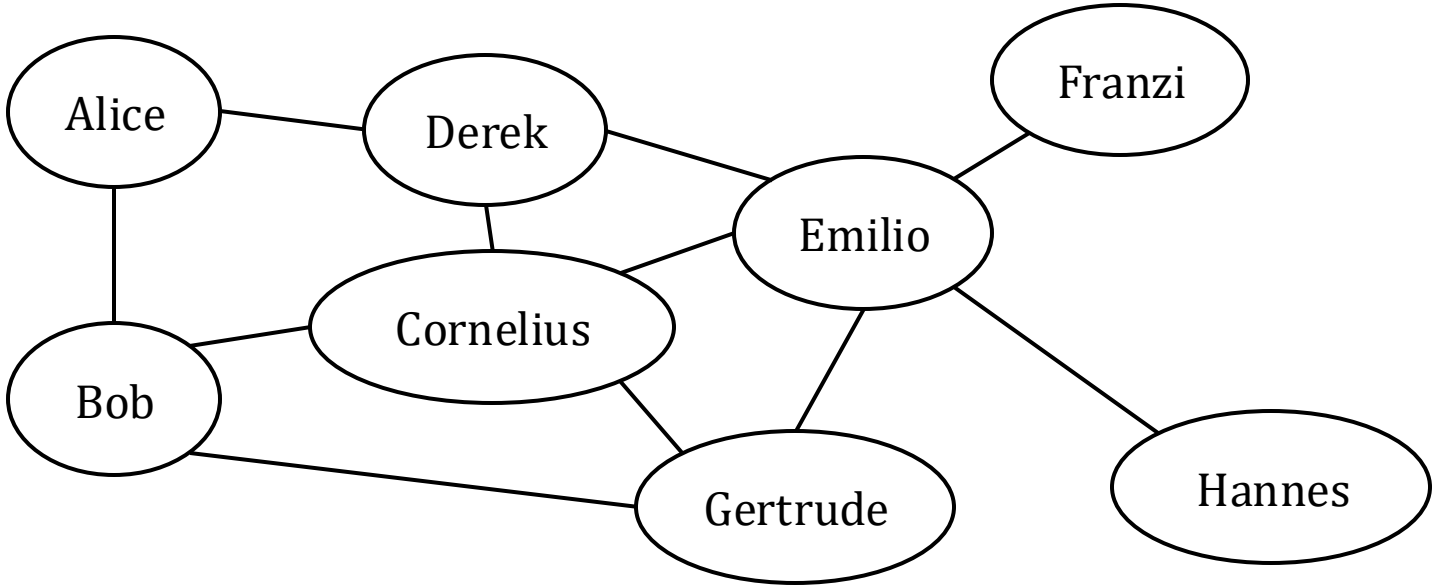
- Betrachte eine beliebige Kante $\{u, v\} \in E$. Einer der beiden Endknoten **muss** im Vertex Cover liegen.
- Angenommen, v ist im Vertex Cover, dann können wir v samt Kanten entfernen und testen, ob der restliche Graph eine Lösung mit $k' = k - 1$ besitzt.
- Angenommen, u ist im Vertex Cover, dann können wir u samt Kanten entfernen und testen, ob der restliche Graph eine Lösung mit $k' = k - 1$ besitzt.

Rekursionstiefe ist maximal k, Verzweigungsgrad der Rekursion ist 2.
→ $O\left(2^k\right)$ Knoten im Suchbaum.
Den jeweils neuen Graphen zu berechnen dauert maximal $O(n + m) = O(kn)$ Zeit.

Ansatz 3: Finden eines Kerns

Idee: Betrachte Knoten von hohem Grad.
Hier: Emilio **muss** im Vertex Cover sein.
Claim: Ist $\delta(v) > k$ für einen Knoten $v \in V$, dann muss v im Vertex Cover sein.

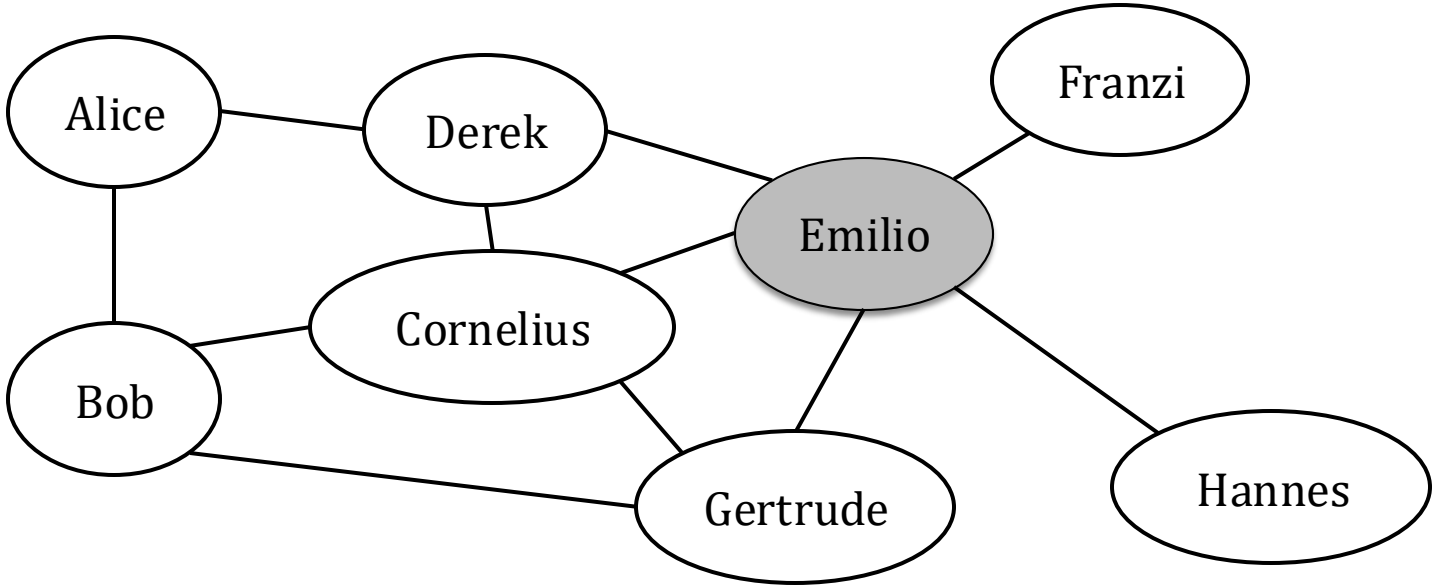
k = 4



Ansatz 3: Finden eines Kerns

Idee: Betrachte Knoten von hohem Grad.
Hier: Emilio **muss** im Vertex Cover sein.
Claim: Ist $\delta(v) > k$ für einen Knoten $v \in V$, dann muss v im Vertex Cover sein.

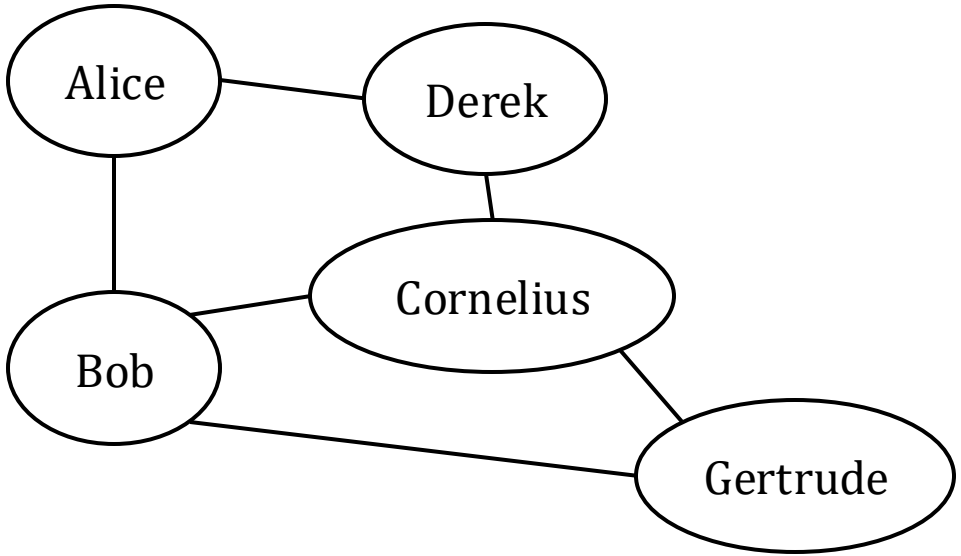
k = 4



Ansatz 3: Finden eines Kerns

Claim: Ist $\delta(v) > k$ für einen Knoten $v \in V$, dann muss v im Vertex Cover sein.
Entferne diesen Knoten samt Kanten und isolierten Knoten.
Prüfe, ob der Restgraph eine Lösung mit $k' = k - 1$ besitzt.

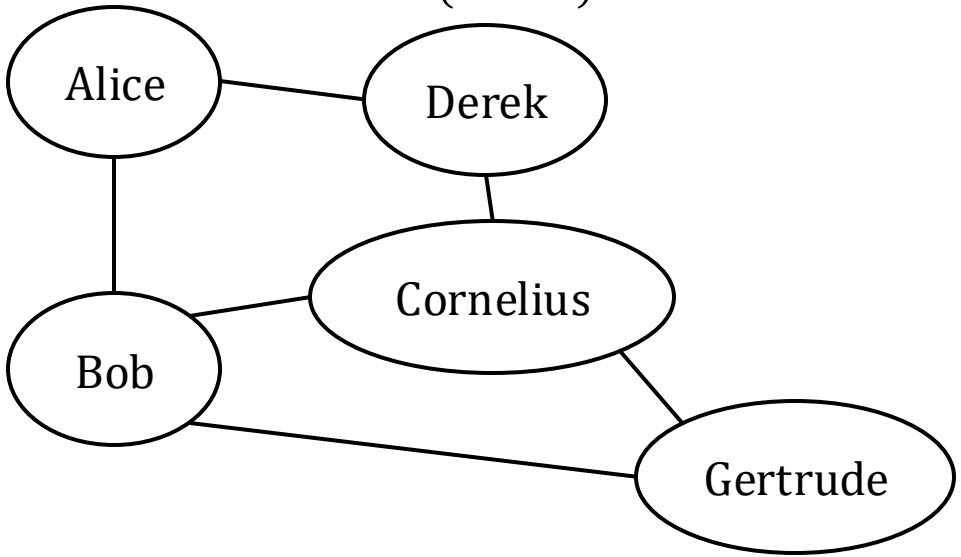
k = 3



Ansatz 3: Finden eines Kerns

Claim: Ist $\delta(v) > k$ für einen Knoten $v \in V$, dann muss v im Vertex Cover sein.
Durch wiederholtes Ausführen entsteht ein Graph, in welchem jeder Knoten Grad höchstens k besitzt. → Maximal $O(k^2)$ Knoten und Kanten.
→ Bounded Tree Search liefert in $O(2^k \cdot k^2)$ eine Antwort.

k = 3



Ansatz 3: Finden eines Kerns

Lemma 1.4:
Das Bar-Fight-Problem lässt sich in Zeit $O\left(2^k \cdot k^2 + n + m\right)$ lösen.

Idee:
1. Reduziere den Graphen mit der angegebenen Regel, bis sie nicht mehr anwendbar ist.
2. Führe auf dem Rest Ansatz 2 (Bounded Tree Search) aus.

Schritt 1:
Kann in Linearzeit erfolgen. Liefert einen Graphen mit $n', m' \in O(k^2)$ vielen Knoten und Kanten.

Schritt 2:
Rekursionstiefe ist maximal k, Verzweigungsgrad der Rekursion ist 2.
→ $O\left(2^k\right)$ Knoten im Suchbaum.
Den jeweils neuen Graphen zu berechnen dauert maximal $O(n' + m') = O(k^2)$ Zeit.

Laufzeiten für Vertex Cover

Ansatz	Laufzeit
Brute Force	$O(n^{k+1} \cdot k)$
Bounded Tree Search	$O(2^k \cdot k \cdot n)$
Kern + Bounded Tree Search	$O(2^k \cdot k^2 + n + m)$

Damit haben wir zwei Arten von Laufzeiten:

1. Die Form $O(n^k)$
2. Die Form $O(f(k) \cdot \text{poly}(n))$

Beide sind zwar polynomiell für fixiertes k, allerdings ist die zweite Variante immer polynomiell in der Größe des Graphen.

Kapitel 1.3 – Formale Definitionen

Parametrisierte Probleme

Definition 1.5
Ein **parametrisiertes Problem** ist eine Sprache $L \subseteq \Sigma^* \times \mathbb{N}$, wobei Σ ein beliebiges, aber festes Alphabet ist.

Für eine Instanz $(x, k) \in \Sigma^* \times \mathbb{N}$ wird k der **Parameter des Problems** genannt.

Beispiel:

- $L = \{(G, k) \mid G$ ist ein Graph mit einem Vertex Cover der Größe $\leq k. \}$.
  "Vertex Cover parametrisiert mit der Größe des Vertex Covers"
- $L = \{(G_k, \Delta) \mid G$ ist ein Graph mit Maximalgrad Δ und Vertex Cover der Größe $\leq k. \}$.
  "Vertex Cover parametrisiert mit der Größe des Maximalgrades."

Nebenbemerkung
Sei $\kappa: \Sigma^* \to \mathbb{N}$ eine poly.-Zeit berechenbare Funktion. Dann ist $\big(x, \kappa(x)\big)$ ein parametrisiertes Problem. κ heißt auch die **Parametrisierung** des Problems.

Komplexitätsklasse FPT

Definition 1.6
Ein parametrisiertes Problem $L \subseteq \Sigma^* \times \mathbb{N}$ heißt **fixed-parameter tractable**, wenn ein Algorithmus A, eine berechenbare Funktion $f: \mathbb{N} \to \mathbb{N}$ und eine Konstante c existieren, sodass für ein gegebenes $(x, k) \in \Sigma^* \times \mathbb{N}$ gilt:

- A entscheidet korrekt, ob $(x, k) \in L$
- Die Laufzeit von A liegt in $O(f(k) \cdot |(x, k)|^c)$

Die Klasse aller fixed-parameter tractable Probleme heißt **FPT**.

Das Vertex-Cover-Problem parametrisiert mit der Größe k des Vertex Covers ist in FPT, denn es gibt einen Algorithmus, der das parametrisierte Problem

- Korrekt entscheidet
- Laufzeit $O\left(2^k \cdot k^2 + n + m\right)$ besitzt

Komplexitätsklasse XP

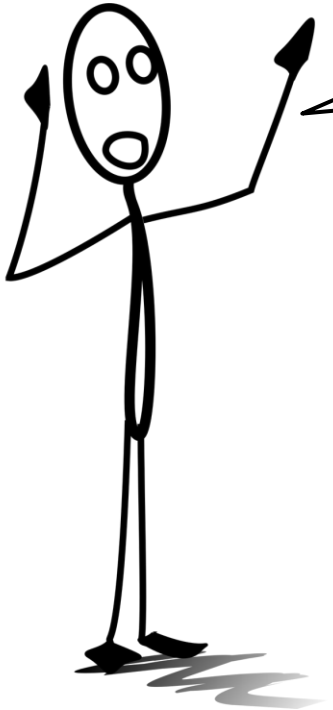
Definition 1.7
Ein parametrisiertes Problem $L \subseteq \Sigma^* \times \mathbb{N}$ heißt **stückweise polynomiell** (slice-wise polynomial), wenn
ein Algorithmus A und zwei berechenbare Funktionen $f, g: \mathbb{N} \to \mathbb{N}$ existieren,
sodass für ein gegebenes $(x, k) \in \Sigma^* \times \mathbb{N}$ gilt:

- A entscheidet korrekt, ob $(x, k) \in L$
- Die Laufzeit von A liegt in $O\left(f(k) \cdot |(x, k)|^{g(k)}\right)$

Die Klasse aller stückweise polynomiellen Probleme heißt **XP**.

Korollar 1.8
FPT $\subseteq$ XP.

Herausforderungen im Algorithmendesign

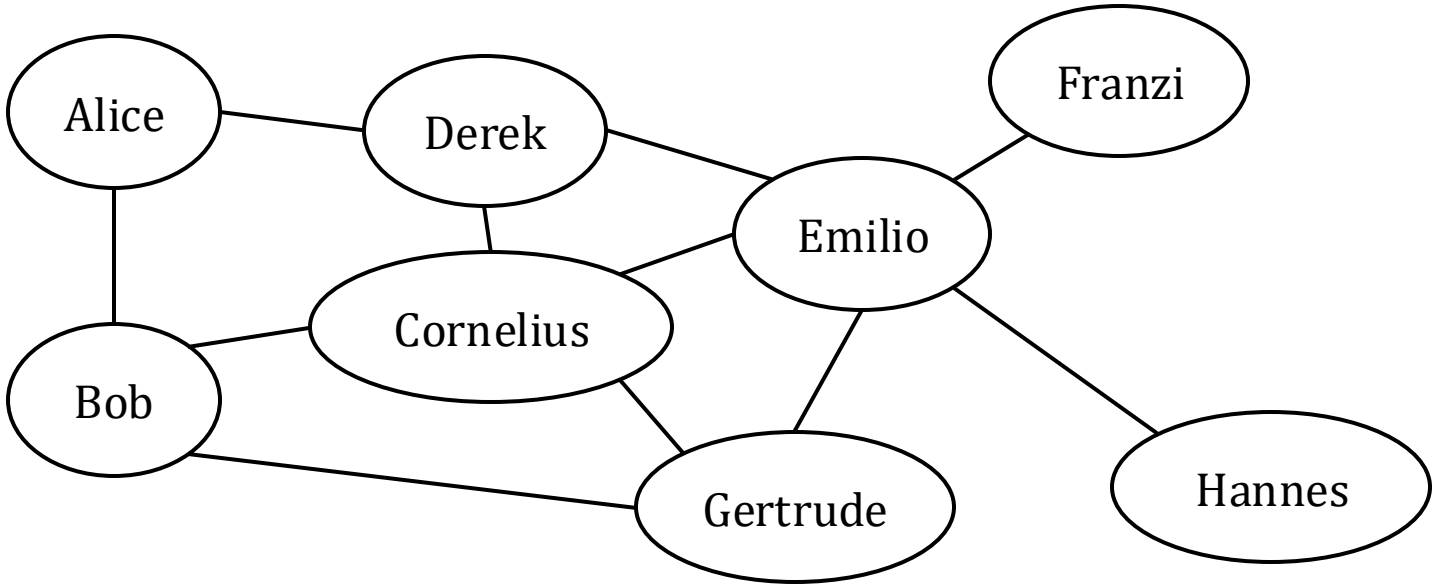


Für FPT Algorithmen kann versucht werden, die Konstante c, $f(k)$ oder beides zu minimieren.

Kapitel 1.4 – Vertex Coloring und Clique

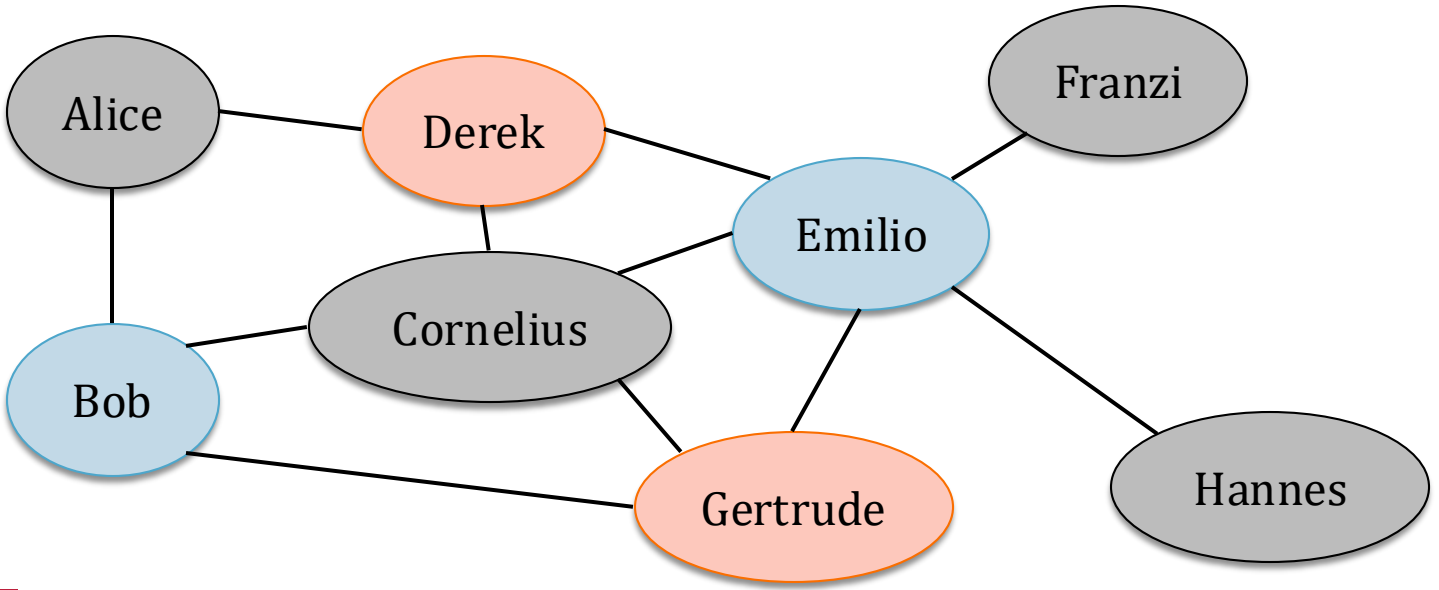
Vertex Coloring

Wie viele Bars benötigen wir, damit alle konfliktfrei in eine Bar gehen können?
Genügen 3?



Vertex Coloring

Wie viele Bars benötigen wir, damit alle konfliktfrei in eine Bar gehen können?
Genügen 3? Ja!



Vertex Coloring – Definition

Problem 1.9
Gegeben: Graph $G = (V, E)$, eine Zahl $k \in \mathbb{N}$
Frage: Existiert eine Funktion $f: V \to \{1, \dots, k\}$, sodass $\forall \{u, v\} \in E: f(u) \neq f(v)$

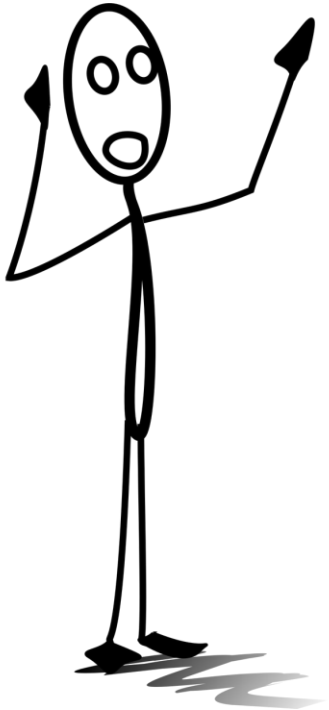
Fragen:

- Ist Vertex Coloring parametrisiert nach der Anzahl der Farben in FPT?
- Falls nein, ist das parametrisierte Problem in XP?

Zunächst:
Vertex Coloring ist NP-schwer. Sogar, wenn man $k = 3$ fixiert!

Damit:
Das par. Problem kann weder in XP noch in FPT liegen!

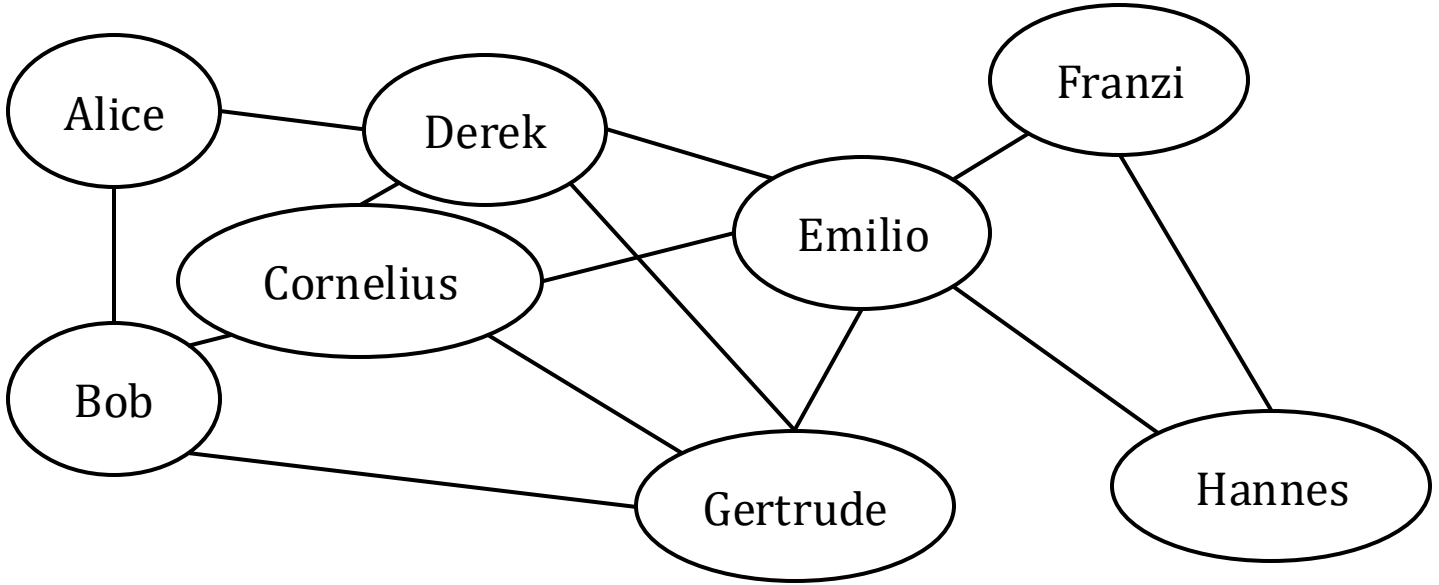
“Schwere NP-schwere Probleme”



Es gibt parametrisierte Probleme, die nicht in polynomieller Laufzeit entschieden werden können.

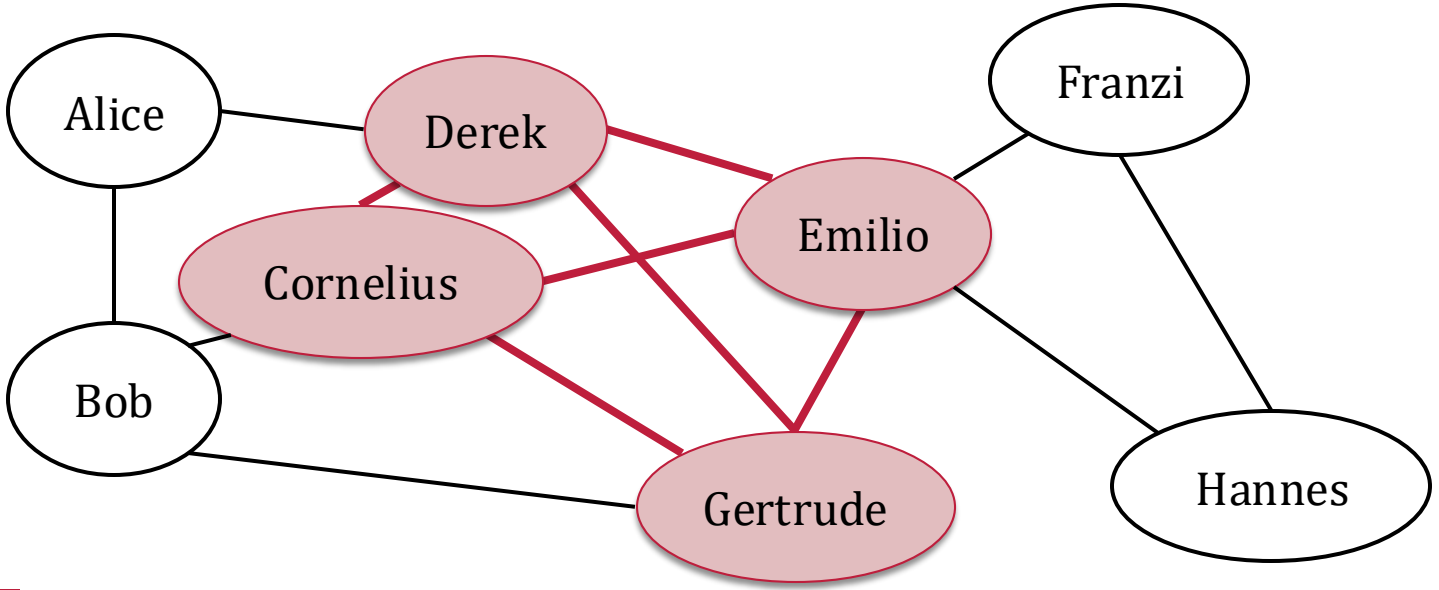
“All for one, one for all”

Besucher wollen mit ihren Freunden in die Bar. Hat man Problem mit einem, hat man ein Problem mit der Freundesgruppe (alle in der Gruppe müssen befreundet sein).
Angenommen, wir können mit maximal 3 Personen umgehen. Können wir alle hineinlassen?



“All for one, one for all”

Besucher wollen mit ihren Freunden in die Bar. Hat man Problem mit einem, hat man ein Problem mit der Freundesgruppe (alle in der Gruppe müssen befreundet sein).
Angenommen, wir können mit maximal 3 Personen umgehen. Können wir alle hineinlassen? Nein!



Clique – Definition

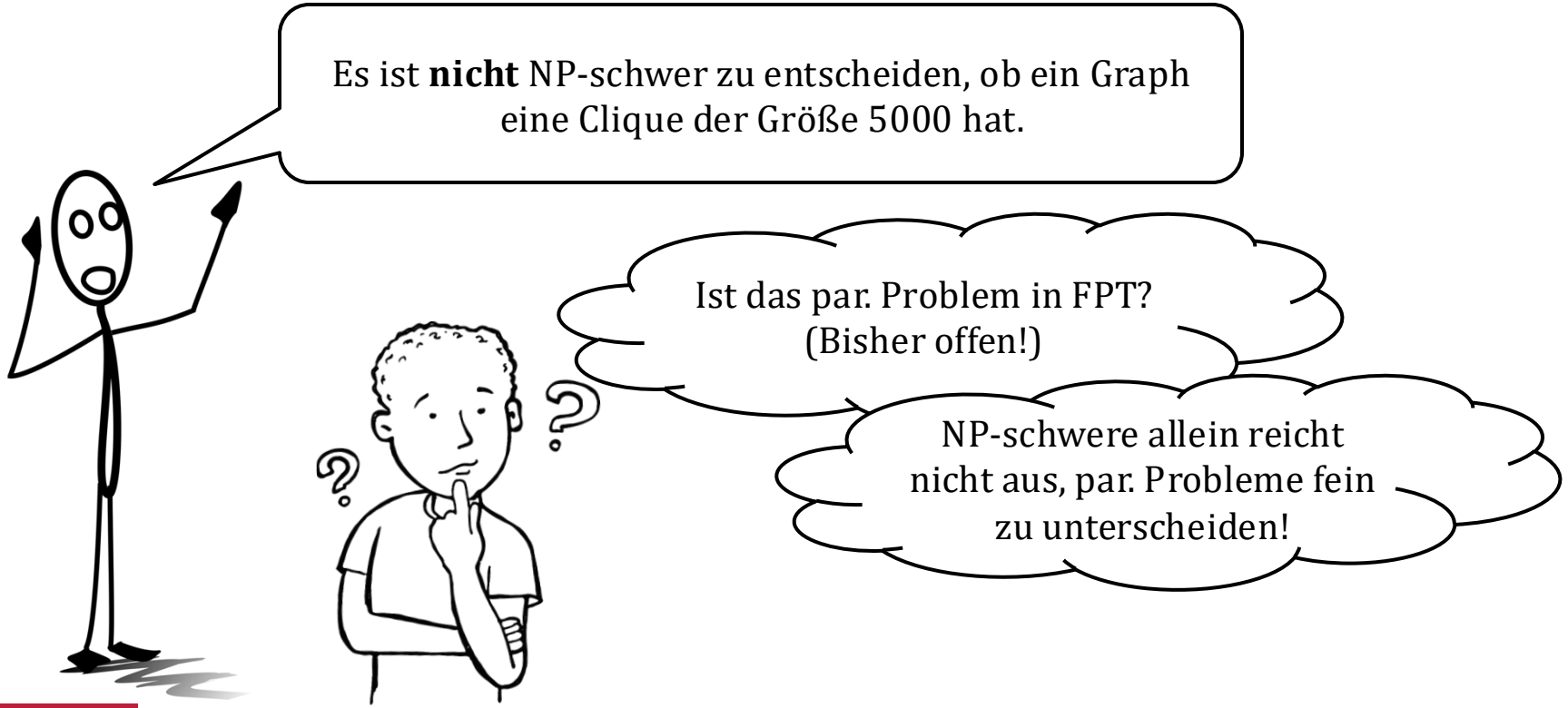
Problem 1.10
Gegeben: Graph $G = (V, E)$, eine Zahl $k \in \mathbb{N}$
Frage: Existiert eine Knotenmenge $C \subseteq V$ mit $|C| \geq k$ und $\forall u, v \in C: \{u, v\} \in E$?

Fragen:
- Ist Clique parametrisiert nach der Mindestgröße k in FPT?
- Falls nein, ist das parametrisierte Problem in XP?

Zunächst:
Das par. Problem ist in XP:
Enumeriere alle $O(n^k)$ Teilmengen von Knoten der Größe k und prüfe auf vollständige Verbundenheit.

Laufzeit: Ungefähr $O(n^k \cdot k^2)$

Konsequenz und offene Fragen



Inhalte der Vorlesung