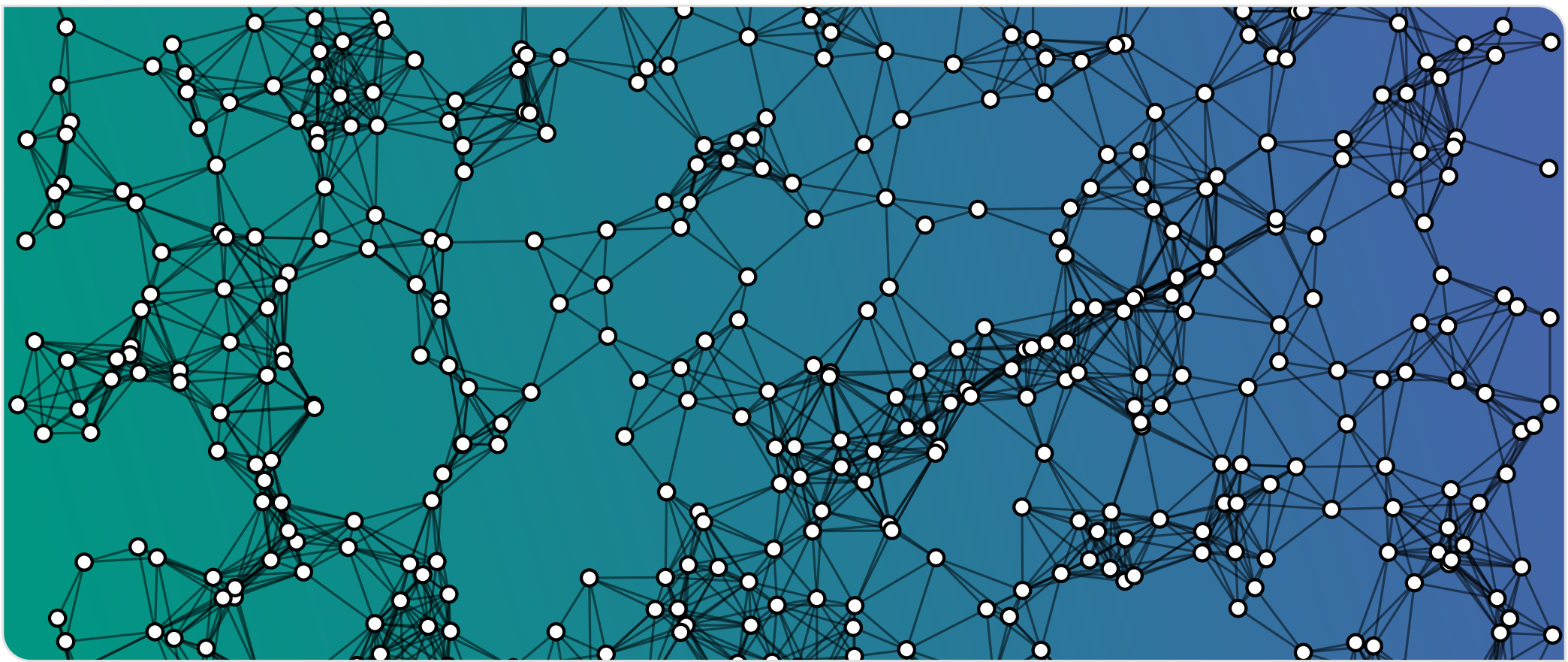
Karlsruher Institut für Technologie

Parametrisierte Algorithmen

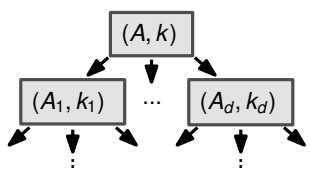
Untere Schranken: ETH und SETH

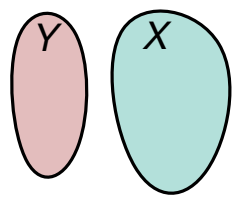


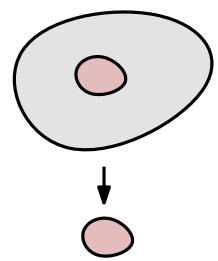
Inhalt

Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung

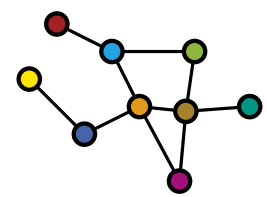


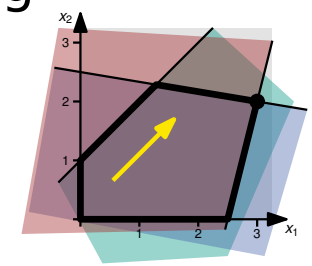


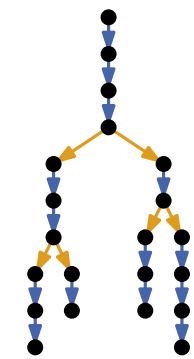


Erweiterte Toolbox

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





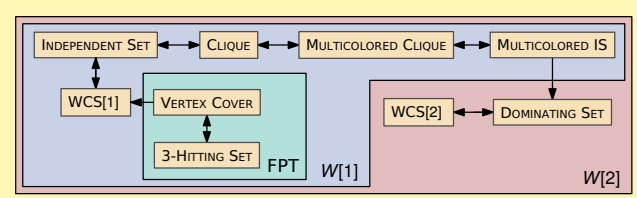


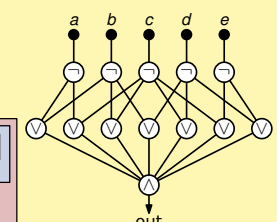
Baumweite

- 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 δ_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 δ_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 δ_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-SAT $\in$ P und 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 δ_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-SAT $\in$ P und 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 δ_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-SAT $\in$ P und 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 δ_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-SAT $\in$ P und 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 δ_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-SAT $\in$ P und 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 δ_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-SAT $\in$ P und 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 $\lim_{q\to\infty} \delta_q = 1$.

(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 δ_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-SAT $\in$ P und 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 $\lim_{q\to\infty} \delta_q = 1$.

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.

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

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

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 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 $\neq W[1]$

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

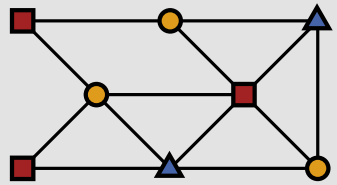
Startproblem: 3-Coloring

- hat keine „Ausgabegröße“ $\rightarrow$ wir geraten nicht in Versuchung es parametrisiert zu betrachten
- lokaler 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.

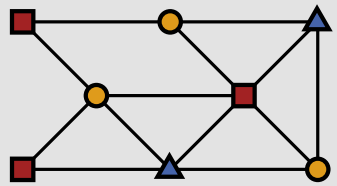
Auswahl geeigneter Teillösungen

Problem: 3-Coloring
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



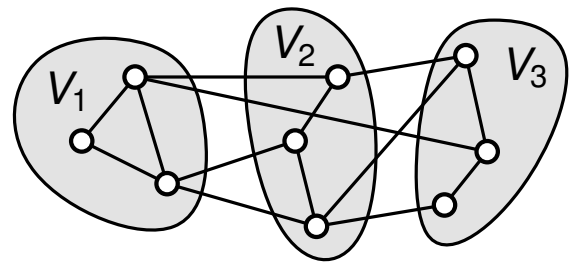
Auswahl geeigneter Teillösungen

Problem: 3-Coloring
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



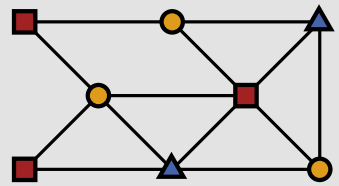
Teillösungen

- partitioniere $V = V_1 \dot\cup \cdots \dot\cup V_k$



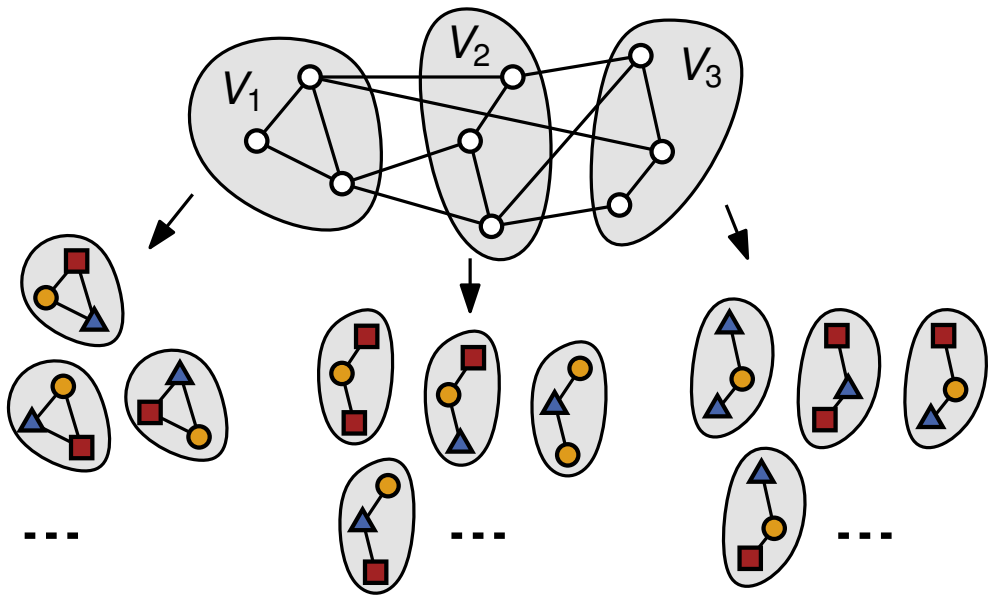
Auswahl geeigneter Teillösungen

Problem: 3-Coloring
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



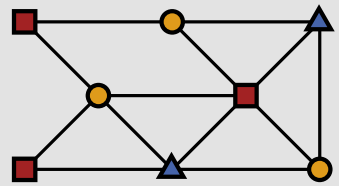
Teillösungen

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



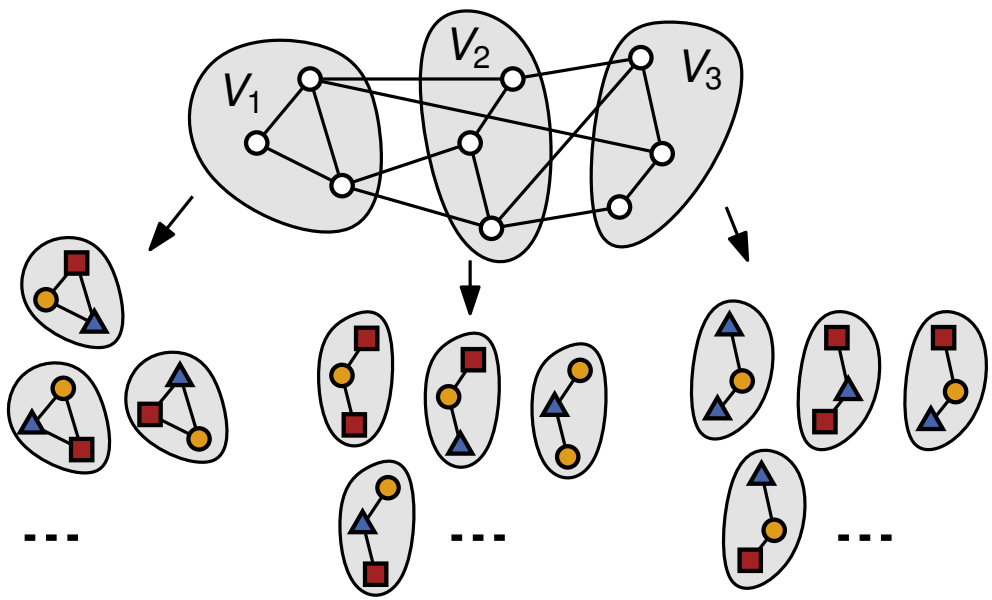
Auswahl geeigneter Teillösungen

Problem: 3-Coloring
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



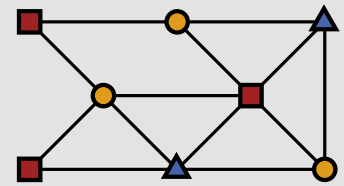
Teillösungen

- partitioniere $V = V_1 \dot\cup \cdots \dot\cup V_k$
- 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



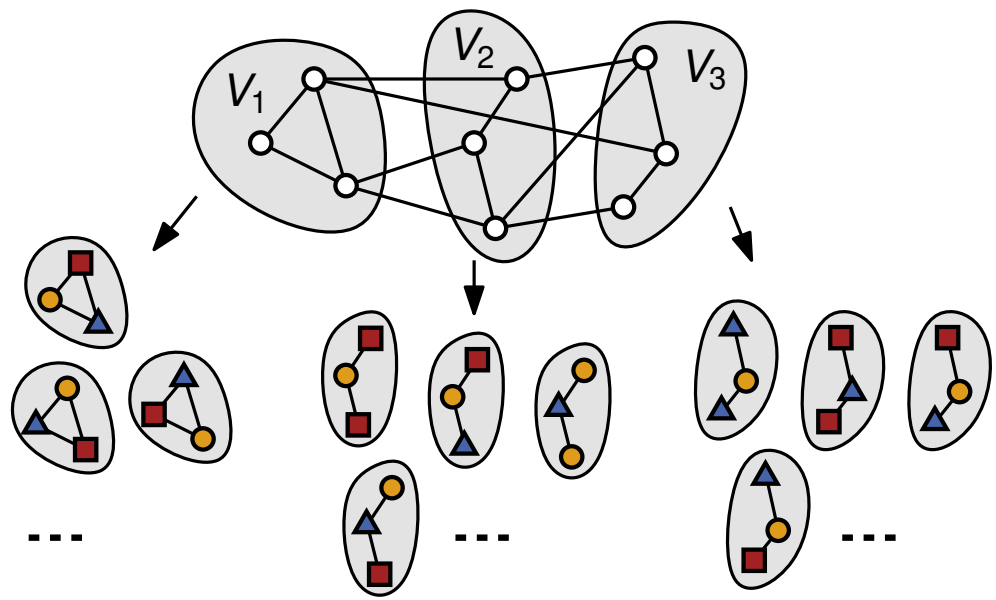
Auswahl geeigneter Teillösungen

Problem: 3-Coloring
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



Teillösungen

- partitioniere $V = V_1 \dot\cup \cdots \dot\cup V_k$
- 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

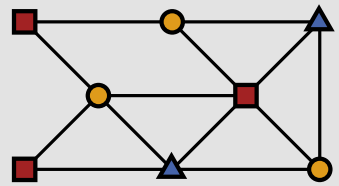


Interpretation

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

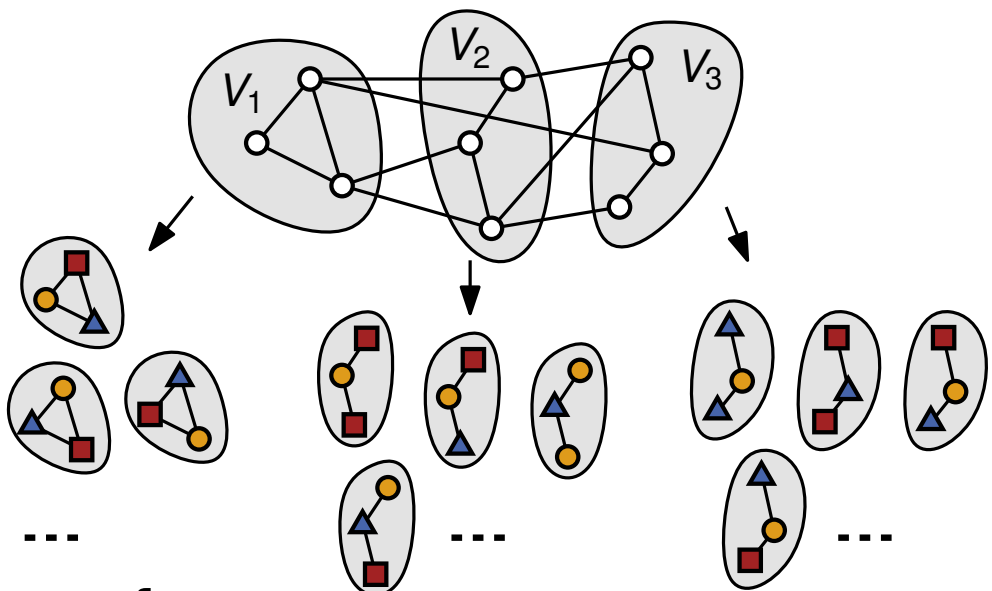
Auswahl geeigneter Teillösungen

Problem: 3-Coloring
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



Teillösungen

- partitioniere $V = V_1 \dot\cup \cdots \dot\cup V_k$
- 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

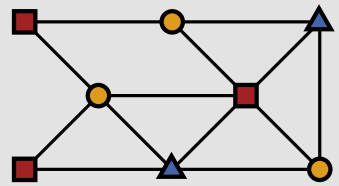


Andere Sichtweise

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

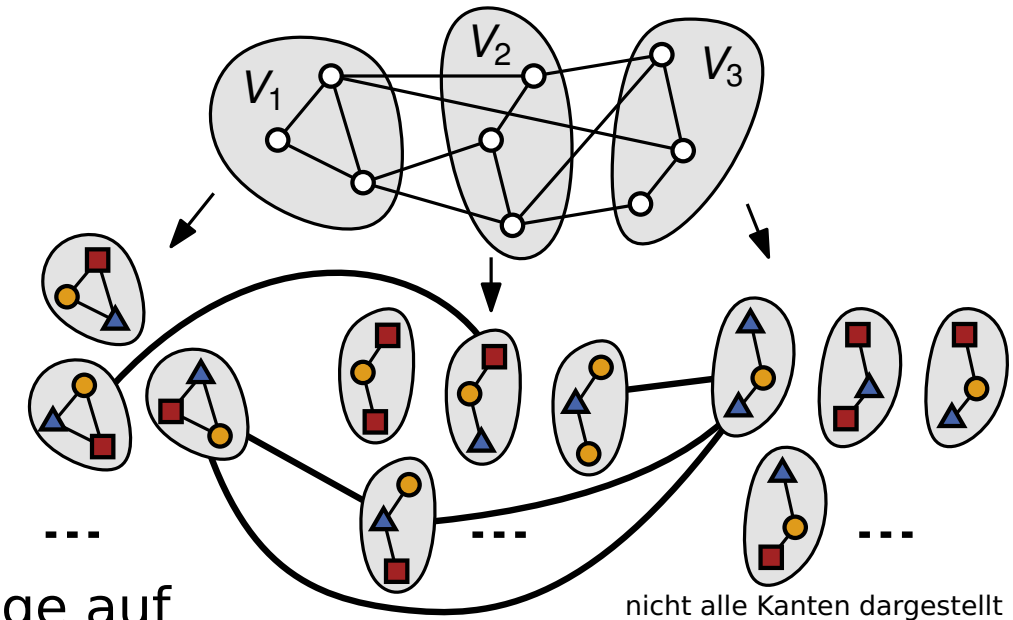
Auswahl geeigneter Teillösungen

Problem: 3-Coloring
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



Teillösungen

- partitioniere $V = V_1 \dot\cup \cdots \dot\cup V_k$
- 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



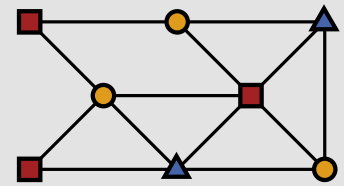
nicht alle Kanten dargestellt

Andere Sichtweise

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

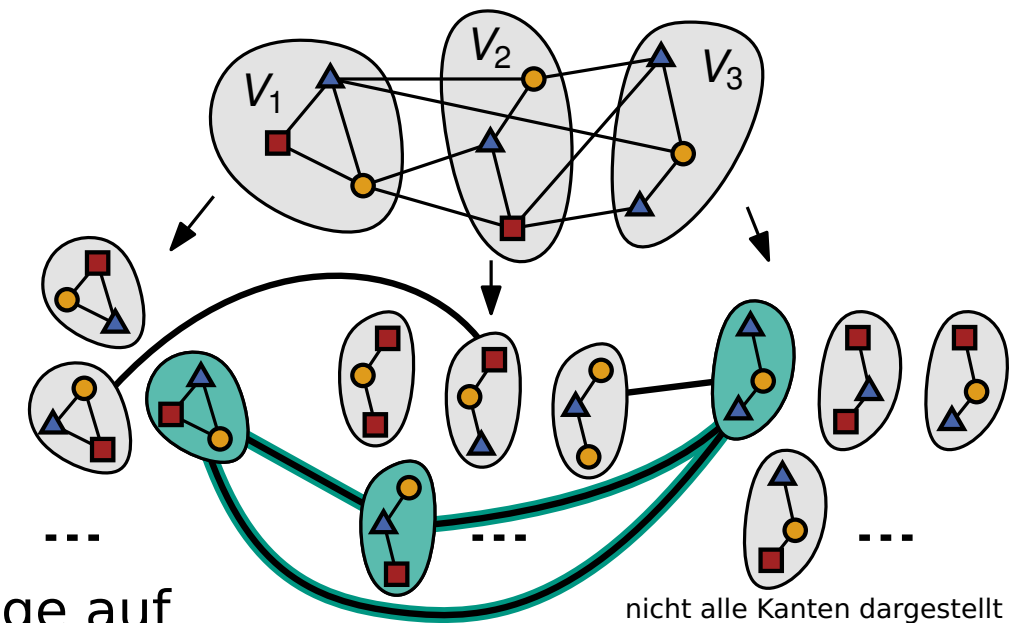
Auswahl geeigneter Teillösungen

Problem: 3-Coloring
Färbe die Knoten eines Graphen mit drei Farben, sodass benachbarte Knoten unterschiedliche Farben haben.



Teillösungen

- partitioniere $V = V_1 \dot\cup \cdots \dot\cup V_k$
- 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 $C_1 \cup \cdots \cup C_k$ 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: $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$

  $f(n)$: potentiell zu groß; $3^{\frac{n}{n} \cdot o(n)}$: 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: $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$
  - $f(n)$: potentiell zu groß
  - $3^{\frac{n}{n} \cdot o(n)}$: 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: $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$

  $f(n)$: potentiell zu groß; $3^{\frac{n}{n} \cdot o(n)}$: klein genug

- zweiter Versuch: k konstant

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

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

  $f(n)$: potentiell zu groß; $3^{\frac{n}{n} \cdot o(n)}$: klein genug

- zweiter Versuch: k konstant

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

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

($o(k)$ 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: $k = n \rightarrow f(n) \cdot 3^{\frac{n}{n} \cdot o(n)}$
  - $f(n)$: potentiell zu groß
  - $3^{\frac{n}{n} \cdot o(n)}$: 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 $n = 5$

$S_{1,1}$ $(1,1)$ $(3,1)$ $(2,4)$	$S_{1,2}$ $(5,1)$ $(1,4)$ $(5,3)$	$S_{1,3}$ $(1,1)$ $(2,4)$ $(3,3)$
$S_{2,1}$ $(2,2)$ $(1,4)$	$S_{2,2}$ $(3,1)$ $(1,2)$	$S_{2,3}$ $(2,2)$ $(2,3)$
$S_{3,1}$ $(1,3)$ $(2,3)$ $(3,3)$	$S_{3,2}$ $(1,1)$ $(1,3)$	$S_{3,3}$ $(2,3)$ $(5,3)$

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.

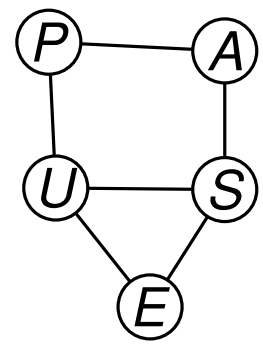
Beispiel

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

$S_{1,1}$ (1, 1) (3, 1) **(2, 4)**	$S_{1,2}$ (5, 1) **(1, 4)** (5, 3)	$S_{1,3}$ (1, 1) **(2, 4)** (3, 3)
$S_{2,1}$ **(2, 2)** (1, 4)	$S_{2,2}$ (3, 1) **(1, 2)**	$S_{2,3}$ **(2, 2)** (2, 3)
$S_{3,1}$ (1, 3) **(2, 3)** (3, 3)	$S_{3,2}$ (1, 1) **(1, 3)**	$S_{3,3}$ **(2, 3)** (5, 3)

Was ist hier passiert?

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



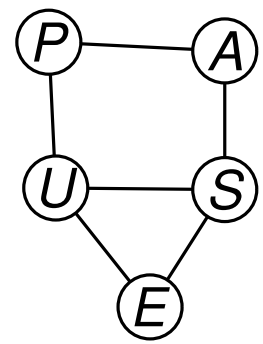
(P,P) (A,A) (U,U) (S,S) (E,E)	(A,P) (U,P) (S,A) (S,U) (E,U) (E,S)	(A,P) (U,P) (S,A) (S,U) (E,U) (E,S)
(P,A) (P,U) (A,S) (U,S) (U,E) (S,E)	(P,P) (A,A) (U,U) (S,S) (E,E)	(A,P) (U,P) (S,A) (S,U) (E,U) (E,S)
(P,A) (P,U) (A,S) (U,S) (U,E) (S,E)	(P,A) (P,U) (A,S) (U,S) (U,E) (S,E)	(P,P) (A,A) (U,U) (S,S) (E,E)

$S_{1,1}$ $(1,1)$ $(3,1)$ $(2,4)$	$S_{1,2}$ $(5,1)$ $(1,4)$ $(5,3)$	$S_{1,3}$ $(1,1)$ $(2,4)$ $(3,3)$
$S_{2,1}$ $(2,2)$ $(1,4)$	$S_{2,2}$ $(3,1)$ $(1,2)$	$S_{2,3}$ $(2,2)$ $(2,3)$
$S_{3,1}$ $(1,3)$ $(2,3)$ $(3,3)$	$S_{3,2}$ $(1,1)$ $(1,3)$	$S_{3,3}$ $(2,3)$ $(5,3)$

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) (U,U) (S,S) (E,E)	(A,P) (U,P) (S,A) (S,U) (E,U) (E,S)	(A,P) (U,P) (S,A) (S,U) (E,U) (E,S)
(P,A) (P,U) (A,S) (U,S) (U,E) (S,E)	(P,P) (A,A) (U,U) (S,S) (E,E)	(A,P) (U,P) (S,A) (S,U) (E,U) (E,S)
(P,A) (P,U) (A,S) (U,S) (U,E) (S,E)	(P,A) (P,U) (A,S) (U,S) (U,E) (S,E)	(P,P) (A,A) (U,U) (S,S) (E,E)

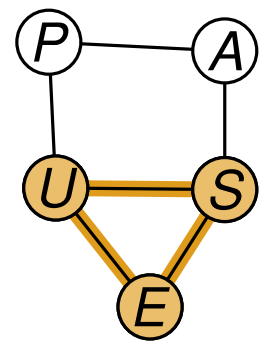
$S_{1,1}$ $(1,1)$ $(3,1)$ $(2,4)$	$S_{1,2}$ $(5,1)$ $(1,4)$ $(5,3)$	$S_{1,3}$ $(1,1)$ $(2,4)$ $(3,3)$
$S_{2,1}$ $(2,2)$ $(1,4)$	$S_{2,2}$ $(3,1)$ $(1,2)$	$S_{2,3}$ $(2,2)$ $(2,3)$
$S_{3,1}$ $(1,3)$ $(2,3)$ $(3,3)$	$S_{3,2}$ $(1,1)$ $(1,3)$	$S_{3,3}$ $(2,3)$ $(5,3)$

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) **(U,U)** (S,S) (E,E)	(A,P) (U,P) (S,A) **(S,U)** (E,U) (E,S)	(A,P) (U,P) (S,A) (S,U) **(E,U)** (E,S)
(P,A) (P,U) (A,S) **(U,S)** (U,E) (S,E)	(P,P) (A,A) (U,U) **(S,S)** (E,E)	(A,P) (U,P) (S,A) (S,U) (E,U) **(E,S)**
(P,A) (P,U) (A,S) (U,S) **(U,E)** (S,E)	(P,A) (P,U) (A,S) (U,S) (U,E) **(S,E)**	(P,P) (A,A) (U,U) (S,S) **(E,E)**

$S_{1,1}$ $(1,1)$ $(3,1)$ **$(2,4)$**	$S_{1,2}$ $(5,1)$ **$(1,4)$** $(5,3)$	$S_{1,3}$ $(1,1)$ **$(2,4)$** $(3,3)$
$S_{2,1}$ **$(2,2)$** $(1,4)$	$S_{2,2}$ $(3,1)$ **$(1,2)$**	$S_{2,3}$ **$(2,2)$** $(2,3)$
$S_{3,1}$ $(1,3)$ **$(2,3)$** $(3,3)$	$S_{3,2}$ $(1,1)$ **$(1,3)$**	$S_{3,3}$ **$(2,3)$** $(5,3)$

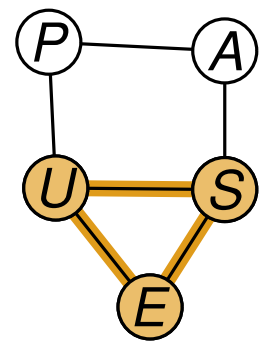
Problem: GRID TILING
Gegeben sei eine $k \times k$ Matrix an Mengen $S_{i,j}$, wobei $S_{i,j} \subseteq [n] \times [n]$. Wähle für jede Zelle ein $s_{i,j} \in S_{i,j}$, sodass $s_{i,j}$ und $s_{i+1,j}$ bzw. $s_{i,j}$ und $s_{i,j+1}$ sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

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) **(U,U)** (S,S) (E,E)	(A,P) (U,P) (S,A) **(S,U)** (E,U) (E,S)	(A,P) (U,P) (S,A) (S,U) **(E,U)** (E,S)
(P,A) (P,U) (A,S) **(U,S)** (U,E) (S,E)	(P,P) (A,A) (U,U) **(S,S)** (E,E)	(A,P) (U,P) (S,A) (S,U) (E,U) **(E,S)**
(P,A) (P,U) (A,S) (U,S) **(U,E)** (S,E)	(P,A) (P,U) (A,S) (U,S) (U,E) **(S,E)**	(P,P) (A,A) (U,U) (S,S) **(E,E)**

$S_{1,1}$ $(1,1)$ $(3,1)$ **$(2,4)$**	$S_{1,2}$ $(5,1)$ **$(1,4)$** $(5,3)$	$S_{1,3}$ $(1,1)$ **$(2,4)$** $(3,3)$
$S_{2,1}$ **$(2,2)$** $(1,4)$	$S_{2,2}$ $(3,1)$ **$(1,2)$**	$S_{2,3}$ **$(2,2)$** $(2,3)$
$S_{3,1}$ $(1,3)$ **$(2,3)$** $(3,3)$	$S_{3,2}$ $(1,1)$ **$(1,3)$**	$S_{3,3}$ **$(2,3)$** $(5,3)$

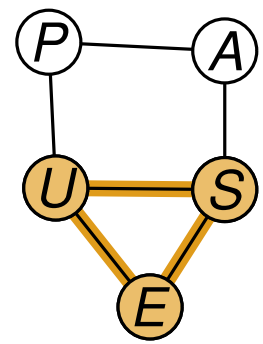
Problem: Grid Tiling
Gegeben sei eine $k \times k$ Matrix an Mengen $S_{i,j}$, wobei $S_{i,j} \subseteq [n] \times [n]$. Wähle für jede Zelle ein $s_{i,j} \in S_{i,j}$, sodass $s_{i,j}$ und $s_{i+1,j}$ bzw. $s_{i,j}$ und $s_{i,j+1}$ sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

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 = #Knoten, k = Cliquengröße
- wir haben also eine parametrisierte Reduktion von Clique zu Grid Tiling
- Parameter k überträgt sich linear

(P,P) (A,A) **(U,U)** (S,S) (E,E)	(A,P) (U,P) (S,A) **(S,U)** (E,U) (E,S)	(A,P) (U,P) (S,A) (S,U) **(E,U)** (E,S)
(P,A) (P,U) (A,S) **(U,S)** (U,E) (S,E)	(P,P) (A,A) (U,U) **(S,S)** (E,E)	(A,P) (U,P) (S,A) (S,U) (E,U) **(E,S)**
(P,A) (P,U) (A,S) (U,S) **(U,E)** (S,E)	(P,A) (P,U) (A,S) (U,S) (U,E) **(S,E)**	(P,P) (A,A) (U,U) (S,S) **(E,E)**

$S_{1,1}$ $(1,1)$ $(3,1)$ **$(2,4)$**	$S_{1,2}$ $(5,1)$ **$(1,4)$** $(5,3)$	$S_{1,3}$ $(1,1)$ **$(2,4)$** $(3,3)$
$S_{2,1}$ **$(2,2)$** $(1,4)$	$S_{2,2}$ $(3,1)$ **$(1,2)$**	$S_{2,3}$ **$(2,2)$** $(2,3)$
$S_{3,1}$ $(1,3)$ **$(2,3)$** $(3,3)$	$S_{3,2}$ $(1,1)$ **$(1,3)$**	$S_{3,3}$ **$(2,3)$** $(5,3)$

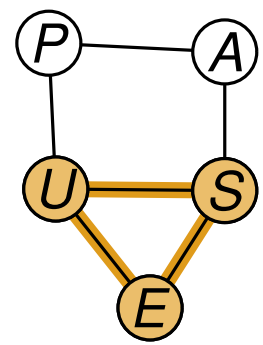
Problem: Grid Tiling
Gegeben sei eine $k \times k$ Matrix an Mengen $S_{i,j}$, wobei $S_{i,j} \subseteq [n] \times [n]$. Wähle für jede Zelle ein $s_{i,j} \in S_{i,j}$, sodass $s_{i,j}$ und $s_{i+1,j}$ bzw. $s_{i,j}$ und $s_{i,j+1}$ sich in der ersten bzw. in der zweiten Koordinate nicht unterscheiden.

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 = #Knoten, k = Cliquengröße
- wir haben also eine parametrisierte Reduktion von Clique zu Grid Tiling
- Parameter k überträgt sich linear

(P,P) (A,A) **(U,U)** (S,S) (E,E)	(A,P) (U,P) (S,A) **(S,U)** (E,U) (E,S)	(A,P) (U,P) (S,A) (S,U) **(E,U)** (E,S)
(P,A) (P,U) (A,S) **(U,S)** (U,E) (S,E)	(P,P) (A,A) (U,U) **(S,S)** (E,E)	(A,P) (U,P) (S,A) (S,U) (E,U) **(E,S)**
(P,A) (P,U) (A,S) (U,S) **(U,E)** (S,E)	(P,A) (P,U) (A,S) (U,S) (U,E) **(S,E)**	(P,P) (A,A) (U,U) (S,S) **(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.

$S_{1,1}$ $(1,1)$ $(3,1)$ **$(2,4)$**	$S_{1,2}$ $(5,1)$ **$(1,4)$** $(5,3)$	$S_{1,3}$ $(1,1)$ **$(2,4)$** $(3,3)$
$S_{2,1}$ **$(2,2)$** $(1,4)$	$S_{2,2}$ $(3,1)$ **$(1,2)$**	$S_{2,3}$ **$(2,2)$** $(2,3)$
$S_{3,1}$ $(1,3)$ **$(2,3)$** $(3,3)$	$S_{3,2}$ $(1,1)$ **$(1,3)$**	$S_{3,3}$ **$(2,3)$** $(5,3)$

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

$\Rightarrow$ Grid Tiling eignet sich gut um Schwere auf planaren Graphen zu zeigen

Anmerkungen zu Grid Tiling

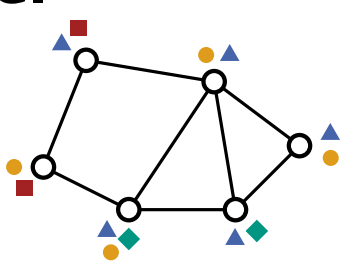
Reduktionen von Grid Tiling

- 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

$\Rightarrow$ 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

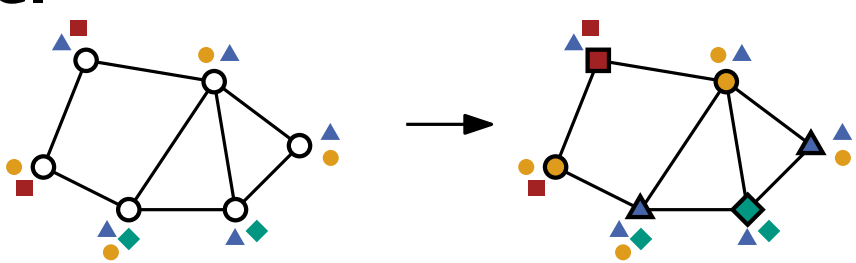
Reduktionen von Grid Tiling

- 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

$\Rightarrow$ 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

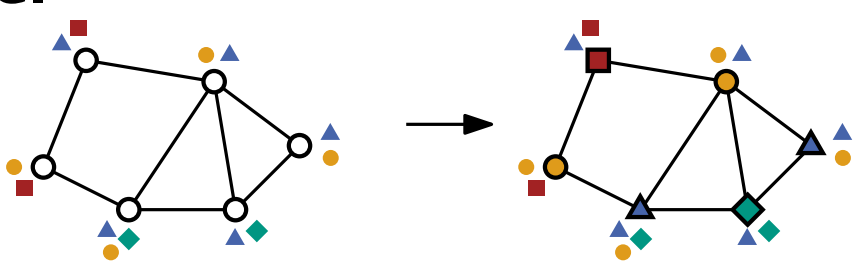
Reduktionen von Grid Tiling

- 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

$\Rightarrow$ 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

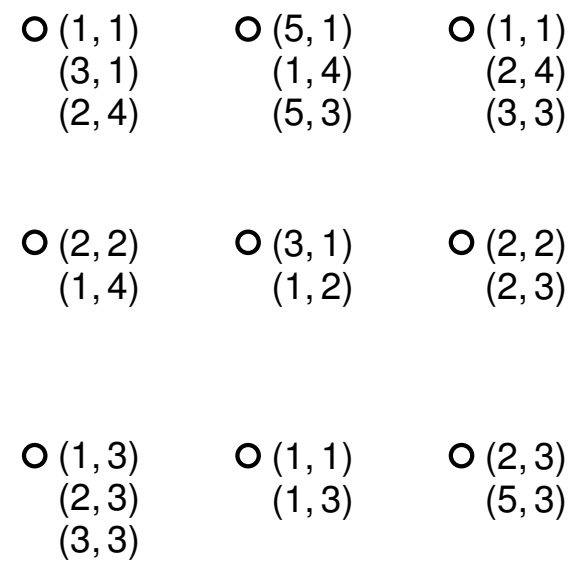
Grid Tiling $\rightarrow$ Planar List Coloring

$S_{1,1}$ $(1,1)$ $(3,1)$ $(2,4)$	$S_{1,2}$ $(5,1)$ $(1,4)$ $(5,3)$	$S_{1,3}$ $(1,1)$ $(2,4)$ $(3,3)$
$S_{2,1}$ $(2,2)$ $(1,4)$	$S_{2,2}$ $(3,1)$ $(1,2)$	$S_{2,3}$ $(2,2)$ $(2,3)$
$S_{3,1}$ $(1,3)$ $(2,3)$ $(3,3)$	$S_{3,2}$ $(1,1)$ $(1,3)$	$S_{3,3}$ $(2,3)$ $(5,3)$

Wahl der Knoten und Farben

Grid Tiling $\rightarrow$ Planar List Coloring

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (2,3)
$S_{3,1}$ (1,3) (2,3) (3,3)	$S_{3,2}$ (1,1) (1,3)	$S_{3,3}$ (2,3) (5,3)

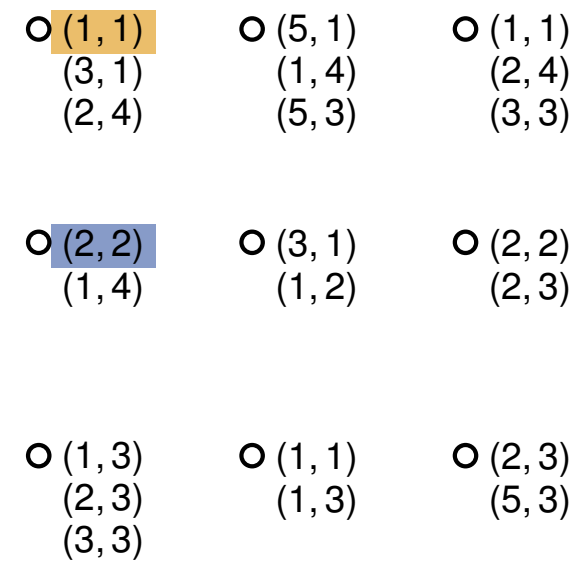


Wahl der Knoten und Farben

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

Grid Tiling $\rightarrow$ Planar List Coloring

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (2,3)
$S_{3,1}$ (1,3) (2,3) (3,3)	$S_{3,2}$ (1,1) (1,3)	$S_{3,3}$ (2,3) (5,3)



Wahl der Knoten und Farben

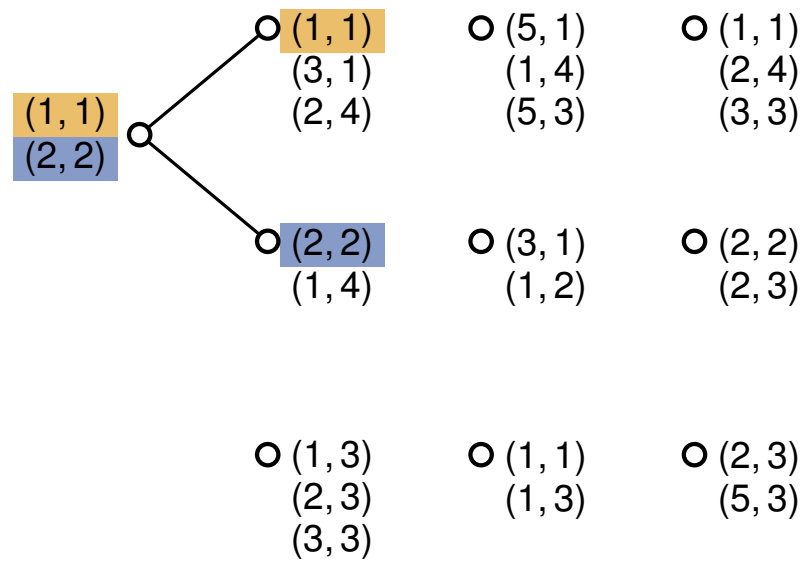
- naheliegender Versuch: ein Knoten pro Zelle, eine Farbe pro Paar

Verbiete gewisse Farbwahlen

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

Grid Tiling → Planar List Coloring

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (2,3)
$S_{3,1}$ (1,3) (2,3) (3,3)	$S_{3,2}$ (1,1) (1,3)	$S_{3,3}$ (2,3) (5,3)



Wahl der Knoten und Farben

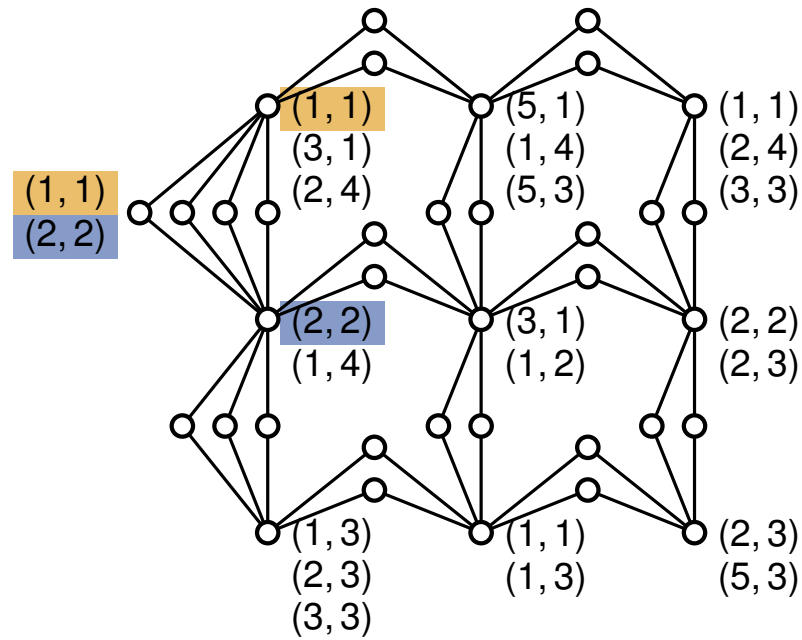
- naheliegender Versuch: ein Knoten pro Zelle, eine Farbe pro Paar

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)

Grid Tiling → Planar List Coloring

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (2,3)
$S_{3,1}$ (1,3) (2,3) (3,3)	$S_{3,2}$ (1,1) (1,3)	$S_{3,3}$ (2,3) (5,3)



Wahl der Knoten und Farben

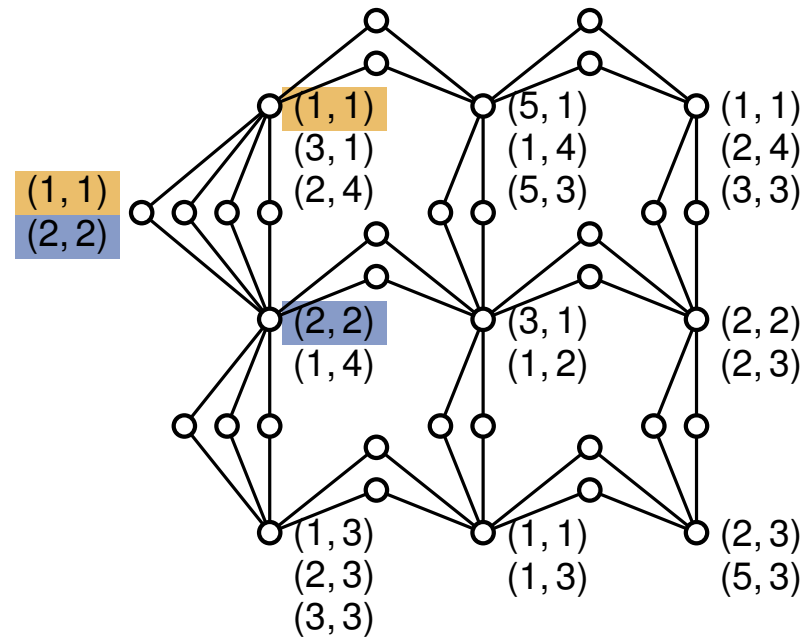
- naheliegender Versuch: ein Knoten pro Zelle, eine Farbe pro Paar

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 → Planar List Coloring

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (2,3)
$S_{3,1}$ (1,3) (2,3) (3,3)	$S_{3,2}$ (1,1) (1,3)	$S_{3,3}$ (2,3) (5,3)



Wahl der Knoten und Farben

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

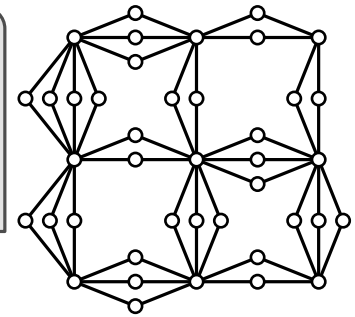
Verbiete gewisse Farbwahlen

- 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?

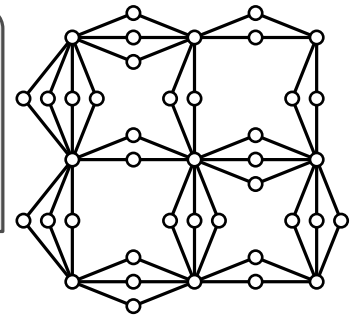
Theorem
ETH impliziert, dass es keinen $f(k) \cdot n^{o(k)}$-Algorithmus für Planar List Coloring gibt (außerdem ist es $W[1]$-schwer).



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?

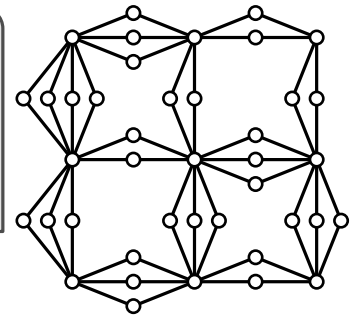
Theorem
ETH impliziert, dass es keinen $f(k) \cdot n^{o(k)}$-Algorithmus für Grid Tiling gibt (außerdem ist es $W[1]$-schwer).

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).



Für welchen Parameter k?

- Beobachtung: der resultierende Graph hat Baumweite k

Warum?

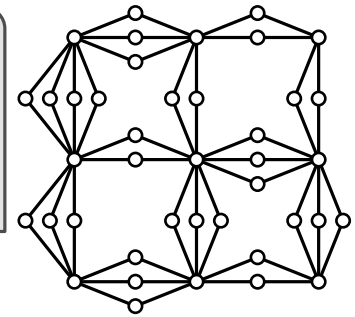
Theorem
ETH impliziert, dass es keinen $f(k) \cdot n^{o(k)}$-Algorithmus für Grid Tiling gibt (außerdem ist es $W[1]$-schwer).

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?

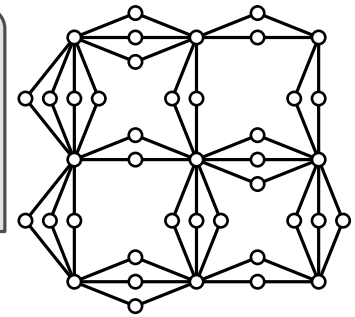
Theorem
ETH impliziert, dass es keinen $f(k) \cdot n^{o(k)}$-Algorithmus für Grid Tiling gibt (außerdem ist es $W[1]$-schwer).

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?

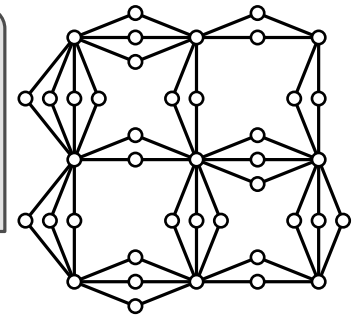
Theorem
ETH impliziert, dass es keinen $f(k) \cdot n^{o(k)}$-Algorithmus für Grid Tiling gibt (außerdem ist es $W[1]$-schwer).

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 $n = 5$

$S_{1,1}$ **(1, 1)** (3, 1) (2, 4)	$S_{1,2}$ (5, 1) **(1, 4)** (5, 3)	$S_{1,3}$ (1, 1) **(2, 4)** (3, 3)
$S_{2,1}$ **(2, 2)** (1, 4)	$S_{2,2}$ (3, 1) **(1, 2)**	$S_{2,3}$ **(2, 2)** (2, 3)
$S_{3,1}$ (1, 3) **(2, 3)** (3, 3)	$S_{3,2}$ (1, 1) **(1, 3)**	$S_{3,3}$ (2, 3) **(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

$S_{1,1}$ **(1,1)** (3,1) (2,4)	$S_{1,2}$ (5,1) **(1,4)** (5,3)	$S_{1,3}$ (1,1) **(2,4)** (3,3)
$S_{2,1}$ **(2,2)** (1,4)	$S_{2,2}$ (3,1) **(1,2)**	$S_{2,3}$ **(2,2)** (2,3)
$S_{3,1}$ (1,3) **(2,3)** (3,3)	$S_{3,2}$ (1,1) **(1,3)**	$S_{3,3}$ (2,3) **(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 $n = 5$

$S_{1,1}$ **(1, 1)** (3, 1) (2, 4)	$S_{1,2}$ (5, 1) **(1, 4)** (5, 3)	$S_{1,3}$ (1, 1) **(2, 4)** (3, 3)
$S_{2,1}$ **(2, 2)** (1, 4)	$S_{2,2}$ (3, 1) **(1, 2)**	$S_{2,3}$ **(2, 2)** (2, 3)
$S_{3,1}$ (1, 3) **(2, 3)** (3, 3)	$S_{3,2}$ (1, 1) **(1, 3)**	$S_{3,3}$ (2, 3) **(5, 3)**

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

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 $n = 5$

$S_{1,1}$ **(1, 1)** (3, 1) (2, 4)	$S_{1,2}$ (5, 1) **(1, 4)** (5, 3)	$S_{1,3}$ (1, 1) **(2, 4)** (3, 3)
$S_{2,1}$ **(2, 2)** (1, 4)	$S_{2,2}$ (3, 1) **(1, 2)**	$S_{2,3}$ **(2, 2)** (2, 3)
$S_{3,1}$ (1, 3) **(2, 3)** (3, 3)	$S_{3,2}$ (1, 1) **(1, 3)**	$S_{3,3}$ (2, 3) **(5, 3)**

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

Theorem (ohne Beweis)

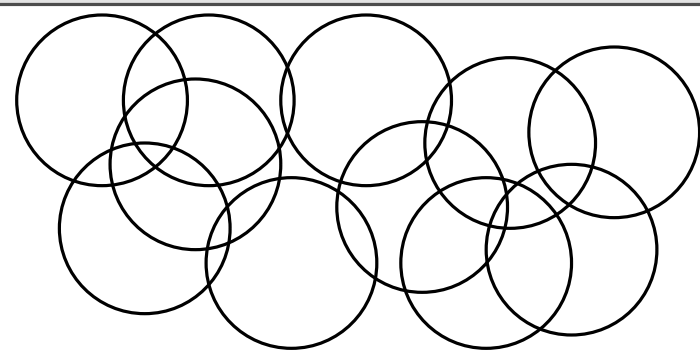
ETH impliziert, dass es keinen $f(k) \cdot n^{o(k)}$-Algorithmus für Grid Tiling with $\leq$ gibt (außerdem ist es $W[1]$-schwer).

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?

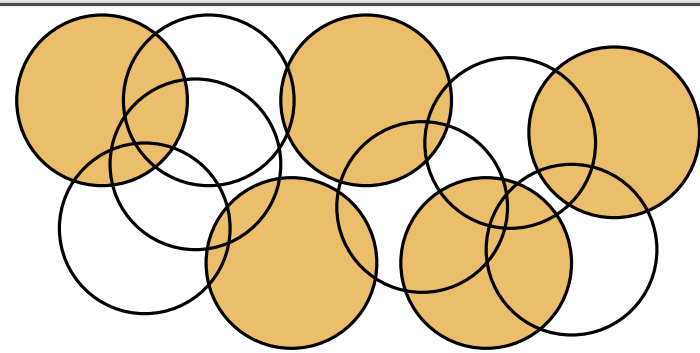


Unabhängige Einheitskreise

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

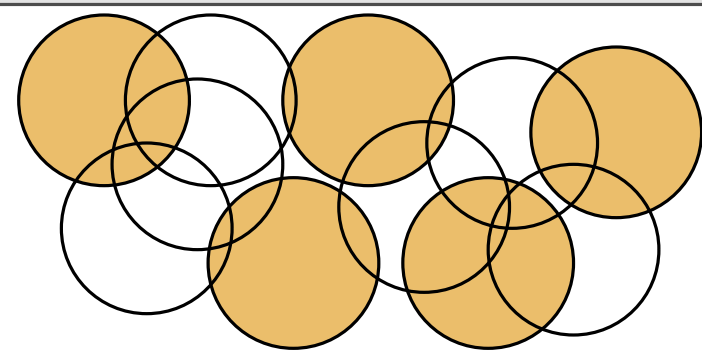


Unabhängige Einheitskreise

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

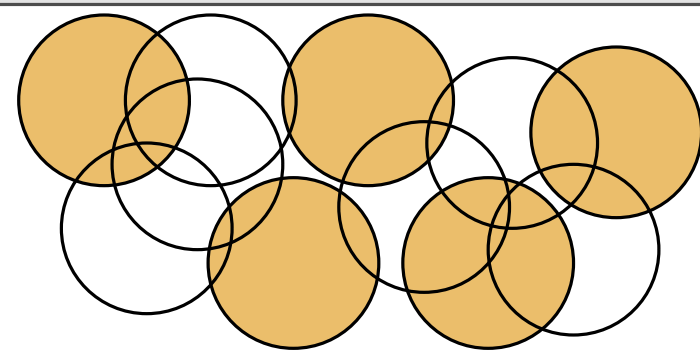
- äquivalent zu Independent Set eingeschränkt auf „unit disk graphs“
- also potentiell echt leichter als Independent Set

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

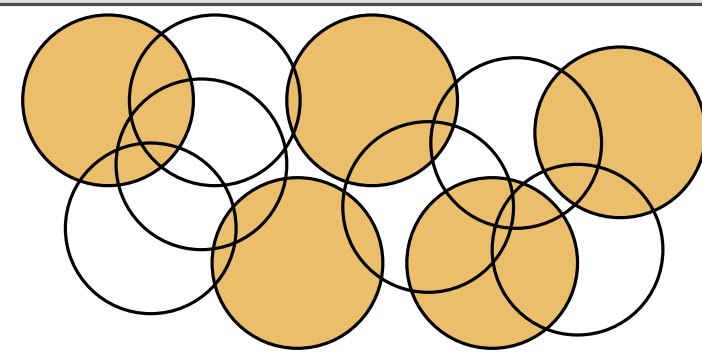
- äquivalent zu Independent Set eingeschränkt auf „unit disk graphs“
- also potentiell echt leichter als Independent Set
- tatsächlich: Unit Disk Independent Set kann in $n^{O(\sqrt{k})}$ gelöst werden

  (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

$S_{1,1}$ **(1,1)** (3,1) (2,4)	$S_{1,2}$ (5,1) **(1,4)** (5,3)	$S_{1,3}$ (1,1) **(2,4)** (3,3)
$S_{2,1}$ **(2,2)** (1,4)	$S_{2,2}$ (3,1) **(1,2)**	$S_{2,3}$ **(2,2)** (2,3)
$S_{3,1}$ (1,3) **(2,3)** (3,3)	$S_{3,2}$ (1,1) **(1,3)**	$S_{3,3}$ (2,3) **(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

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (2,3)
$S_{3,1}$ (1,3) (2,3) (3,3)	$S_{3,2}$ (1,1) (1,3)	$S_{3,3}$ (2,3) (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

$S_{1,1}$ **(1,1)** (3,1) (2,4)	$S_{1,2}$ (5,1) **(1,4)** (5,3)	$S_{1,3}$ (1,1) **(2,4)** (3,3)
$S_{2,1}$ **(2,2)** (1,4)	$S_{2,2}$ (3,1) **(1,2)**	$S_{2,3}$ **(2,2)** (2,3)
$S_{3,1}$ (1,3) **(2,3)** (3,3)	$S_{3,2}$ (1,1) **(1,3)**	$S_{3,3}$ (2,3) **(5,3)**

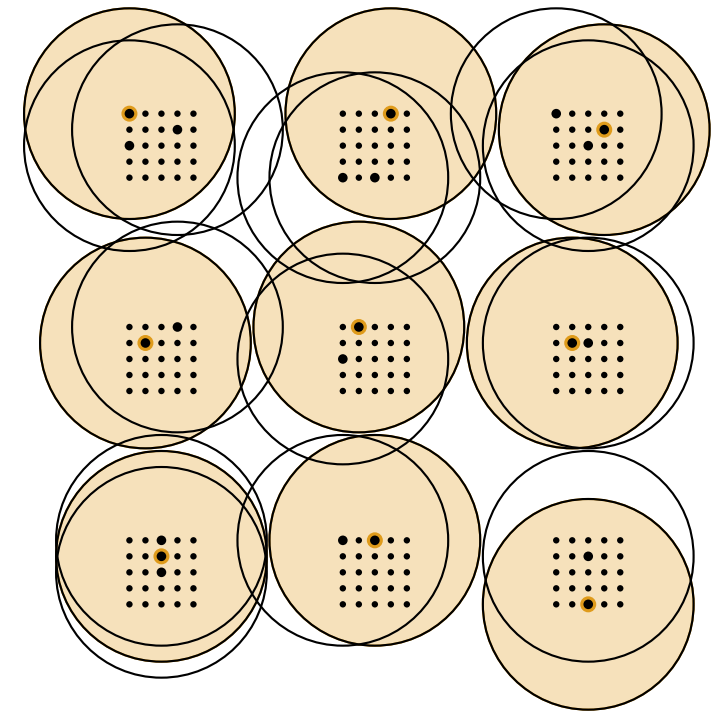
Plan: erstelle Instanz von Unit Disk Independent Set, die Lösung der Größe k^2 hat $\Leftrightarrow$ Grid Tiling with $\leq$ hat Lösung (im Beispiel ist $k = 3$)

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

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (2,3)
$S_{3,1}$ (1,3) (2,3) (3,3)	$S_{3,2}$ (1,1) (1,3)	$S_{3,3}$ (2,3) (5,3)



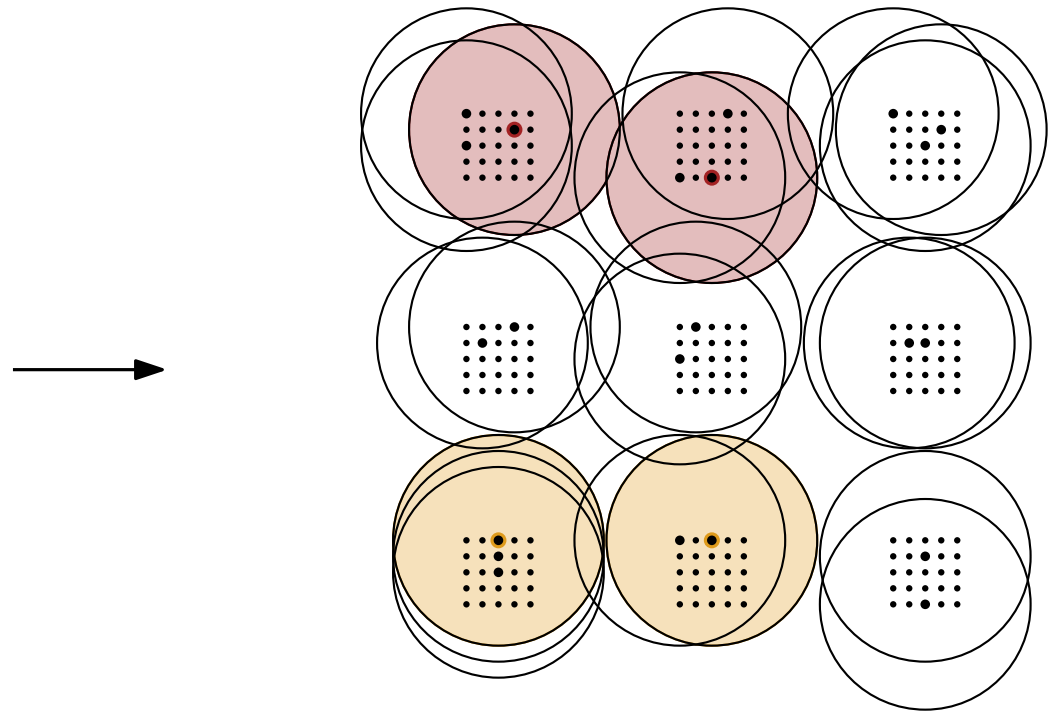
Plan: erstelle Instanz von Unit Disk Independent Set, die Lösung der Größe k^2 hat $\Leftrightarrow$ Grid Tiling with $\leq$ hat Lösung (im Beispiel ist $k = 3$)

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

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (2,3)
$S_{3,1}$ (1,3) (2,3) (3,3)	$S_{3,2}$ (1,1) (1,3)	$S_{3,3}$ (2,3) (5,3)



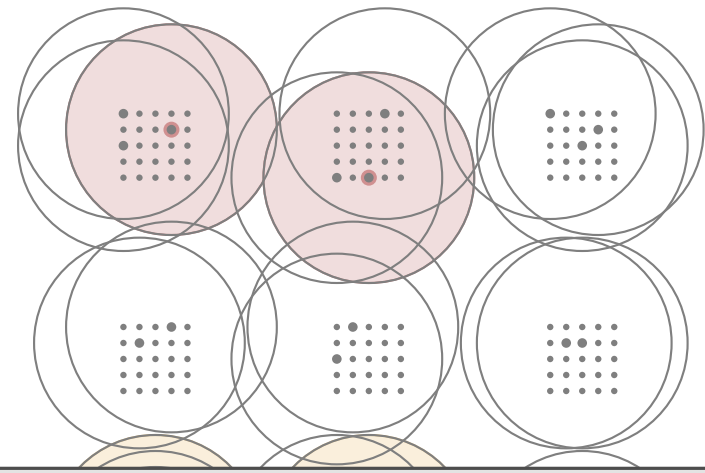
Plan: erstelle Instanz von Unit Disk Independent Set, die Lösung der Größe k^2 hat $\Leftrightarrow$ Grid Tiling with $\leq$ hat Lösung (im Beispiel ist $k = 3$)

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

$S_{1,1}$ (1,1) (3,1) (2,4)	$S_{1,2}$ (5,1) (1,4) (5,3)	$S_{1,3}$ (1,1) (2,4) (3,3)
$S_{2,1}$ (2,2) (1,4)	$S_{2,2}$ (3,1) (1,2)	$S_{2,3}$ (2,2) (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 $O(k^{2-\varepsilon}) \Rightarrow \text{NP} \subseteq \text{coNP/poly}$

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 $O(k^{2-\varepsilon}) \Rightarrow \text{NP} \subseteq \text{coNP/poly}$

Untere Schranken in P

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