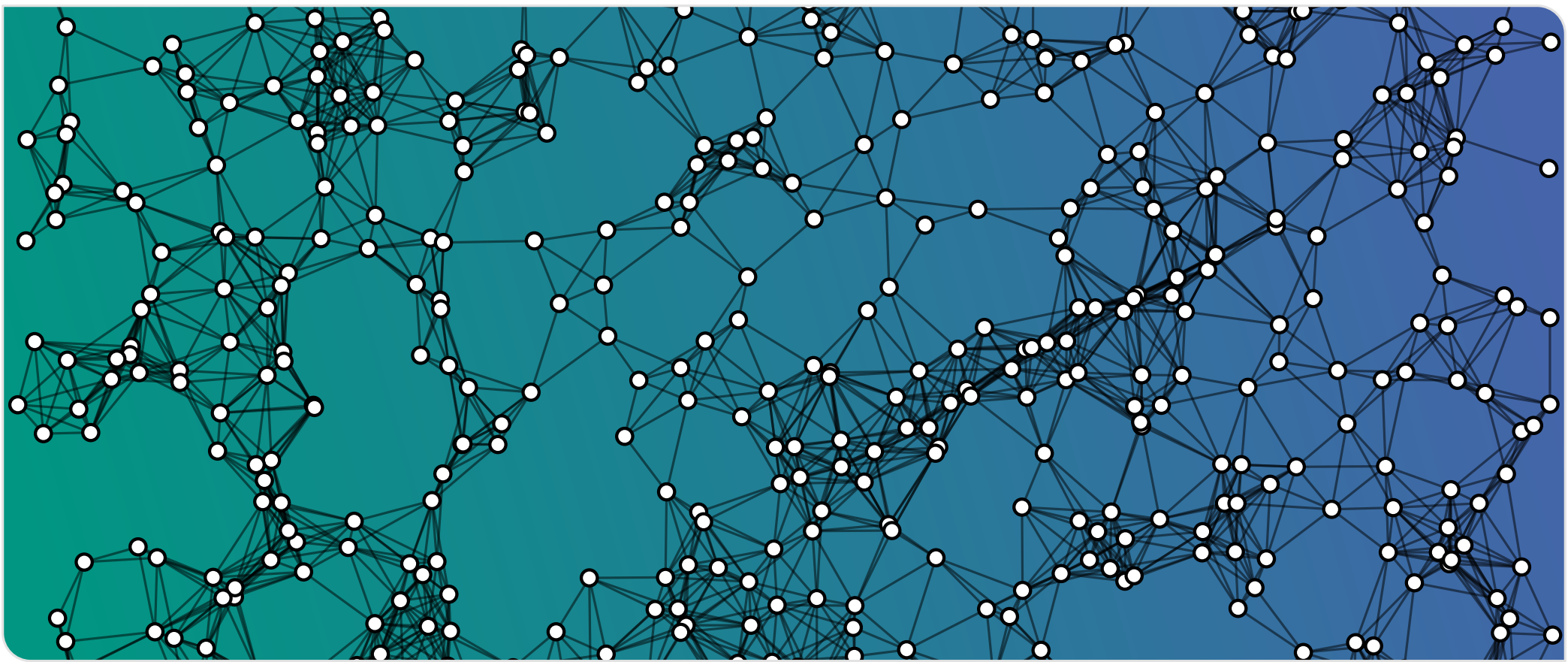
Karlsruher Institut für Technologie

Parametrisierte Algorithmen

Zusammenfassung



Wiederholung: 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)**

Wiederholung: 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)**

Wiederholung: 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

$S_{1,1}$ **(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)**

Wiederholung: 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

$S_{1,1}$ **(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)**

Theorem (ohne Beweis)

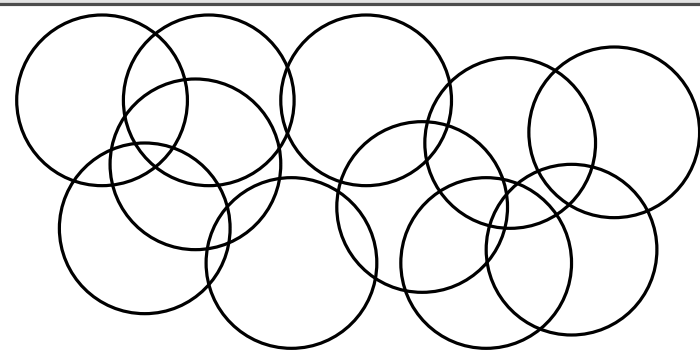
ETH impliziert, dass es keinen $f(k) \cdot n^{o(k)}$-Algorithmus für Grid Tiling with $\leq$ gibt (außerdem ist es $W[1]$-schwer).

Nachtrag: 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?

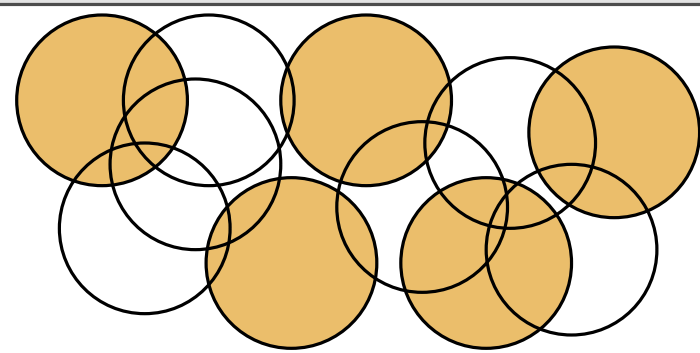


Nachtrag: Unabhängige Einheitskreise

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

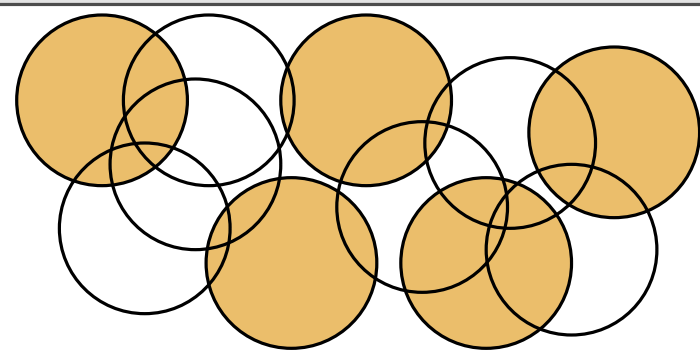


Nachtrag: Unabhängige Einheitskreise

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

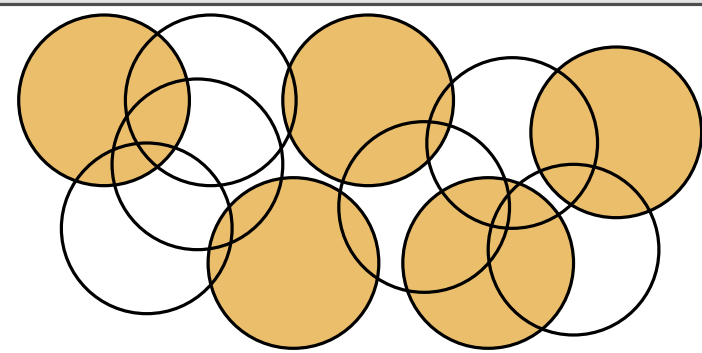
- äquivalent zu Independent Set eingeschränkt auf „unit disk graphs“
- also potentiell echt leichter als Independent Set

Nachtrag: 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

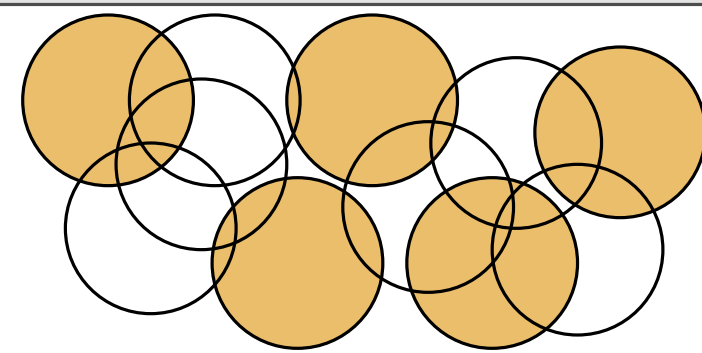
(untere Schranke für allgemeines IS (basierend auf ETH): $n^{\Omega(k)}$)

Nachtrag: 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 $\leq$

Nachtrag: 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 $k = 3$)

Nachtrag: 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

Nachtrag: 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)

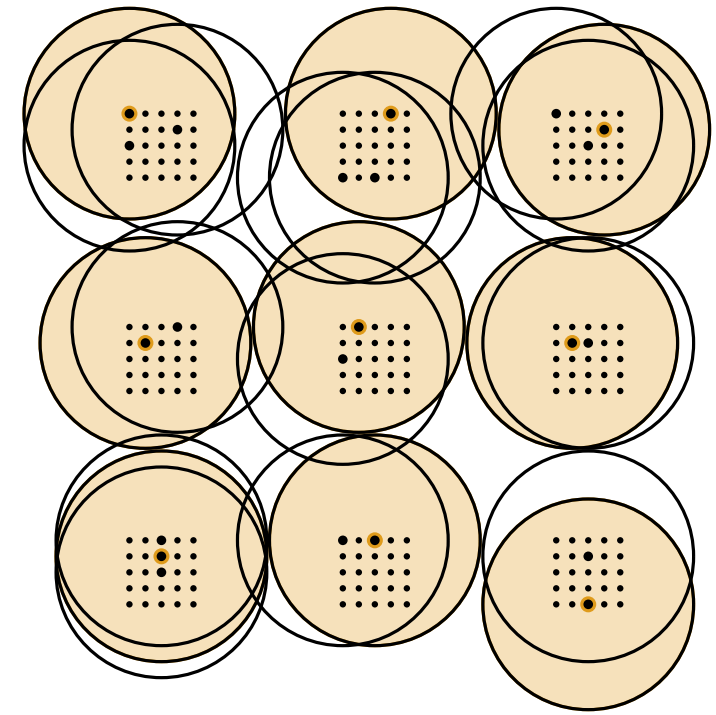
Plan: erstelle Instanz von Unit Disk Independent Set, die Lösung der Größe k^2 hat $\Leftrightarrow$ Grid Tiling with $\leq$ hat Lösung (im Beispiel ist $k = 3$)

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

Nachtrag: 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)



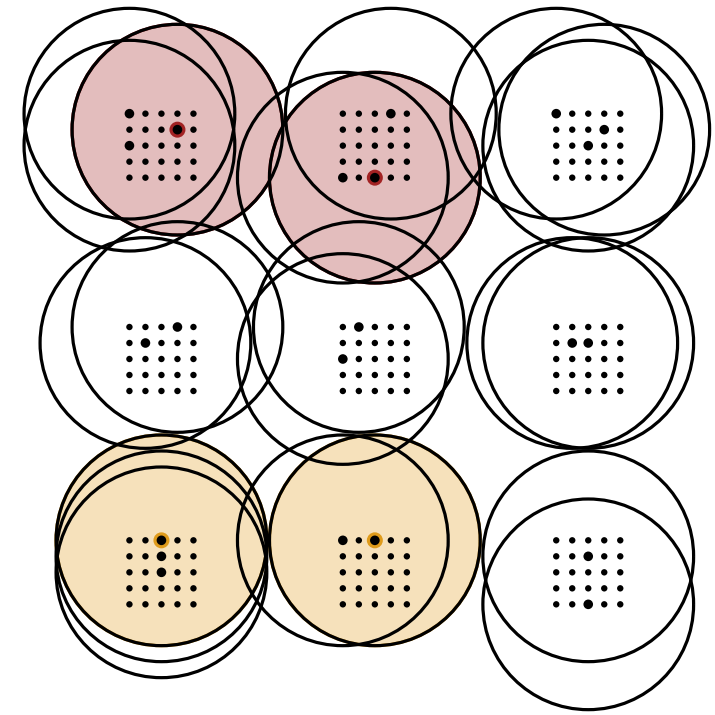
Plan: erstelle Instanz von Unit Disk Independent Set, die Lösung der Größe k^2 hat $\Leftrightarrow$ Grid Tiling with $\leq$ hat Lösung (im Beispiel ist $k = 3$)

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

Nachtrag: 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)



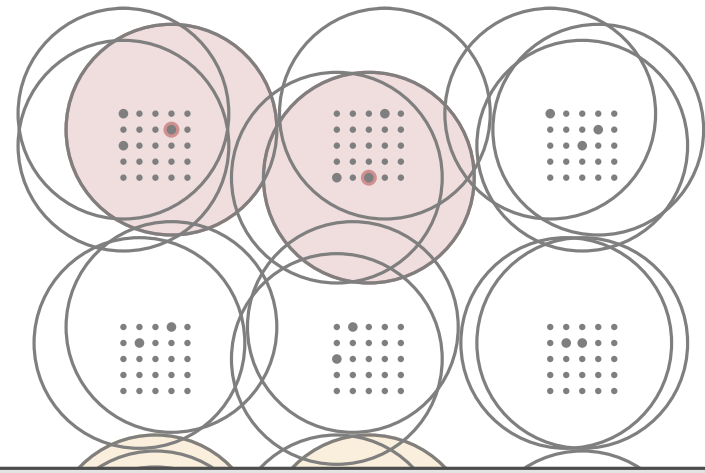
Plan: erstelle Instanz von Unit Disk Independent Set, die Lösung der Größe k^2 hat $\Leftrightarrow$ Grid Tiling with $\leq$ hat Lösung (im Beispiel ist $k = 3$)

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

Nachtrag: 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

Grundsätzliche Hinweise

Ziel der Prüfung

- wir müssen euch bescheinigen, dass ihr
  - den Stoff der Vorlesung kennt
  - die Inhalte verstanden habt
  - mit den gelernten Techniken neue Algorithmen entwerfen und analysieren könnt

Grundsätzliche Hinweise

Ziel der Prüfung

- wir müssen euch bescheinigen, dass ihr
  - den Stoff der Vorlesung kennt
  - die Inhalte verstanden habt
  - mit den gelernten Techniken neue Algorithmen entwerfen und analysieren könnt
- mündliche Pr.: besonders gut geeignet für die ersten beiden Punkte
- der Fokus wird auf dem zweiten Punkt liegen

(Punkt eins und drei werden aber auch vertreten sein)

Grundsätzliche Hinweise

Ziel der Prüfung

- wir müssen euch bescheinigen, dass ihr
  - den Stoff der Vorlesung kennt
  - die Inhalte verstanden habt
  - mit den gelernten Techniken neue Algorithmen entwerfen und analysieren könnt
- mündliche Pr.: besonders gut geeignet für die ersten beiden Punkte
- der Fokus wird auf dem zweiten Punkt liegen

(Punkt eins und drei werden aber auch vertreten sein)

Vorbereitung

- Schritt 1: rekapituliert und versteht die Inhalte

Grundsätzliche Hinweise

Ziel der Prüfung

- wir müssen euch bescheinigen, dass ihr
  - den Stoff der Vorlesung kennt
  - die Inhalte verstanden habt
  - mit den gelernten Techniken neue Algorithmen entwerfen und analysieren könnt
- mündliche Pr.: besonders gut geeignet für die ersten beiden Punkte
- der Fokus wird auf dem zweiten Punkt liegen

(Punkt eins und drei werden aber auch vertreten sein)

Vorbereitung

- Schritt 1: rekapituliert und versteht die Inhalte
- Schritt 2: erklärt jemandem die Inhalte

Grundsätzliche Hinweise

Ziel der Prüfung

- wir müssen euch bescheinigen, dass ihr
  - den Stoff der Vorlesung kennt
  - die Inhalte verstanden habt
  - mit den gelernten Techniken neue Algorithmen entwerfen und analysieren könnt
- mündliche Pr.: besonders gut geeignet für die ersten beiden Punkte
- der Fokus wird auf dem zweiten Punkt liegen

(Punkt eins und drei werden aber auch vertreten sein)

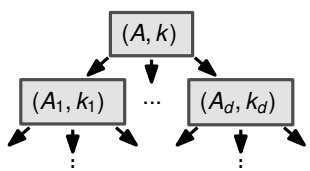
Vorbereitung

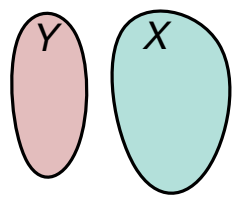
- Schritt 1: rekapituliert und versteht die Inhalte
- Schritt 2: erklärt jemandem die Inhalte
- **Achtung:** Schritt 2 ist genau das, was in der Prüfung passiert. Das solltet ihr definitiv vorher üben!

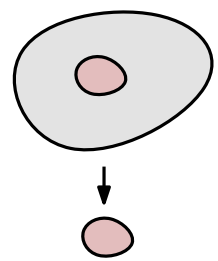
Inhalt

Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung

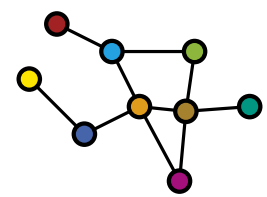


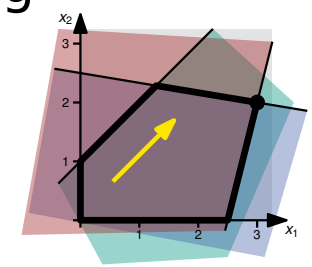


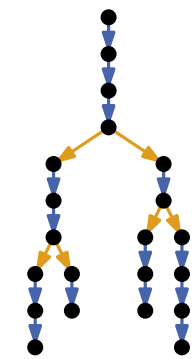


Erweiterte Toolbox

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





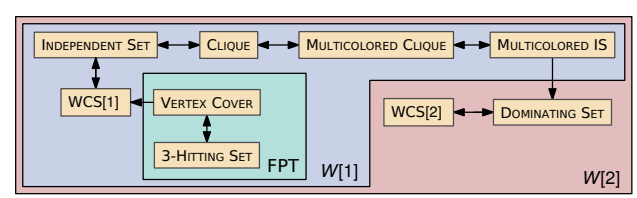


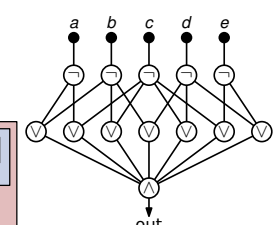
Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

Untere Schranken

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

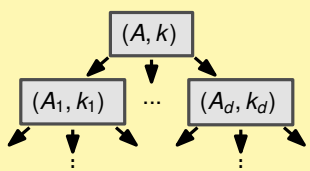


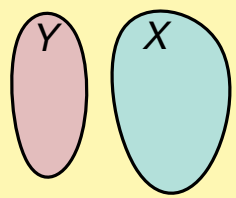


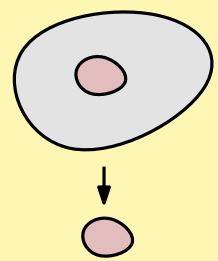
Inhalt

Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung

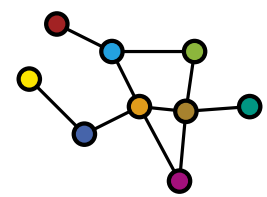


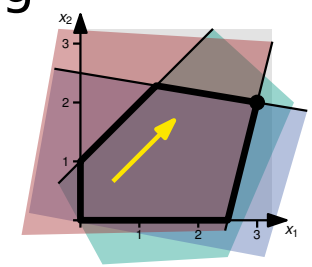


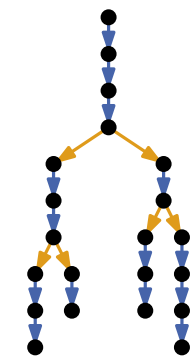


Erweiterte Toolbox

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





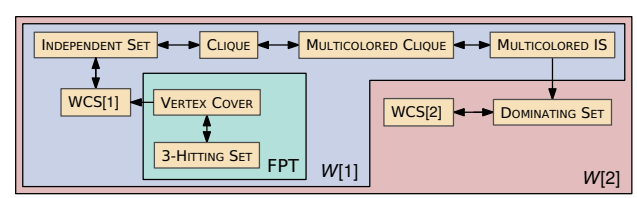


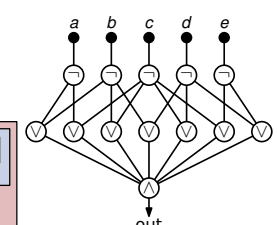
Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

Untere Schranken

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





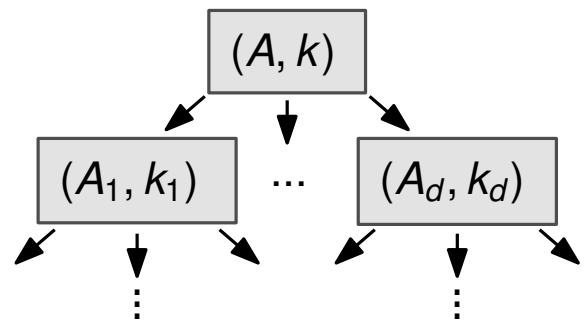
FPT-Techniken Grundtechniken

Grundbegriffe: Parametrisierung

- Was sind parametrisierte Probleme?
- Was ist FPT?

Suchbäume

- Wie funktionieren Suchbäume?
- Wie bekomm man FPT-Laufzeit?
- Was ist wichtig um die Tiefe zu beschränken?

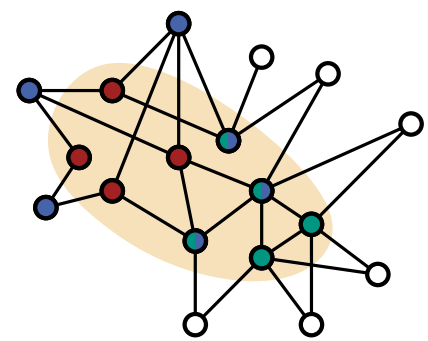


Kernbildung

- Was ist ein Kern?
- Was ist die Standardmethode um Kerne zu berechnen?
- Wann ist Kernbildung für ein Problem möglich?

Iterative Kompression

- Was ist die Grundidee dieser Technik?
- Woher kommt die zu große Lösung?
- Was ist das Kompressions-Problem?

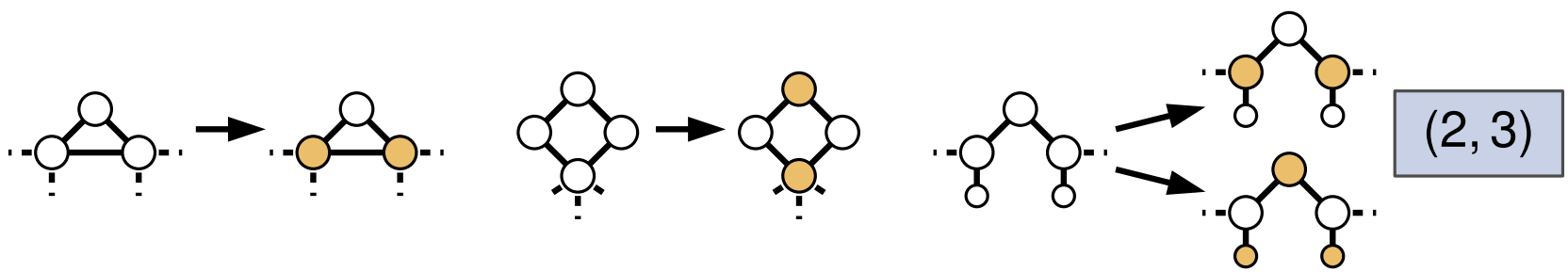


Wie lassen sich diese Techniken auf Vertex Cover anwenden?

Beschränkte Suchbäume

Allgemeine Verzweigung

- Was ist ein Verzweigungsvektor?
- Wie findet man anhand des Verzweigungsvektors die Baumgröße?
- Warum muss man da die Nullstelle eines Polynoms berechnen?



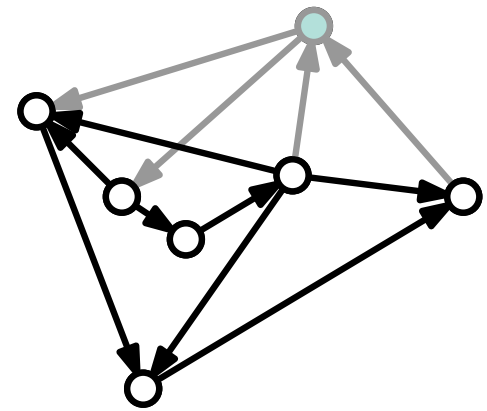
Bessere Verzweigung bei Vertex Cover

- Warum helfen Knoten mit großen Grad?
- Kann man immer einen Knoten mit Grad 4 finden?
- Wie wird man Knoten mit Grad 2 los?
- Was ist die Idee für Grad 3 Knoten?
- Wie sieht es mit Grad 4 Knoten aus?
- Warum kann ich den Verzweigungsvektor $(1, 4)$ ignorieren?

Iterative Kompression

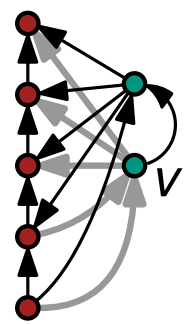
Feedback Vertex Set

- Was ist das Problem?
- Wie reduziert man FVS auf FVS Compression?
- Wie reduziert man FVS Compression auf Disjoint FVS?
- Welche Laufzeit verliert man unterwegs?



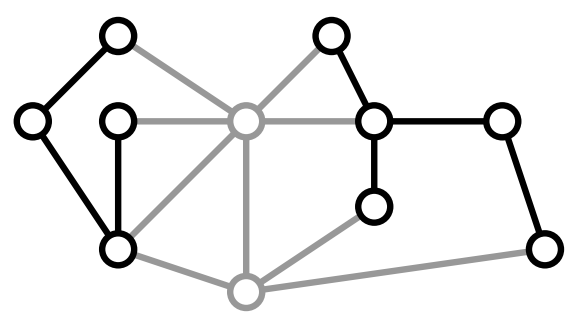
Disjoint Feedback Vertex Set auf Turniergraphen

- Was sind Turniergraphen?
- Wie reduziert sich das Problem dabei auf Longest Common Subsequence?



Ungerichtetes FVS

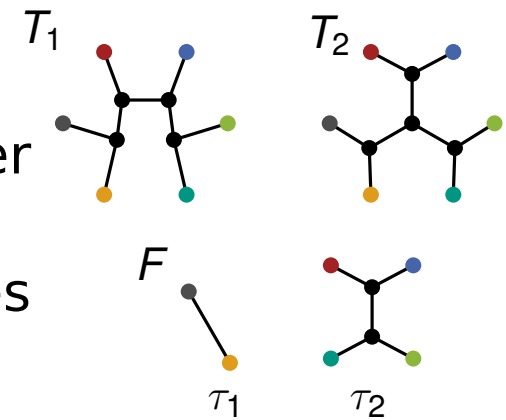
- Reduktion auf die disjunkte Variante?
- Wie löst man diese?

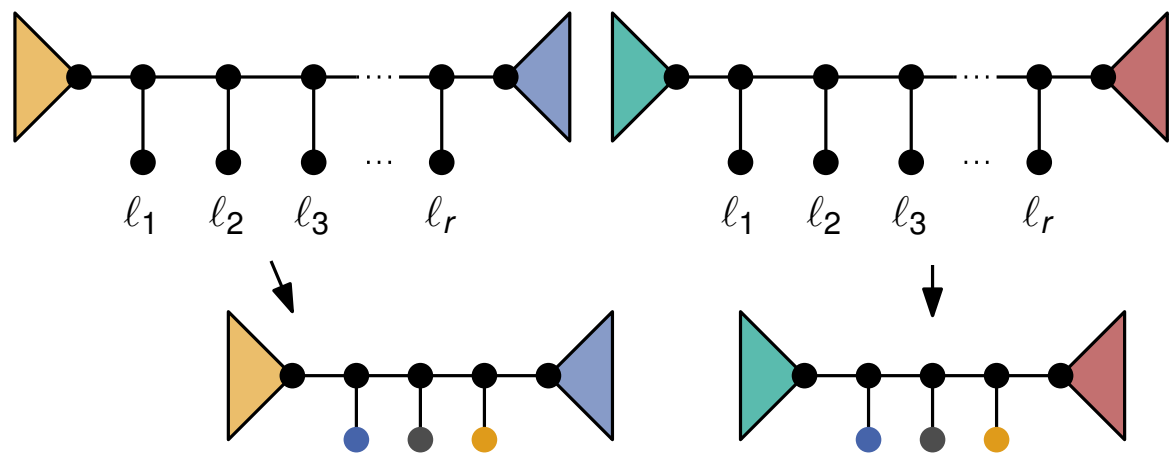


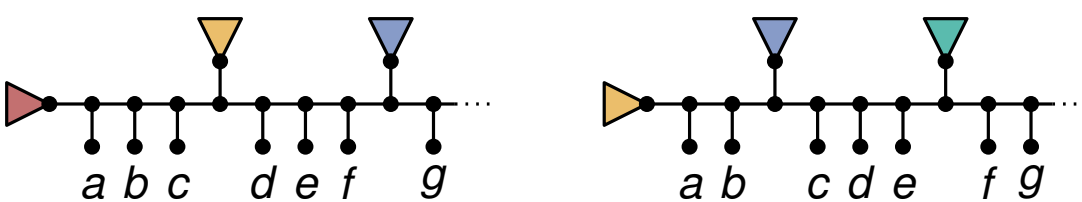
Kernbildung: Ähnliche Bäume

Maximum Agreement Forest

- Was ist das Problem?
- Warum würde es helfen, wenn jeder Baum in der Lösung wenige Blätter enthält?
- Welche Gegenbeispiele für diesen Wunsch gibt es und wie werden wir sie los?
- Wo muss man bei der Reduktion aufpassen?
- Wie hilft die amortisierte Sichtweise?



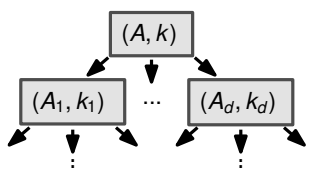


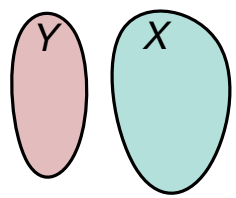


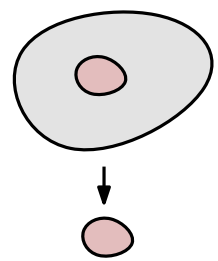
Inhalt

Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung

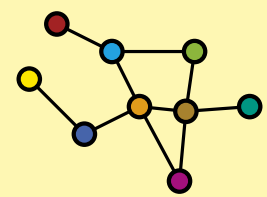


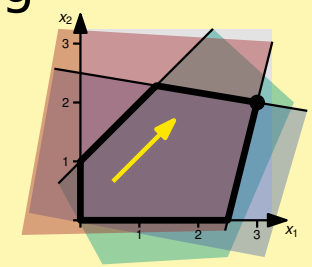


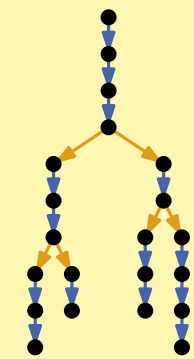


Erweiterte Toolbox

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





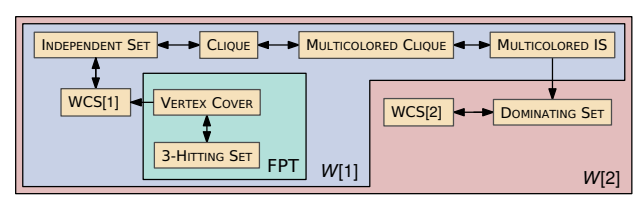


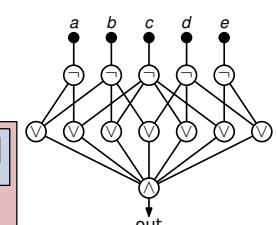
Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

Untere Schranken

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

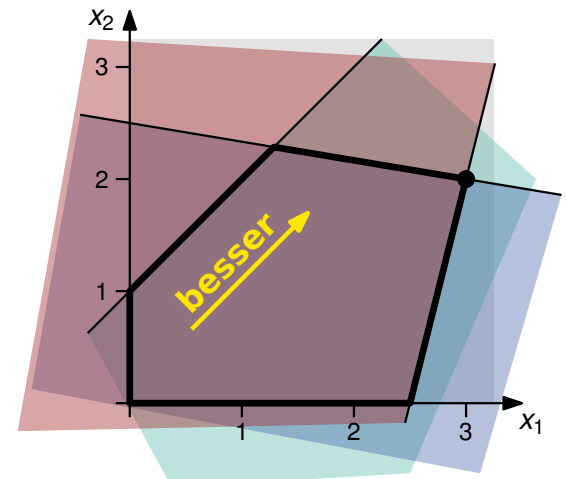




Lineare Programmierung

Grundlagen

- Was ist lineare Programmierung?
- Welche Schreibweise wird für LPs verwendet?
- Wie lassen sich LPs umformen z.B. zu $Ax \leq b$?
- Wie lässt sich manchmal ein Betrag durch lineare Bedingungen abbilden?



Dualität

- Was bedeutet Dualität für LP?
- Was sagt der Dualitätssatz?
- Wofür ist Dualität nützlich?

Theorem von Lenstra

- Was sagt uns Lenstras Theorem?
- Wie kann man damit einen FPT-Algo für Closest String bekommen?

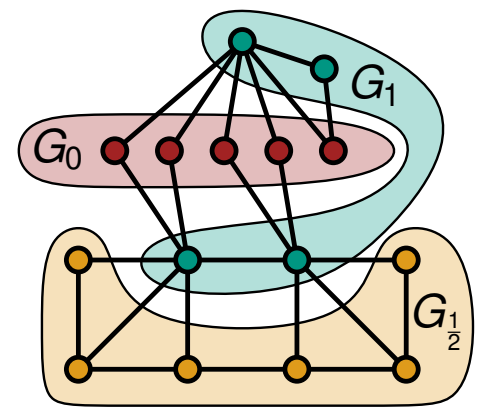
ILP Relaxierung

ILP

- Warum sind ILPs interessant?
- Was ist Relaxierung und was hat man davon?
- Wann ist Relaxierung exakt?

Unimodularität

- Wie ist totale Unimodularität definiert?
- Wie beweist man totale Unimodularität?
- Beispiele total unimodularer Matrizen?



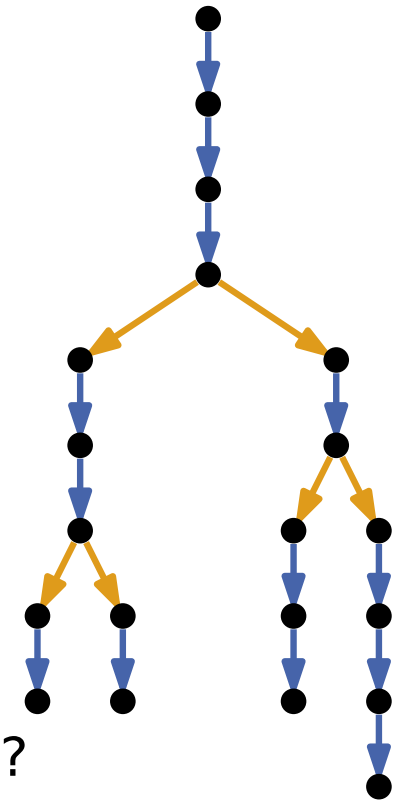
Kernbildung für Vertex Cover

- Welche Eigenschaften hat eine Optimallösung der Relaxierung?
- Wie bildet man mit einer LP Lösung einen Kern?
- Wie zeigt man, dass diese Reduktion sicher ist?
- Muss man wiklich ein LP lösen?

Branch-and-Reduce

Branch-and-Reduce

- Was versteht man unter diesem Begriff?



Above Lower Bound Parametrisierung

- Welche Parameter werden hier betrachtet?
- Warum sind solche Parameter interessant?
- Beispiele für diese Parametrisierungen?

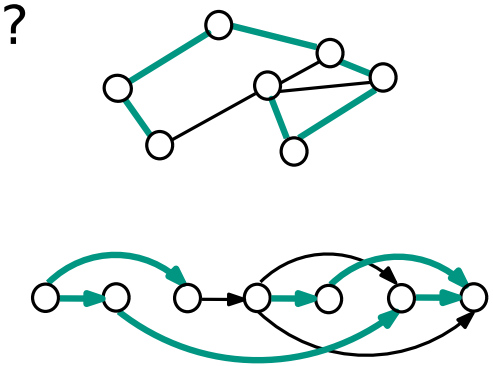
Vertex Cover above LP

- Was ist die Grundidee zur Lösung dieses Problems?
- Warum braucht man hier Reduktion vor dem Branching?
- Wie funktioniert die Parameteranpassung?

Color Coding

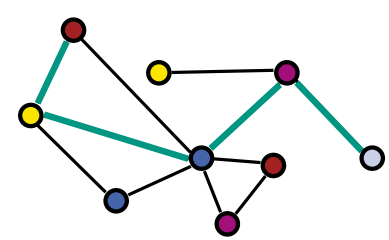
Longest Path: Randomisierung

- Welche zusätzliche Struktur haben wir uns geraten?
- Wie hilft diese zusätzliche Struktur beim Lösen?
- Was muss man über die Erfolgswahrscheinlichkeit zeigen, um FPT-Laufzeit zu erhalten?



Color Coding

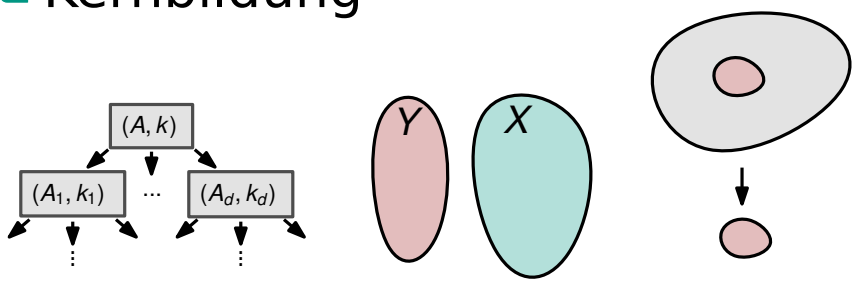
- Welche zusätzliche Struktur raten wir hier?
- Wie hilft uns das?
- Wie können wir das derandomisieren?
- Was sind perfekte Familien von Hash-Funktionen?
- Was sind universelle Mengen?



Inhalt

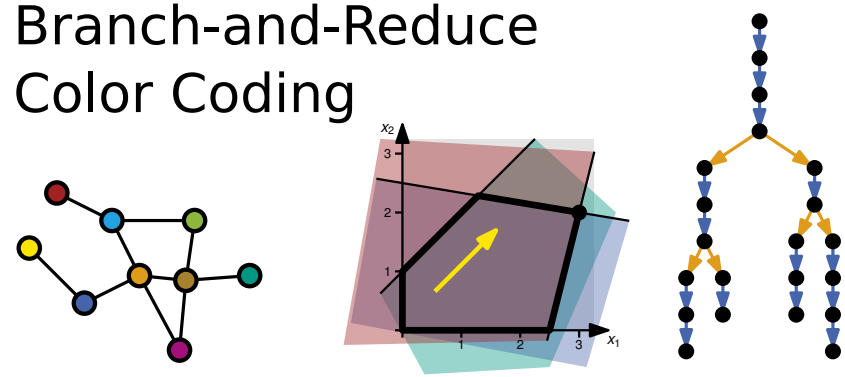
Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung



Erweiterte Toolbox

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

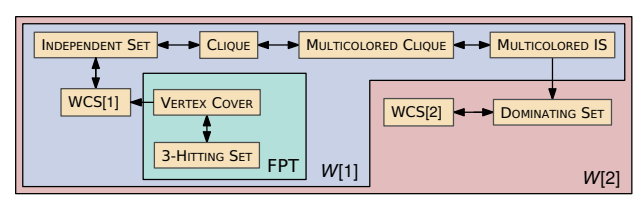


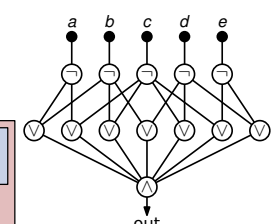
Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

Untere Schranken

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

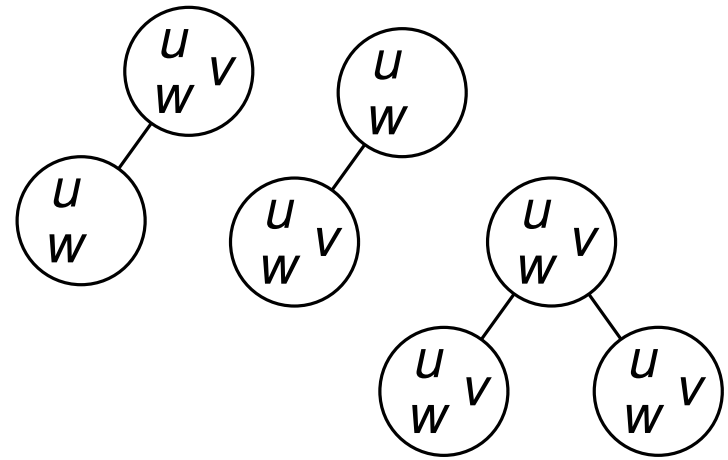




Baumweite

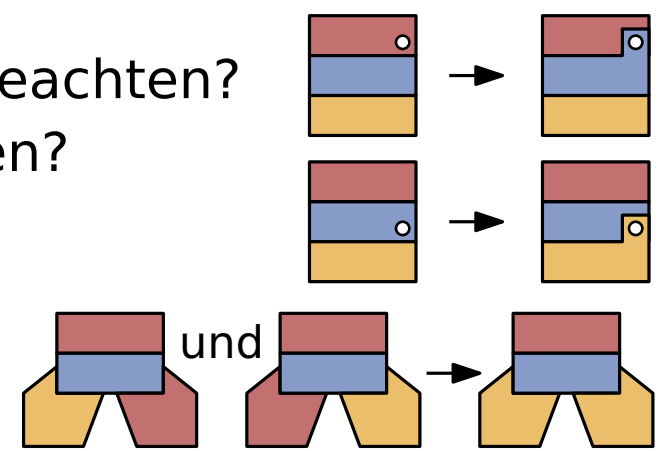
Pfad- und Baumzerlegungen

- Was ist eine Pfad- bzw. Baumzerlegung?
- Was ist die Pfad- bzw. Baumweite?
- Wann ist eine Baumzerlegung schön?
- Warum kann man immer von schönen Baumzerlegungen ausgehen?



Dynamische Programme

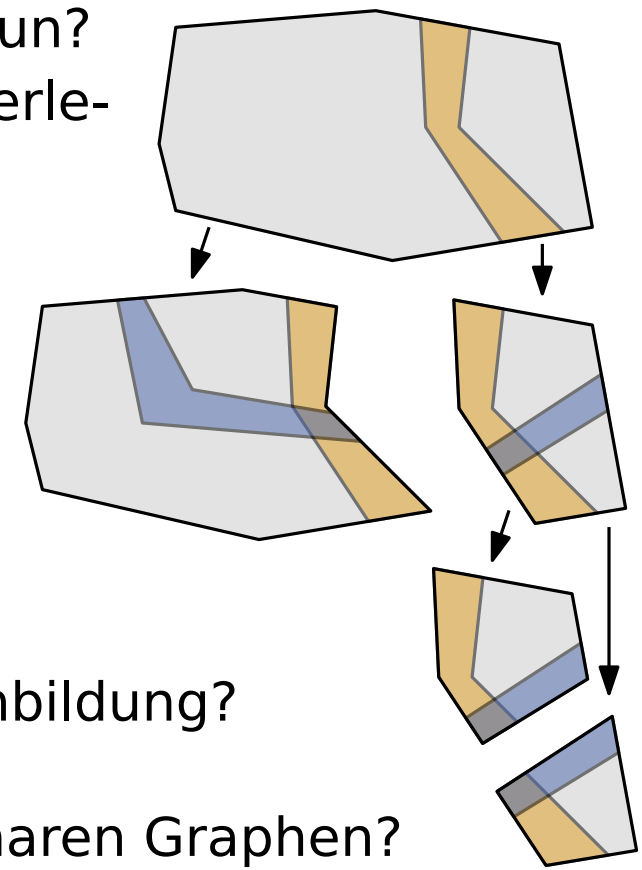
- Wie kann man Probleme mittels DB auf Baumzerlegungen bauen?
- Beispiele: INDEPENDENT SET und HAMILTONKREIS
- Wie schafft man es globale Bedingungen zu beachten?
- Kannst du das auf andere Probleme übertragen?



Baumzerlegungen und planare Graphen

Approximativer FPT-Algo für Baumzerlegungen

- Was heißt hier Approximation und was bedeutet das für unser DP?
- Was ist die Idee?
- Was hat das mit balancierten Separatoren zu tun?
- Warum können wir den Separator balanciert zerlegen, aber nicht den ganzen Graph?
- Warum genügt eine balancierte Aufteilung des Separators?
- Warum gibt es balancierte Separatoren?



Win-Win in planaren Graphen

- Was sind Minoren?
- Was passiert mit der Baumweite unter Minorenbildung?
- Was hat das sonst mit der Baumweite zu tun?
- Wie hilft das für schnellere Algorithmen in planaren Graphen?

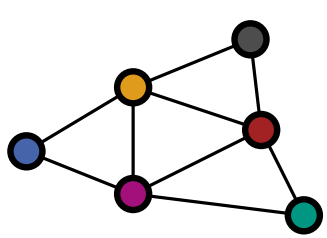
Courcelle & Chordale Graphen

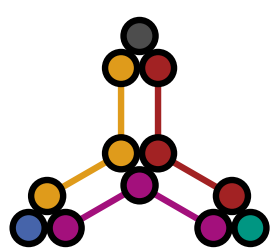
Courcelles Theorem

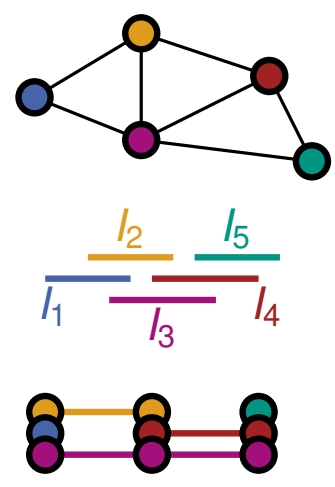
- Was erlaubt MSO_2?
- Wie kann man Zusammenhang modellieren?
- Modellierung von HAMILTONKREIS bzw. VERTEX COVER?
- Wie sieht es mit anderen Problemen aus?
- Was besagt Courcelles Theorem?
- Wozu brauchen wir die Optimierungsvariante?

Chordale Graphen

- Was sind Intervallgraphen bzw. chordale Graphen?
- Was ist die Intervallweite bzw. Chordalweite?
- Was hat das mit Pfadweite bzw. Baumweite zu tun?



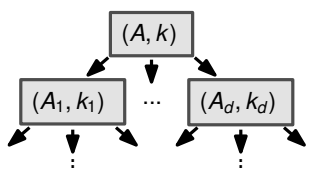


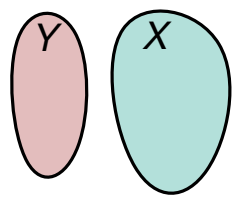


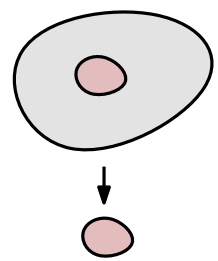
Inhalt

Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung

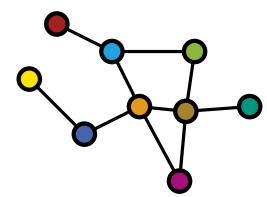


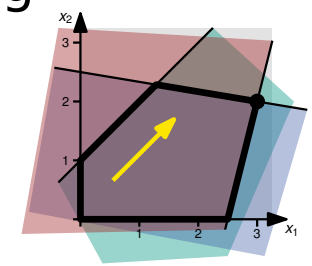


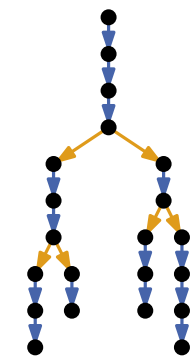


Erweiterte Toolbox

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





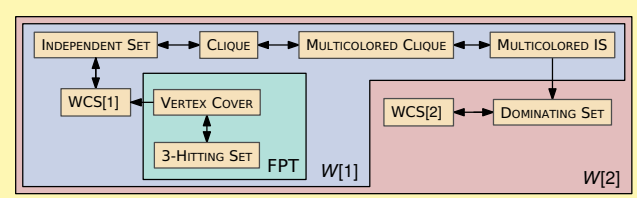


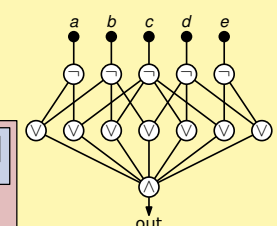
Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

Untere Schranken

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





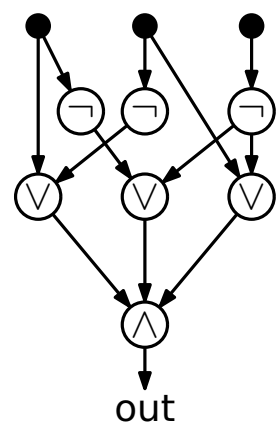
W-Hierarchie

Parametrisierte Reduktion

- Was ist eine parametrisierte Reduktion?
- Warum ist diese Reduktion so definiert?
- Reduktionen zwischen Multicolored Clique und Clique?
- Weitere interessante Reduktionen?
  - Multicolored IS $\rightarrow$ Dominating Set
  - Independent Set $\rightarrow$ Weighted Circuit Satisfiability
  - Dominating Set $\rightarrow$ Weighted Circuit Satisfiability

W-Hierarchie

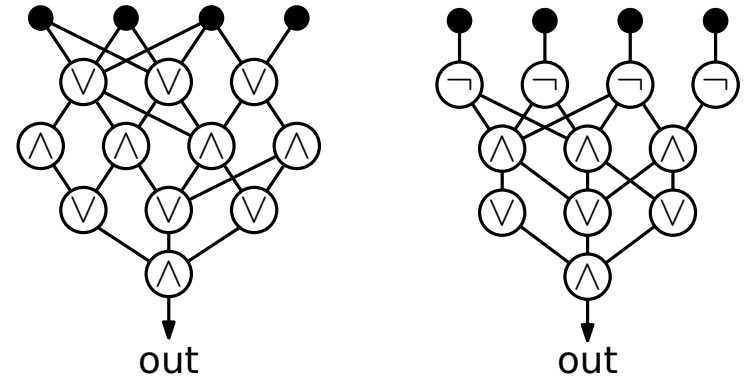
- Was ist WCS?
- Wie sind die Klassen der W-Hierarchie definiert?
- Wie zeigt man Härte/Zugehörigkeit zu W[t]?
- Vollständige Probleme für W[1], W[2]?
- Warum ist FPT eine Teilmenge von W[1]?



Relationale Datenbanken

Normalisierte Schaltkreise

- Was sind normalisierte Schaltkreise?
- Monoton vs. antimonoton?
- Warum normalisieren wir?

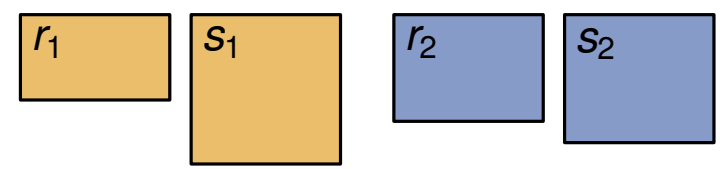


UNIQUE

- Was ist das Problem?
- Wie reduziert man von und zu HITTING SET?

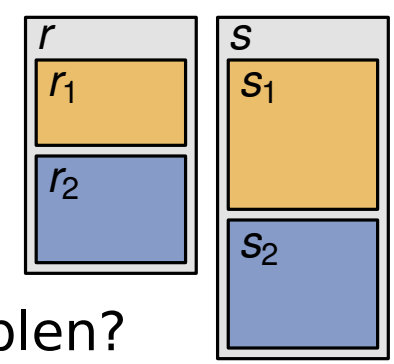
FD

- Was sind funktionale Abhängigkeiten?
- Wie reduziert man das auf WCS[2]?



$\text{IND}_{\text{FIXED}}$

- Wie kann man das als Boolesche Funktionen auffassen?
- Wie kann man das verunden und warum hilft das?
- Was fehlt dann noch für die Reduktion von WANS[3]?
- Wie modellieren wir Disj. von Konj. von negierten Variablen?

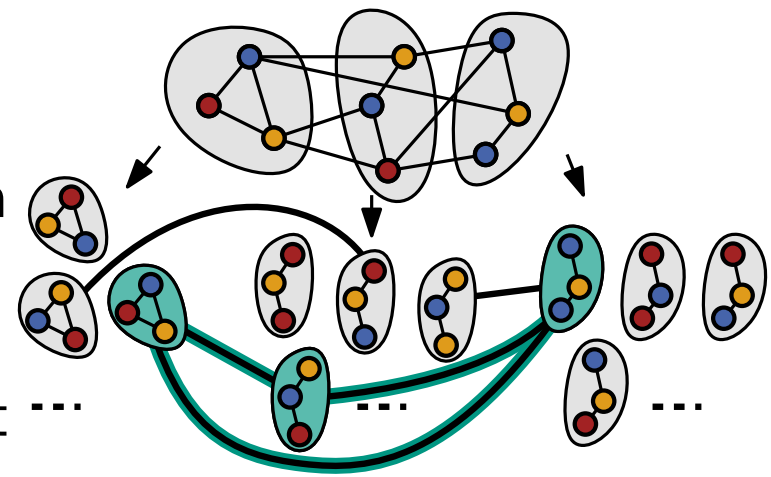


ETH und SETH

Grundlagen

- Was ist ETH? Was ist SETH?
- Wie verhalten sich die beiden zueinander?
- Wie erhält man untere Schranken für andere Probleme?
- Genügt dazu eine polynomielle Reduktion?

Reduktionen so weit das Auge reicht



- Wie erhält man daraus untere Schranken für parametrisierte Probleme?
- Was bedeutet das für $W[1]$ und FPT?
- Übertragung auf andere Probleme: genügt eine parametrisierte Reduktion?
- Was ist Grid Tiling?
- Wie reduziert man Clique auf Grid Tiling?
- Grid Tiling $\rightarrow$ Planar List Coloring? Was passiert mit dem Parameter?
- Grid Tiling $\rightarrow$ Unit Disk Indpendent Set?

Viel Erfolg!