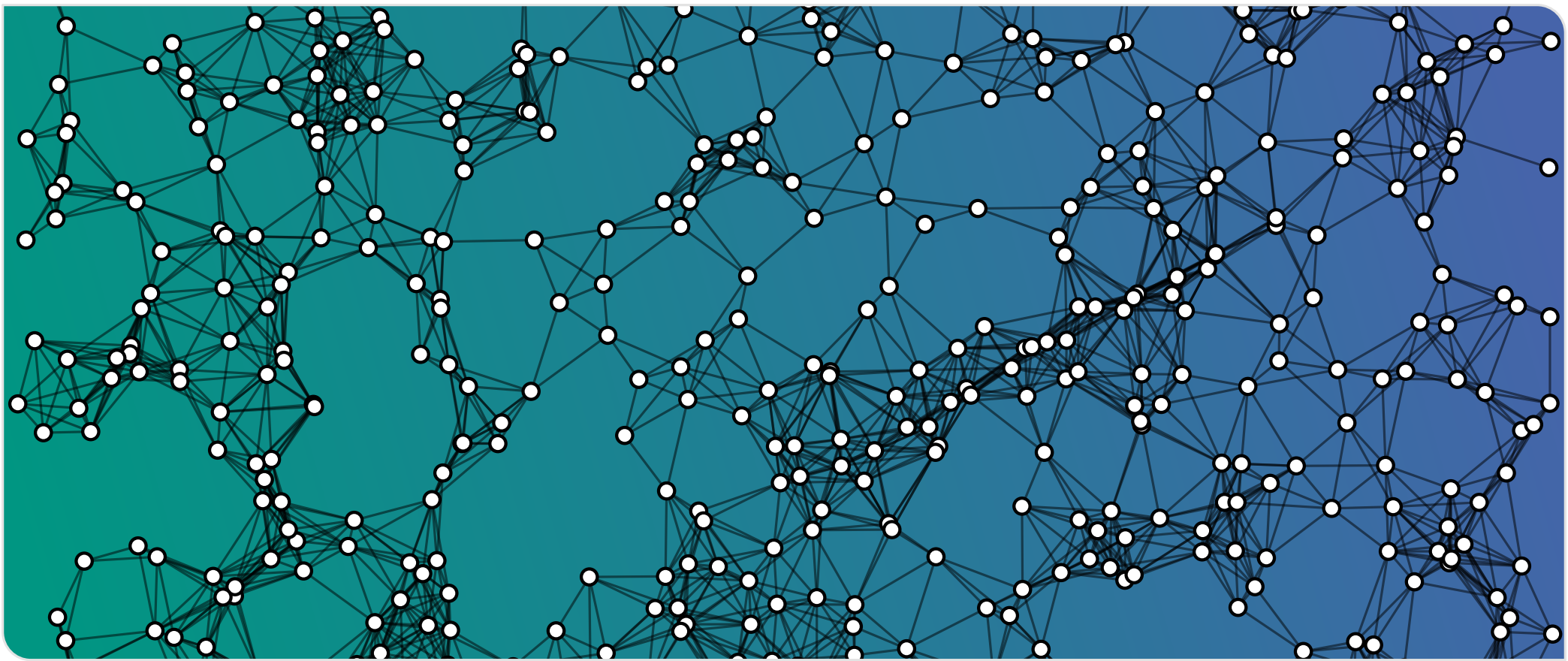
Karlsruher Institut für Technologie

Parametrisierte Algorithmen

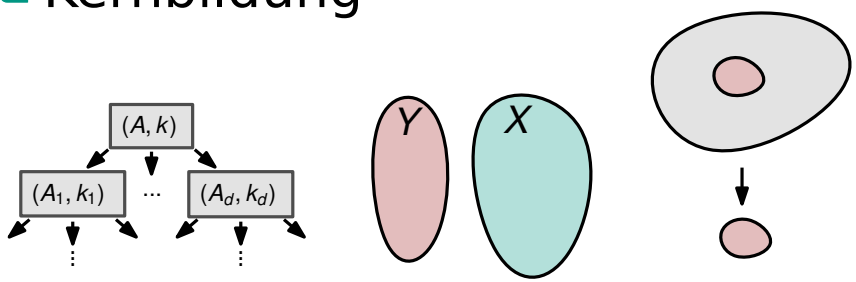
Baumweite: Courcelles Theorem und chordale Graphen



Inhalt

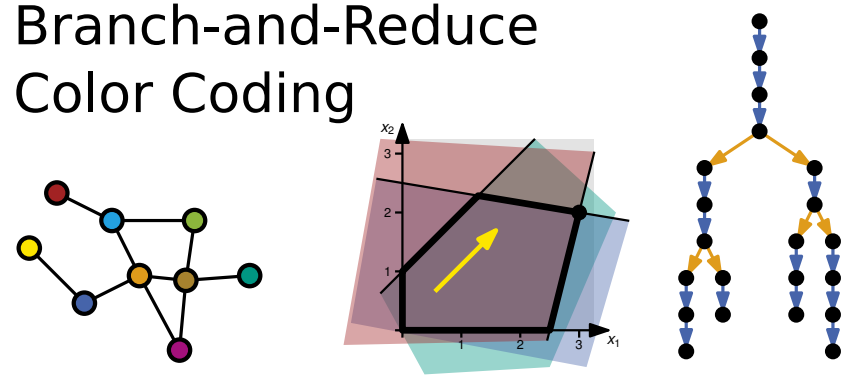
Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung



Erweiterte Toolbox

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

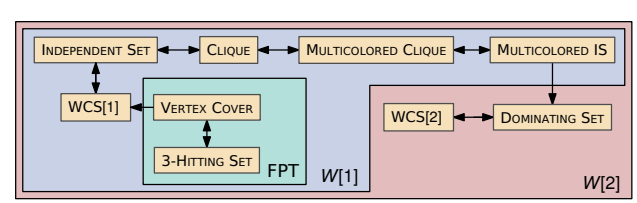


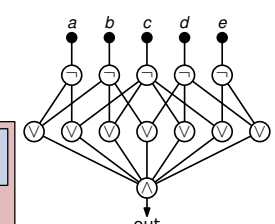
Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

Untere Schranken

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

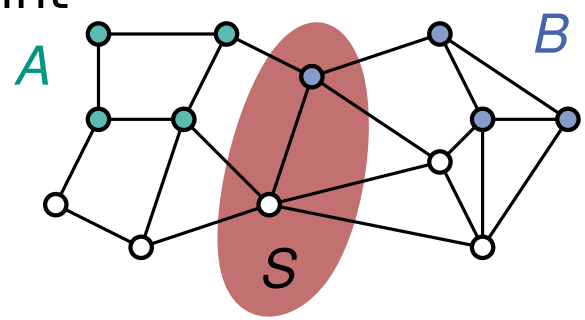




Nachtrag: Separator mittels Fluss

Situation

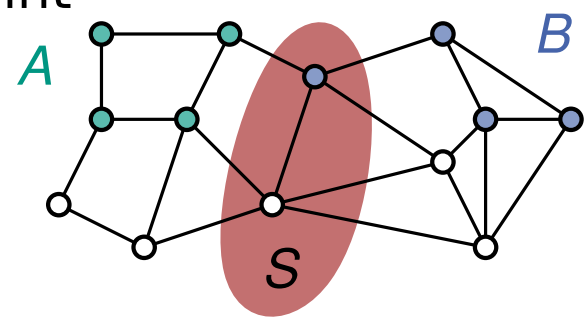
- gegeben: Graph $G = (V, E)$ und zwei Knotenmengen $A, B \subseteq V$ $(A \cap B = \emptyset)$
- finde kleinen Separator S in G, der A und B trennt
- S darf Knoten aus A und B enthalten



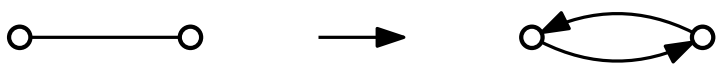
Nachtrag: Separator mittels Fluss

Situation

- gegeben: Graph $G = (V, E)$ und zwei Knotenmengen $A, B \subseteq V$ $(A \cap B = \emptyset)$
- finde kleinen Separator S in G, der A und B trennt
- S darf Knoten aus A und B enthalten



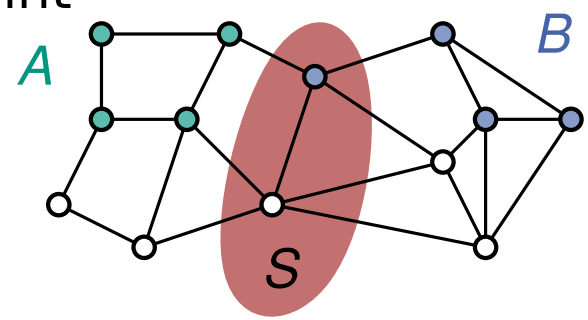
Schritt 1: betrachte gerichteten Graphen



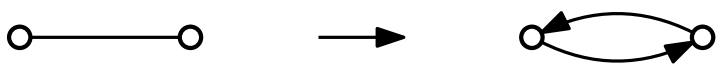
Nachtrag: Separator mittels Fluss

Situation

- gegeben: Graph $G = (V, E)$ und zwei Knotenmengen $A, B \subseteq V$ $(A \cap B = \emptyset)$
- finde kleinen Separator S in G, der A und B trennt
- S darf Knoten aus A und B enthalten



Schritt 1: betrachte gerichteten Graphen

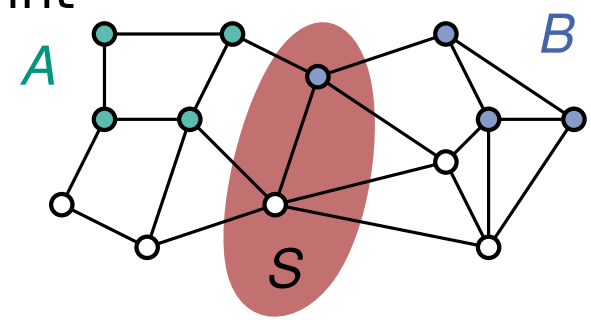


Schritt 2: Superquelle und -senke für A und B

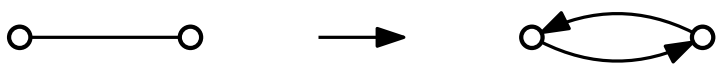
Nachtrag: Separator mittels Fluss

Situation

- gegeben: Graph $G = (V, E)$ und zwei Knotenmengen $A, B \subseteq V$ $(A \cap B = \emptyset)$
- finde kleinen Separator S in G, der A und B trennt
- S darf Knoten aus A und B enthalten

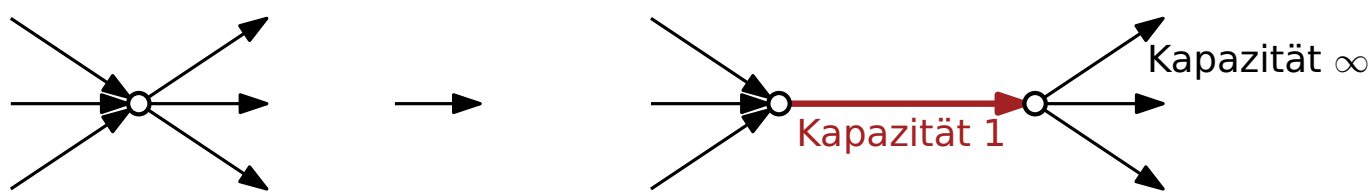


Schritt 1: betrachte gerichteten Graphen



Schritt 2: Superquelle und -senke für A und B

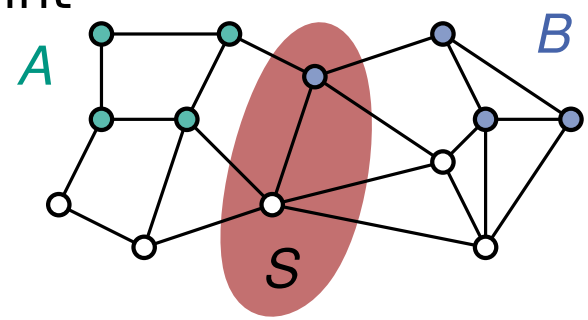
Schritt 3: Knotenkapazitäten durch Aufspaltung jedes Knotens in zwei



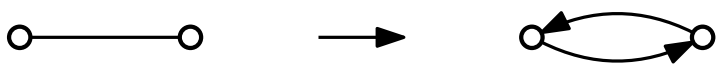
Nachtrag: Separator mittels Fluss

Situation

- gegeben: Graph $G = (V, E)$ und zwei Knotenmengen $A, B \subseteq V$ $(A \cap B = \emptyset)$
- finde kleinen Separator S in G, der A und B trennt
- S darf Knoten aus A und B enthalten

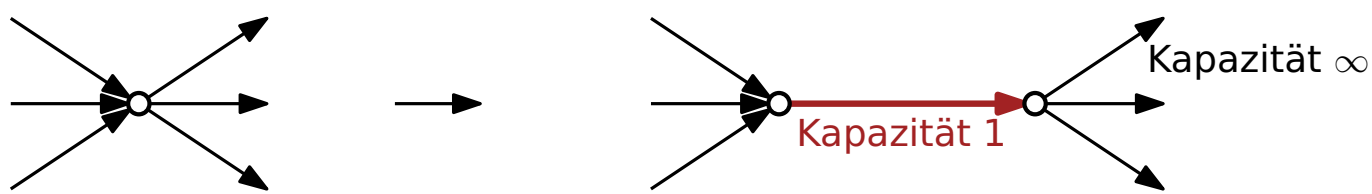


Schritt 1: betrachte gerichteten Graphen



Schritt 2: Superquelle und -senke für A und B

Schritt 3: Knotenkapazitäten durch Aufspaltung jedes Knotens in zwei



Schritt 4: maximaler *ab*-Fluss (minimaler *ab*-Schnitt)

Courcelles Theorem

Vorherige Vorlesungen

- dynamische Programme über Baumzerlegungen

Courcelles Theorem

Vorherige Vorlesungen

- dynamische Programme über Baumzerlegungen
- liefert FPT-Algorithmus (bezgl. Baumweite) für sehr viele Probleme:

Vertex Cover MaxCut Steiner Tree Hamilton Cycle Longest Cycle Cycle Packing Dominating Set Odd Cycle Transversal Longest Path Connected Dominating Set Independent Set Feedback Vertex Set Hamilton Path Connected Vertex Cover Chromatic Number Connected Feedback Vertex Set

Courcelles Theorem

Vorherige Vorlesungen

- dynamische Programme über Baumzerlegungen
- liefert FPT-Algorithmus (bezgl. Baumweite) für sehr viele Probleme:

Vertex Cover MaxCut Steiner Tree Hamilton Cycle Longest Cycle Cycle Packing Dominating Set Odd Cycle Transversal Longest Path Connected Dominating Set Independent Set Feedback Vertex Set Hamilton Path Connected Vertex Cover Chromatic Number Connected Feedback Vertex Set

Heute

- Metatheorem der Form: wenn ein Problem Eigenschaft *XY* hat, dann gibt es einen FPT-Algorithmus (bezgl. Baumweite)

Courcelles Theorem

Vorherige Vorlesungen

- dynamische Programme über Baumzerlegungen
- liefert FPT-Algorithmus (bezgl. Baumweite) für sehr viele Probleme:

Vertex Cover MaxCut Steiner Tree Hamilton Cycle Longest Cycle Cycle Packing Dominating Set Odd Cycle Transversal Longest Path Connected Dominating Set Independent Set Feedback Vertex Set Hamilton Path Connected Vertex Cover Chromatic Number Connected Feedback Vertex Set

Heute

- Metatheorem der Form: wenn ein Problem Eigenschaft *XY* hat, dann gibt es einen FPT-Algorithmus (bezgl. Baumweite)
- Theorem wird hier nicht bewiesen

Courcelles Theorem

Vorherige Vorlesungen

- dynamische Programme über Baumzerlegungen
- liefert FPT-Algorithmus (bezgl. Baumweite) für sehr viele Probleme:

Vertex Cover MaxCut Steiner Tree Hamilton Cycle Longest Cycle Cycle Packing Dominating Set Odd Cycle Transversal Longest Path Connected Dominating Set Independent Set Feedback Vertex Set Hamilton Path Connected Vertex Cover Chromatic Number Connected Feedback Vertex Set

Heute

- Metatheorem der Form: wenn ein Problem Eigenschaft *XY* hat, dann gibt es einen FPT-Algorithmus (bezgl. Baumweite)
- Theorem wird hier nicht bewiesen
- aber ich verrate euch, was *XY* ist

$\mathbf{MSO_2}$ auf Graphen

(MSO = Monadische Prädikatenlogik zweiter Ordnung)

Beispiel: Was drückt die folgende Formel aus?

- $G = (V, E)$ ist ein Graph und $X \subseteq V$
- $\text{inc}(v, e)$ für $v \in V$ und $e \in E$ ist wahr $\Leftrightarrow$ v ist ein Endpunkt von e

$$A(X) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in X} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X} \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \Big]$$

$\mathbf{MSO_2}$ auf Graphen

(MSO = Monadische Prädikatenlogik zweiter Ordnung)

Beispiel: Was drückt die folgende Formel aus?

- $G = (V, E)$ ist ein Graph und $X \subseteq V$
- $\text{inc}(v, e)$ für $v \in V$ und $e \in E$ ist wahr $\Leftrightarrow$ v ist ein Endpunkt von e

$$A(X) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in X} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X} \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \Big]$$

- Lösung: $A(X)$ = wahr $\Leftrightarrow$ $X \subseteq V$ induziert zusammenh. Graphen

$\mathbf{MSO_2}$ auf Graphen

(MSO = Monadische Prädikatenlogik zweiter Ordnung)

Beispiel: Was drückt die folgende Formel aus?

- $G = (V, E)$ ist ein Graph und $X \subseteq V$
- $\text{inc}(v, e)$ für $v \in V$ und $e \in E$ ist wahr $\Leftrightarrow$ v ist ein Endpunkt von e

$$A(X) = \forall_{Y \subseteq V} \Big[(\exists_{u,v \in X} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X} \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \Big]$$

- Lösung: $A(X) =$ wahr $\Leftrightarrow$ $X \subseteq V$ induziert zusammenh. Graphen

Was erlaubt $\mathbf{MSO_2}$?

$\mathbf{MSO_2}$ auf Graphen

(MSO = Monadische Prädikatenlogik zweiter Ordnung)

Beispiel: Was drückt die folgende Formel aus?

- $G = (V, E)$ ist ein Graph und $X \subseteq V$
- $\text{inc}(v, e)$ für $v \in V$ und $e \in E$ ist wahr $\Leftrightarrow$ v ist ein Endpunkt von e

$$A(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in X} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X} \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \big]$$

- Lösung: $A(X) =$ wahr $\Leftrightarrow$ $X \subseteq V$ induziert zusammenh. Graphen

Was erlaubt $\mathbf{MSO_2}$?

Second Order: zusätzlich zur Quantifizierung über Elemente (first order) auch über Relationen

first order: $\forall_{v \in V}$

second order: $\forall_{Y \subseteq V}$ $\forall_{Y \subseteq V \times V}$ $\forall_{Y \subseteq V \times V \times V}$

$\mathbf{MSO_2}$ auf Graphen

(MSO = Monadische Prädikatenlogik zweiter Ordnung)

Beispiel: Was drückt die folgende Formel aus?

- $G = (V, E)$ ist ein Graph und $X \subseteq V$
- $\text{inc}(v, e)$ für $v \in V$ und $e \in E$ ist wahr $\Leftrightarrow$ v ist ein Endpunkt von e

$$A(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in X} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X} \text{inc}(u, e) \wedge \text{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \big]$$

- Lösung: $A(X) =$ wahr $\Leftrightarrow$ $X \subseteq V$ induziert zusammenh. Graphen

Was erlaubt $\mathbf{MSO_2}$?

Second Order: zusätzlich zur Quantifizierung über Elemente (first order) auch über Relationen

first order: $\forall_{v \in V}$

second order: $\forall_{Y \subseteq V}$ ~~$\forall_{Y \subseteq V \times V}$~~ ~~$\forall_{Y \subseteq V \times V \times V}$~~

Monadic: nur einstellig Relationen

MSO_2 auf Graphen

(MSO = Monadische Prädikatenlogik zweiter Ordnung)

Beispiel: Was drückt die folgende Formel aus?

- $G = (V, E)$ ist ein Graph und $X \subseteq V$
- $\mathrm{inc}(v, e)$ für $v \in V$ und $e \in E$ ist wahr $\Leftrightarrow$ v ist ein Endpunkt von e

$$A(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in X} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in E} \exists_{u,v \in X} \mathrm{inc}(u, e) \wedge \mathrm{inc}(v, e) \wedge u \in Y \wedge v \notin Y) \big]$$

- Lösung: $A(X) =$ wahr $\Leftrightarrow$ $X \subseteq V$ induziert zusammenh. Graphen

Was erlaubt MSO_2?

Second Order: zusätzlich zur Quantifizierung über Elemente (first order) auch über Relationen

first order: $\forall_{v \in V}$

second order: $\forall_{Y \subseteq V}$ ~~$\forall_{Y \subseteq V \times V}$~~ ~~$\forall_{Y \subseteq V \times V \times V}$~~

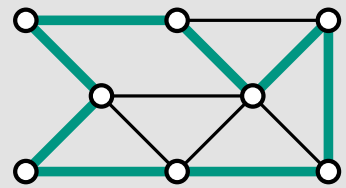
Monadic: nur einstellig Relationen

$_2$: Quantifizierung über V und über E erlaubt $\forall_{v \in V}$ $\forall_{e \in E}$ $\forall_{V' \subseteq V}$ $\forall_{E' \subseteq E}$

(MSO_1 erlaubt nur Quantifizierung über V)

Probleme in MSO_2

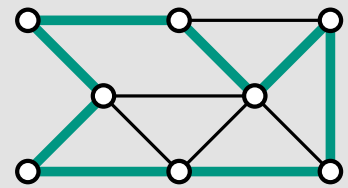
Problem: HAMILTONKREIS
Gibt es in einem gegebenen Graphen einen Kreis, der alle Knoten enthält?



Drücke die Existenz eines Hamiltonkreises in MSO_2 aus

Probleme in MSO_2

Problem: HAMILTONKREIS
Gibt es in einem gegebenen Graphen einen Kreis, der alle Knoten enthält?

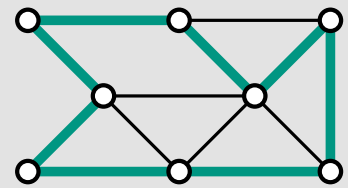


Drücke die Existenz eines Hamiltonkreises in MSO_2 aus

- beachte: $C \subseteq E$ ist Lösung $\Leftrightarrow$ $G[C]$ ist zusammenh. und 2-regulär

Probleme in MSO_2

Problem: HAMILTONKREIS
Gibt es in einem gegebenen Graphen einen Kreis, der alle Knoten enthält?



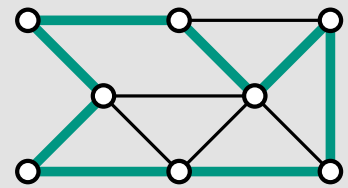
Drücke die Existenz eines Hamiltonkreises in MSO_2 aus

- beachte: $C \subseteq E$ ist Lösung $\Leftrightarrow$ $G[C]$ ist zusammenh. und 2-regulär
- grobe Formel:

$$\text{hamiltonisch} = \exists_{C \subseteq E} \text{conn}(C) \wedge \forall_{v \in V} \text{deg2}(v, C)$$

Probleme in MSO_2

Problem: HAMILTONKREIS
Gibt es in einem gegebenen Graphen einen Kreis, der alle Knoten enthält?



Drücke die Existenz eines Hamiltonkreises in MSO_2 aus

- beachte: $C \subseteq E$ ist Lösung $\Leftrightarrow$ $G[C]$ ist zusammenh. und 2-regulär
- grobe Formel:

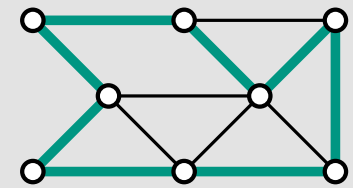
$$\text{hamiltonisch} = \exists_{C \subseteq E} \text{conn}(C) \wedge \forall_{v \in V} \text{deg2}(v, C)$$

- Zusammenhang:

$$\text{conn}(C) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in V} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in C} \exists_{u \in Y} \exists_{v \notin Y} \text{inc}(u, e) \wedge \text{inc}(v, e)) \big]$$

Probleme in MSO_2

Problem: HAMILTONKREIS
Gibt es in einem gegebenen Graphen einen Kreis, der alle Knoten enthält?



Drücke die Existenz eines Hamiltonkreises in MSO_2 aus

- beachte: $C \subseteq E$ ist Lösung $\Leftrightarrow$ $G[C]$ ist zusammenh. und 2-regulär
- grobe Formel:

$$\text{hamiltonisch} = \exists_{C \subseteq E} \text{conn}(C) \wedge \forall_{v \in V} \text{deg2}(v, C)$$

- Zusammenhang:

$$\text{conn}(C) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in V} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{e \in C} \exists_{u \in Y} \exists_{v \notin Y} \text{inc}(u, e) \wedge \text{inc}(v, e)) \big]$$

- Knotengrad 2:

$$\text{deg2}(v, C) = \exists_{e_1, e_2 \in C} \big[e_1 \neq e_2 \wedge \text{inc}(v, e_1) \wedge \text{inc}(v, e_2) \wedge \big(\forall_{e_3 \in C} \text{inc}(v, e_3) \Rightarrow (e_3 = e_1 \vee e_3 = e_2) \big) \big]$$

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Beispiel

- die Eigenschaft einen Hamiltonkreis zu haben kann mit einer konstant großen MSO_2 Formel ausgedrückt werden
- $\Rightarrow$ HAMILTONKREIS hat einen FPT-Algorithmus bezgl. Baumweite

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Beispiel

- die Eigenschaft einen Hamiltonkreis zu haben kann mit einer konstant großen MSO_2 Formel ausgedrückt werden
- $\Rightarrow$ HAMILTONKREIS hat einen FPT-Algorithmus bezgl. Baumweite

Was machen die „freien Variablen“?

- nützlich, falls Eingabe z.B. aus $G = (V, E)$ und $X \subseteq V$ besteht

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Beispiel

- die Eigenschaft einen Hamiltonkreis zu haben kann mit einer konstant großen MSO_2 Formel ausgedrückt werden
- $\Rightarrow$ HAMILTONKREIS hat einen FPT-Algorithmus bezgl. Baumweite

Was machen die „freien Variablen“?

- nützlich, falls Eingabe z.B. aus $G = (V, E)$ und $X \subseteq V$ besteht
- Beispiel: bei STEINER BAUM ist zusätzlich zum Graphen eine Menge von Terminalknoten gegeben

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Beispiel 2: k-Vertex Cover

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Beispiel 2: k-Vertex Cover

- grobe Formel:

$$\text{k-vc} = \exists_{X \subseteq V}(\text{vc}(X) \wedge \text{size-k}(X))$$

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Beispiel 2: k-Vertex Cover

- grobe Formel:

$$\text{k-vc} = \exists_{X \subseteq V}(\text{vc}(X) \wedge \text{size-k}(X))$$

- Abdeckung aller Kanten:

$$\text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \text{inc}(v, e)$$

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Beispiel 2: k-Vertex Cover

- grobe Formel:

$$\text{k-vc} = \exists_{X \subseteq V}(\text{vc}(X) \wedge \text{size-k}(X))$$

- Abdeckung aller Kanten:

$$\text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \text{inc}(v, e)$$

- maximal k Knoten:

$$\text{size-k}(X) = \forall_{v_0,\ldots,v_k \in V} v_0 \notin X \vee \cdots \vee v_k \notin X \vee v_0 = v_1 \vee v_0 = v_2 \vee \cdots \vee v_{k-1} = v_k$$

(für $k+1$ Knoten ist einer nicht in X oder zwei der Knoten sind gleich)

Courcelles Theorem

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel und sei G ein Graph zusammen mit einer Auswertung für alle freien Variablen von φ und mit einer Baumzerlegung der Weite t. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit testet ob φ von G erfüllt wird (für eine berechenbare Funktion f).

Beispiel 2: k-Vertex Cover

- grobe Formel:

$$\text{k-vc} = \exists_{X \subseteq V}(\text{vc}(X) \wedge \text{size-k}(X))$$

- Abdeckung aller Kanten:

$$\text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \text{inc}(v, e)$$

- maximal k Knoten:

$$\text{size-k}(X) = \forall_{v_0,\ldots,v_k \in V} v_0 \notin X \vee \cdots \vee v_k \notin X \vee v_0 = v_1 \vee v_0 = v_2 \vee \cdots \vee v_{k-1} = v_k$$

(für $k+1$ Knoten ist einer nicht in X oder zwei der Knoten sind gleich)

Achtung: $|\varphi| = |\text{k-vc}|$ hängt von k ab

$\Rightarrow$ Courcelles Theorem liefert nur einen FPT-Algorithmus bezgl. beider Parameter k und t

Courcelles Theorem – Optimierung

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel mit p freien monadischen Variablen $X_1, \dots X_p$ und sei $\alpha(x_1, \dots, x_p)$ eine affine Funktion. Sei G ein Graph zusammen mit einer Baumzerlegung der Weite t und mit einer Auswertung für alle freien Variablen von φ außer $X_1, \dots, X_p$. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit Auswertungen für $X_1, \dots X_p$ findet, sodass $\varphi(X_1, \dots, X_p) = \texttt{true}$ und $\alpha(|X_1|, \dots, |X_p|)$ maximal oder minimal.

Erinnerung affine Funktion: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Courcelles Theorem – Optimierung

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel mit p freien monadischen Variablen $X_1, \dots X_p$ und sei $\alpha(x_1, \dots, x_p)$ eine affine Funktion. Sei G ein Graph zusammen mit einer Baumzerlegung der Weite t und mit einer Auswertung für alle freien Variablen von φ außer $X_1, \dots, X_p$. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit Auswertungen für $X_1, \dots X_p$ findet, sodass $\varphi(X_1, \dots, X_p) = \texttt{true}$ und $\alpha(|X_1|, \dots, |X_p|)$ maximal oder minimal.

Beispiel: Vertex Cover

Erinnerung affine Funktion: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Courcelles Theorem – Optimierung

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel mit p freien monadischen Variablen $X_1, \dots X_p$ und sei $\alpha(x_1, \dots, x_p)$ eine affine Funktion. Sei G ein Graph zusammen mit einer Baumzerlegung der Weite t und mit einer Auswertung für alle freien Variablen von φ außer $X_1, \dots, X_p$. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit Auswertungen für $X_1, \dots X_p$ findet, sodass $\varphi(X_1, \dots, X_p) = \texttt{true}$ und $\alpha(|X_1|, \dots, |X_p|)$ maximal oder minimal.

Beispiel: Vertex Cover

- eine freie Variable $X \subseteq V$
- Ziel 1: X ist Vertex Cover: $\varphi(X) = \text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \text{inc}(v, e)$

Erinnerung affine Funktion: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Courcelles Theorem – Optimierung

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel mit p freien monadischen Variablen $X_1, \dots X_p$ und sei $\alpha(x_1, \dots, x_p)$ eine affine Funktion. Sei G ein Graph zusammen mit einer Baumzerlegung der Weite t und mit einer Auswertung für alle freien Variablen von φ außer $X_1, \dots, X_p$. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit Auswertungen für $X_1, \dots X_p$ findet, sodass $\varphi(X_1, \dots, X_p) = \texttt{true}$ und $\alpha(|X_1|, \dots, |X_p|)$ maximal oder minimal.

Beispiel: Vertex Cover

- eine freie Variable $X \subseteq V$
- Ziel 1: X ist Vertex Cover: $\varphi(X) = \text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \text{inc}(v, e)$
- Ziel 2: X ist minimal unter allen Vertex Covern: $\alpha(x) = x$

Erinnerung affine Funktion: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Courcelles Theorem – Optimierung

Theorem (ohne Beweis)
Sei φ eine MSO_2 Formel mit p freien monadischen Variablen $X_1, \dots X_p$ und sei $\alpha(x_1, \dots, x_p)$ eine affine Funktion. Sei G ein Graph zusammen mit einer Baumzerlegung der Weite t und mit einer Auswertung für alle freien Variablen von φ außer $X_1, \dots, X_p$. Dann gibt es einen Algorithmus, der in $f(|\varphi|, t) \cdot n$ Zeit Auswertungen für $X_1, \dots X_p$ findet, sodass $\varphi(X_1, \dots, X_p) = \texttt{true}$ und $\alpha(|X_1|, \dots, |X_p|)$ maximal oder minimal.

Beispiel: Vertex Cover

- eine freie Variable $X \subseteq V$
- Ziel 1: X ist Vertex Cover: $\varphi(X) = \text{vc}(X) = \forall_{e \in E} \exists_{v \in X} \text{inc}(v, e)$
- Ziel 2: X ist minimal unter allen Vertex Covern: $\alpha(x) = x$

$\Rightarrow |\varphi|$ ist konstant

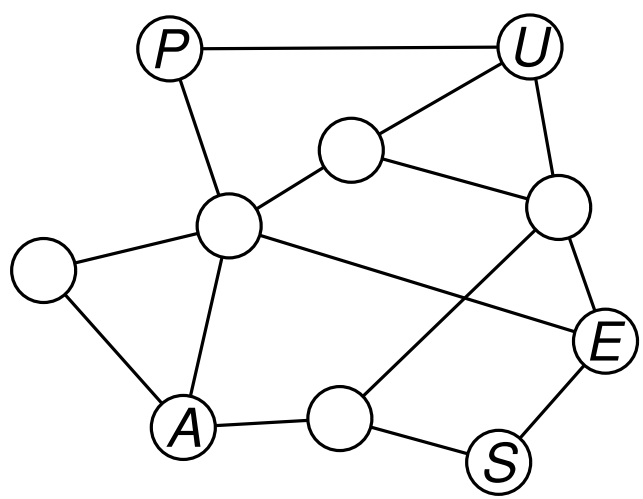
$\Rightarrow$ FPT-Algorithmus für Vertex Cover bezgl. Baumweite

Erinnerung affine Funktion: $\alpha(x_1, \dots, x_p) = a_0 + \sum_{i=1}^{p} a_i x_i$

Was ist das Problem?

Noch ein Beispiel

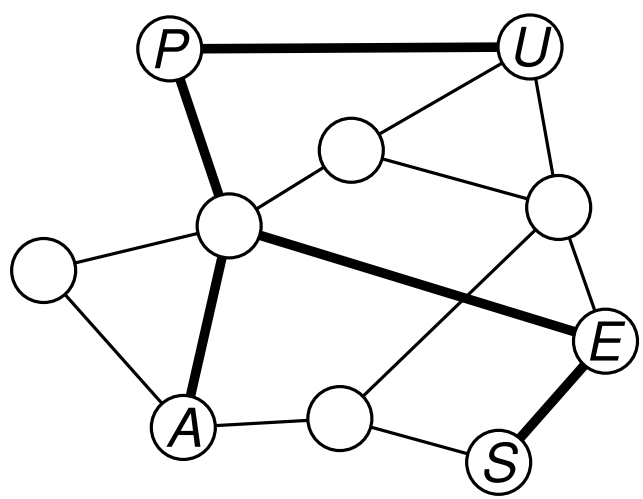
- $G = (V, E)$ ist der gezeigte Graph und $T = \{P, A, U, S, E\} \subseteq V$
- $\varphi(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in T} u \in Y \wedge v \notin Y)$
  $\Rightarrow (\exists_{u,v \in V} \exists_{e \in X} u \in Y \wedge v \notin Y \wedge \text{inc}(u, e) \wedge \text{inc}(v, e))\big]$
- $\alpha(x) = x$
- **Aufgabe:** finde $X \subseteq E$, sodass $\alpha(|X|)$ minimal ist



Was ist das Problem?

Noch ein Beispiel

- $G = (V, E)$ ist der gezeigte Graph und $T = \{P, A, U, S, E\} \subseteq V$
- $\varphi(X) = \forall_{Y \subseteq V} \big[(\exists_{u,v \in T} u \in Y \wedge v \notin Y) \Rightarrow (\exists_{u,v \in V} \exists_{e \in X} u \in Y \wedge v \notin Y \wedge \mathrm{inc}(u, e) \wedge \mathrm{inc}(v, e))\big]$
- $\alpha(x) = x$
- **Aufgabe:** finde $X \subseteq E$, sodass $\alpha(|X|)$ minimal ist

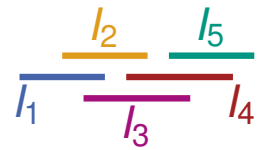


- **Lösung:** zusammenh. Teilgraph mit minimaler Kantenzahl, der alle Knoten aus T enthält $\Rightarrow$ Steiner Baum

Intervallgraphen

Definition

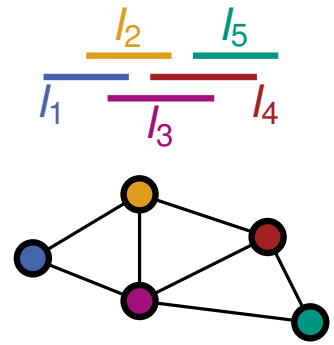
- Menge von n Intervallen $\{I_1, \ldots, I_n\}$ in $\mathbb{R}$



Intervallgraphen

Definition

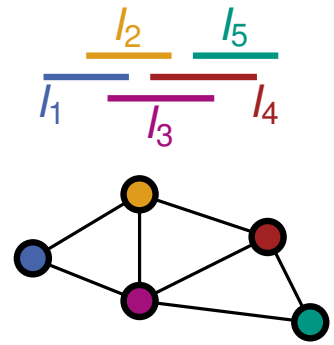
- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$



Intervallgraphen

Definition

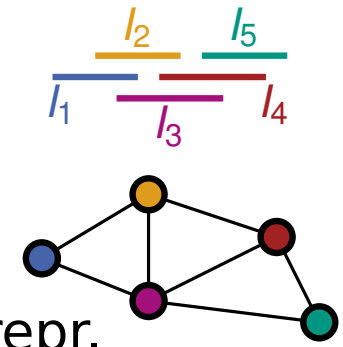
- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ heißt **Intervallrepräsentation** von G



Intervallgraphen

Definition

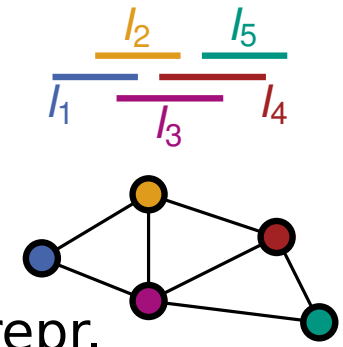
- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ heißt **Intervallrepräsentation** von G
- ein Graph ist ein **Intervallgraph** $\Leftrightarrow$ er hat eine Intervallrepr.



Intervallgraphen

Definition

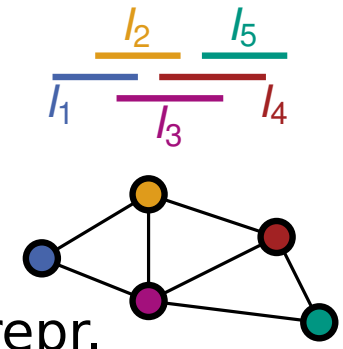
- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ heißt **Intervallrepräsentation** von G
- ein Graph ist ein **Intervallgraph** $\Leftrightarrow$ er hat eine Intervallrepr.



Was ist die Pfadweite eines Intervallgraphen?

Intervallgraphen

Definition



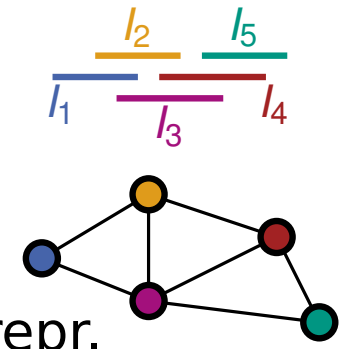
- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ heißt **Intervallrepräsentation** von G
- ein Graph ist ein **Intervallgraph** $\Leftrightarrow$ er hat eine Intervallrepr.

Was ist die Pfadweite eines Intervallgraphen?

- sei $\omega(G)$ die Cliquengröße und $\mathrm{pw}(G)$ die Pfadweite von G

Intervallgraphen

Definition



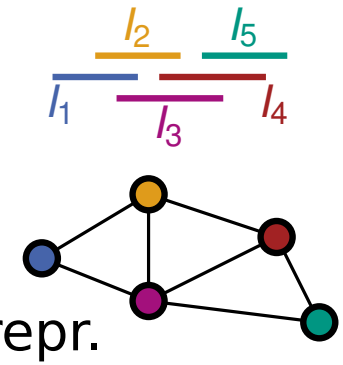
- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ heißt **Intervallrepräsentation** von G
- ein Graph ist ein **Intervallgraph** $\Leftrightarrow$ er hat eine Intervallrepr.

Was ist die Pfadweite eines Intervallgraphen?

- sei $\omega(G)$ die Cliquengröße und $\mathrm{pw}(G)$ die Pfadweite von G
- es gilt: $\mathrm{pw}(G) \leq \omega(G) - 1$ (Intervallrepräsentation liefert Pfadzerlegung)

Intervallgraphen

Definition



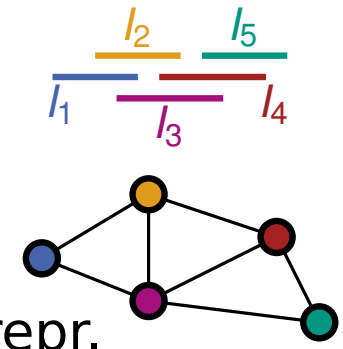
- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ heißt **Intervallrepräsentation** von G
- ein Graph ist ein **Intervallgraph** $\Leftrightarrow$ er hat eine Intervallrepr.

Was ist die Pfadweite eines Intervallgraphen?

- sei $\omega(G)$ die Cliquengröße und $\mathrm{pw}(G)$ die Pfadweite von G
- es gilt: $\mathrm{pw}(G) \leq \omega(G) - 1$ (Intervallrepräsentation liefert Pfadzerlegung)
- und: $\mathrm{pw}(G) \geq \omega(G) - 1$ (Knoten einer Clique teilen sich eine Bag)

Intervallgraphen

Definition



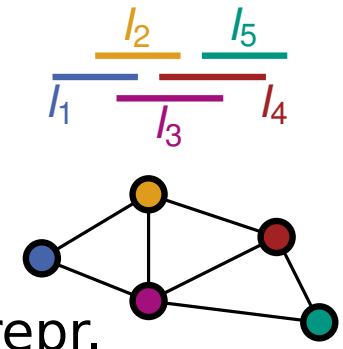
- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ heißt **Intervallrepräsentation** von G
- ein Graph ist ein **Intervallgraph** $\Leftrightarrow$ er hat eine Intervallrepr.

Was ist die Pfadweite eines Intervallgraphen?

- sei $\omega(G)$ die Cliquengröße und $\mathrm{pw}(G)$ die Pfadweite von G
- es gilt: $\mathrm{pw}(G) \leq \omega(G) - 1$ (Intervallrepräsentation liefert Pfadzerlegung)
- und: $\mathrm{pw}(G) \geq \omega(G) - 1$ (Knoten einer Clique teilen sich eine Bag)
- $\Rightarrow \mathrm{pw}(G) = \omega(G) - 1$

Intervallgraphen

Definition



- Menge von n Intervallen $\{I_1, \dots, I_n\}$ in $\mathbb{R}$
- Graph $G = (V, E)$, $V = \{v_1, \dots, v_n\}$, $E = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$
- $I_1, \dots, I_n$ heißt **Intervallrepräsentation** von G
- ein Graph ist ein **Intervallgraph** $\Leftrightarrow$ er hat eine Intervallrepr.

Was ist die Pfadweite eines Intervallgraphen?

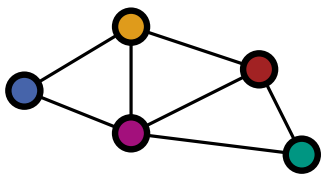
- sei $\omega(G)$ die Cliquengröße und $\text{pw}(G)$ die Pfadweite von G
- es gilt: $\text{pw}(G) \leq \omega(G) - 1$ (Intervallrepräsentation liefert Pfadzerlegung)
- und: $\text{pw}(G) \geq \omega(G) - 1$ (Knoten einer Clique teilen sich eine Bag)
- $\Rightarrow \text{pw}(G) = \omega(G) - 1$

Intervallweite

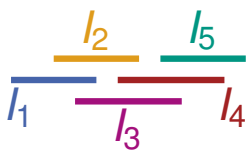
- $\text{interval-width}(G) = \min\{\omega(G') \mid G \subseteq G' \text{ und } G' \text{ ist Intervallgraph}\}$
- $\Rightarrow \text{pw}(G) = \text{interval-width}(G) - 1$

Chordale Graphen

Definition
Ein Graph G ist **chordal**, wenn er eine Schnittrepräsentation mit Teilbäumen eines Baumes hat. (statt Teilpfade eines Pfades, wie bei Intervallgraphen)

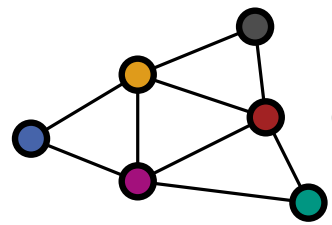


Intervallgraph

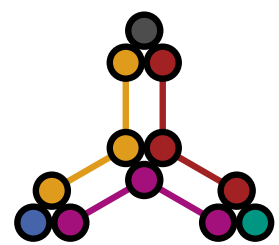


Intervallrepräsentation

Schnittrepräsentation mit Teilpfaden eines Pfads



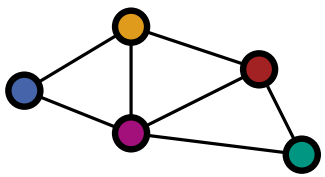
chordaler Graph



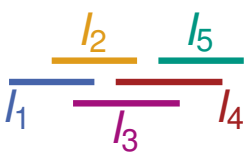
Schnittrepräsentation mit Teilbäumen eines Baums

Chordale Graphen

Definition
Ein Graph G ist **chordal**, wenn er eine Schnittrepräsentation mit Teilbäumen eines Baumes hat. (statt Teilpfade eines Pfades, wie bei Intervallgraphen)

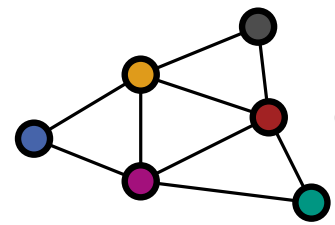


Intervallgraph

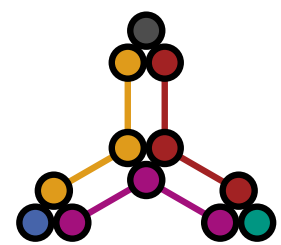


Intervallrepräsentation

Schnittrepräsentation mit Teilpfaden eines Pfads



chordaler Graph



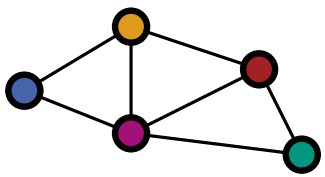
Schnittrepräsentation mit Teilbäumen eines Baums

Äquivalente Definition

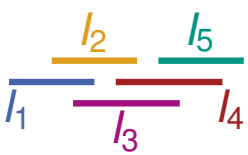
- G chordal $\Leftrightarrow$ G hat keinen induzierten Kreis der Länge 4 oder mehr

Chordale Graphen

Definition
Ein Graph G ist **chordal**, wenn er eine Schnittrepräsentation mit Teilbäumen eines Baumes hat. (statt Teilpfade eines Pfades, wie bei Intervallgraphen)

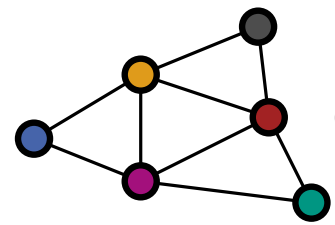


Intervallgraph

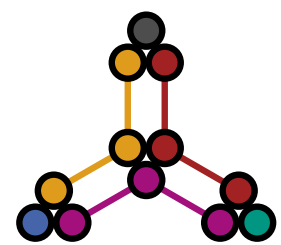


Intervallrepräsentation

Schnittrepräsentation mit Teilpfaden eines Pfads



chordaler Graph



Schnittrepräsentation mit Teilbäumen eines Baums

Äquivalente Definition

- G chordal $\Leftrightarrow$ G hat keinen induzierten Kreis der Länge 4 oder mehr

Chordalweite

- chordal-width$(G) = \min\{\omega(G') \mid G \subseteq G' \text{ und } G' \text{ ist chordal}\}$
- $\Rightarrow$ tw(G) = chordal-width$(G) - 1$

Zusammenfassung

Courcelles Theorem

- Problem in MSO_2 ausdrückbar $\Rightarrow$ FPT bezüglich Baumweite

Zusammenfassung

Courcelles Theorem

- Problem in MSO_2 ausdrückbar $\Rightarrow$ FPT bezüglich Baumweite
- mächtiges Tool, da sich viele Probleme in MSO_2 ausdrücken lassen

Zusammenfassung

Courcelles Theorem

- Problem in MSO_2 ausdrückbar $\Rightarrow$ FPT bezüglich Baumweite
- mächtiges Tool, da sich viele Probleme in MSO_2 ausdrücken lassen
- **Achtung:** Courcelle liefert nur sehr schlechte Laufzeiten:

Zusammenfassung

Courcelles Theorem

- Problem in MSO_2 ausdrückbar $\Rightarrow$ FPT bezüglich Baumweite
- mächtiges Tool, da sich viele Probleme in MSO_2 ausdrücken lassen
- **Achtung:** Courcelle liefert nur sehr schlechte Laufzeiten:
  - $f(k)$ ist mehrfach exponentiell; denk an: $2^{2^{\cdots^{2^k}}}$

Zusammenfassung

Courcelles Theorem

- Problem in MSO_2 ausdrückbar $\Rightarrow$ FPT bezüglich Baumweite
- mächtiges Tool, da sich viele Probleme in MSO_2 ausdrücken lassen
- **Achtung:** Courcelle liefert nur sehr schlechte Laufzeiten:
  - $f(k)$ ist mehrfach exponentiell; denk an: $2^{2^{\cdots^{2^k}}}$
  - Höhe des Potenzturms nicht durch eine Konstante beschränkt
    (linear in der Anzahl alternierender $\forall$ und $\exists$ Quantoren)

Zusammenfassung

Courcelles Theorem

- Problem in MSO_2 ausdrückbar $\Rightarrow$ FPT bezüglich Baumweite
- mächtiges Tool, da sich viele Probleme in MSO_2 ausdrücken lassen
- **Achtung:** Courcelle liefert nur sehr schlechte Laufzeiten:
  - $f(k)$ ist mehrfach exponentiell; denk an: $2^{2^{\cdots^{2^k}}}$
  - Höhe des Potenzturms nicht durch eine Konstante beschränkt
    (linear in der Anzahl alternierender $\forall$ und $\exists$ Quantoren)

Chordale Graphen

- Baumweite = minimale Cliquenzahl von chordalem Supergraphen

Zusammenfassung

Courcelles Theorem

- Problem in MSO_2 ausdrückbar $\Rightarrow$ FPT bezüglich Baumweite
- mächtiges Tool, da sich viele Probleme in MSO_2 ausdrücken lassen
- **Achtung:** Courcelle liefert nur sehr schlechte Laufzeiten:
  - $f(k)$ ist mehrfach exponentiell; denk an: $2^{2^{\cdots^{2^k}}}$
  - Höhe des Potenzturms nicht durch eine Konstante beschränkt
    (linear in der Anzahl alternierender $\forall$ und $\exists$ Quantoren)

Chordale Graphen

- Baumweite = minimale Cliquenzahl von chordalem Supergraphen
- übrigens: die Cliquenzahl eines chordalen Graphen (und eine entsprechende Baumzerlegung) kann man leicht ausrechnen

Literaturhinweise

The monadic second-order logic of graphs. I. Recognizable sets of finite graphs

- Bruno Courcelle [1990]
- Courcelles Theorem

https://doi.org/10.1016/0890-5401(90)90043-H

Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families

- Richard B. Borie, R. Gary Parker, Craig A. Tovey [1992]
- unabhängige Wiederentdeckung

https://doi.org/10.1007/BF01758777

Courcelle's theorem—A game-theoretic approach

- Joachim Kneis, Alexander Langer, Peter Rossmanith [2011]
- Courcelles Theorem in der „Praxis“

https://doi.org/10.1016/j.disopt.2011.06.001