

# Einführung und Definitionen

Caspar Nagy

27. Mai 2019

# Gliederung

- ▶ Motivation
- ▶ Beispiele für parametrisierte Probleme und ihre Komplexität
  - ▶ BAR FIGHT PREVENTION
    - ▶ Brute Force
    - ▶ Kernelization
    - ▶ Bounded Search Trees
  - ▶ VERTEX COLORING
  - ▶ CLIQUE
    - ▶ mit  $k$
    - ▶ mit  $\Delta$
- ▶ Definitionen
- ▶ Komplexitätsbeweise für parametrisierte Probleme
  - ▶ Theoretische Grundlagen
  - ▶ CLIQUE für reguläre Graphen  $\in W[1]$

# Motivation – Wo wir bei TGI stehen geblieben sind

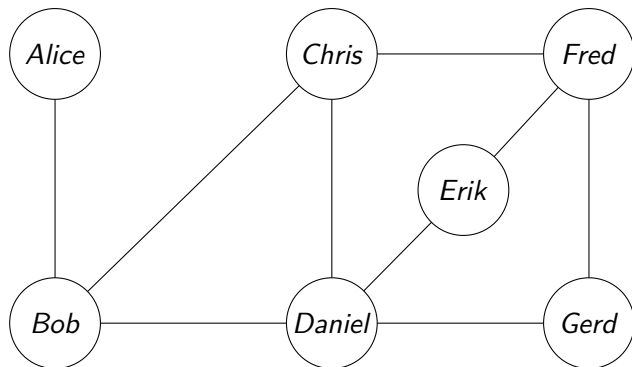
## Viele Interessante Probleme $\in NP$

- ▶ Lösungen für BAR FIGHT PREVENTION (aka VERTEX COVER) schon für  $n = 1000$  sehr unhandlich
- ▶ Laufzeit kann drastisch reduziert werden, wenn wir den Lösungsraum einschränken

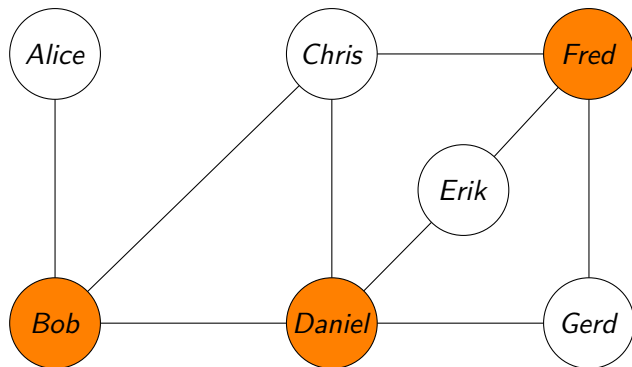
## Frage:

- ▶ Welche Parameter vereinfachen unser Problem tatsächlich?
- ▶ Welche Laufzeit kann man mit Parametrisierung erreichen?

## Beispiel BAR FIGHT PREVENTION



## Beispiel BAR FIGHT PREVENTION



# Beispiel – BAR FIGHT PREVENTION – Brute Force

```
min_size = INFINITY
for candidate in potenzmenge(g.V):
    if solution(candidate) and len(candidate) < min_size:
        best, min_size = candidate, len(candidate)
return best
```

- ▶ `solution(candidate)` wird  $2^n$  mal aufgerufen
- ▶ sei  $n = 1000$ :  $2^{1000} \approx 10^{301}$ 
  - ▶ terminiert nach Ende des Universums

Ansatz: Parameter  $k$  einführen, der Größe der Lösung beschränkt

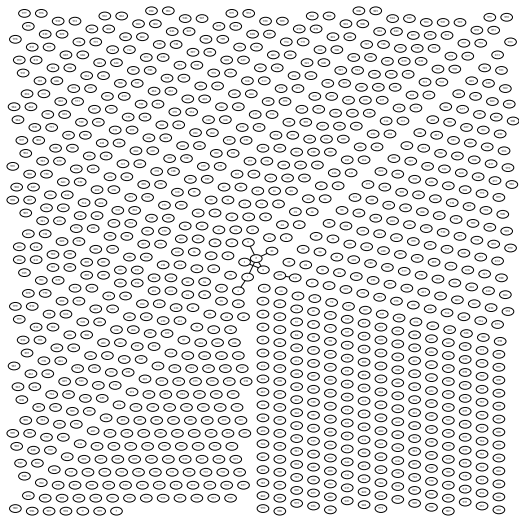
# Beispiel – BAR FIGHT PREVENTION – Brute Force

```
for candidate in k_teilmengen(g.V, k):  
    if solution(candidate):  
        return candidate  
return None
```

- ▶ 'solution(candidate)' wird  $\binom{n}{k}$  mal aufgerufen
- ▶ sei  $n = 1000, k = 10$  :  $\binom{1000}{10} \approx 2,63 * 10^{23}$ 
  - ▶ terminiert nach einigen Jahren auf einem state-of-the-art Supercomputer

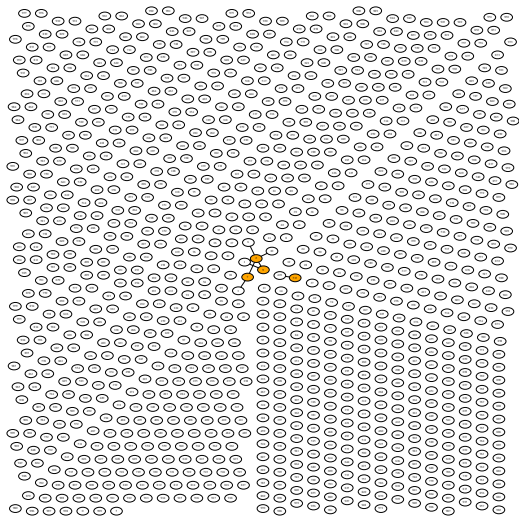
Noch immer unbefriedigend. Können wir den Suchraum weiter einschränken?

# Beispiel – BAR FIGHT PREVENTION

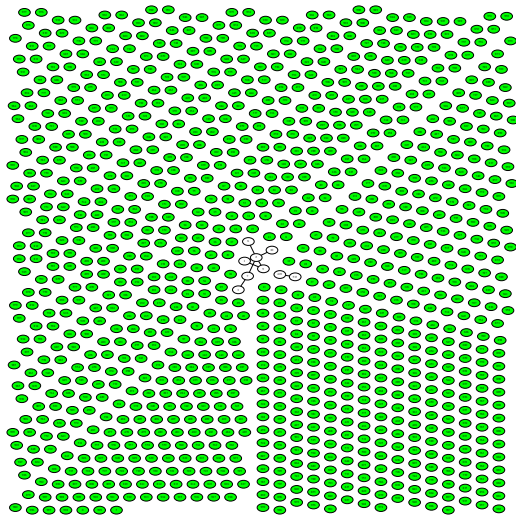




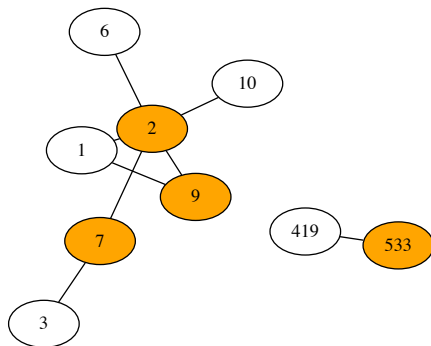
# Beispiel – BAR FIGHT PREVENTION



# Beispiel – BAR FIGHT PREVENTION

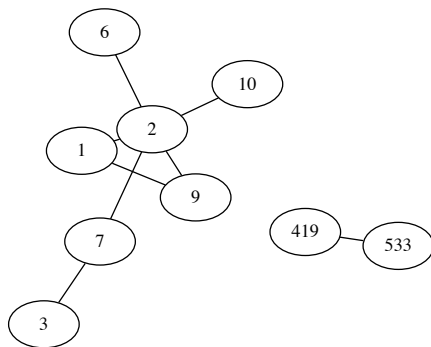


# Beispiel – BAR FIGHT PREVENTION – Kernelization



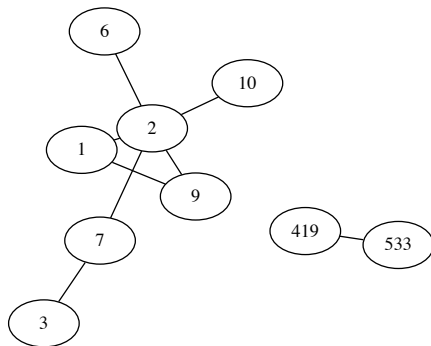
- ▶ Ist effizient berechnenbar
- ▶ Kann viele Instanzen wesentlich verkleinern
  - ▶ hilft uns aber nicht im worst-case
  - ▶ nutzt noch nicht unseren Parameter  $k$

# Beispiel – BAR FIGHT PREVENTION – Kernelization



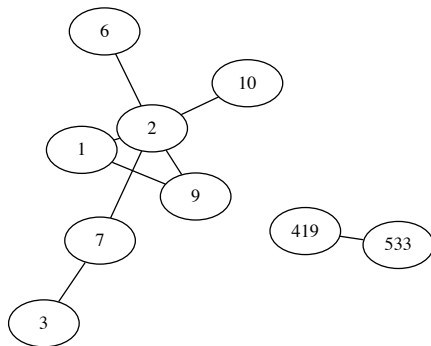
Frage: Wie können wir die verbleibenden Knoten entscheiden?

## Beispiel – BAR FIGHT PREVENTION – Kernelization



sei  $k = 4$

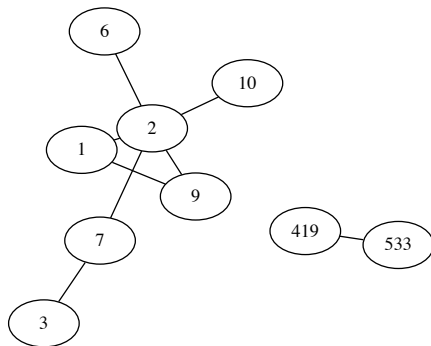
# Beispiel – BAR FIGHT PREVENTION – Kernelization



sei  $k = 4$

- Es ist egal ob wir 419 oder 533 herauswerfen.

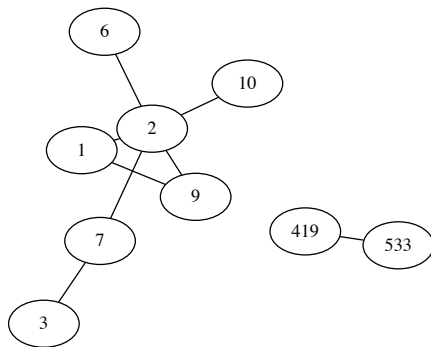
# Beispiel – BAR FIGHT PREVENTION – Kernelization



sei  $k = 4$

- ▶ Es ist egal ob wir 419 oder 533 herauswerfen.
- ▶ 2 muss immer herausgeworfen werden.

# Beispiel – BAR FIGHT PREVENTION – Kernelization



## Allgemein:

- ▶ Knoten mit  $\text{degree}(n) = 1$  können wir hereinlassen, solange dadurch nicht direkt ein Konflikt entsteht.
- ▶ Knoten mit  $\text{degree}(n) > k$  immer werden heraus geworfen.



# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v):
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

► Läuft in Polynomialzeit

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
```

```
denied = list()
```

```
unknown = list()
```

```
for v in G.V:
```

```
    if degree(v) == 0:
```

```
        accepted += v
```

```
    elif degree(v) == 1:
```

```
        if conflict(v:
```

```
            rejected += v
```

```
            k -= 1
```

```
        else: accepted += v
```

```
    elif degree(v) > k:
```

```
        rejected += v
```

```
        k -= 1
```

```
    else:
```

```
        unknown += v
```

```
    if k < 0:
```

```
        fail()
```

```
if len(G.subgraph(unknown + accepted).edges) > k*k:
```

```
    fail()
```

► Läuft in Polynomialzeit

► Was tun mit unknown?

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v):
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in unknown : degree(v) \leq k$

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v):
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v):
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v:
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte
    - ▶  $2 \cdot k^2$  Konfliktparteien

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v:
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte
    - ▶  $2 \cdot k^2$  Konfliktparteien
    - ▶  $\binom{2k^2}{k}$  Checks

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v:
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in unknown : degree(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte
    - ▶  $2 \cdot k^2$  Konfliktparteien
    - ▶  $\binom{2k^2}{k}$  Checks
    - ▶  $k = 10, \binom{200}{10} \approx 2,24 \cdot 10^{16}$



# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v:
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte
    - ▶  $2 \cdot k^2$  Konfliktparteien
    - ▶  $\binom{2k^2}{k}$  Checks
    - ▶  $k = 10, \binom{200}{10} \approx 2,24 \cdot 10^{16}$
  - ▶ da außerdem:  
 $\forall v \in \text{unknown} : \text{degree}(v) > 1$

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v:
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte
    - ▶  $2 \cdot k^2$  Konfliktparteien
    - ▶  $\binom{2k^2}{k}$  Checks
    - ▶  $k = 10, \binom{200}{10} \approx 2,24 * 10^{16}$
  - ▶ da außerdem:
    - $\forall v \in \text{unknown} : \text{degree}(v) > 1$
    - ▶ im Worst Case (Kreis) 1 Mensch pro Konflikt

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v:
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte
    - ▶  $2 \cdot k^2$  Konfliktparteien
    - ▶  $\binom{2k^2}{k}$  Checks
    - ▶  $k = 10, \binom{200}{10} \approx 2,24 * 10^{16}$
  - ▶ da außerdem:
    - $\forall v \in \text{unknown} : \text{degree}(v) > 1$ 
      - ▶ im Worst Case (Kreis) 1 Mensch pro Konflikt
      - ▶ also nur noch  $\binom{k^2}{k}$  Checks

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v:
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte
    - ▶  $2 \cdot k^2$  Konfliktparteien
    - ▶  $\binom{2k^2}{k}$  Checks
    - ▶  $k = 10, \binom{200}{10} \approx 2,24 * 10^{16}$
  - ▶ da außerdem:
    - $\forall v \in \text{unknown} : \text{degree}(v) > 1$
    - ▶ im Worst Case (Kreis) 1 Mensch pro Konflikt
    - ▶ also nur noch  $\binom{k^2}{k}$  Checks
    - ▶  $k = 10, \binom{100}{10} \approx 1,73 * 10^{13}$  Checks

# Beispiel – BAR FIGHT PREVENTION – Kernelization

```
accepted = list()
denied = list()
unknown = list()
for v in G.V:
    if degree(v) == 0:
        accepted += v
    elif degree(v) == 1:
        if conflict(v:
            rejected += v
            k -= 1
        else: accepted += v
    elif degree(v) > k:
        rejected += v
        k -= 1
    else:
        unknown += v
    if k < 0:
        fail()
if len(G.subgraph(unknown + accepted).edges) > k*k:
    fail()
```

- ▶ Läuft in Polynomialzeit
- ▶ Was tun mit unknown?
  - ▶  $\forall v \in \text{unknown} : \text{degree}(v) \leq k$
  - ▶  $\Rightarrow$  also höchstens...
    - ▶  $k^2$  ungelöste Konflikte
    - ▶  $2 \cdot k^2$  Konfliktparteien
    - ▶  $\binom{2k^2}{k}$  Checks
    - ▶  $k = 10, \binom{200}{10} \approx 2,24 * 10^{16}$
  - ▶ da außerdem:
    - $\forall v \in \text{unknown} : \text{degree}(v) > 1$ 
      - ▶ im Worst Case (Kreis) 1 Mensch pro Konflikt
      - ▶ also nur noch  $\binom{k^2}{k}$  Checks
      - ▶  $k = 10, \binom{100}{10} \approx 1,73 * 10^{13}$  Checks
      - ▶  $\Rightarrow$  einige Stunden auf einem Laptop

# Definitionen

# Definitionen 1/2

## Parametrisiertes Problem

- ▶  $(X, k) \in \Sigma^* \times \mathbb{N}$ , wobei  $X$  die Instanz des Problems und  $k$  die unäre Kodierung des Parameters ist. \*

## FPT (*Fixed Parameter Tractable*)

- ▶ Menge der parametrisierten Probleme, für die ein Algorithmus  $\mathcal{A}$  existiert, der Instanzen in Zeit  $f(k) \cdot |(x, k)|^c$  entscheidet.

## XP (*slice-wise polynomial*)

- ▶ Menge der parametrisierten Probleme, für die ein Algorithmus  $\mathcal{A}$  existiert, der Instanzen in Zeit  $f(k) \cdot |(x, k)|^{g(k)}$  entscheidet.

## Definitionen 2/2

Aus TGI kennen wir die Mengen  $P$  und  $NP$ . Für parametrisierte Probleme gibt es analog  $FPT/XP$  und  $W[1]$

- ▶  $W[1]$  ist die Menge aller parametrisierten Probleme, die mindestens so komplex sind wie das Finden einer  $CLIQUE$  der Größe  $k$ .
- ▶ Analog zu  $NP$  wird die  $W[1]$ -Vollständigkeit über polynomielle Transformationen gezeigt.
- ▶ Das alles ist natürlich sinnlos, sollte  $P = NP$  oder  $CLIQUE \in FPT$  sein.



Fragen?