Kombinatorische Optimierung auf Graphen mit beschränkter Baumweite

Luca Oeljeklaus

13. Juli 2018

Übersicht

Kombinatorische Optimierung

- Gegeben durch ein Paar (L, f).
- L eine Menge von Elementen (z.B. Knoten, Kanten, etc.).
- $f : L \to S$ eine Zielfunktion.
- Ziel: Teilmenge von L wählen, die die Zielfunktion maximiert (bzw. minimiert).

Beispiele:

- Problem des Handlungsreisenden (TSP)
- Behälterproblem (BPP)
- Minimaler Spannbaum (MST)
- Größte Stabile Menge (MIS)
- Steinerbaumproblem

Anwendungen von Maximum Independent Set

- Scheduling
- Kombinatorischen Auktionen
- Kartenbeschriftung
- Molekulare Biologie
- Pattern Recognition

Baumzerlegung

Definition: Baumzerlegung

$(\{ X_i \mid i \in I \}, T = (I, F))$

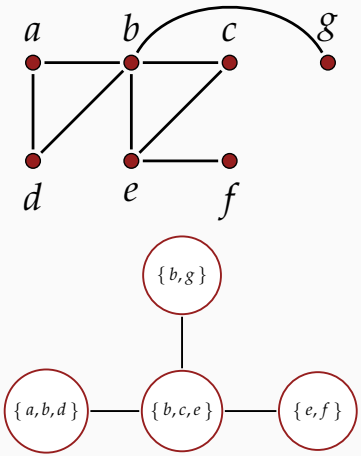
- $\bigcup_{i \in I} X_i = V$.
- $\forall (v, w) \in E, \exists i \in I : v, w \in X_i$.
- $\forall v \in V$ induziert $\{ i \in I \mid v \in X_i \}$ einen Teilbaum in T.

Baumzerlegung

Definition: Baumzerlegung

$(\{ X_i \mid i \in I \}, T = (I, F))$

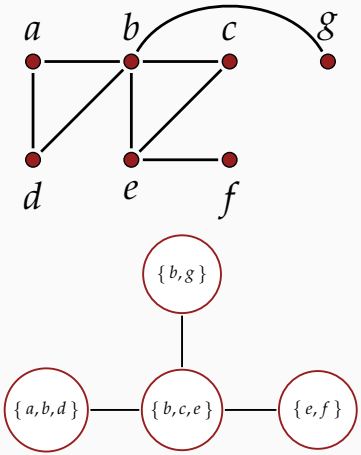
- $\bigcup_{i \in I} X_i = V$.
- $\forall (v, w) \in E, \exists i \in I : v, w \in X_i$.
- $\forall v \in V$ induziert $\{ i \in I \mid v \in X_i \}$ einen Teilbaum in T.



Baumweite

Definition: Baumweite

$\mathrm{BW} = \min_{\mathrm{BZ}} \max_{i \in I} |X_i| - 1$

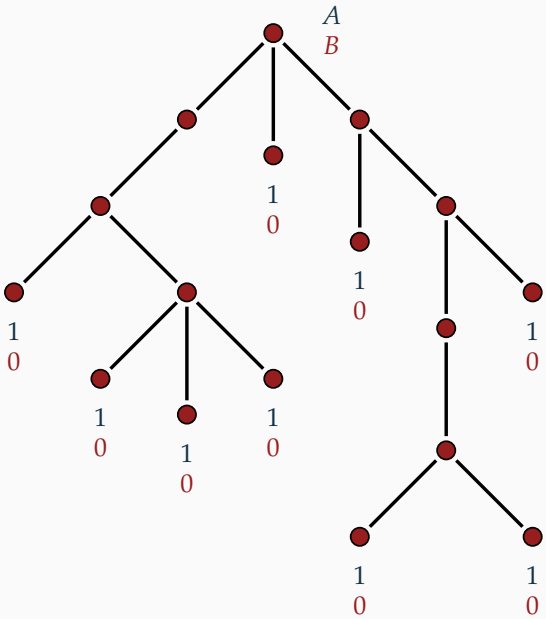


MIS auf Bäumen

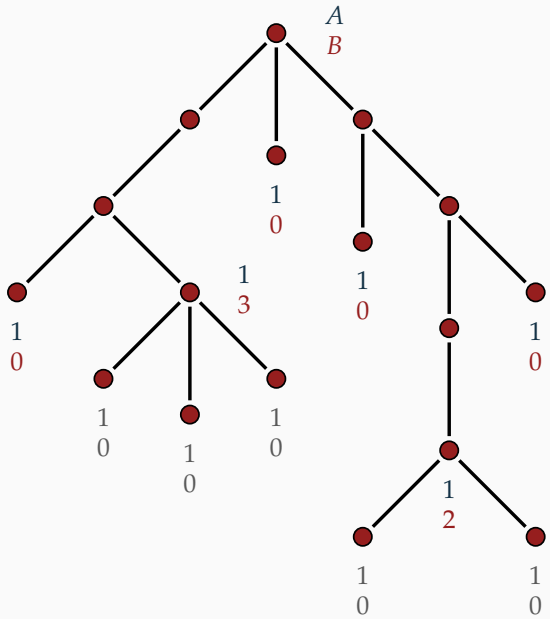
MIS auf Bäumen

```
data Tree = Tree [Tree] | Leaf

mis, computeA, computeB :: Tree -> [Tree]

mis x = max (computeA x) (computeB x)

computeA x = case x of
        Leaf -> [Leaf]
        Tree children ->
            x : map computeB children

computeB x = case x of
        Leaf -> [Leaf]
        Tree children ->
            x : map mis children
```

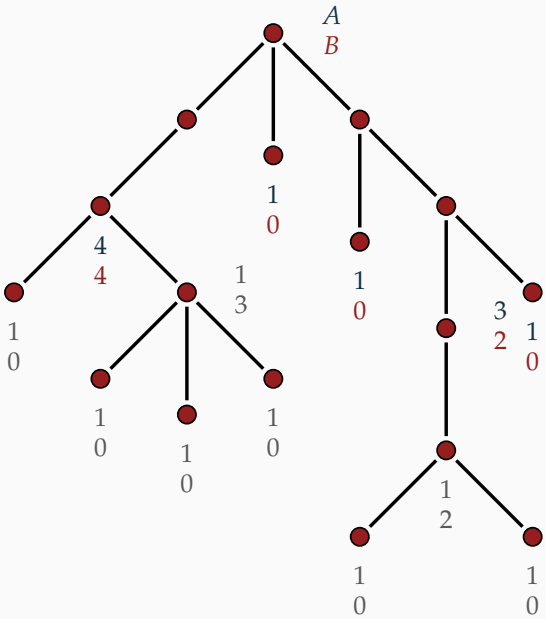
Beispiel



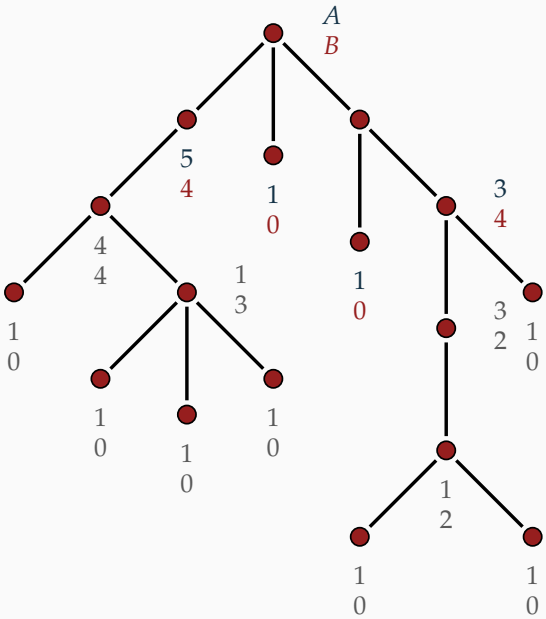
Beispiel



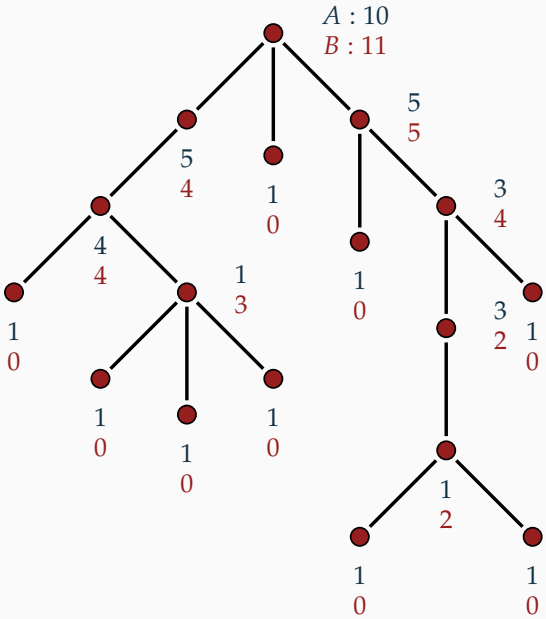
Beispiel



Beispiel

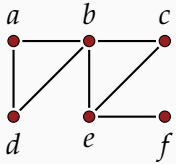


Beispiel



MIS auf SP-Graphen

SP-Graph



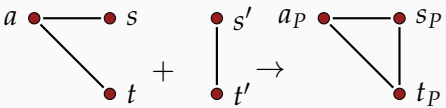
Definition: SP-Graph

Ein *SP-Graph* ist gegeben durch ein Tupel (G, s, t):

- $G = (V, E)$, einem ungerichteten Graphen.
- s, einer Quelle.
- t, einer Senke.

Ein Graph ist ein SP-Graph falls er durch rekursives parallelisieren und serialisieren entstehen kann.

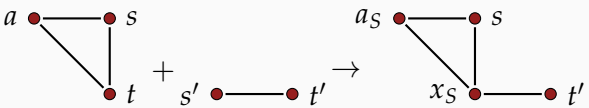
Parallelisierung



Definition: Parallelisierung

Beide Quellen und beide Senken miteinander fusionieren.

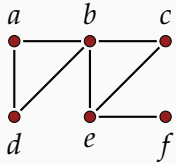
Serialisierung

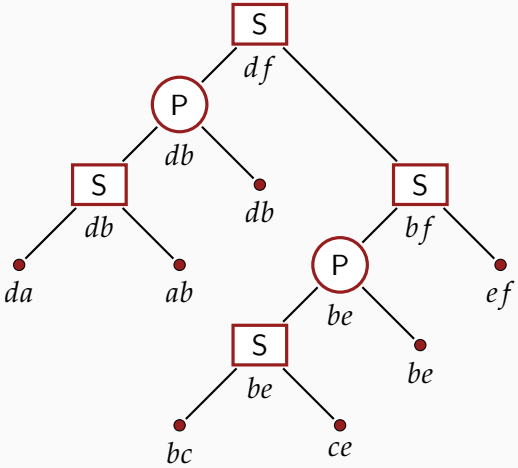


Definition: Serialisierung

Die Senke des einen Graphen mit der Quelle des Anderen fusionieren.

SP-Baum





MIS auf SP-Graphen

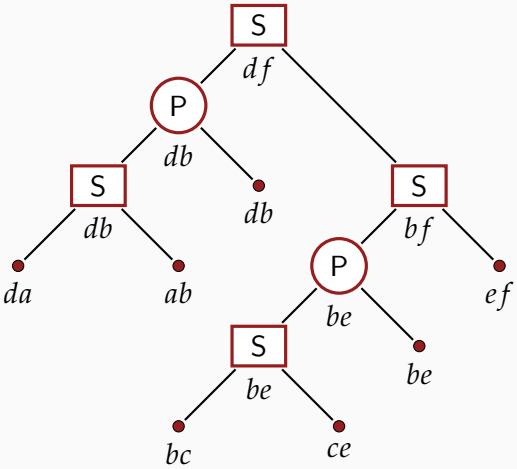
Berechne Rekursiv vier Werte:

- AA, Größe des MIS das sowohl s als auch t enthält.
- AB, Größe des MIS das s aber nicht t enthält.
- BA, Größe des MIS das nicht s aber t enthält.
- BB, Größe des MIS das weder s noch t enthält.

Werte für Blätter

Die Werte sind für Blätter trivial:

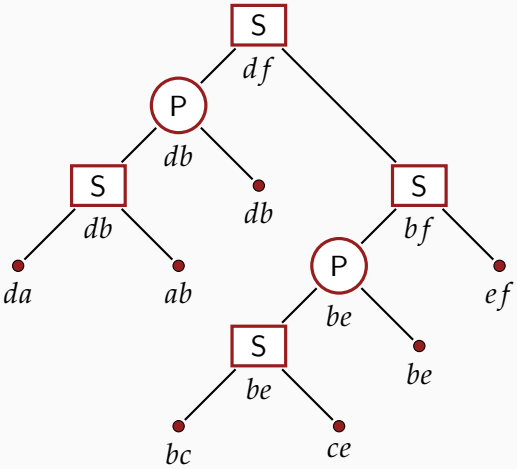
- $AA = -\infty$
- $AB = 1$
- $BA = 1$
- $BB = 0$



Werte für S Knoten

Die Werte für S-Knoten sind:

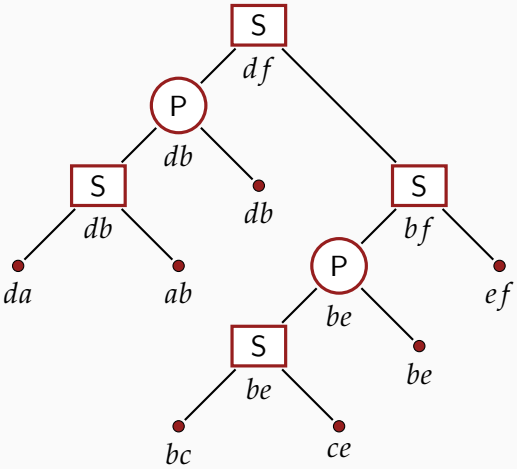
- $AA = \max\{AA(l) + AA(r) - 1, AB(l) + BA(r)\}$
- $AB = \max\{AA(l) + AB(r) - 1, AB(l) + BB(r)\}$
- $BA = \max\{BA(l) + AA(r) - 1, BB(l) + BA(r)\}$
- $BB = \max\{BA(l) + AB(r) - 1, BB(l) + BB(r)\}$



Werte für P Knoten

Die Werte für P-Knoten sind:

- $AA = AA(l) + AA(r) - 2$
- $AB = AB(l) + AB(r) - 1$
- $BA = BA(l) + BA(r) - 1$
- $BB = BB(l) + BB(r)$



MIS auf Graphen mit geringer Baumweite

Schöne Baumzerlegung: Definition

Definition: Schöne Baumzerlegung

Eine *Schöne Baumzerlegung* (*nice tree decomposition*) ist eine Baumzerlegung sodass jeder Baumknoten eine der folgenden Eigenschaften besitzt:

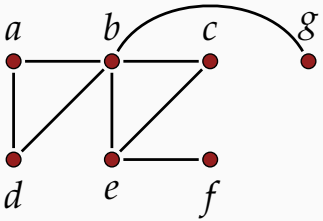
- *Blatt*: i ist ein Blatt in T und $|X_i| = 1$.
- *Verbindend*: i hat zwei Kinder j_1 und j_2 und es gilt $X_i = X_{j_1} = X_{j_2}$.
- *Einführend*: i hat genau ein Kind j und $\exists v \in V : X_i = X_j \cup \{ v \}$.
- *Vergessend*: i hat genau ein Kind j und $\exists v \in V : X_j = X_i \cup \{ v \}$.

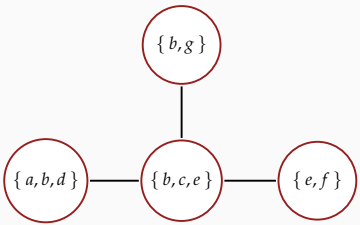
Lemma: Schöne Baumzerlegung in linearer Laufzeit

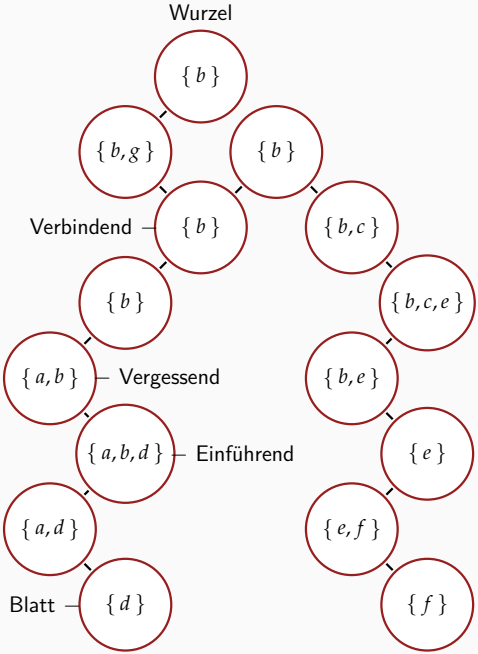
Gegeben einen beliebige Baumzerlegung mit Weite k kann eine schöne Baumzerlegung mit Weite k in $\mathcal{O}(n)$ berechnet werden.[1]

[1] Kloks (1994) - Treewidth. Computations and Approximations.

Schöne Baumzerlegung: Beispiel





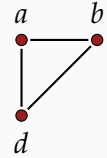


Algorithmus: Vorbereitungen 1

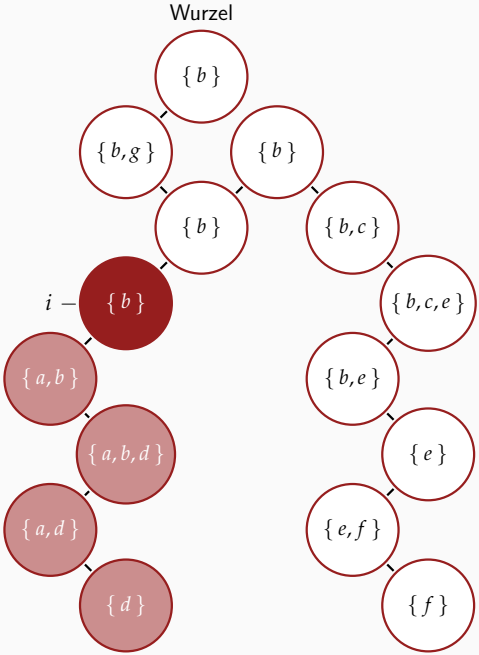
Definition: Subgraph

Wir definieren, ausgehend von $G(V, E)$ und der Baumzerlegung $(\{ X_i \mid i \in I \}, T)$, $\forall i \in I : G_i(V_i, E_i)$ sodass

- $V_i = \bigcup X_j$ mit $j = i$ oder ein Nachfahre von i in T.
- $E_i = E \cap (V_i \times V_i)$



$G_i = (V_i, E_i)$

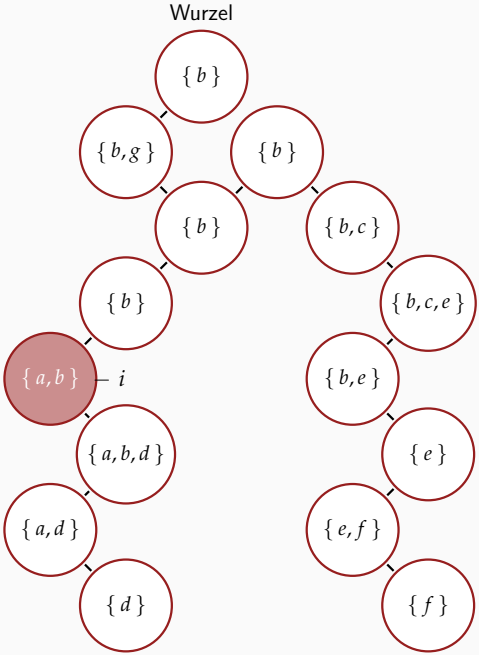


Algorithmus: Vorbereitungen 2

Definition: Tabelle C_i

Für jeden Knoten i berechnen wir eine Tabelle C_i mit Einträgen $C_i(S)$ für alle $S \subseteq X_i$.
Sie drückt aus, wie groß das MIS ist, welches die Knoten aus S enthält.

$C_i(\varnothing)$	?
$C_i(\{a\})$	?
$C_i(\{b\})$	?
$C_i(\{a,b\})$	?



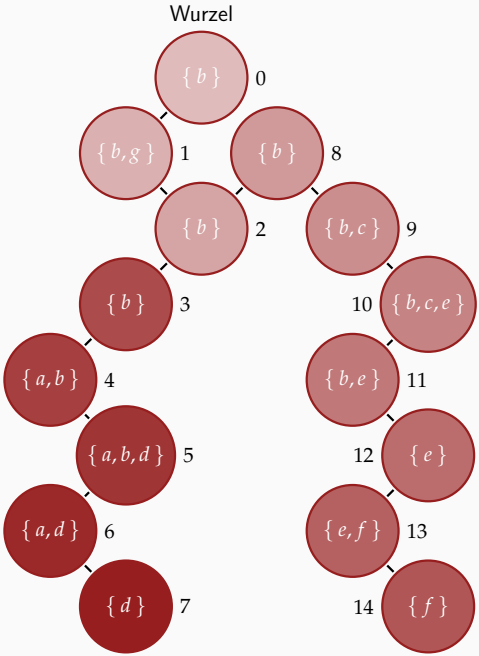
Algorithmus: Vorbereitungen 3

Definition: Postorder (LRN)

Eine LRN Traversierung bearbeitet einen Baum in dieser Reihenfolge:

- linker Teilbaum L.
- rechter Teilbaum R.
- Wurzel N.

$$[7, 6, 5, 4, 3, 14, 13, 12$$
$$11, 10, 9, 8, 2, 1, 0]$$



Knoten 7 - Blatt

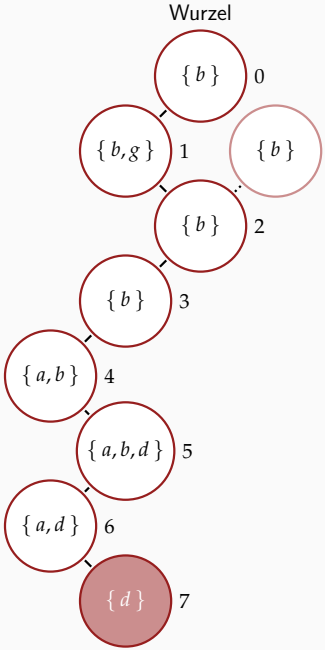
Definition: Tabelle - Blatt

Sei i ein Blatt. Dann gilt $|X_i| = 1$. Sei $v \in X_i$:

- $C_i(\varnothing) = 0$.
- $C_i(v) = 1$.

$$\left|\begin{array}{c|c} C_7(\varnothing) & 0 \\ C_7(\{d\}) & 1 \end{array}\right|$$

G_7



Knoten 6 - Einführend

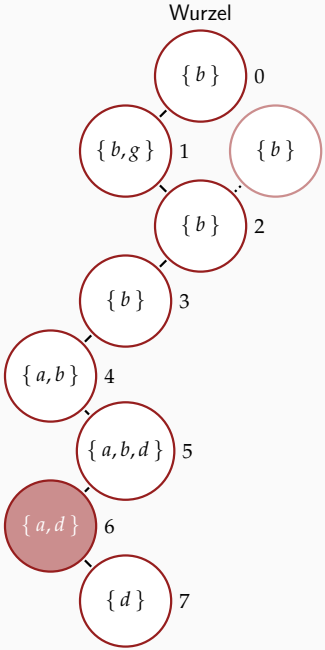
Definition: Tabelle - Einführend

Sei i ein Einführender Knoten mit Kind j, sei $X_i = X_j \cup \{ v \}$ und $S \subseteq X_j$:

- $C_i(S) = C_j(S)$
- $C_i(S \cup \{ v \}) =$
  - $-\infty$, $\exists w \in S : \{ v, w \} \in E$
  - $C_j(S) + 1$, sonst.

$C_6(\varnothing)$	0
$C_6(\{ a \})$	1
$C_6(\{ d \})$	1
$C_6(\{ a, d \})$	$-\infty$

G_6



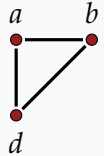
Knoten 5 - Einführend

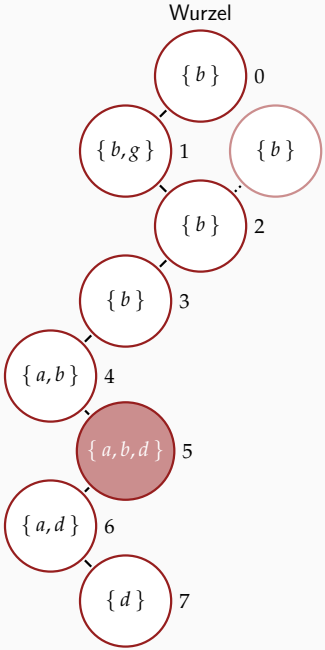
Definition: Tabelle - Einführend

- $C_i(S) = C_j(S)$
- $C_i(S \cup \{ v \}) =$
  - $-\infty$, $\exists w \in S : \{ v, w \} \in E$
  - $C_j(S) + 1$, sonst.

$C_5(\varnothing)$	0
$C_5(\{ a \})$	1
$C_5(\{ b \})$	1
$C_5(\{ d \})$	1
$C_5(\{ a, b \})$	$-\infty$
$C_5(\{ a, d \})$	$-\infty$
$C_5(\{ b, d \})$	$-\infty$
$C_5(\{ a, b, d \})$	$-\infty$

G_5





Knoten 4 - Vergessend

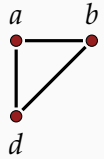
Definition: Tabelle - Vergessend

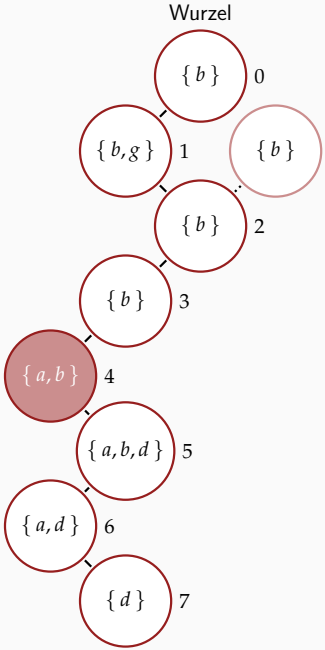
Sei i ein Vergessender Knoten mit Kind j sodass $S \subseteq X_i$ und $X_j = X_i \cup v$:

- $C_i(S) =$
  $\max \{ C_j(S), C_j(S \cup \{ v \}) \}$

$C_4(\varnothing)$	1
$C_4(\{ a \})$	1
$C_4(\{ b \})$	1
$C_4(\{ a, b \})$	$-\infty$

G_4





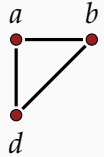
Knoten 3 - Vergessend

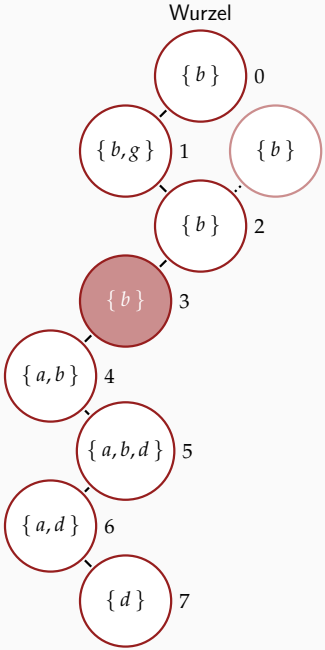
Definition: Tabelle - Vergessend

- $C_i(S) = \max\{ C_j(S), C_j(S \cup \{ v \}) \}$

$$\begin{array}{|c|c|} \hline C_3(\varnothing) & 1 \\ C_3(\{ b \}) & 1 \\ \hline \end{array}$$

G_3





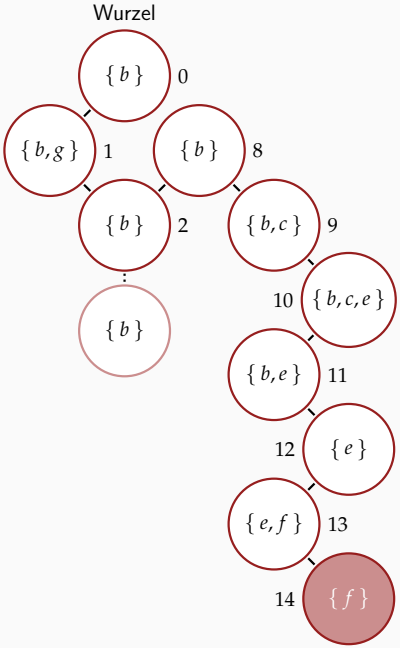
Knoten 14 - Blatt

Definition: Tabelle - Blatt

- $C_i(\varnothing) = 0$.
- $C_i(v) = 1$.

$C_{14}(\varnothing)$	0
$C_{14}(\{f\})$	1

G_{14}
f



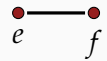
Knoten 13 - Einführend

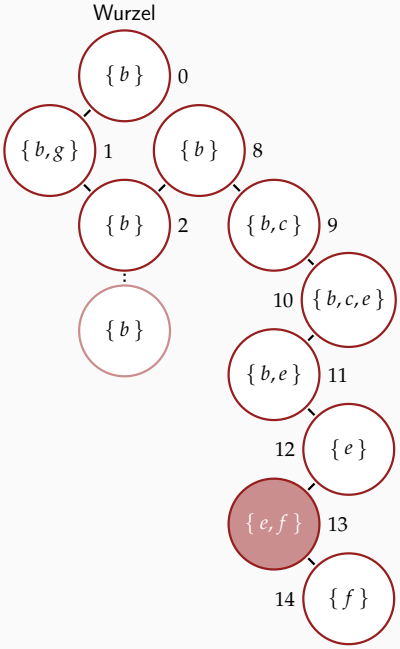
Definition: Tabelle - Einführend

- $C_i(S) = C_j(S)$
- $C_i(S \cup \{ v \}) =$
  - $-\infty$, $\exists w \in S : \{ v, w \} \in E$
  - $C_j(S) + 1$, sonst.

$C_{13}(\varnothing)$	0
$C_{13}(\{ e \})$	1
$C_{13}(\{ f \})$	1
$C_{13}(\{ e, f \})$	$-\infty$

G_{13}





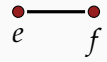
Knoten 12 - Vergessend

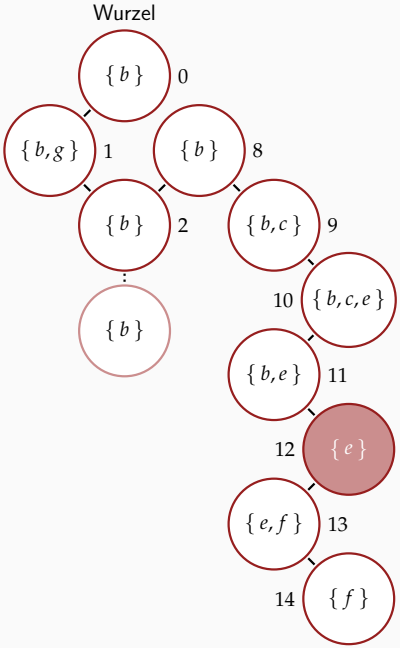
Definition: Tabelle - Vergessend

- $C_i(S) = \max\{ C_j(S), C_j(S \cup \{ v \}) \}$

$$\begin{vmatrix} C_{12}(\varnothing) \\ C_{12}(\{ e \}) \end{vmatrix} \begin{vmatrix} 1 \\ 1 \end{vmatrix}$$

G_{12}





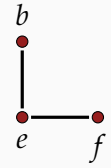
Knoten 11 - Einführend

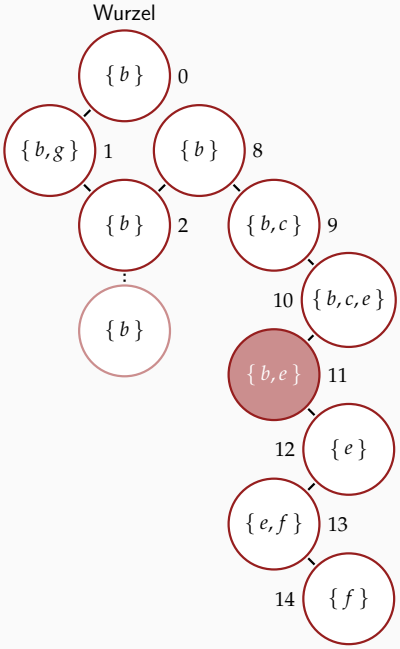
Definition: Tabelle - Einführend

- $C_i(S) = C_j(S)$
- $C_i(S \cup \{ v \}) =$
  - $-\infty$, $\exists w \in S : \{ v, w \} \in E$
  - $C_j(S) + 1$, sonst.

$C_{11}(\varnothing)$	1
$C_{11}(\{ b \})$	2
$C_{11}(\{ e \})$	1
$C_{11}(\{ b, e \})$	$-\infty$

G_{11}





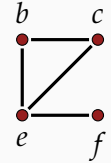
Knoten 10 - Einführend

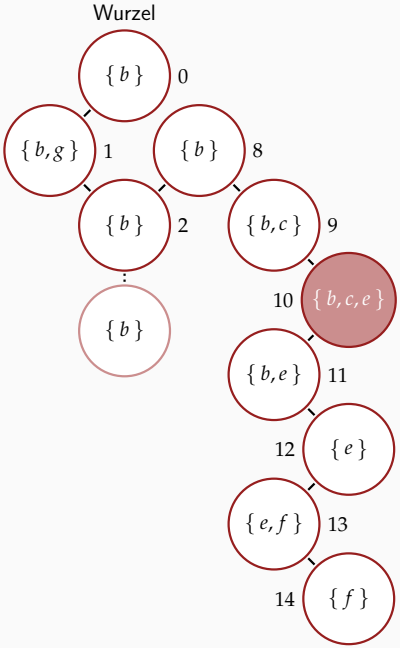
Definition: Tabelle - Einführend

- $C_i(S) = C_j(S)$
- $C_i(S \cup \{ v \}) =$
  - $-\infty$, $\exists w \in S : \{ v, w \} \in E$
  - $C_j(S) + 1$, sonst.

$C_{10}(\varnothing)$	1
$C_{10}(\{ b \})$	2
$C_{10}(\{ c \})$	2
$C_{10}(\{ e \})$	1
$C_{10}(\{ b, c \})$	$-\infty$
$C_{10}(\{ b, e \})$	$-\infty$
$C_{10}(\{ c, e \})$	$-\infty$
$C_{10}(\{ b, c, e \})$	$-\infty$

G_{10}





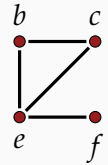
Knoten 9 - Vergessend

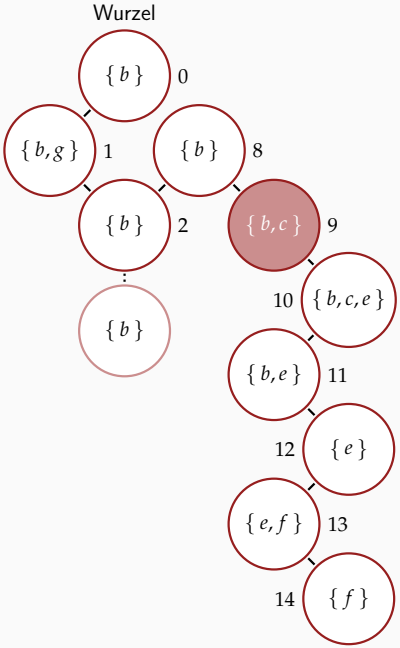
Definition: Tabelle - Vergessend

- $C_i(S) =$
  $\max\{ C_j(S), C_j(S \cup \{ v \}) \}$

$C_9(\varnothing)$	1
$C_9(\{ b \})$	2
$C_9(\{ c \})$	2
$C_9(\{ b, c \})$	$-\infty$

G_9





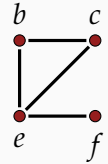
Knoten 8 - Vergessend

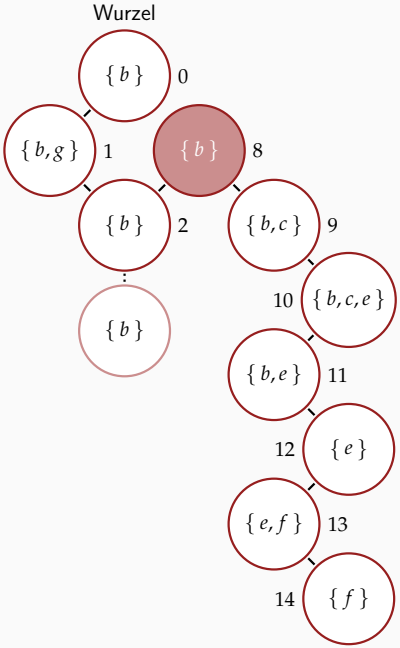
Definition: Tabelle - Vergessend

- $C_i(S) =$
  $\max\{ C_j(S), C_j(S \cup \{ v \}) \}$

$$\begin{array}{|c|c|} \hline C_8(\varnothing) & 2 \\ C_8(\{ b \}) & 2 \\ \hline \end{array}$$

G_8





Knoten 2 - Verbindend

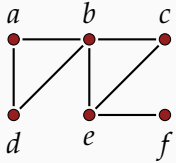
Definition: Tabelle - Verbindend

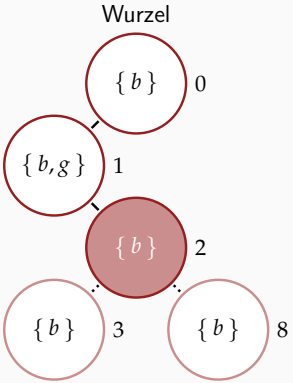
Sei i ein Verbindender Knoten mit Kindern j_1 und j_2 sodass $S \subseteq X_i$:

- $C_i(S) = C_{j_1}(S) + C_{j_2}(S) - |S|$.

$C_2(\varnothing)$	3
$C_2(\{b\})$	2

G_2





$C_3(\varnothing)$	1
$C_3(\{b\})$	1

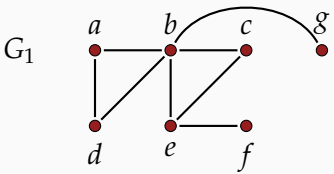
$C_8(\varnothing)$	2
$C_8(\{b\})$	2

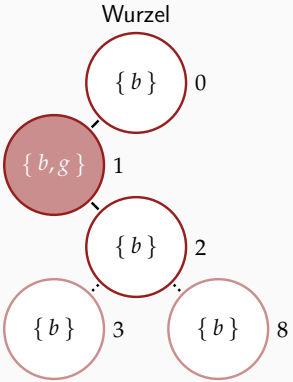
Knoten 1 - Einführend

Definition: Tabelle - Einführend

- $C_i(S) = C_j(S)$
- $C_i(S \cup \{ v \}) =$
  - $-\infty$, $\exists w \in S : \{ v, w \} \in E$
  - $C_j(S) + 1$, sonst.

$C_1(\varnothing)$	3
$C_1(\{ b \})$	2
$C_1(\{ g \})$	4
$C_1(\{ b, g \})$	$-\infty$



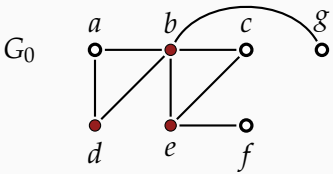


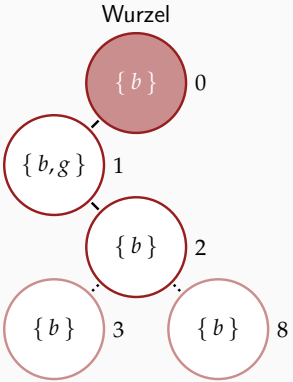
Knoten 0 - Vergessend

Definition: Tabelle - Vergessend

- $C_i(S) =$
  $\max \{ C_j(S), C_j(S \cup \{ v \}) \}$

$$\begin{vmatrix} C_0(\varnothing) & 4 \\ C_0(\{ b \}) & 2 \end{vmatrix}$$





Laufzeit

Schöne Baumzerlegung	$\mathcal{O}(n)$
LRN Traversierung berechnen	$\mathcal{O}(n)$
Traversierung	$\mathcal{O}(n)$
Berechnung von C_i	$\mathcal{O}(2^{k+1})$
Gesamt	$\mathcal{O}(2n + 2^{k+1}n)$

Theorem: Laufzeit des MIS-Algorithmus'

Gegeben eine beliebige Baumzerlegung mit Weite k ergibt sich eine Komplexität von $\mathcal{O}(2^{k+1}n)$.

Fixed Parameter Tractability

Definition: FPT

Falls für ein Problem ein Algorithmus mit Laufzeit $\mathcal{O}(f(k) \cdot p(n))^a$ existiert, dann liegt es in der Komplexitätsklasse FPT.

[a] f eine berechenbare Funktion, p ein Polynom und k ein Parameter

Fixed Parameter Tractability

Definition: FPT

Falls für ein Problem ein Algorithmus mit Laufzeit $\mathcal{O}(f(k) \cdot p(n))^a$ existiert, dann liegt es in der Komplexitätsklasse FPT.

[a] f eine berechenbare Funktion, p ein Polynom und k ein Parameter

Problem: Baumzerlegung liegt in FPT.

Polynomialzeitapproximationsschema für MIS auf planaren Graphen

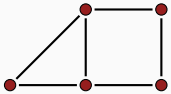
Polynomialzeitapproximationsschema

Definition: Polynomialzeitapproximationsschema - PTAS

Ein *PTAS* ist ein Approximationsalgorithmus, der in Polynomialzeit für ein beliebiges $\varepsilon > 0$ eine $(1-\varepsilon)$[a]-Approximation berechnet.

[a] bzw. $(1+\varepsilon)$ bei Minimierungsproblemen

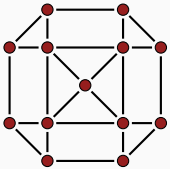
Außerplanarität



außerplanar

~~außer~~planar

2-außerplanar



3-außerplanar

Definition: Außerplanarer Graph

Ein Graph $G(V, E)$ ist $(1-)$außerplanar falls für ihn eine planare Einbettung existiert sodass alle seine Knoten auf der Außenseite liegen.

Definition: k-außerplanarer Graph

Ein Graph $G(V, E)$ ist k-außerplanar falls für ihn eine planare Einbettung existiert, sodass das Entfernen der äußeren Knoten in einem $(k-1)$-außerplanaren Graphen resultiert.

Außerplanarität berechnen

Lemma: Außerplanarität

Die Außerplanarität eines Graphen kann in Laufzeit $\mathcal{O}(n^2)$ berechnet werden.[1]

[1] Kammer (2007) - Determining the smallest k such that G is k-outerplanar

Planare Graphen und Baumzerlegungen

Lemma: Baumweite

Die Baumweite eines k-außerplanaren Graphen ist beschränkt durch $3k - 1$.[1]

Lemma: Baumzerlegung

Die entsprechende Baumzerlegung kann mit einer Laufzeit von $\mathcal{O}(kn)$ gefunden werden.[1]

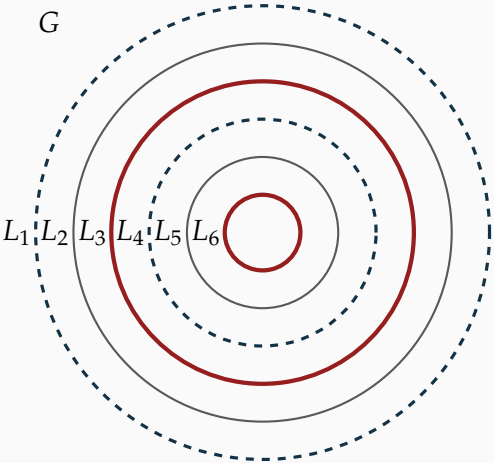
[1] Bodlaender (1998) - A partial k-arboretum of graphs with bounded treewidth

PTAS Beschreibung

- *Eingabe:* planarer Graph $G = (V, E)$, Schranke $\varepsilon > 0$.
- Setze $k = \lceil \frac{1}{\varepsilon} \rceil$.
- Seien $L_1 \dots L_m$ die Schichten von G.
- G_i mit $i \in \{ 1 \dots k \}$ entsteht, wenn man die Knoten von $L_i, L_{i+k}, L_{i+2k} \dots$ aus G entfernt.
- Berechne MIS für jedes G_i.
- Gebe größtes MIS zurück.

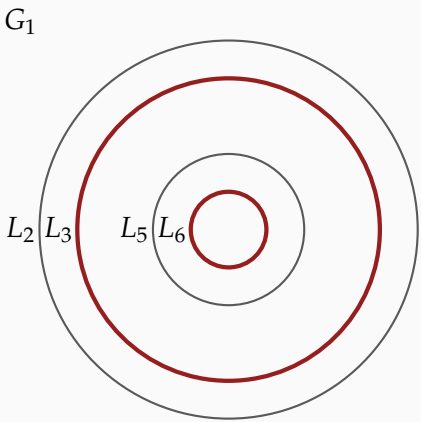
Beispiel

- $G = (V, E)$
- $\varepsilon = \frac{1}{3}$
- $\rightarrow k = 3$
- Definiere G_1, G_2, G_3



Beispiel

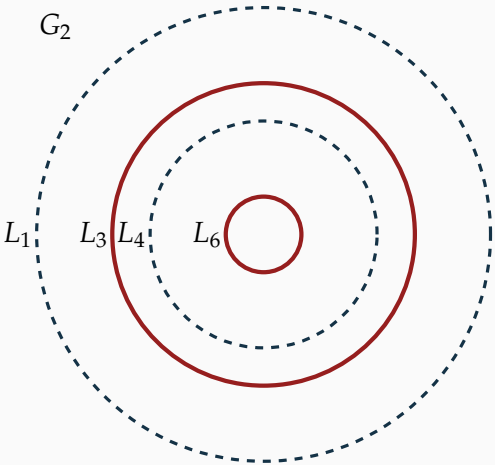
- $G = (V, E)$
- $\varepsilon = \frac{1}{3}$
- $\rightarrow k = 3$
- Definiere G_1, G_2, G_3
- $G_1 = G \setminus \{ L_1, L_4 \}$



Beispiel

- $G = (V, E)$
- $\varepsilon = \frac{1}{3}$
- $\rightarrow k = 3$
- Definiere G_1, G_2, G_3

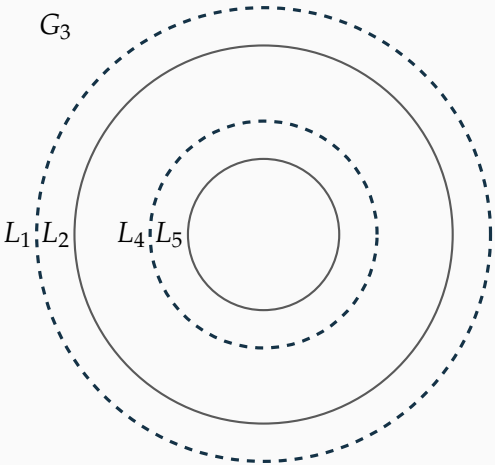
- $G_1 = G \setminus \{ L_1, L_4 \}$
- $G_2 = G \setminus \{ L_2, L_5 \}$



Beispiel

- $G = (V, E)$
- $\varepsilon = \frac{1}{3}$
- $\rightarrow k = 3$
- Definiere G_1, G_2, G_3

- $G_1 = G \setminus \{ L_1, L_4 \}$
- $G_2 = G \setminus \{ L_2, L_5 \}$
- $G_3 = G \setminus \{ L_3, L_6 \}$



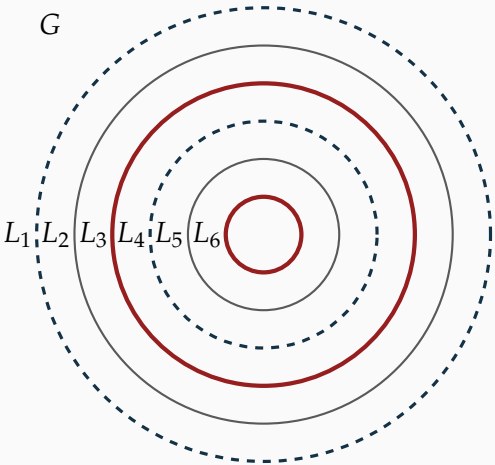
Beispiel

- $G = (V, E)$
- $\varepsilon = \frac{1}{3}$
- $\rightarrow k = 3$
- Definiere G_1, G_2, G_3

- $G_1 = G \setminus \{ L_1, L_4 \}$
- $G_2 = G \setminus \{ L_2, L_5 \}$
- $G_3 = G \setminus \{ L_3, L_6 \}$

Anmerkung: Außerplanarität

Jedes G_i ist aller höchstens $(k-1)$-außerplanar.
Daher hat G_i höchstens Baumweite $3k - 1$.



Laufzeit und Güte

Lemma: Laufzeit von MIS auf G_i

MIS lässt sich mit einer Laufzeit von $\mathcal{O}(2^{3k-1} \cdot n)$ auf G_i lösen.

Wir berechnen für $G_i, i \in \{1 \dots k\}$ jeweils das MIS.

Theorem: Laufzeit des gesamten PTAS

Die Laufzeit des PTAS beträgt $\mathcal{O}(2^{3k-1} \cdot k \cdot n)$.

Theorem: Approximationsgüte

Das PTAS liefert eine $(1-\varepsilon)$-Approximation.

Konklusion

Beschränkte Baumweite

- Gut geeignet um viele NP-schwere Probleme in FPT bearbeiten.
- Kombinatorische Optimierung: Viele Algorithmen polynomiell in der Graphgröße, exponentiell in der Baumweite.

  $\rightarrow$ Schranke für Baumweite bietet polynomielle Algorithmen.
- Es gibt gute obere und untere Schranken sowie Heuristiken, um Baumweite zu approximieren.

Übersicht