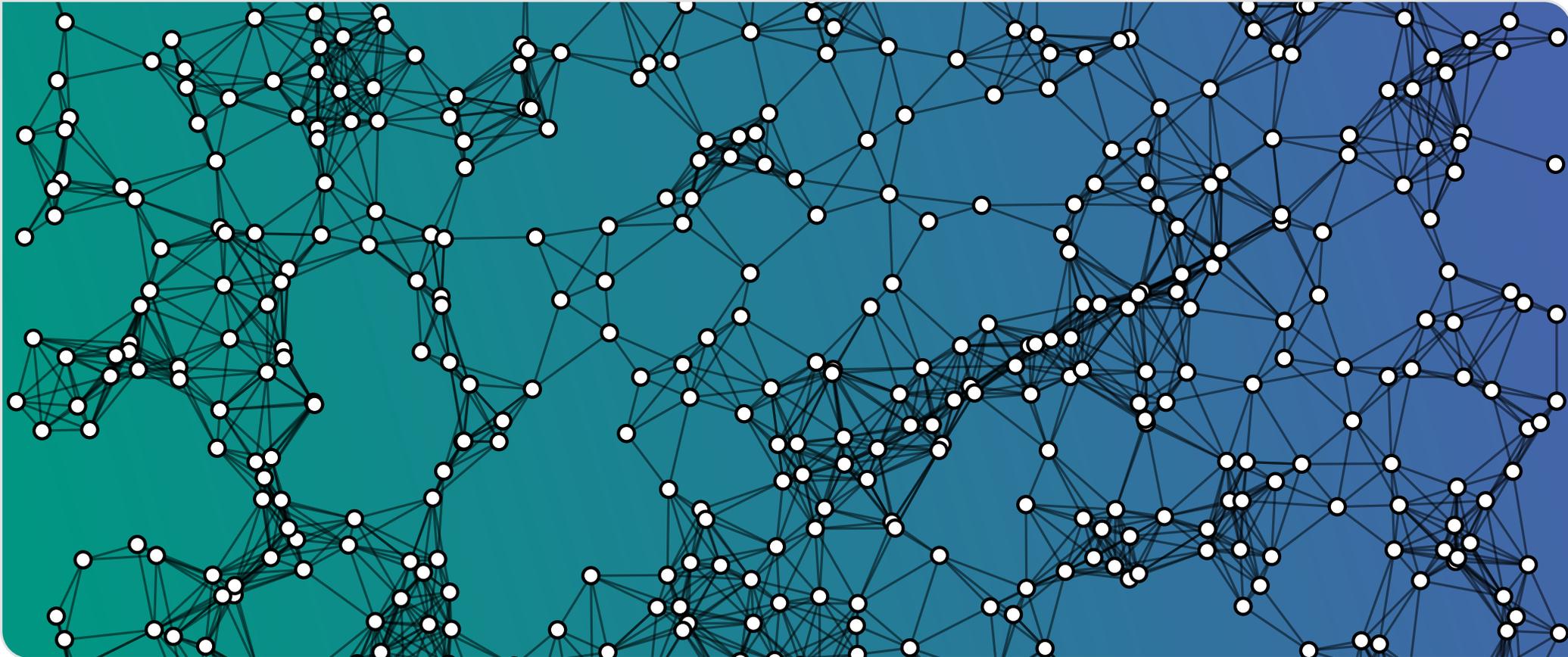


# Parametrisierte Algorithmen

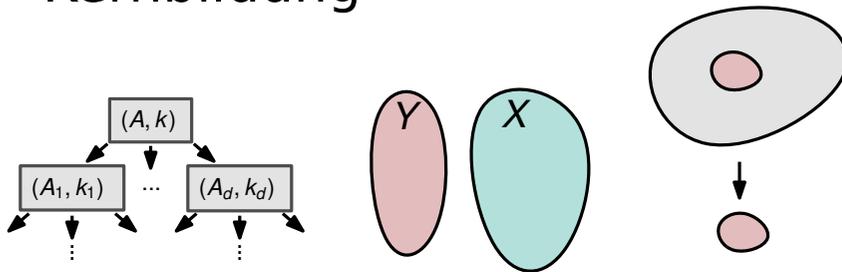
## Untere Schranken: ETH und SETH



# Inhalt

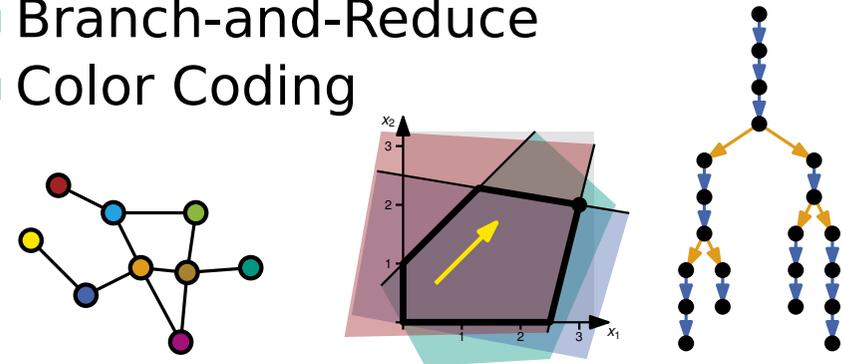
## Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung



## Erweiterte Toolbox

- lineare Programme
- Branch-and-Reduce
- Color Coding



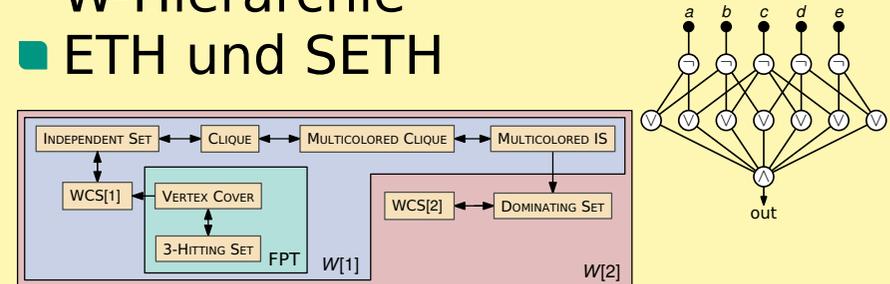
## Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem



## Untere Schranken

- parametrisierte Reduktionen
- boolesche Schaltkreise und die W-Hierarchie
- ETH und SETH



# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

## Was wissen wir?

- $q = 2$ :

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

## Was wissen wir?

- $q = 2$ :  $\delta_q = 0$ , denn  $2\text{-SAT} \in P$  und  $\text{poly}(n) \in O(2^{cn})$  für alle  $c > 0$

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

## Was wissen wir?

- $q = 2$ :  $\delta_q = 0$ , denn  $2\text{-SAT} \in P$  und  $\text{poly}(n) \in O(2^{cn})$  für alle  $c > 0$
- $q > 2$ :

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

## Was wissen wir?

- $q = 2$ :  $\delta_q = 0$ , denn  $2\text{-SAT} \in P$  und  $\text{poly}(n) \in O(2^{cn})$  für alle  $c > 0$
- $q > 2$ :  $\delta_q \leq 1$ , denn ein  $O(2^n)$ -Algorithmus existiert für jedes  $q$

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

## Was wissen wir?

- $q = 2$ :  $\delta_q = 0$ , denn  $2\text{-SAT} \in P$  und  $\text{poly}(n) \in O(2^{cn})$  für alle  $c > 0$
- $q > 2$ :  $\delta_q \leq 1$ , denn ein  $O(2^n)$ -Algorithmus existiert für jedes  $q$

## ETH

Es gilt  $\delta_3 > 0$ .

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

## Was wissen wir?

- $q = 2$ :  $\delta_q = 0$ , denn  $2\text{-SAT} \in P$  und  $\text{poly}(n) \in O(2^{cn})$  für alle  $c > 0$
- $q > 2$ :  $\delta_q \leq 1$ , denn ein  $O(2^n)$ -Algorithmus existiert für jedes  $q$

## ETH

Es gilt  $\delta_3 > 0$ .

**Also:** kein subexponentieller Algo für 3-SAT

**Vermutung:** die Hypothese stimmt

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

## Was wissen wir?

- $q = 2$ :  $\delta_q = 0$ , denn  $2\text{-SAT} \in P$  und  $\text{poly}(n) \in O(2^{cn})$  für alle  $c > 0$
- $q > 2$ :  $\delta_q \leq 1$ , denn ein  $O(2^n)$ -Algorithmus existiert für jedes  $q$

## ETH

Es gilt  $\delta_3 > 0$ .

**Also:** kein subexponentieller Algo für 3-SAT

**Vermutung:** die Hypothese stimmt

## SETH

Es gilt  $\lim_{q \rightarrow \infty} \delta_q = 1$ .

# (Strong) Exponential-Time Hypothesis

## Ziel

- $n^{\log \log k}$  ist zwar keine FPT-Laufzeit, aber deutlich besser als  $n^k$
- zeige:  $f(k) \cdot n^{o(k)}$  geht nicht für manche Probleme in  $W[1]$
- wir brauchen dazu eine etwas stärkere Annahme als  $FPT \neq W[1]$

## Definition

Für  $q \geq 2$ , sei  $\delta_q$  das Infimum aller  $c \in \mathbb{R}$ , für die es einen  $O(2^{cn})$  Algorithmus für  $q$ -SAT mit  $n$  Variablen gibt.

## Was wissen wir?

- $q = 2$ :  $\delta_q = 0$ , denn  $2\text{-SAT} \in P$  und  $\text{poly}(n) \in O(2^{cn})$  für alle  $c > 0$
- $q > 2$ :  $\delta_q \leq 1$ , denn ein  $O(2^n)$ -Algorithmus existiert für jedes  $q$

## ETH

Es gilt  $\delta_3 > 0$ .

**Also:** kein subexponentieller Algo für 3-SAT

**Vermutung:** die Hypothese stimmt

## SETH

Es gilt  $\lim_{q \rightarrow \infty} \delta_q = 1$ .

**Also:** für SAT geht es nicht besser als Brute-Force

**Vermutung:** unklar, aber besserer Algorithmus würde einen sehr großen Durchbruch bedeuten

# Ein paar Implikationen

**Theorem:** SETH impliziert ETH.

**Beweisidee**

**ETH:** kein subexponentieller Algo für 3-SAT  
**SETH:** für SAT geht es nicht besser als  $2^n$

# Ein paar Implikationen

**Theorem:** SETH impliziert ETH.

**ETH:** kein subexponentieller Algo für 3-SAT  
**SETH:** für SAT geht es nicht besser als  $2^n$

## Beweisidee

- reduziere  $q$ -SAT auf 3-SAT
- zeige: die Formel wird nicht so viel größer, dass aus etwas subexponentiellen etwas exponentielles wird

# Ein paar Implikationen

**Theorem:** SETH impliziert ETH.

**ETH:** kein subexponentieller Algo für 3-SAT  
**SETH:** für SAT geht es nicht besser als  $2^n$

## Beweisidee

- reduziere  $q$ -SAT auf 3-SAT
- zeige: die Formel wird nicht so viel größer, dass aus etwas subexponentiellen etwas exponentielles wird

**Theorem:** ETH impliziert  $FPT \neq W[1]$ .

## Beweisidee

# Ein paar Implikationen

**Theorem:** SETH impliziert ETH.

**ETH:** kein subexponentieller Algo für 3-SAT  
**SETH:** für SAT geht es nicht besser als  $2^n$

## Beweisidee

- reduziere  $q$ -SAT auf 3-SAT
- zeige: die Formel wird nicht so viel größer, dass aus etwas subexponentiellen etwas exponentielles wird

**Theorem:** ETH impliziert  $\text{FPT} \neq W[1]$ .

## Beweisidee

- zeige später: untere Schranke von  $n^{\Omega(k)}$  für ein Problem in  $W[1]$  (basierend auf ETH)
- das impliziert direkt  $\text{FPT} \neq W[1]$

# Ein paar Implikationen

**Theorem:** SETH impliziert ETH.

**ETH:** kein subexponentieller Algo für 3-SAT  
**SETH:** für SAT geht es nicht besser als  $2^n$

## Beweisidee

- reduziere  $q$ -SAT auf 3-SAT
- zeige: die Formel wird nicht so viel größer, dass aus etwas subexponentiellen etwas exponentielles wird

**Theorem:** ETH impliziert  $FPT \neq W[1]$ .

## Beweisidee

- zeige später: untere Schranke von  $n^{\Omega(k)}$  für ein Problem in  $W[1]$  (basierend auf ETH)
- das impliziert direkt  $FPT \neq W[1]$

## Theorem

ETH impliziert, dass es für VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, 3-COLORING und HAMILTONIAN CYCLE keinen  $2^{o(n+m)}$ -Algorithmus gibt.

## Beweisidee

# Ein paar Implikationen

**Theorem:** SETH impliziert ETH.

**ETH:** kein subexponentieller Algo für 3-SAT  
**SETH:** für SAT geht es nicht besser als  $2^n$

## Beweisidee

- reduziere  $q$ -SAT auf 3-SAT
- zeige: die Formel wird nicht so viel größer, dass aus etwas subexponentiellen etwas exponentielles wird

**Theorem:** ETH impliziert  $FPT \neq W[1]$ .

## Beweisidee

- zeige später: untere Schranke von  $n^{\Omega(k)}$  für ein Problem in  $W[1]$  (basierend auf ETH)
- das impliziert direkt  $FPT \neq W[1]$

## Theorem

ETH impliziert, dass es für VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, 3-COLORING und HAMILTONIAN CYCLE keinen  $2^{o(n+m)}$ -Algorithmus gibt.

## Beweisidee

- lineare Reduktionen von 3-SAT auf diese Probleme

# ETH und Parametrisierung

## Was haben wir?

- untere Schranken der Form  $2^{\Omega(n+m)}$

### **Theorem**

ETH impliziert, dass es für VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, 3-COLORING und HAMILTONIAN CYCLE keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# ETH und Parametrisierung

## Was haben wir?

- untere Schranken der Form  $2^{\Omega(n+m)}$

## Was wollen wir?

- untere Schranken der Form  $f(k) \cdot n^{\Omega(k)}$  (für ein parametrisiertes Problem)

### **Theorem**

ETH impliziert, dass es für VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, 3-COLORING und HAMILTONIAN CYCLE keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# ETH und Parametrisierung

## Was haben wir?

- untere Schranken der Form  $2^{\Omega(n+m)}$

## Was wollen wir?

- untere Schranken der Form  $f(k) \cdot n^{\Omega(k)}$  (für ein parametrisiertes Problem)

## Grundsätzlicher Plan

- reduziere ein „normales“ Problem auf ein parametrisiertes

### Theorem

ETH impliziert, dass es für VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, 3-COLORING und HAMILTONIAN CYCLE keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# ETH und Parametrisierung

## Was haben wir?

- untere Schranken der Form  $2^{\Omega(n+m)}$

## Was wollen wir?

- untere Schranken der Form  $f(k) \cdot n^{\Omega(k)}$  (für ein parametrisiertes Problem)

## Grundsätzlicher Plan

- reduziere ein „normales“ Problem auf ein parametrisiertes
- zähle bei der Reduktion mögliche Teillösungen auf und lasse das Zielproblem  $k$  Teillösungen „auswählen“

### Theorem

ETH impliziert, dass es für VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, 3-COLORING und HAMILTONIAN CYCLE keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# ETH und Parametrisierung

## Was haben wir?

- untere Schranken der Form  $2^{\Omega(n+m)}$

## Was wollen wir?

- untere Schranken der Form  $f(k) \cdot n^{\Omega(k)}$  (für ein parametrisiertes Problem)

## Grundsätzlicher Plan

- reduziere ein „normales“ Problem auf ein parametrisiertes
- zähle bei der Reduktion mögliche Teillösungen auf und lasse das Zielproblem  $k$  Teillösungen „auswählen“
- versteckt einen Teil der Schwierigkeit in der Eingabgröße und einen anderen Teil in der Auswahl von  $k$  Teillösungen

### Theorem

ETH impliziert, dass es für VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, 3-COLORING und HAMILTONIAN CYCLE keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# ETH und Parametrisierung

## Was haben wir?

- untere Schranken der Form  $2^{\Omega(n+m)}$

## Was wollen wir?

- untere Schranken der Form  $f(k) \cdot n^{\Omega(k)}$  (für ein parametrisiertes Problem)

## Grundsätzlicher Plan

- reduziere ein „normales“ Problem auf ein parametrisiertes
- zähle bei der Reduktion mögliche Teillösungen auf und lasse das Zielproblem  $k$  Teillösungen „auswählen“
- versteckt einen Teil der Schwierigkeit in der Eingabgröße und einen anderen Teil in der Auswahl von  $k$  Teillösungen

## Startproblem: 3-COLORING

- hat keine „Ausgabegröße“ → wir geraten nicht in Versuchung es parametrisiert zu betrachten
- lokale Teillösungen sind kombinierbar (sehen wir gleich)

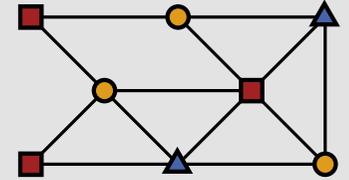
### Theorem

ETH impliziert, dass es für VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, 3-COLORING und HAMILTONIAN CYCLE keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# Auswahl geeigneter Teillösungen

## Problem: 3-COLORING

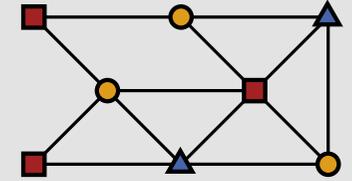
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



# Auswahl geeigneter Teillösungen

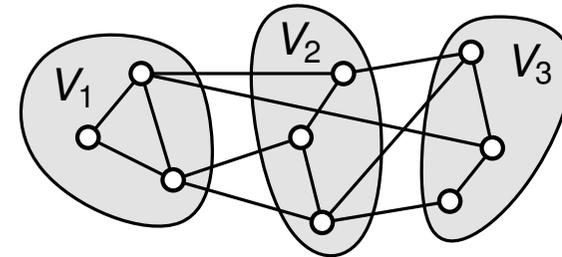
## Problem: 3-COLORING

Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



## Teillösungen

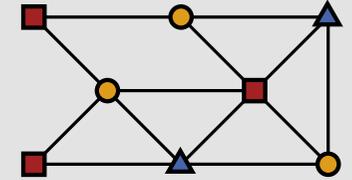
- partitioniere  $V = V_1 \cup \dots \cup V_k$



# Auswahl geeigneter Teillösungen

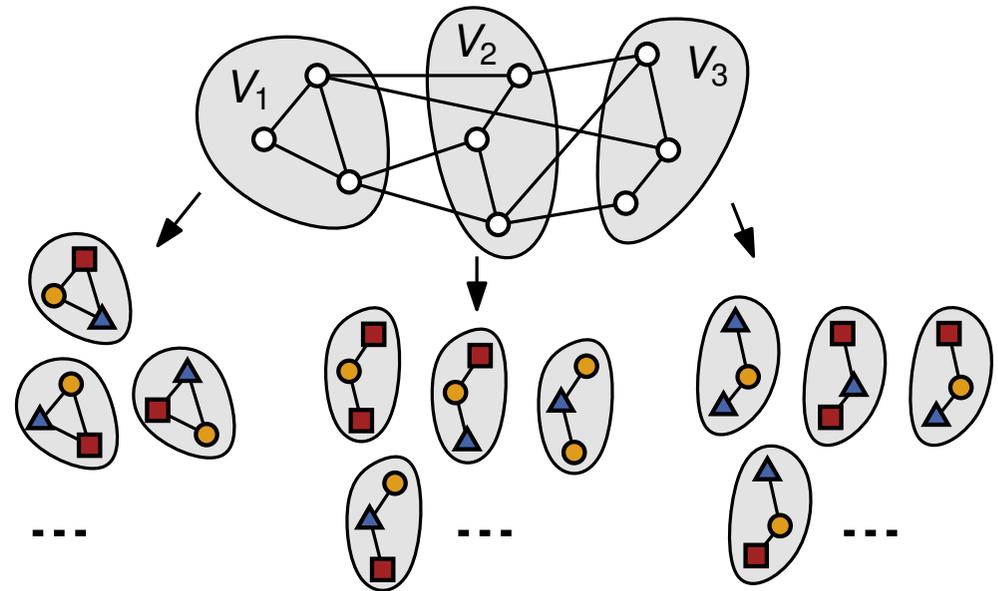
## Problem: 3-COLORING

Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



## Teillösungen

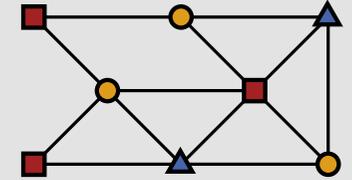
- partitioniere  $V = V_1 \cup \dots \cup V_k$
- betrachte Menge  $C_i$  aller 3-Färbungen für  $G[V_i]$



# Auswahl geeigneter Teillösungen

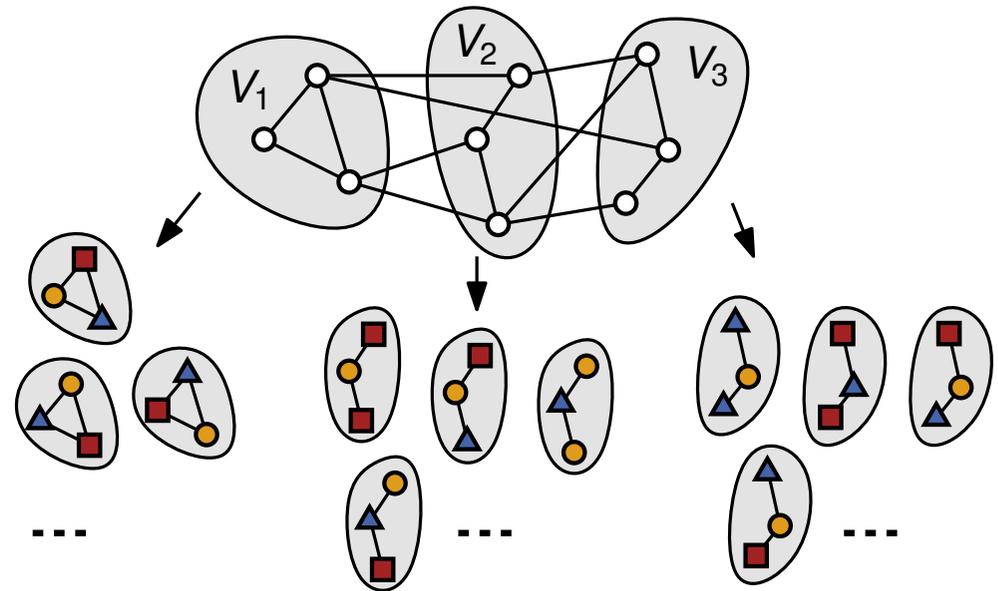
## Problem: 3-COLORING

Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



## Teillösungen

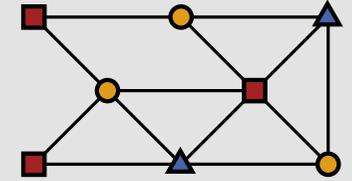
- partitioniere  $V = V_1 \cup \dots \cup V_k$
- betrachte Menge  $C_i$  aller 3-Färbungen für  $G[V_i]$
- wähle für jedes  $i$  eine Färbung  $c_i \in C_i$ , sodass je zwei  $c_i$  und  $c_j$  zusammenpassen



# Auswahl geeigneter Teillösungen

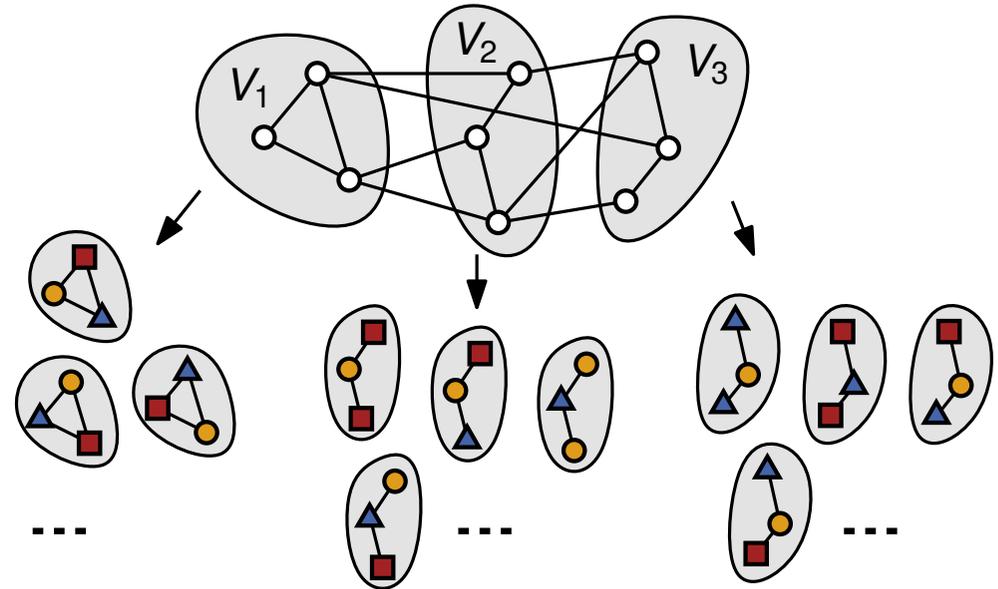
## Problem: 3-COLORING

Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



## Teillösungen

- partitioniere  $V = V_1 \cup \dots \cup V_k$
- betrachte Menge  $C_i$  aller 3-Färbungen für  $G[V_i]$
- wähle für jedes  $i$  eine Färbung  $c_i \in C_i$ , sodass je zwei  $c_i$  und  $c_j$  zusammenpassen



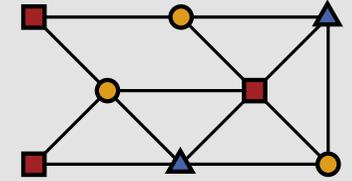
## Interpretation

- einen Teil der Schwierigkeit verstecken wir in der Problemgröße (wir zählen viele Färbungen auf)
- einen anderen Teil in der Auswahl von  $k$  passenden Teilfärbungen

# Auswahl geeigneter Teillösungen

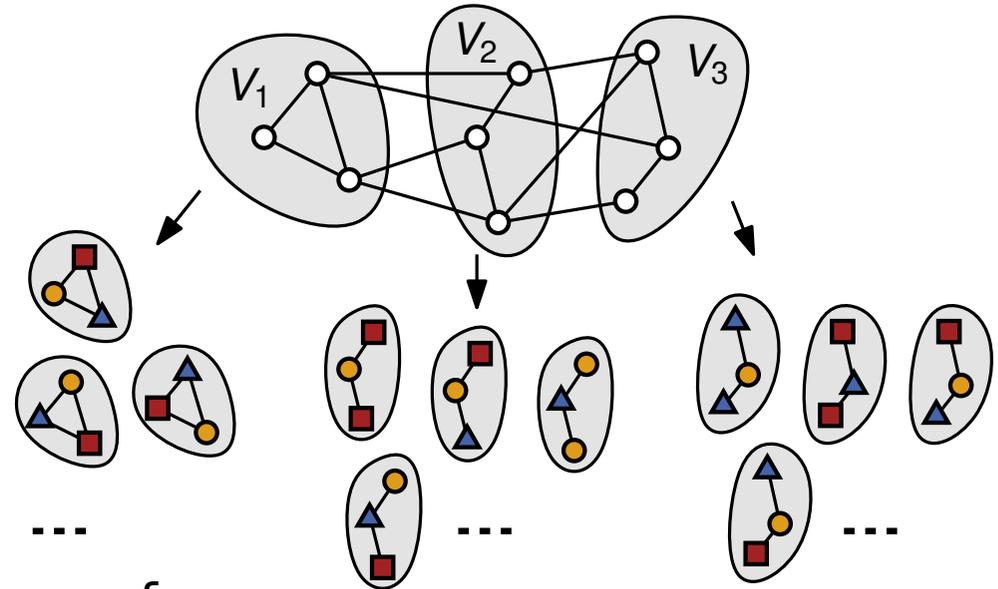
## Problem: 3-COLORING

Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



## Teillösungen

- partitioniere  $V = V_1 \cup \dots \cup V_k$
- betrachte Menge  $C_i$  aller 3-Färbungen für  $G[V_i]$
- wähle für jedes  $i$  eine Färbung  $c_i \in C_i$ , sodass je zwei  $c_i$  und  $c_j$  zusammenpassen



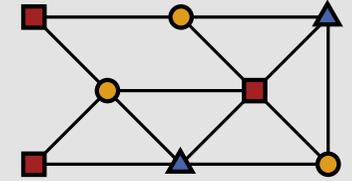
## Andere Sichtweise

- fasse  $C_1 \cup \dots \cup C_k$  als Knotenmenge auf

# Auswahl geeigneter Teillösungen

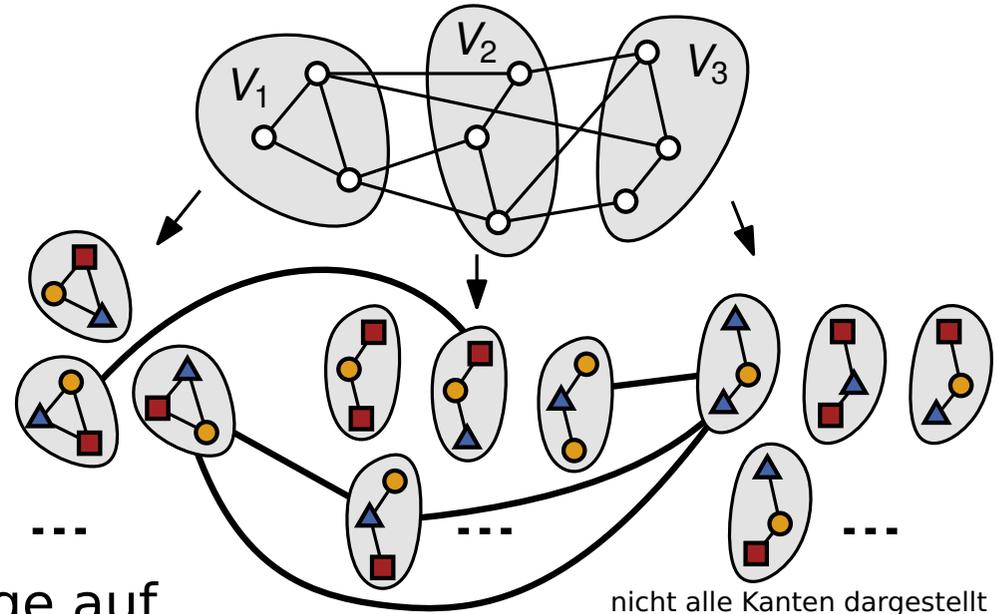
## Problem: 3-COLORING

Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



## Teillösungen

- partitioniere  $V = V_1 \cup \dots \cup V_k$
- betrachte Menge  $C_i$  aller 3-Färbungen für  $G[V_i]$
- wähle für jedes  $i$  eine Färbung  $c_i \in C_i$ , sodass je zwei  $c_i$  und  $c_j$  zusammenpassen



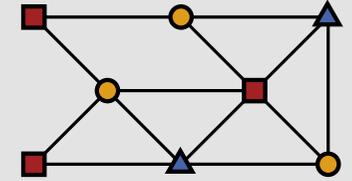
## Andere Sichtweise

- fasse  $C_1 \cup \dots \cup C_k$  als Knotenmenge auf
- füge Kante  $\{c_i, c_j\}$  ein, wenn  $c_i$  und  $c_j$  zusammenpassen ( $c_i \in C_i, c_j \in C_j$ )

# Auswahl geeigneter Teillösungen

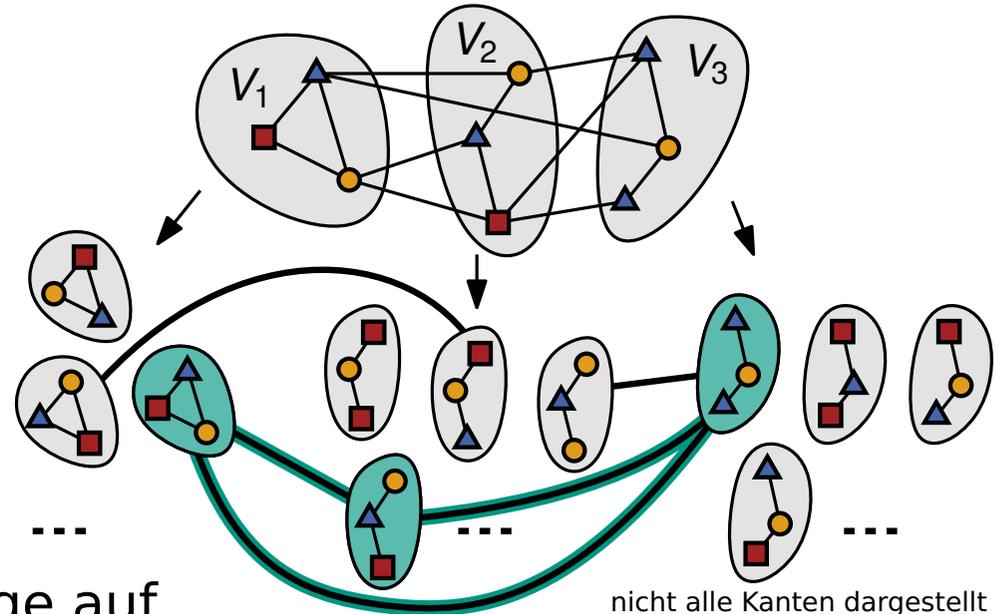
## Problem: 3-COLORING

Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



## Teillösungen

- partitioniere  $V = V_1 \cup \dots \cup V_k$
- betrachte Menge  $C_i$  aller 3-Färbungen für  $G[V_i]$
- wähle für jedes  $i$  eine Färbung  $c_i \in C_i$ , sodass je zwei  $c_i$  und  $c_j$  zusammenpassen



nicht alle Kanten dargestellt

## Andere Sichtweise

- fasse  $C_1 \cup \dots \cup C_k$  als Knotenmenge auf
- füge Kante  $\{c_i, c_j\}$  ein, wenn  $c_i$  und  $c_j$  zusammenpassen ( $c_i \in C_i, c_j \in C_j$ )

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

## Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

■ erster Versuch:  $k = n$

## Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
 ( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

■ erster Versuch:  $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$

## Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
 ( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

■ erster Versuch:  $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$   
 potentiell zu groß      klein genug

## Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
 ( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

■ erster Versuch:  $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$   
 potentiell zu groß      klein genug

■ zweiter Versuch:  $k$  konstant

**Theorem**  
 ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
 ( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

■ erster Versuch:  $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$   
 potentiell zu groß      klein genug

### Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

■ zweiter Versuch:  $k$  konstant

**Achtung:**  $o(k)$  wird nicht zu  $o(1)$

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
 ( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

■ erster Versuch:  $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$   
 potentiell zu groß      klein genug

### Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

■ zweiter Versuch:  $k$  konstant

**Achtung:**  $o(k)$  wird nicht zu  $o(1)$

( $o(k)$  wächst sublinear mit wachsendem  $k$ ; wenn  $k$  nicht wächst, dann ist auch  $o(k)$  konstant (und insbesondere nicht fallend bezüglich  $n$ ))

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
 ( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

■ erster Versuch:  $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$   
 potentiell zu groß      klein genug

### Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

■ zweiter Versuch:  $k$  konstant

**Achtung:**  $o(k)$  wird nicht zu  $o(1)$

( $o(k)$  wächst sublinear mit wachsendem  $k$ ; wenn  $k$  nicht wächst, dann ist auch  $o(k)$  konstant (und insbesondere nicht fallend bezüglich  $n$ ))

■ dritter Versuch:  $k = g(n) = f^{-1}(n) \rightarrow n \cdot 3^{\frac{n}{g(n)} \cdot o(g(n))} \leq 2^{o(n)}$

(wir können annehmen, dass  $g(n)$  streng monoton wächst)

# Eine untere Schranke für Clique

## Lemma

Der resultierend Graph mit  $3^{\frac{n}{k}} \cdot k$  Knoten hat eine Clique der Größe  $k$  genau dann, wenn  $G$  3-färbbar ist.

**Annahme:** es gibt einen  $f(k) \cdot n_c^{o(k)}$ -Algorithmus für CLIQUE  
 ( $n_c = 3^{\frac{n}{k}} = \#$ Knoten in der CLIQUE-Instanz)

## Resultierende Laufzeit für 3-COLORING

$f(k) \cdot 3^{\frac{n}{k} \cdot o(k)}$  (Ziel: wähle  $k$  so, dass die Laufzeit in  $2^{o(n)}$  liegt)

■ erster Versuch:  $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$   
 potentiell zu groß      klein genug

### Theorem

ETH impliziert, dass es für 3-COLORING keinen  $2^{o(n+m)}$ -Algorithmus gibt.

■ zweiter Versuch:  $k$  konstant

**Achtung:**  $o(k)$  wird nicht zu  $o(1)$

( $o(k)$  wächst sublinear mit wachsendem  $k$ ; wenn  $k$  nicht wächst, dann ist auch  $o(k)$  konstant (und insbesondere nicht fallend bezüglich  $n$ ))

■ dritter Versuch:  $k = g(n) = f^{-1}(n) \rightarrow n \cdot 3^{\frac{n}{g(n)} \cdot o(g(n))} \leq 2^{o(n)}$

(wir können annehmen, dass  $g(n)$  streng monoton wächst)

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für CLIQUE gibt.

# Implikationen

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für CLIQUE gibt.

## Mehr untere Schranken

- parametrisierte Reduktionen von CLIQUE liefern untere Schranken für andere Probleme
- einfachstes Beispiel: ETH  $\Rightarrow$  INDEPENDENT SET hat keinen  $f(k)n^{o(k)}$ -Algo

# Implikationen

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für CLIQUE gibt.

## Mehr untere Schranken

- parametrisierte Reduktionen von CLIQUE liefern untere Schranken für andere Probleme
- einfachstes Beispiel: ETH  $\Rightarrow$  INDEPENDENT SET hat keinen  $f(k)n^{o(k)}$ -Algo
- gilt ebenfalls für MULTICOLORED CLIQUE, MULTICOLORED INDEPENDENT SET, DOMINATING SET, CONNECTED DOMINATING SET, ...

# Implikationen

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für CLIQUE gibt.

## Mehr untere Schranken

- parametrisierte Reduktionen von CLIQUE liefern untere Schranken für andere Probleme
- einfachstes Beispiel: ETH  $\Rightarrow$  INDEPENDENT SET hat keinen  $f(k)n^{o(k)}$ -Algo
- gilt ebenfalls für MULTICOLORED CLIQUE, MULTICOLORED INDEPENDENT SET, DOMINATING SET, CONNECTED DOMINATING SET, ...
- **Achtung:** man muss bei der Reduktion aufpassen, ob sich der Parameter verändert

# Implikationen

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für CLIQUE gibt.

## Mehr untere Schranken

- parametrisierte Reduktionen von CLIQUE liefern untere Schranken für andere Probleme
- einfachstes Beispiel: ETH  $\Rightarrow$  INDEPENDENT SET hat keinen  $f(k)n^{o(k)}$ -Algo
- gilt ebenfalls für MULTICOLORED CLIQUE, MULTICOLORED INDEPENDENT SET, DOMINATING SET, CONNECTED DOMINATING SET, ...
- **Achtung:** man muss bei der Reduktion aufpassen, ob sich der Parameter verändert

## Die W-Hierarchie

- wir erhalten:

**Theorem:** ETH impliziert  $FPT \neq W[1]$ .

# Implikationen

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für CLIQUE gibt.

## Mehr untere Schranken

- parametrisierte Reduktionen von CLIQUE liefern untere Schranken für andere Probleme
- einfachstes Beispiel: ETH  $\Rightarrow$  INDEPENDENT SET hat keinen  $f(k)n^{o(k)}$ -Algo
- gilt ebenfalls für MULTICOLORED CLIQUE, MULTICOLORED INDEPENDENT SET, DOMINATING SET, CONNECTED DOMINATING SET, ...
- **Achtung:** man muss bei der Reduktion aufpassen, ob sich der Parameter verändert

## Die W-Hierarchie

- wir erhalten:

**Theorem:** ETH impliziert  $FPT \neq W[1]$ .

## Gleich

- weitere (leicht anders geartete) Implikationen
- wichtiges Zwischenproblem: GRID TILING

# GRID TILING

## Problem: GRID TILING

 $([n] = [1, n])$ 

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass  $s_{i,j}$  und  $s_{i+1,j}$  bzw.  $s_{i,j}$  und  $s_{i,j+1}$  sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

## Beispiel

- $k = 3$  und  $n = 5$

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

# GRID TILING

## Problem: GRID TILING

 $([n] = [1, n])$ 

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass  $s_{i,j}$  und  $s_{i+1,j}$  bzw.  $s_{i,j}$  und  $s_{i,j+1}$  sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

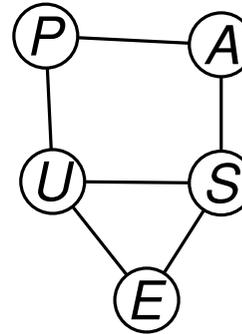
## Beispiel

- $k = 3$  und  $n = 5$

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

# Was ist hier passiert?

**Ist die GRID TILING-Instanz lösbar?**  
**Was bedeutet das für den Graphen?**



|   |   |   |
|---|---|---|
| $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          |

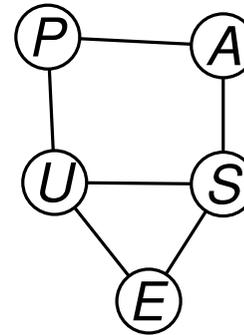
|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |

## Problem: GRID TILING

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass  $s_{i,j}$  und  $s_{i+1,j}$  bzw.  $s_{i,j}$  und  $s_{i,j+1}$  sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

# Was ist hier passiert?

**Ist die GRID TILING-Instanz lösbar?**  
**Was bedeutet das für den Graphen?**



|   |   |   |
|---|---|---|
| $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          |

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |

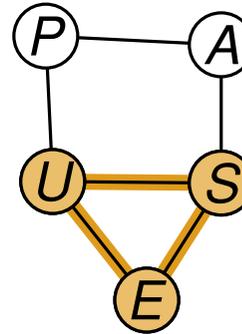
## Problem: GRID TILING

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass  $s_{i,j}$  und  $s_{i+1,j}$  bzw.  $s_{i,j}$  und  $s_{i,j+1}$  sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

# Was ist hier passiert?

## Ist die GRID TILING-Instanz lösbar? Was bedeutet das für den Graphen?

- eine 3-Clique in  $G$  liefert eine Lösung für GRID TILING



|   |   |   |
|---|---|---|
| $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          |

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |

### Problem: GRID TILING

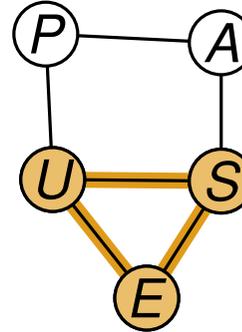
Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass  $s_{i,j}$  und  $s_{i+1,j}$  bzw.  $s_{i,j}$  und  $s_{i,j+1}$  sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

# Was ist hier passiert?

## Ist die GRID TILING-Instanz lösbar? Was bedeutet das für den Graphen?

- eine 3-Clique in  $G$  liefert eine Lösung für GRID TILING
- Umkehrung gilt auch

Warum?



|   |   |   |
|---|---|---|
| $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          |

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |

### Problem: GRID TILING

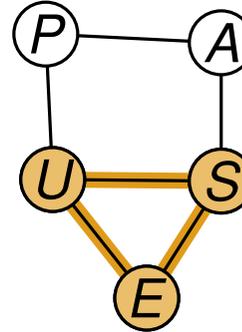
Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass  $s_{i,j}$  und  $s_{i+1,j}$  bzw.  $s_{i,j}$  und  $s_{i,j+1}$  sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

# Was ist hier passiert?

## Ist die GRID TILING-Instanz lösbar? Was bedeutet das für den Graphen?

- eine 3-Clique in  $G$  liefert eine Lösung für GRID TILING
- Umkehrung gilt auch

Warum?



### Beachte

- $n = \# \text{Knoten}$ ,  $k = \text{Cliquengröße}$
- wir haben also eine parametrisierte Reduktion von CLIQUE ZU GRID TILING
- Parameter  $k$  überträgt sich linear

|   |   |   |
|---|---|---|
| $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          |

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |

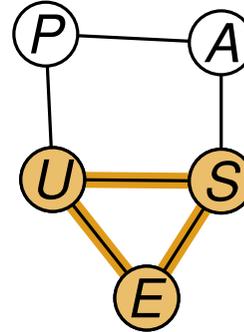
### Problem: GRID TILING

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass  $s_{i,j}$  und  $s_{i+1,j}$  bzw.  $s_{i,j}$  und  $s_{i,j+1}$  sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

# Was ist hier passiert?

## Ist die GRID TILING-Instanz lösbar? Was bedeutet das für den Graphen?

- eine 3-Clique in  $G$  liefert eine Lösung für GRID TILING
- Umkehrung gilt auch Warum?



### Beachte

- $n = \# \text{Knoten}$ ,  $k = \text{Cliquengröße}$
- wir haben also eine parametrisierte Reduktion von CLIQUE ZU GRID TILING
- Parameter  $k$  überträgt sich linear

|   |   |   |
|---|---|---|
| $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          | $(A, P)$ $(U, P)$<br>$(S, A)$ $(S, U)$<br>$(E, U)$ $(E, S)$ |
| $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, A)$ $(P, U)$<br>$(A, S)$ $(U, S)$<br>$(U, E)$ $(S, E)$ | $(P, P)$ $(A, A)$<br>$(U, U)$ $(S, S)$<br>$(E, E)$          |

### Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für GRID TILING gibt (außerdem ist es  $W[1]$ -schwer).

### Problem: GRID TILING

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass  $s_{i,j}$  und  $s_{i+1,j}$  bzw.  $s_{i,j}$  und  $s_{i,j+1}$  sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |

# Anmerkungen zu GRID TILING

## Reduktionen von GRID TILING

- die grobe Struktur von GRID TILING ist ein Gitter
- alle Bedingungen sind lokal: nur zwischen benachbarten Zellen
- die Bedingungen sind sehr einfach: Gleichheit in einer Koordinate

# Anmerkungen zu GRID TILING

## Reduktionen von GRID TILING

- die grobe Struktur von GRID TILING ist ein Gitter
  - alle Bedingungen sind lokal: nur zwischen benachbarten Zellen
  - die Bedingungen sind sehr einfach: Gleichheit in einer Koordinate
- ⇒ GRID TILING eignet sich gut um Schwere auf planaren Graphen zu zeigen

# Anmerkungen zu GRID TILING

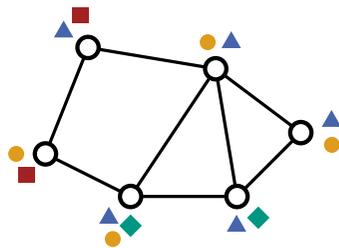
## Reduktionen von GRID TILING

- die grobe Struktur von GRID TILING ist ein Gitter
  - alle Bedingungen sind lokal: nur zwischen benachbarten Zellen
  - die Bedingungen sind sehr einfach: Gleichheit in einer Koordinate
- ⇒ GRID TILING eignet sich gut um Schwere auf planaren Graphen zu zeigen

### Problem: PLANAR LIST COLORING

Gegeben sei ein planarer Graph und eine Liste von Farben für jeden Knoten. Wähle für jeden Knoten eine Farbe aus seiner Liste, sodass benachbarte Knoten unterschiedlich gefärbt sind.

## Beispiel



# Anmerkungen zu GRID TILING

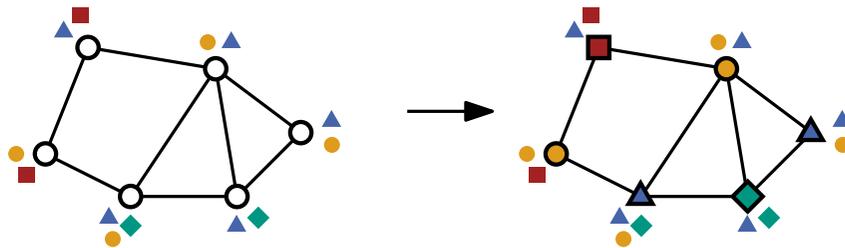
## Reduktionen von GRID TILING

- die grobe Struktur von GRID TILING ist ein Gitter
  - alle Bedingungen sind lokal: nur zwischen benachbarten Zellen
  - die Bedingungen sind sehr einfach: Gleichheit in einer Koordinate
- ⇒ GRID TILING eignet sich gut um Schwere auf planaren Graphen zu zeigen

### Problem: PLANAR LIST COLORING

Gegeben sei ein planarer Graph und eine Liste von Farben für jeden Knoten. Wähle für jeden Knoten eine Farbe aus seiner Liste, sodass benachbarte Knoten unterschiedlich gefärbt sind.

## Beispiel



# Anmerkungen zu GRID TILING

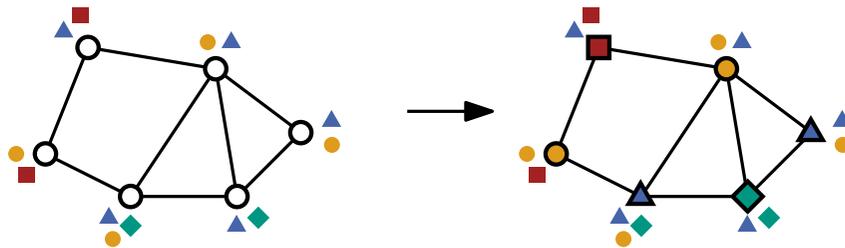
## Reduktionen von GRID TILING

- die grobe Struktur von GRID TILING ist ein Gitter
  - alle Bedingungen sind lokal: nur zwischen benachbarten Zellen
  - die Bedingungen sind sehr einfach: Gleichheit in einer Koordinate
- ⇒ GRID TILING eignet sich gut um Schwere auf planaren Graphen zu zeigen

### Problem: PLANAR LIST COLORING

Gegeben sei ein planarer Graph und eine Liste von Farben für jeden Knoten. Wähle für jeden Knoten eine Farbe aus seiner Liste, sodass benachbarte Knoten unterschiedlich gefärbt sind.

### Beispiel



**Ziel:** reduziere GRID TILING auf PLANAR LIST COLORING

# GRID TILING $\rightarrow$ PLANAR LIST COLORING

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

## Wahl der Knoten und Farben

# GRID TILING $\rightarrow$ PLANAR LIST COLORING

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

|                              |                              |                              |
|------------------------------|------------------------------|------------------------------|
| ○ (1, 1)<br>(3, 1)<br>(2, 4) | ○ (5, 1)<br>(1, 4)<br>(5, 3) | ○ (1, 1)<br>(2, 4)<br>(3, 3) |
|------------------------------|------------------------------|------------------------------|

|                    |                    |                    |
|--------------------|--------------------|--------------------|
| ○ (2, 2)<br>(1, 4) | ○ (3, 1)<br>(1, 2) | ○ (2, 2)<br>(2, 3) |
|--------------------|--------------------|--------------------|

|                              |                    |                    |
|------------------------------|--------------------|--------------------|
| ○ (1, 3)<br>(2, 3)<br>(3, 3) | ○ (1, 1)<br>(1, 3) | ○ (2, 3)<br>(5, 3) |
|------------------------------|--------------------|--------------------|

## Wahl der Knoten und Farben

- naheliegender Versuch: ein Knoten pro Zelle, eine Farbe pro Paar

# GRID TILING $\rightarrow$ PLANAR LIST COLORING

|   |   |   |
|---|---|---|
| $S_{1,1}$<br><span style="background-color: #f4a460;">(1, 1)</span><br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br><span style="background-color: #6699cc;">(2, 2)</span><br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3)   | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

|  |          |          |
|--|----------|----------|
| ○ <span style="background-color: #f4a460;">(1, 1)</span> | ○ (5, 1) | ○ (1, 1) |
| (3, 1)   | (1, 4)   | (2, 4)   |
| (2, 4)   | (5, 3)   | (3, 3)   |

|  |          |          |
|--|----------|----------|
| ○ <span style="background-color: #6699cc;">(2, 2)</span> | ○ (3, 1) | ○ (2, 2) |
| (1, 4)   | (1, 2)   | (2, 3)   |

|          |          |          |
|----------|----------|----------|
| ○ (1, 3) | ○ (1, 1) | ○ (2, 3) |
| (2, 3)   | (1, 3)   | (5, 3)   |
| (3, 3)   |          |          |

## Wahl der Knoten und Farben

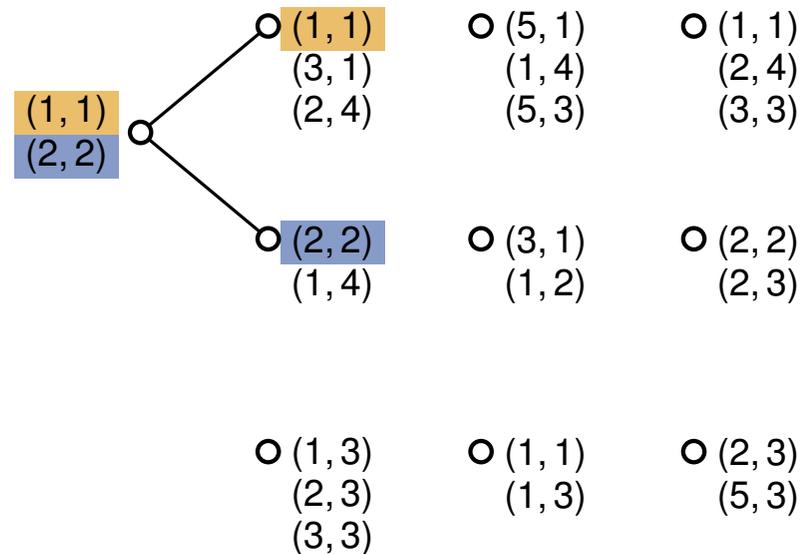
- naheliegender Versuch: ein Knoten pro Zelle, eine Farbe pro Paar

## Verbiete gewisse Farbwahlen

- verhindere, dass (1, 1) für  $S_{1,1}$  und (2, 2) für  $S_{2,1}$  ausgewählt wird

# GRID TILING $\rightarrow$ PLANAR LIST COLORING

|   |   |   |
|---|---|---|
| $S_{1,1}$<br><span style="background-color: #f4a460;">(1, 1)</span><br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br><span style="background-color: #6699cc;">(2, 2)</span><br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3)   | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |



## Wahl der Knoten und Farben

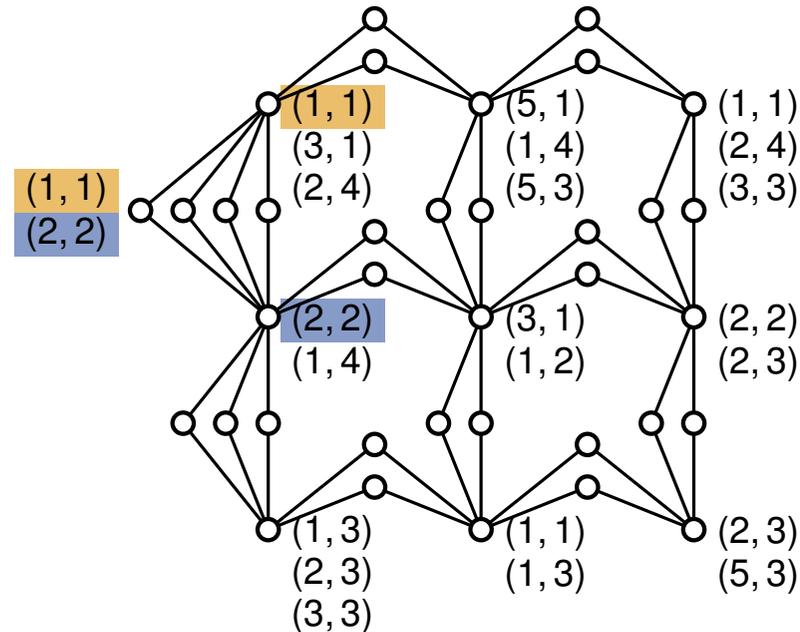
- naheliegender Versuch: ein Knoten pro Zelle, eine Farbe pro Paar

## Verbiere gewisse Farbwahlen

- verhindere, dass (1, 1) für  $S_{1,1}$  und (2, 2) für  $S_{2,1}$  ausgewählt wird
- verbinde beide mit neuem Knoten mit den Farben (1, 1) und (2, 2)

# GRID TILING $\rightarrow$ PLANAR LIST COLORING

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |



## Wahl der Knoten und Farben

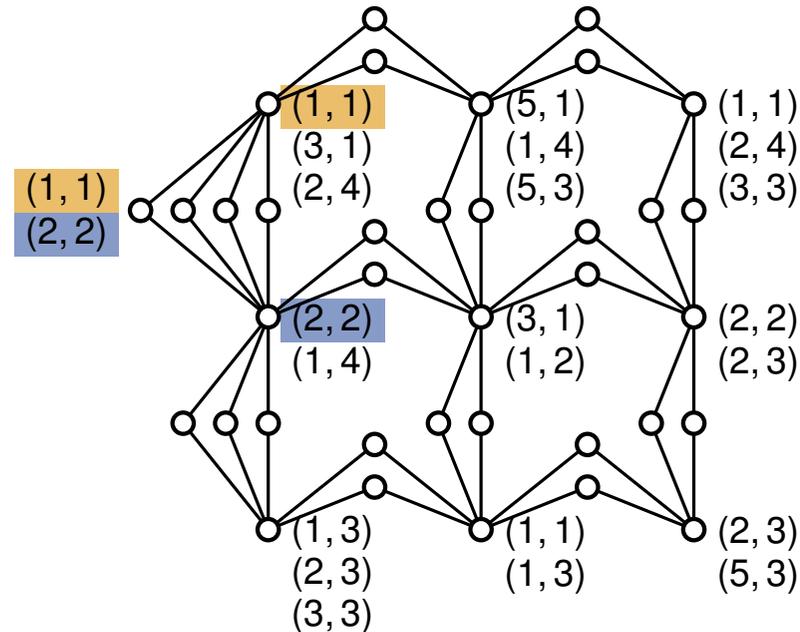
- naheliegender Versuch: ein Knoten pro Zelle, eine Farbe pro Paar

## Verbiete gewisse Farbwahlen

- verhindere, dass  $(1, 1)$  für  $S_{1,1}$  und  $(2, 2)$  für  $S_{2,1}$  ausgewählt wird
- verbinde beide mit neuem Knoten mit den Farben  $(1, 1)$  und  $(2, 2)$
- analoges Vorgehen für alle unerwünschten Paare

# GRID TILING $\rightarrow$ PLANAR LIST COLORING

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |



## Wahl der Knoten und Farben

- naheliegender Versuch: ein Knoten pro Zelle, eine Farbe pro Paar

## Verbiete gewisse Farbwahlen

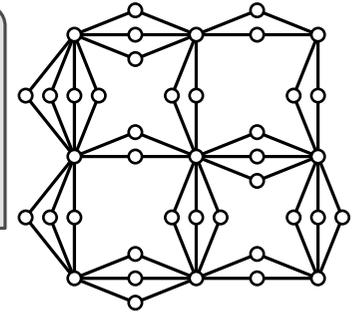
- verhindere, dass  $(1, 1)$  für  $S_{1,1}$  und  $(2, 2)$  für  $S_{2,1}$  ausgewählt wird
- verbinde beide mit neuem Knoten mit den Farben  $(1, 1)$  und  $(2, 2)$
- analoges Vorgehen für alle unerwünschten Paare

**Beachte:** Graph ist planar und die Instanzen sind äquivalent

# Was zeigen wir hier eigentlich?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für PLANAR LIST COLORING gibt (außerdem ist es  $W[1]$ -schwer).



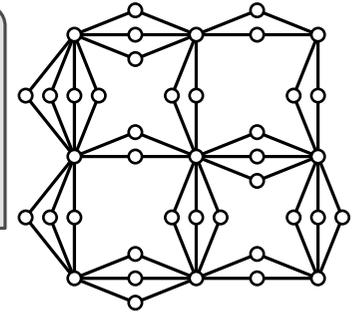
## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für GRID TILING gibt (außerdem ist es  $W[1]$ -schwer).

# Was zeigen wir hier eigentlich?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für PLANAR LIST COLORING gibt (außerdem ist es  $W[1]$ -schwer).



## Für welchen Parameter $k$ ?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für GRID TILING gibt (außerdem ist es  $W[1]$ -schwer).

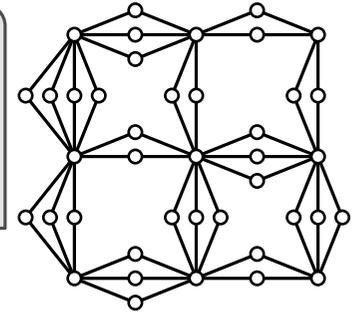
## Erinnerung

■  $k$  war die Gittergröße

# Was zeigen wir hier eigentlich?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für PLANAR LIST COLORING gibt (außerdem ist es  $W[1]$ -schwer).



Warum?

## Für welchen Parameter $k$ ?

- Beobachtung: der resultierende Graph hat Baumweite  $k$

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für GRID TILING gibt (außerdem ist es  $W[1]$ -schwer).

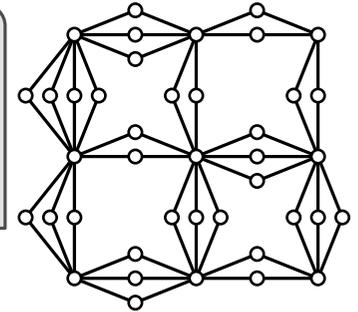
## Erinnerung

- $k$  war die Gittergröße

# Was zeigen wir hier eigentlich?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für PLANAR LIST COLORING gibt (außerdem ist es  $W[1]$ -schwer).



## Für welchen Parameter $k$ ?

- Beobachtung: der resultierende Graph hat Baumweite  $k$
- das Theorem gilt also für die Parametrisierung mittels Baumweite

Warum?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für GRID TILING gibt (außerdem ist es  $W[1]$ -schwer).

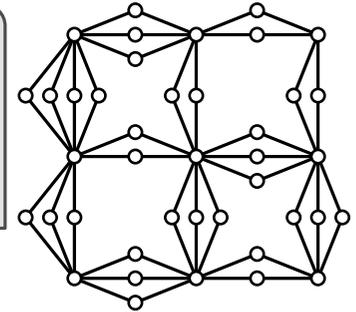
## Erinnerung

- $k$  war die Gittergröße

# Was zeigen wir hier eigentlich?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für PLANAR LIST COLORING gibt (außerdem ist es  $W[1]$ -schwer).



## Für welchen Parameter $k$ ?

- Beobachtung: der resultierende Graph hat Baumweite  $k$
- das Theorem gilt also für die Parametrisierung mittels Baumweite

Warum?

Kann man LIST COLORING nicht mittels DP auf einer Baumzerlegung lösen?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für GRID TILING gibt (außerdem ist es  $W[1]$ -schwer).

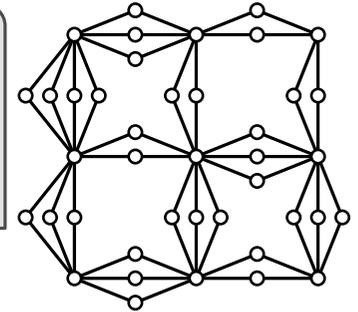
## Erinnerung

- $k$  war die Gittergröße

# Was zeigen wir hier eigentlich?

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für PLANAR LIST COLORING gibt (außerdem ist es  $W[1]$ -schwer).



## Für welchen Parameter $k$ ?

- Beobachtung: der resultierende Graph hat Baumweite  $k$
- das Theorem gilt also für die Parametrisierung mittels Baumweite

Warum?

Kann man LIST COLORING nicht mittels DP auf einer Baumzerlegung lösen?

**Antwort:** doch, aber es ist zu langsam

## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für GRID TILING gibt (außerdem ist es  $W[1]$ -schwer).

## Erinnerung

- $k$  war die Gittergröße

# Relaxiertes GRID TILING

**Problem: GRID TILING WITH  $\leq$**  ( $[n] = [1, n]$ )  
 Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass für jede Spalte bzw. Zeile die ersten bzw. zweiten Koordinaten nicht-absteigend sortiert sind.

## Beispiel

- $k = 3$  und  $n = 5$

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |

# Relaxiertes GRID TILING

**Problem: GRID TILING WITH  $\leq$**  ( $[n] = [1, n]$ )

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass für jede Spalte bzw. Zeile die ersten bzw. zweiten Koordinaten nicht-absteigend sortiert sind.

## Beispiel

- $k = 3$  und  $n = 5$

## Beachte

- die Instanzen von GRID TILING und GRID TILING WITH  $\leq$  sind gleich
- jede Lösung von GRID TILING löst auch GRID TILING WITH  $\leq$ , aber nicht umgekehrt

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

# Relaxiertes GRID TILING

## Problem: GRID TILING WITH $\leq$

$([n] = [1, n])$

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass für jede Spalte bzw. Zeile die ersten bzw. zweiten Koordinaten nicht-absteigend sortiert sind.

## Beispiel

- $k = 3$  und  $n = 5$

## Beachte

- die Instanzen von GRID TILING und GRID TILING WITH  $\leq$  sind gleich
- jede Lösung von GRID TILING löst auch GRID TILING WITH  $\leq$ , aber nicht umgekehrt
- das heißt aber nicht, dass GRID TILING WITH  $\leq$  leichter zu lösen ist

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

# Relaxiertes GRID TILING

## Problem: GRID TILING WITH $\leq$

$([n] = [1, n])$

Gegeben sei eine  $k \times k$  Matrix an Mengen  $S_{i,j}$ , wobei  $S_{i,j} \subseteq [n] \times [n]$ . Wähle für jede Zelle ein  $s_{i,j} \in S_{i,j}$ , sodass für jede Spalte bzw. Zeile die ersten bzw. zweiten Koordinaten nicht-absteigend sortiert sind.

## Beispiel

- $k = 3$  und  $n = 5$

## Beachte

- die Instanzen von GRID TILING und GRID TILING WITH  $\leq$  sind gleich
- jede Lösung von GRID TILING löst auch GRID TILING WITH  $\leq$ , aber nicht umgekehrt
- das heißt aber nicht, dass GRID TILING WITH  $\leq$  leichter zu lösen ist

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>$(1, 1)$<br>$(3, 1)$<br>$(2, 4)$ | $S_{1,2}$<br>$(5, 1)$<br>$(1, 4)$<br>$(5, 3)$ | $S_{1,3}$<br>$(1, 1)$<br>$(2, 4)$<br>$(3, 3)$ |
| $S_{2,1}$<br>$(2, 2)$<br>$(1, 4)$             | $S_{2,2}$<br>$(3, 1)$<br>$(1, 2)$             | $S_{2,3}$<br>$(2, 2)$<br>$(2, 3)$             |
| $S_{3,1}$<br>$(1, 3)$<br>$(2, 3)$<br>$(3, 3)$ | $S_{3,2}$<br>$(1, 1)$<br>$(1, 3)$             | $S_{3,3}$<br>$(2, 3)$<br>$(5, 3)$             |

## Theorem

(ohne Beweis)

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(k)}$ -Algorithmus für GRID TILING WITH  $\leq$  gibt (außerdem ist es  $W[1]$ -schwer).

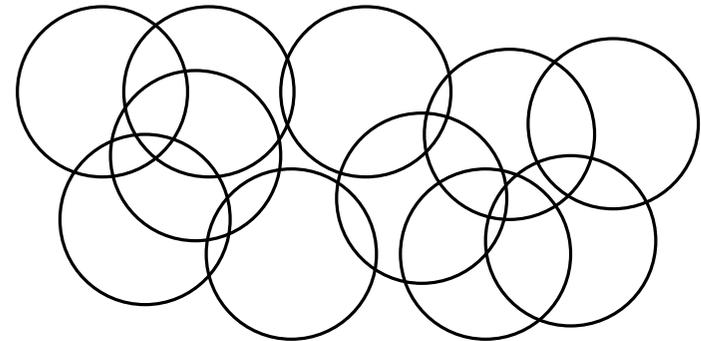
# Unabhängige Einheitskreise

## Problem: UNIT DISK INDEPENDENT SET

Gegeben sei eine Menge von Kreisen mit Radius 1 und ein Parameter  $k$ .  
Gibt es unter ihnen  $k$  disjunkte Kreise?

## Beispiel

- gibt es  $k = 6$  disjunkte Kreise?



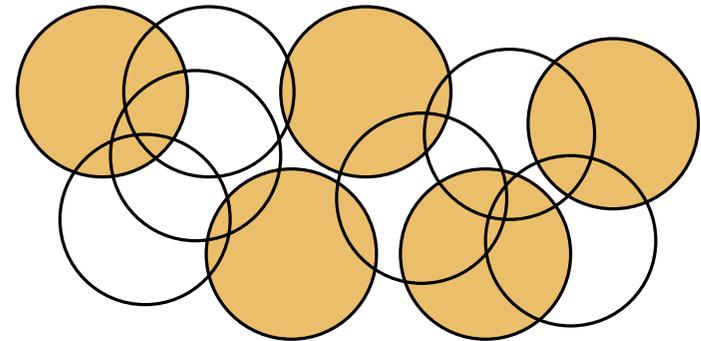
# Unabhängige Einheitskreise

## Problem: UNIT DISK INDEPENDENT SET

Gegeben sei eine Menge von Kreisen mit Radius 1 und ein Parameter  $k$ .  
Gibt es unter ihnen  $k$  disjunkte Kreise?

## Beispiel

- gibt es  $k = 6$  disjunkte Kreise?
- nein, 5 ist das Maximum



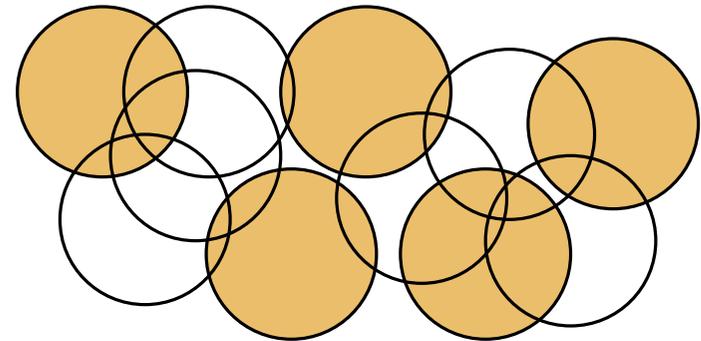
# Unabhängige Einheitskreise

## Problem: UNIT DISK INDEPENDENT SET

Gegeben sei eine Menge von Kreisen mit Radius 1 und ein Parameter  $k$ .  
Gibt es unter ihnen  $k$  disjunkte Kreise?

### Beispiel

- gibt es  $k = 6$  disjunkte Kreise?
- nein, 5 ist das Maximum



### Beachte

- äquivalent zu INDEPENDENT SET eingeschränkt auf „unit disk graphs“
- also potentiell echt leichter als INDEPENDENT SET

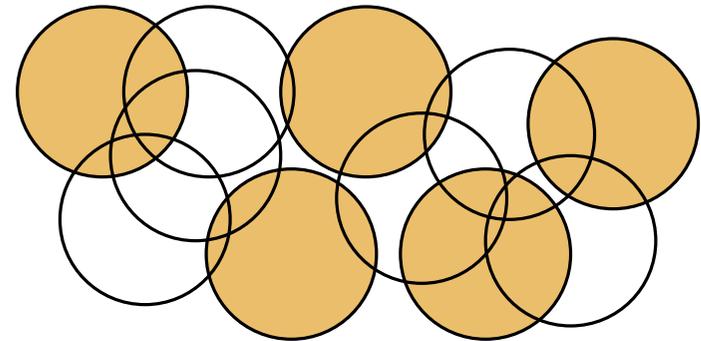
# Unabhängige Einheitskreise

## Problem: UNIT DISK INDEPENDENT SET

Gegeben sei eine Menge von Kreisen mit Radius 1 und ein Parameter  $k$ .  
Gibt es unter ihnen  $k$  disjunkte Kreise?

### Beispiel

- gibt es  $k = 6$  disjunkte Kreise?
- nein, 5 ist das Maximum



### Beachte

- äquivalent zu INDEPENDENT SET eingeschränkt auf „unit disk graphs“
- also potentiell echt leichter als INDEPENDENT SET
- tatsächlich: UNIT DISK INDEPENDENT SET kann in  $n^{O(\sqrt{k})}$  gelöst werden  
(untere Schranke für allgemeines IS (basierend auf ETH):  $n^{\Omega(k)}$ )

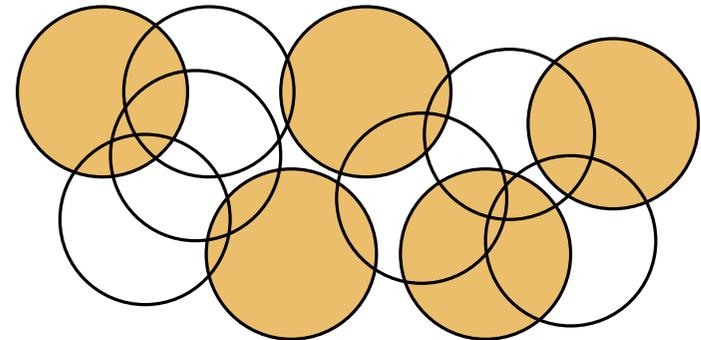
# Unabhängige Einheitskreise

## Problem: UNIT DISK INDEPENDENT SET

Gegeben sei eine Menge von Kreisen mit Radius 1 und ein Parameter  $k$ .  
Gibt es unter ihnen  $k$  disjunkte Kreise?

### Beispiel

- gibt es  $k = 6$  disjunkte Kreise?
- nein, 5 ist das Maximum



### Beachte

- äquivalent zu INDEPENDENT SET eingeschränkt auf „unit disk graphs“
- also potentiell echt leichter als INDEPENDENT SET
- tatsächlich: UNIT DISK INDEPENDENT SET kann in  $n^{O(\sqrt{k})}$  gelöst werden  
(untere Schranke für allgemeines IS (basierend auf ETH):  $n^{\Omega(k)}$ )

### Ziel im Folgenden

- zeige, dass es auch nicht besser geht (untere Schranke:  $n^{\Omega(\sqrt{k})}$ )
- reduziere von GRID TILING WITH  $\leq$

# Untere Schranke für UNIT DISK IS

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

**Plan:** erstelle Instanz von UNIT DISK INDEPENDENT SET, die Lösung der Größe  $k^2$  hat  $\Leftrightarrow$  GRID TILING WITH  $\leq$  hat Lösung (im Beispiel ist  $k = 3$ )

# Untere Schranke für UNIT DISK IS

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

**Plan:** erstelle Instanz von UNIT DISK INDEPENDENT SET, die Lösung der Größe  $k^2$  hat  $\Leftrightarrow$  GRID TILING WITH  $\leq$  hat Lösung (im Beispiel ist  $k = 3$ )

## Idee

- ein Kreis für jedes Paar

# Untere Schranke für UNIT DISK IS

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |

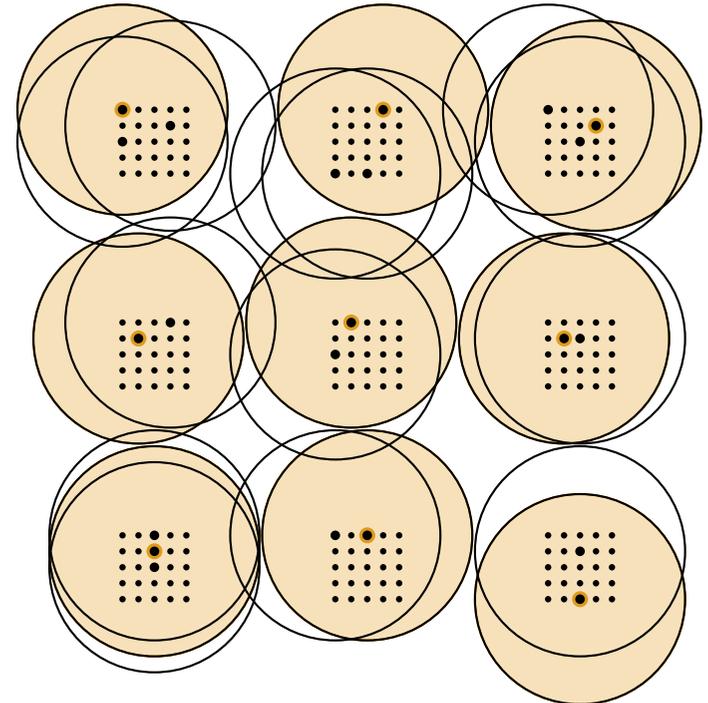
**Plan:** erstelle Instanz von UNIT DISK INDEPENDENT SET, die Lösung der Größe  $k^2$  hat  $\Leftrightarrow$  GRID TILING WITH  $\leq$  hat Lösung (im Beispiel ist  $k = 3$ )

## Idee

- ein Kreis für jedes Paar
- Kreise von Paaren aus der gleichen Zelle schneiden sich immer
- Kreise von Paaren aus benachbarten Zellen schneiden sich, wenn die entsprechende Koordinate kleiner wird

# Untere Schranke für UNIT DISK IS

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |



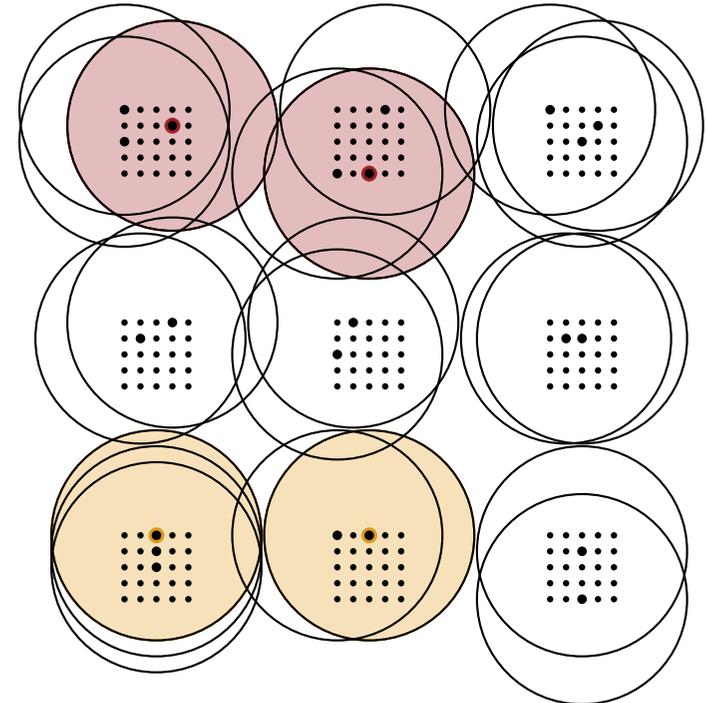
**Plan:** erstelle Instanz von UNIT DISK INDEPENDENT SET, die Lösung der Größe  $k^2$  hat  $\Leftrightarrow$  GRID TILING WITH  $\leq$  hat Lösung (im Beispiel ist  $k = 3$ )

## Idee

- ein Kreis für jedes Paar
- Kreise von Paaren aus der gleichen Zelle schneiden sich immer
- Kreise von Paaren aus benachbarten Zellen schneiden sich, wenn die entsprechende Koordinate kleiner wird

# Untere Schranke für UNIT DISK IS

|   |   |   |
|---|---|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br>(2, 4) | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br>(5, 3) | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)           | $S_{2,2}$<br>(3, 1)<br>(1, 2)           | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |
| $S_{3,1}$<br>(1, 3)<br>(2, 3)<br>(3, 3) | $S_{3,2}$<br>(1, 1)<br>(1, 3)           | $S_{3,3}$<br>(2, 3)<br>(5, 3)           |



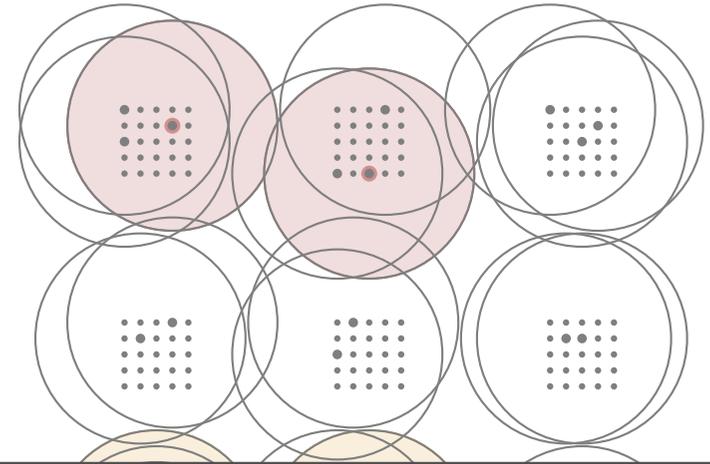
**Plan:** erstelle Instanz von UNIT DISK INDEPENDENT SET, die Lösung der Größe  $k^2$  hat  $\Leftrightarrow$  GRID TILING WITH  $\leq$  hat Lösung (im Beispiel ist  $k = 3$ )

## Idee

- ein Kreis für jedes Paar
- Kreise von Paaren aus der gleichen Zelle schneiden sich immer
- Kreise von Paaren aus benachbarten Zellen schneiden sich, wenn die entsprechende Koordinate kleiner wird

# Untere Schranke für UNIT DISK IS

|  |  |   |
|--|--|---|
| $S_{1,1}$<br>(1, 1)<br>(3, 1)<br><b>(2, 4)</b> | $S_{1,2}$<br>(5, 1)<br>(1, 4)<br><b>(5, 3)</b> | $S_{1,3}$<br>(1, 1)<br>(2, 4)<br>(3, 3) |
| $S_{2,1}$<br>(2, 2)<br>(1, 4)                  | $S_{2,2}$<br>(3, 1)<br>(1, 2)                  | $S_{2,3}$<br>(2, 2)<br>(2, 3)           |



## Theorem

ETH impliziert, dass es keinen  $f(k) \cdot n^{o(\sqrt{k})}$ -Algorithmus für UNIT DISK INDEPENDENT SET gibt (außerdem ist es  $W[1]$ -schwer).

**Plan:** erstelle Instanz von UNIT DISK INDEPENDENT SET, die Lösung der Größe  $k^2$  hat  $\Leftrightarrow$  GRID TILING WITH  $\leq$  hat Lösung (im Beispiel ist  $k = 3$ )

## Idee

- ein Kreis für jedes Paar
- Kreise von Paaren aus der gleichen Zelle schneiden sich immer
- Kreise von Paaren aus benachbarten Zellen schneiden sich, wenn die entsprechende Koordinate kleiner wird

# Zusammenfassung & Ausblick

## Untere Schranken

- Überraschung: stärkere Annahme (ETH statt  $FPT \neq W[1]$ ) liefern stärkere untere Schranken ( $f(k)n^{\Omega(k)}$  statt  $f(k)n^{\Omega(1)}$ )

# Zusammenfassung & Ausblick

## Untere Schranken

- Überraschung: stärkere Annahme (ETH statt  $FPT \neq W[1]$ ) liefern stärkere untere Schranken ( $f(k)n^{\Omega(k)}$  statt  $f(k)n^{\Omega(1)}$ )
- Übertragung von Schranken der Form  $2^{\Omega(n)}$  (für 3-COLORING) auf parametrisierte Schranken der Form  $f(k)n^{\Omega(k)}$  (für CLIQUE)

# Zusammenfassung & Ausblick

## Untere Schranken

- Überraschung: stärkere Annahme (ETH statt  $FPT \neq W[1]$ ) liefern stärkere untere Schranken ( $f(k)n^{\Omega(k)}$  statt  $f(k)n^{\Omega(1)}$ )
- Übertragung von Schranken der Form  $2^{\Omega(n)}$  (für 3-COLORING) auf parametrisierte Schranken der Form  $f(k)n^{\Omega(k)}$  (für CLIQUE)
- liefert feinere Unterteilung von  $W[1]$

# Zusammenfassung & Ausblick

## Untere Schranken

- Überraschung: stärkere Annahme (ETH statt  $FPT \neq W[1]$ ) liefern stärkere untere Schranken ( $f(k)n^{\Omega(k)}$  statt  $f(k)n^{\Omega(1)}$ )
- Übertragung von Schranken der Form  $2^{\Omega(n)}$  (für 3-COLORING) auf parametrisierte Schranken der Form  $f(k)n^{\Omega(k)}$  (für CLIQUE)
- liefert feinere Unterteilung von  $W[1]$

## GRID TILING

- eignet sich gut als Startproblem für Reduktionen
- vor allem dann, wenn das Zielproblem geometrischer Natur

# Zusammenfassung & Ausblick

## Untere Schranken

- Überraschung: stärkere Annahme (ETH statt  $FPT \neq W[1]$ ) liefern stärkere untere Schranken ( $f(k)n^{\Omega(k)}$  statt  $f(k)n^{\Omega(1)}$ )
- Übertragung von Schranken der Form  $2^{\Omega(n)}$  (für 3-COLORING) auf parametrisierte Schranken der Form  $f(k)n^{\Omega(k)}$  (für CLIQUE)
- liefert feinere Unterteilung von  $W[1]$

## GRID TILING

- eignet sich gut als Startproblem für Reduktionen
- vor allem dann, wenn das Zielproblem geometrischer Natur

## Feinere Unterteilung von FPT

- Erinnerung: Problem ist FPT  $\Leftrightarrow$  es gibt Kernbildungsalgo
- untere Schranken für die Kerngröße unterteilen die Klasse FPT

# Zusammenfassung & Ausblick

## Untere Schranken

- Überraschung: stärkere Annahme (ETH statt  $FPT \neq W[1]$ ) liefern stärkere untere Schranken ( $f(k)n^{\Omega(k)}$  statt  $f(k)n^{\Omega(1)}$ )
- Übertragung von Schranken der Form  $2^{\Omega(n)}$  (für 3-COLORING) auf parametrisierte Schranken der Form  $f(k)n^{\Omega(k)}$  (für CLIQUE)
- liefert feinere Unterteilung von  $W[1]$

## GRID TILING

- eignet sich gut als Startproblem für Reduktionen
- vor allem dann, wenn das Zielproblem geometrischer Natur

## Feinere Unterteilung von FPT

- Erinnerung: Problem ist FPT  $\Leftrightarrow$  es gibt Kernbildungsalgo
- untere Schranken für die Kerngröße unterteilen die Klasse FPT
- Beispiel: polynomieller Kernbildungsalgo für VERTEX COVER mit Kerngröße  $O(k^{2-\varepsilon}) \Rightarrow NP \subseteq \text{coNP}/\text{poly}$

# Zusammenfassung & Ausblick

## Untere Schranken

- Überraschung: stärkere Annahme (ETH statt  $FPT \neq W[1]$ ) liefern stärkere untere Schranken ( $f(k)n^{\Omega(k)}$  statt  $f(k)n^{\Omega(1)}$ )
- Übertragung von Schranken der Form  $2^{\Omega(n)}$  (für 3-COLORING) auf parametrisierte Schranken der Form  $f(k)n^{\Omega(k)}$  (für CLIQUE)
- liefert feinere Unterteilung von  $W[1]$

## GRID TILING

- eignet sich gut als Startproblem für Reduktionen
- vor allem dann, wenn das Zielproblem geometrischer Natur

## Feinere Unterteilung von FPT

- Erinnerung: Problem ist FPT  $\Leftrightarrow$  es gibt Kernbildungsalgo
- untere Schranken für die Kerngröße unterteilen die Klasse FPT
- Beispiel: polynomieller Kernbildungsalgo für VERTEX COVER mit Kerngröße  $O(k^{2-\epsilon}) \Rightarrow NP \subseteq coNP/poly$

## Untere Schranken in P

- Beispiel: SETH  $\Rightarrow$  kein subquadratischer Algo für ORTHOGONAL VECTORS
- Stichwort: Fine-Grained Complexity Theory