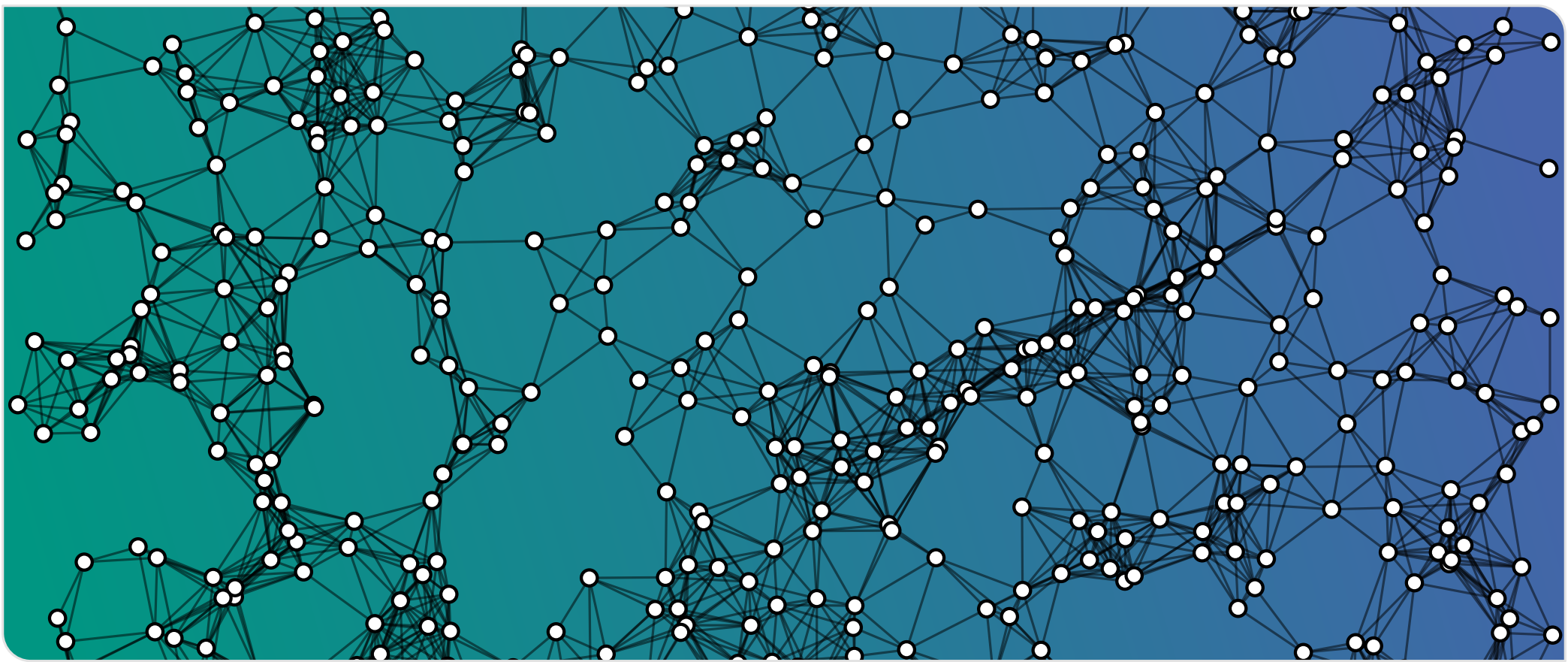
Karlsruher Institut für Technologie

Parametrisierte Algorithmen

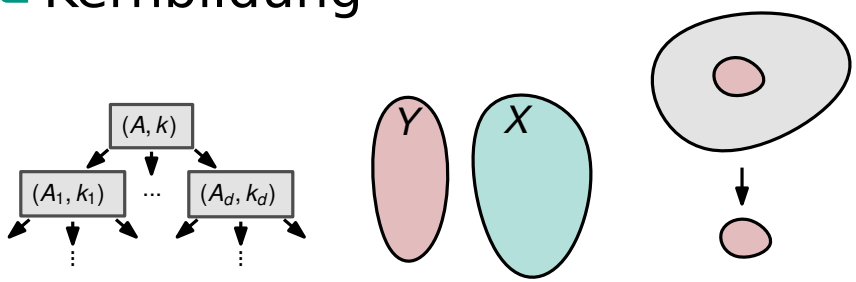
Baumweite: Berechnung einer Baumzerlegung & planare Graphen



Inhalt

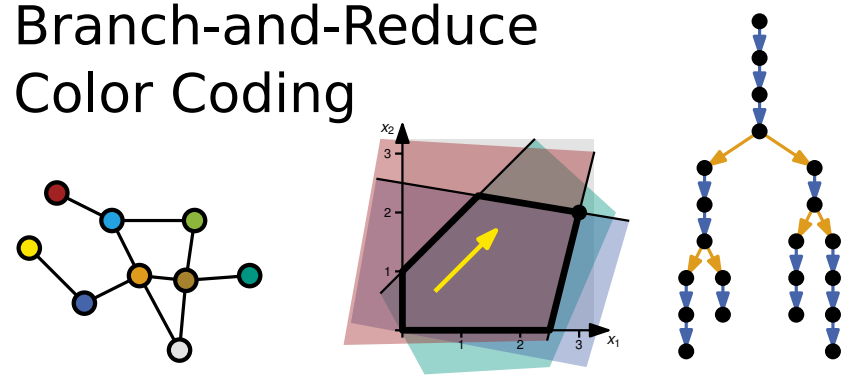
Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung



Erweiterte Toolbox

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

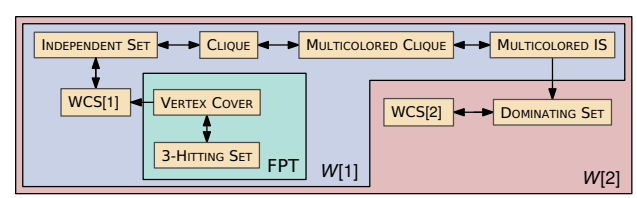


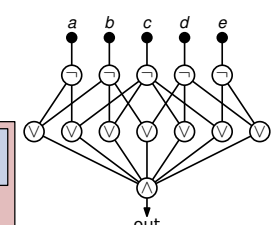
Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

Untere Schranken

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





Berechnung einer Baumzerlegung

Karlsruher Institut für Technologie

Problem

- FPT-Algorithmen mit Baumweite als Parameter nehmen an, dass eine entsprechende Baumzerlegung gegeben ist
- Baumweite für einen Graphen G ausrechnen ist NP-schwer

Berechnung einer Baumzerlegung

Problem

- FPT-Algorithmen mit Baumweite als Parameter nehmen an, dass eine entsprechende Baumzerlegung gegeben ist
- Baumweite für einen Graphen G ausrechnen ist NP-schwer

Theorem
Es gibt einen Algorithmus, der in $k^{O(k^3)} \cdot n$ Zeit eine Baumzerlegung der Weite k berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

Berechnung einer Baumzerlegung

Karlsruher Institut für Technologie

Problem

- FPT-Algorithmen mit Baumweite als Parameter nehmen an, dass eine entsprechende Baumzerlegung gegeben ist
- Baumweite für einen Graphen G ausrechnen ist NP-schwer

Theorem
Es gibt einen Algorithmus, der in $k^{O(k^3)} \cdot n$ Zeit eine Baumzerlegung der Weite k berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

- FPT-Algorithmus
- lineare Laufzeit in n
- Beweis: nicht hier

Berechnung einer Baumzerlegung

Karlsruher Institut für Technologie

Problem

- FPT-Algorithmen mit Baumweite als Parameter nehmen an, dass eine entsprechende Baumzerlegung gegeben ist
- Baumweite für einen Graphen G ausrechnen ist NP-schwer

Theorem
Es gibt einen Algorithmus, der in $k^{O(k^3)} \cdot n$ Zeit eine Baumzerlegung der Weite k berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

- FPT-Algorithmus
- lineare Laufzeit in n
- Beweis: nicht hier

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

Berechnung einer Baumzerlegung

Karlsruher Institut für Technologie

Problem

- FPT-Algorithmen mit Baumweite als Parameter nehmen an, dass eine entsprechende Baumzerlegung gegeben ist
- Baumweite für einen Graphen G ausrechnen ist NP-schwer

Theorem
Es gibt einen Algorithmus, der in $k^{O(k^3)} \cdot n$ Zeit eine Baumzerlegung der Weite k berechnet oder entscheidet, dass $\text{tw}(G) > k$.

- FPT-Algorithmus
- lineare Laufzeit in n
- Beweis: nicht hier

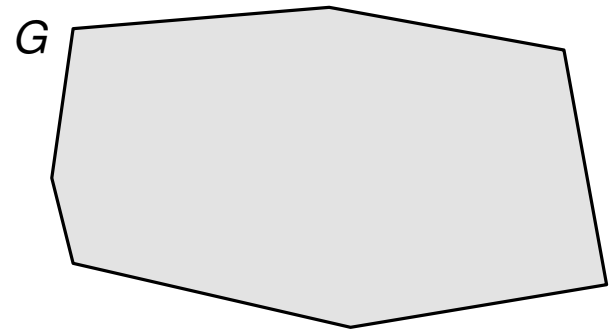
Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\text{tw}(G) > k$.

- approximativer FPT-Algorithmus
- quadratisch in n, dafür bessere Laufzeit in k
- Beweis: gleich

Berechnung einer Baumzerlegung

Grobe Idee

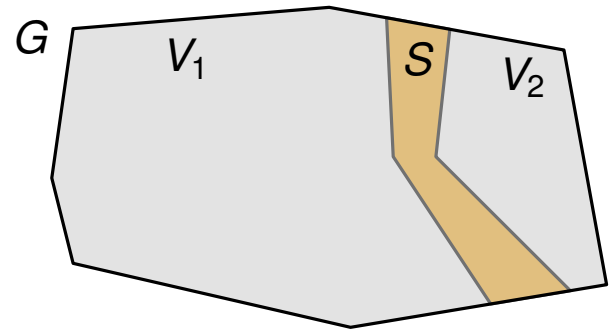
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

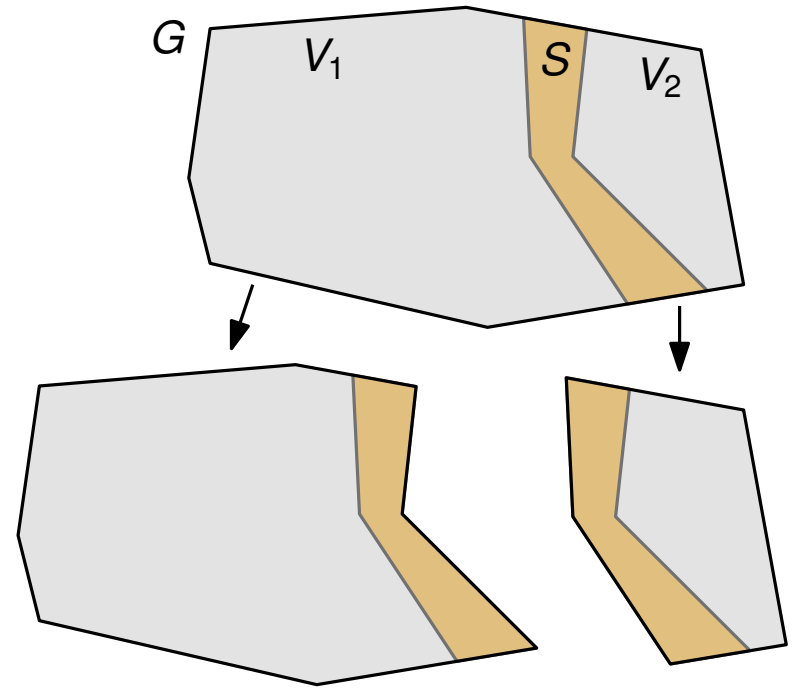
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

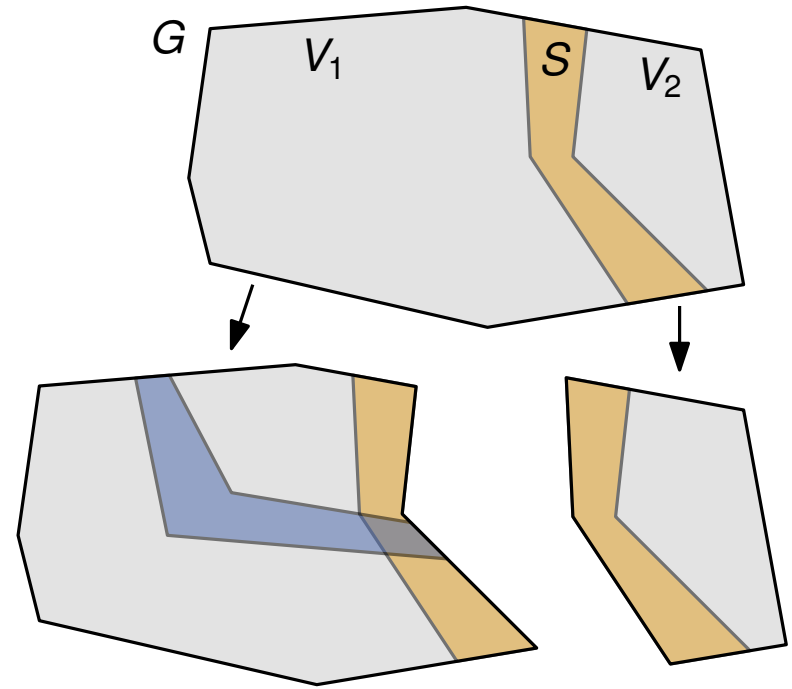
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

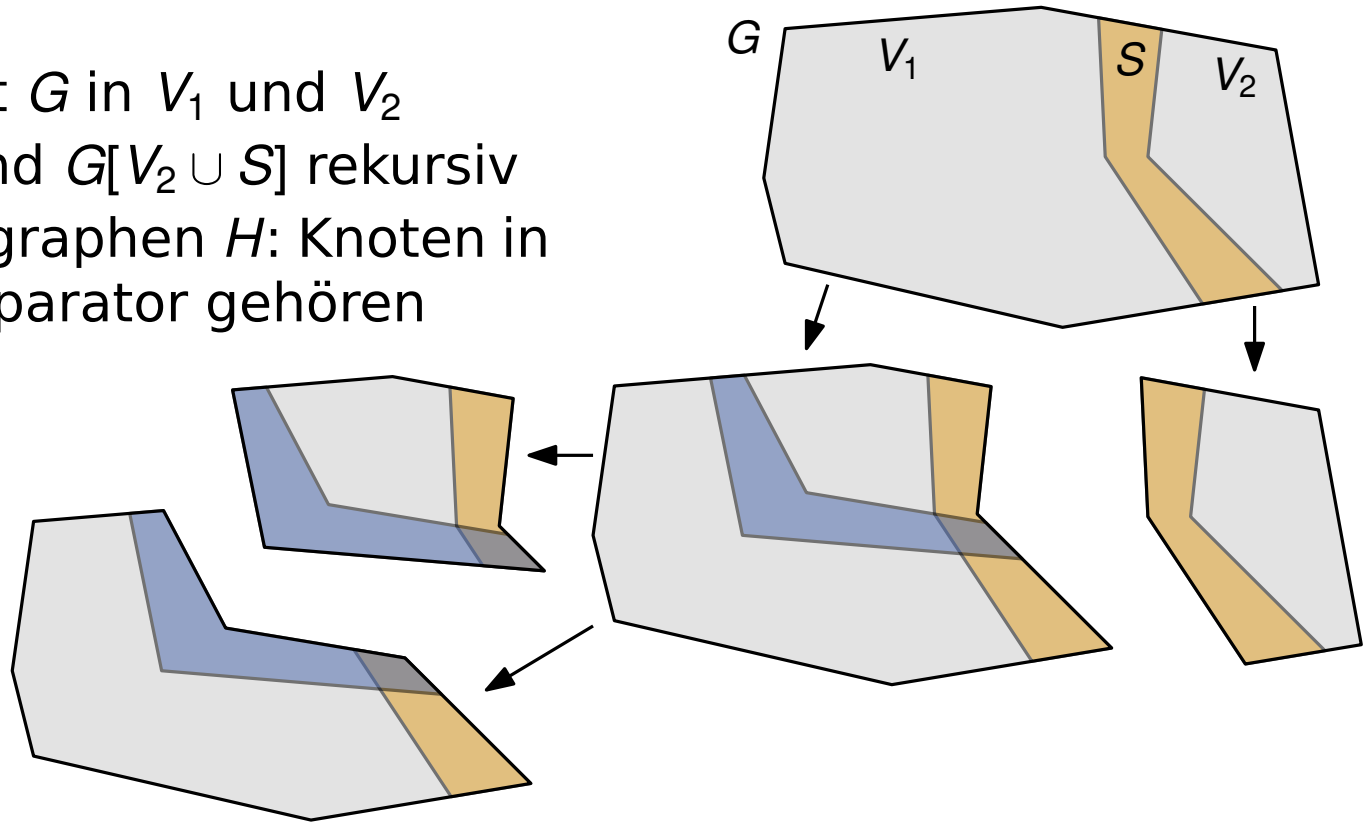
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

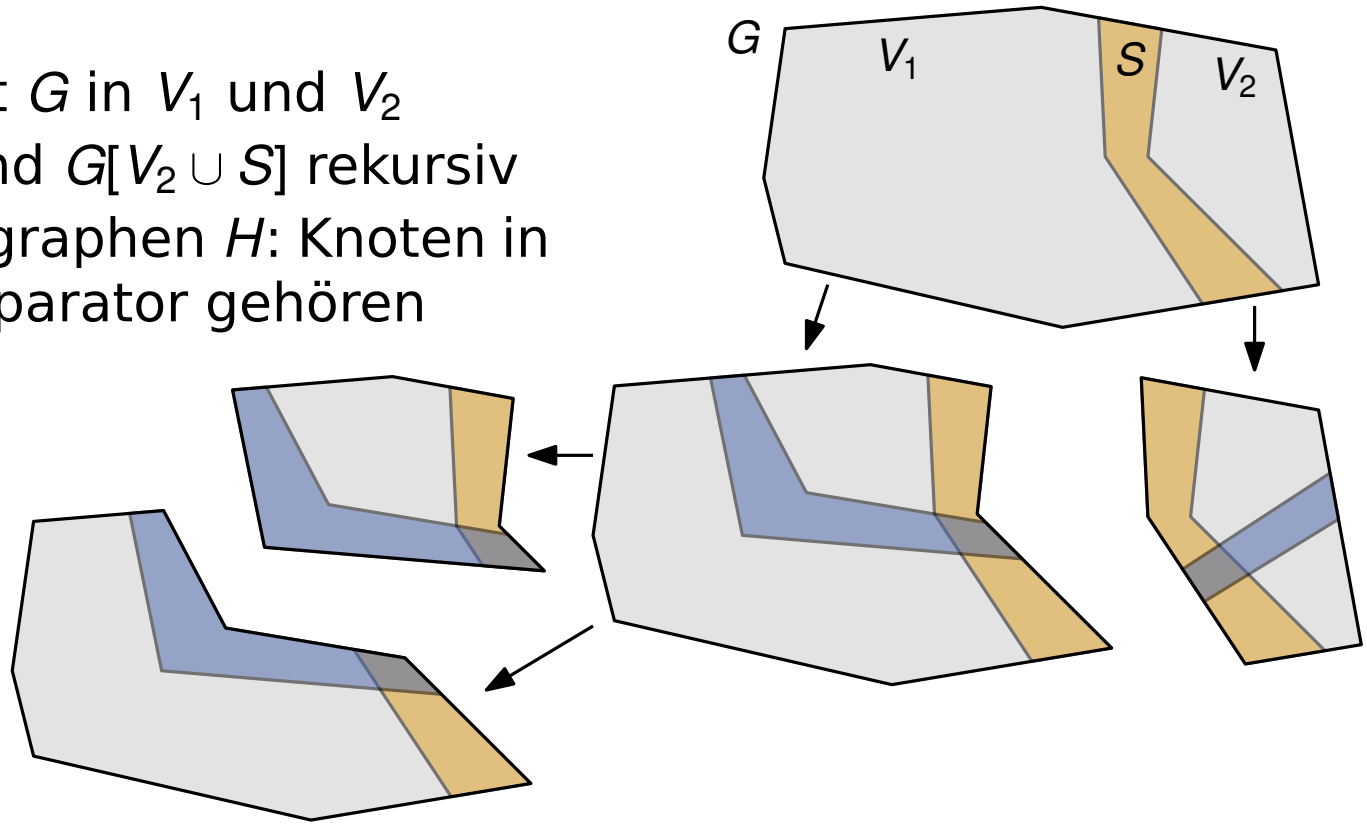
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

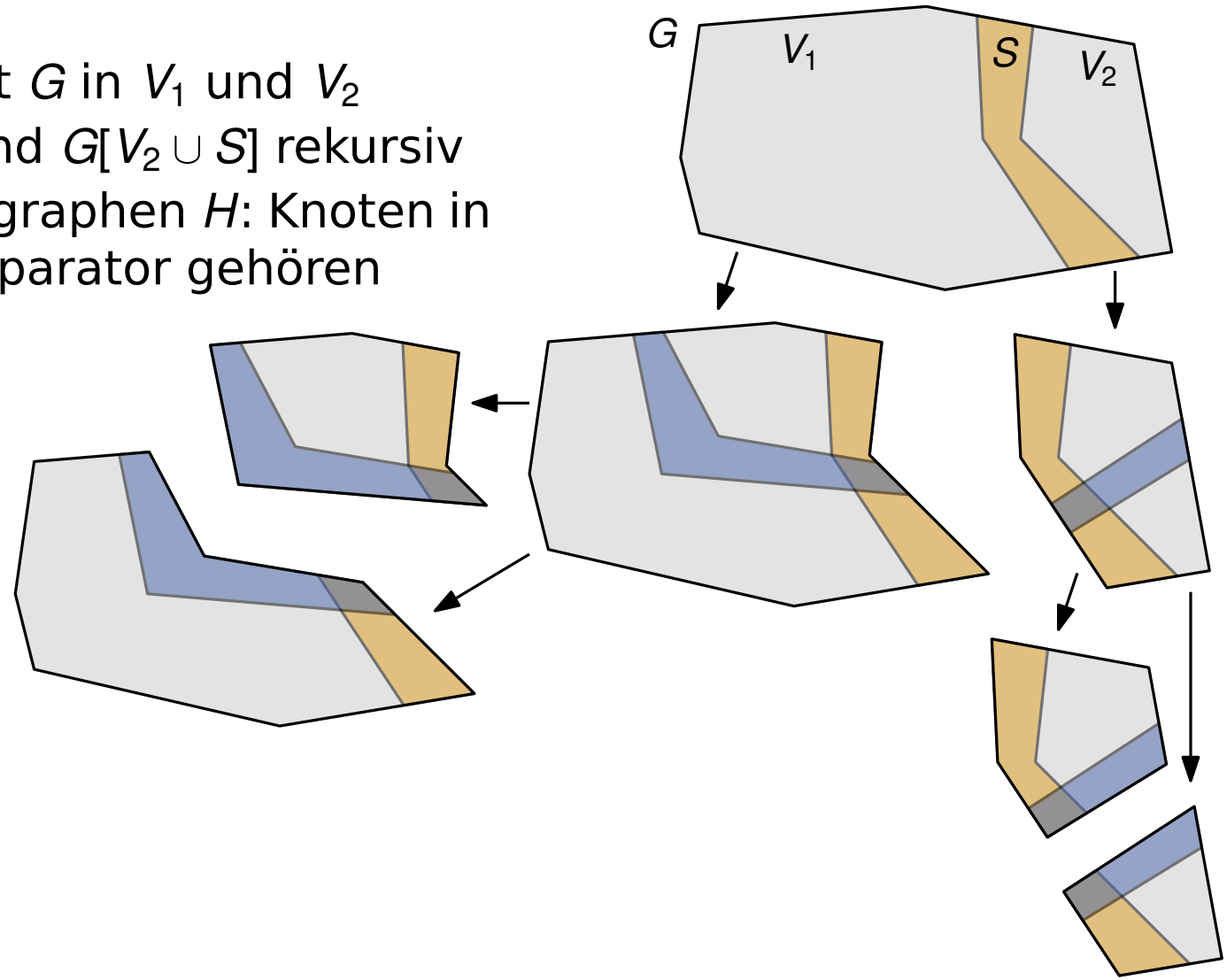
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

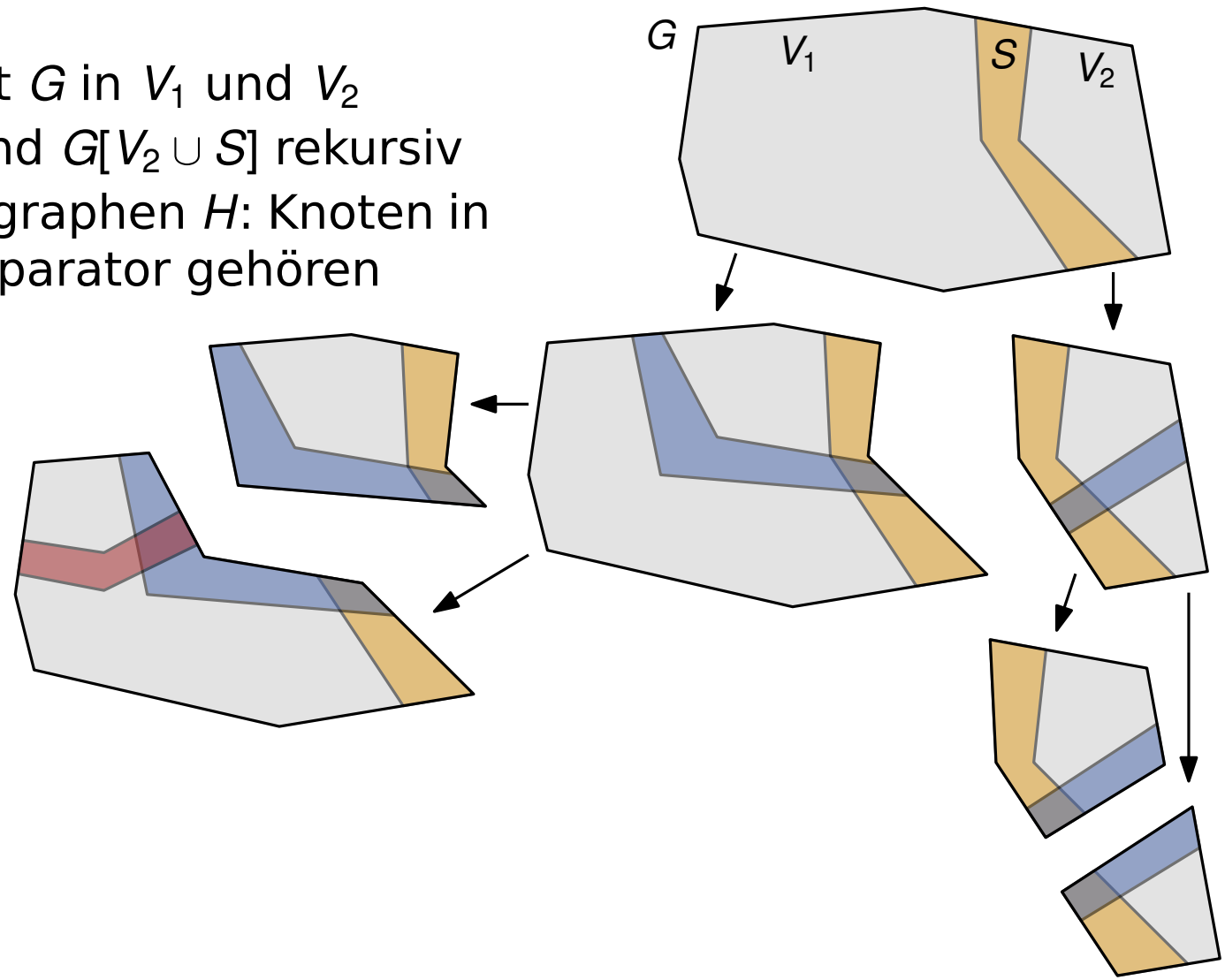
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

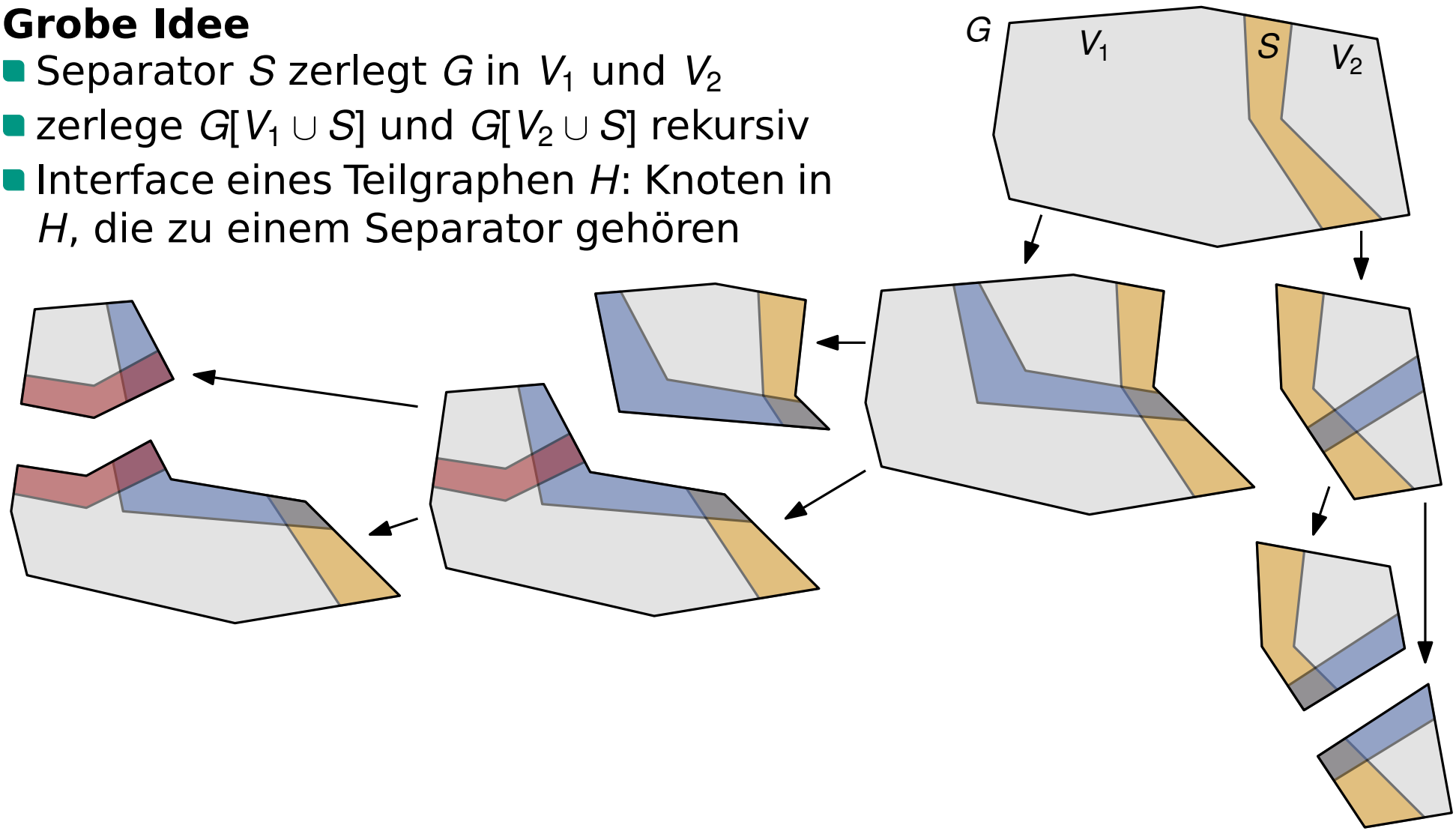
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

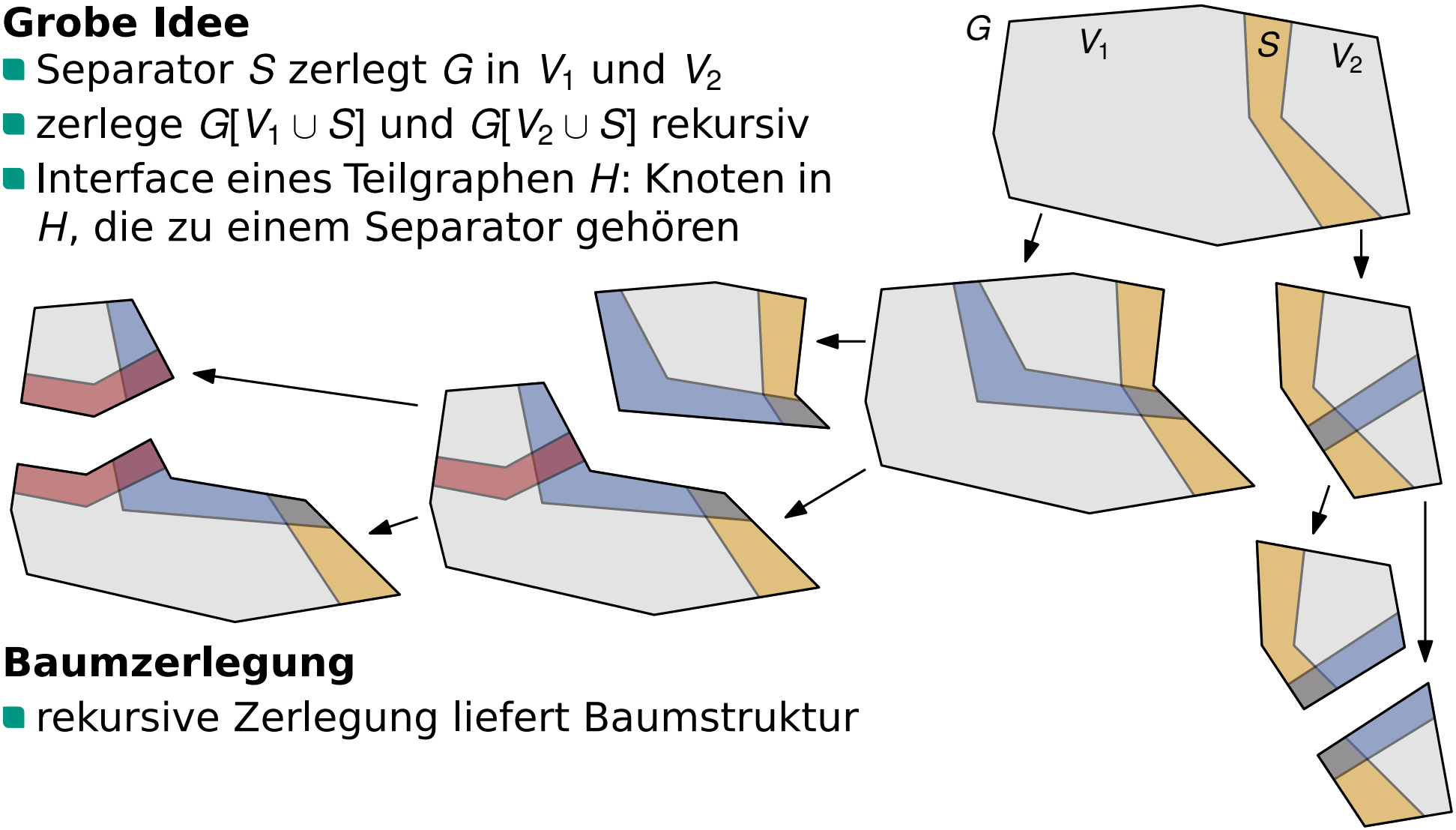
- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



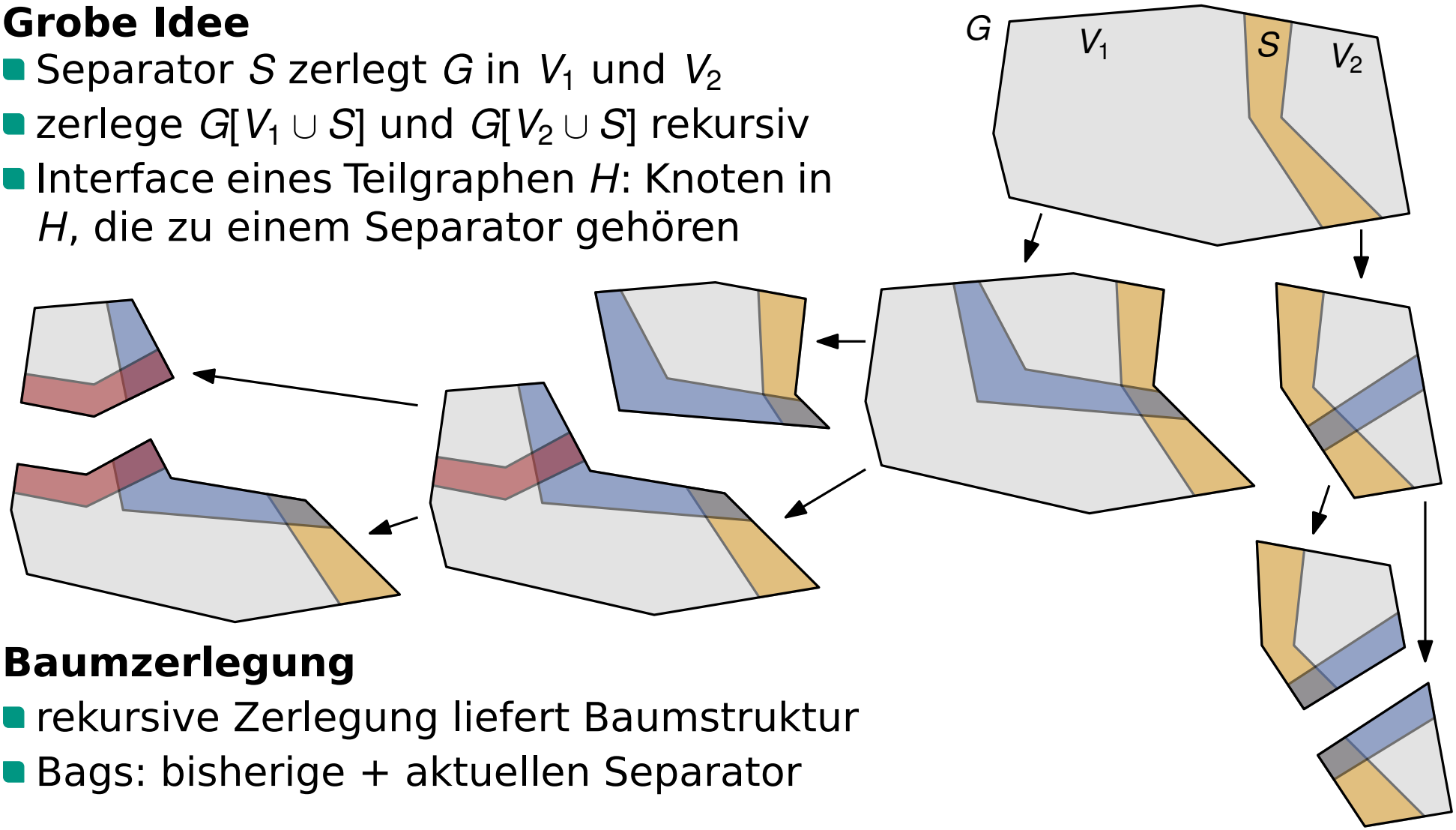
Baumzerlegung

- rekursive Zerlegung liefert Baumstruktur

Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



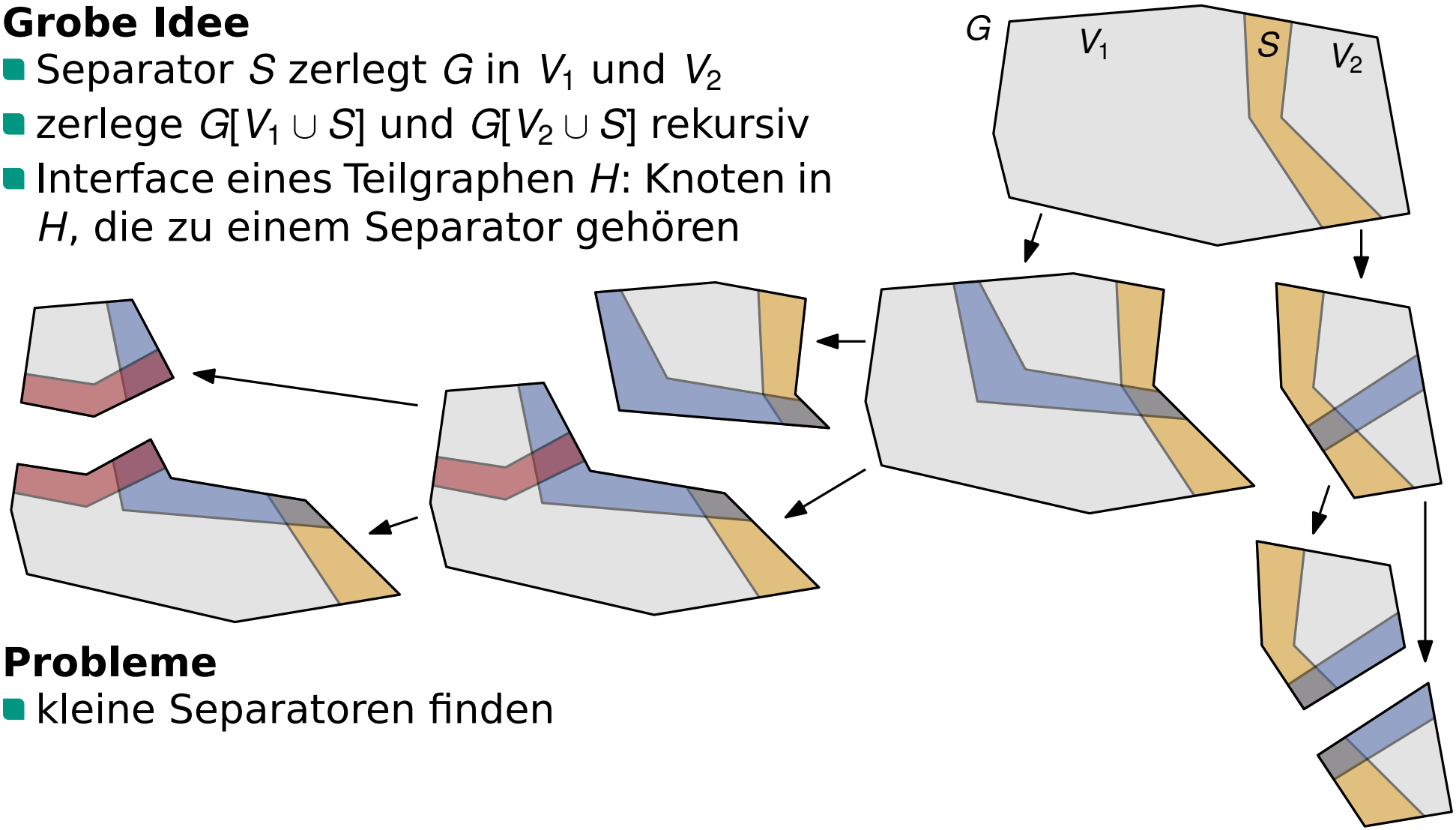
Baumzerlegung

- rekursive Zerlegung liefert Baumstruktur
- Bags: bisherige + aktuellen Separator

Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



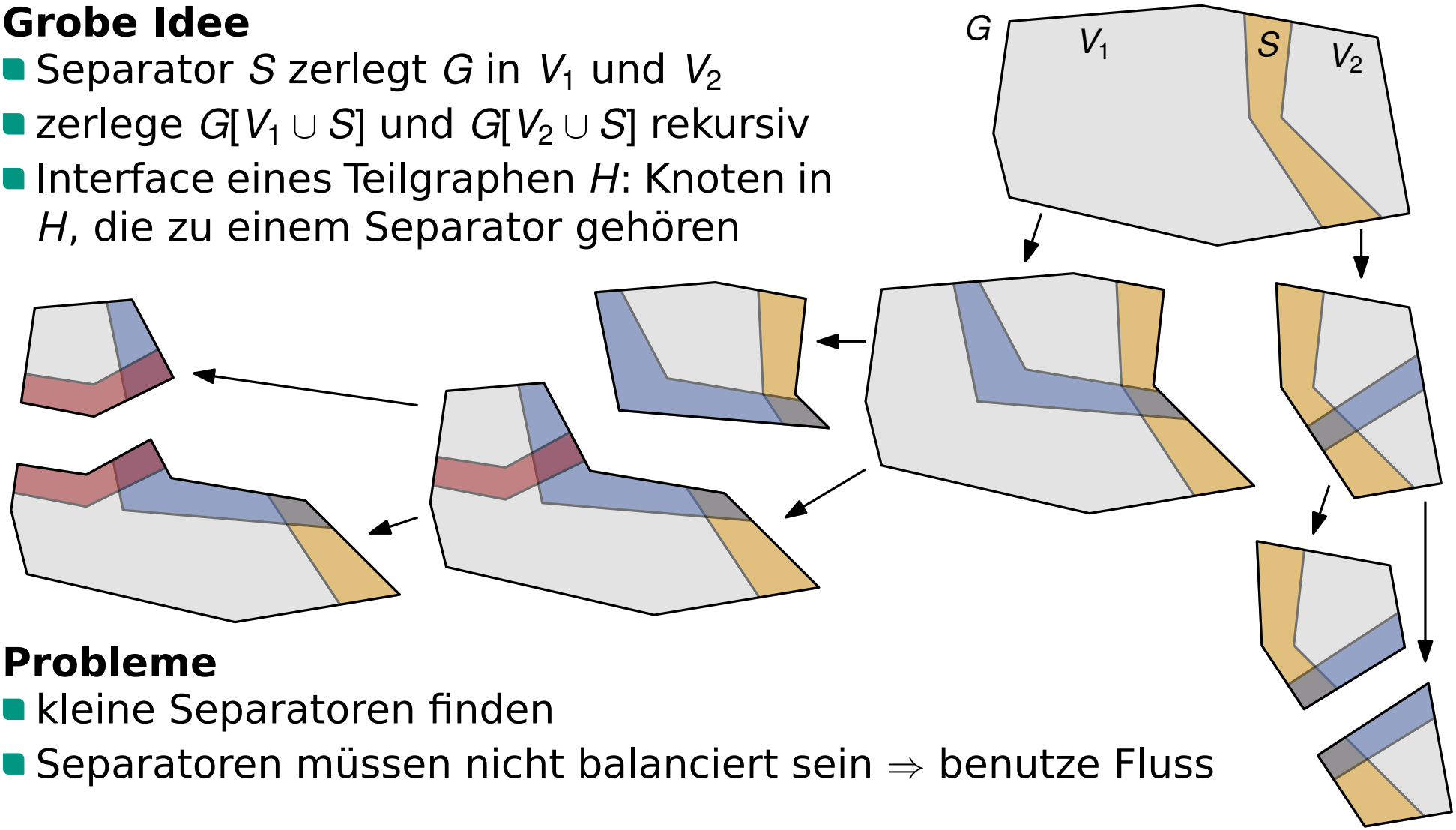
Probleme

- kleine Separatoren finden

Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



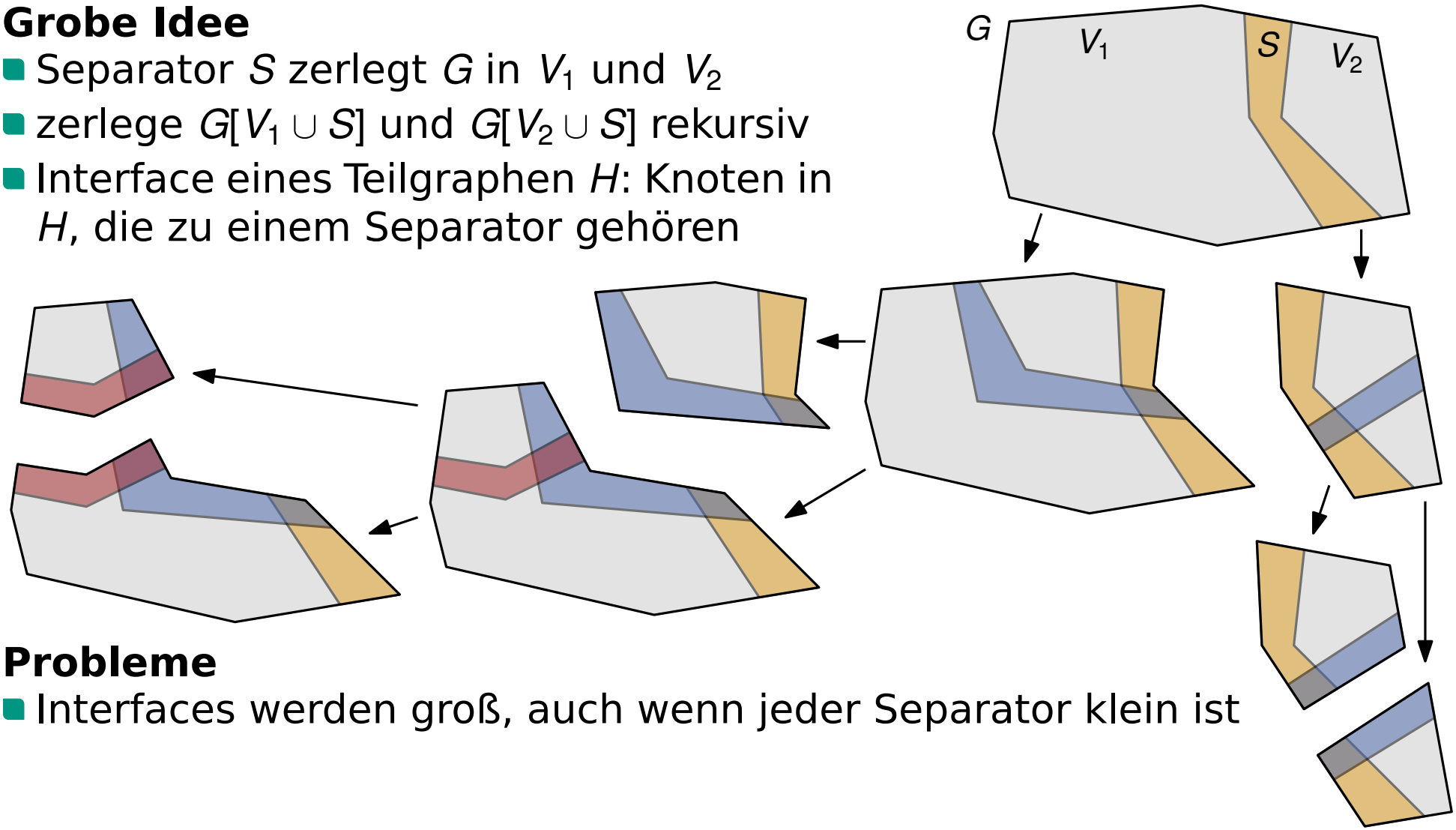
Probleme

- kleine Separatoren finden
- Separatoren müssen nicht balanciert sein $\Rightarrow$ benutze Fluss

Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



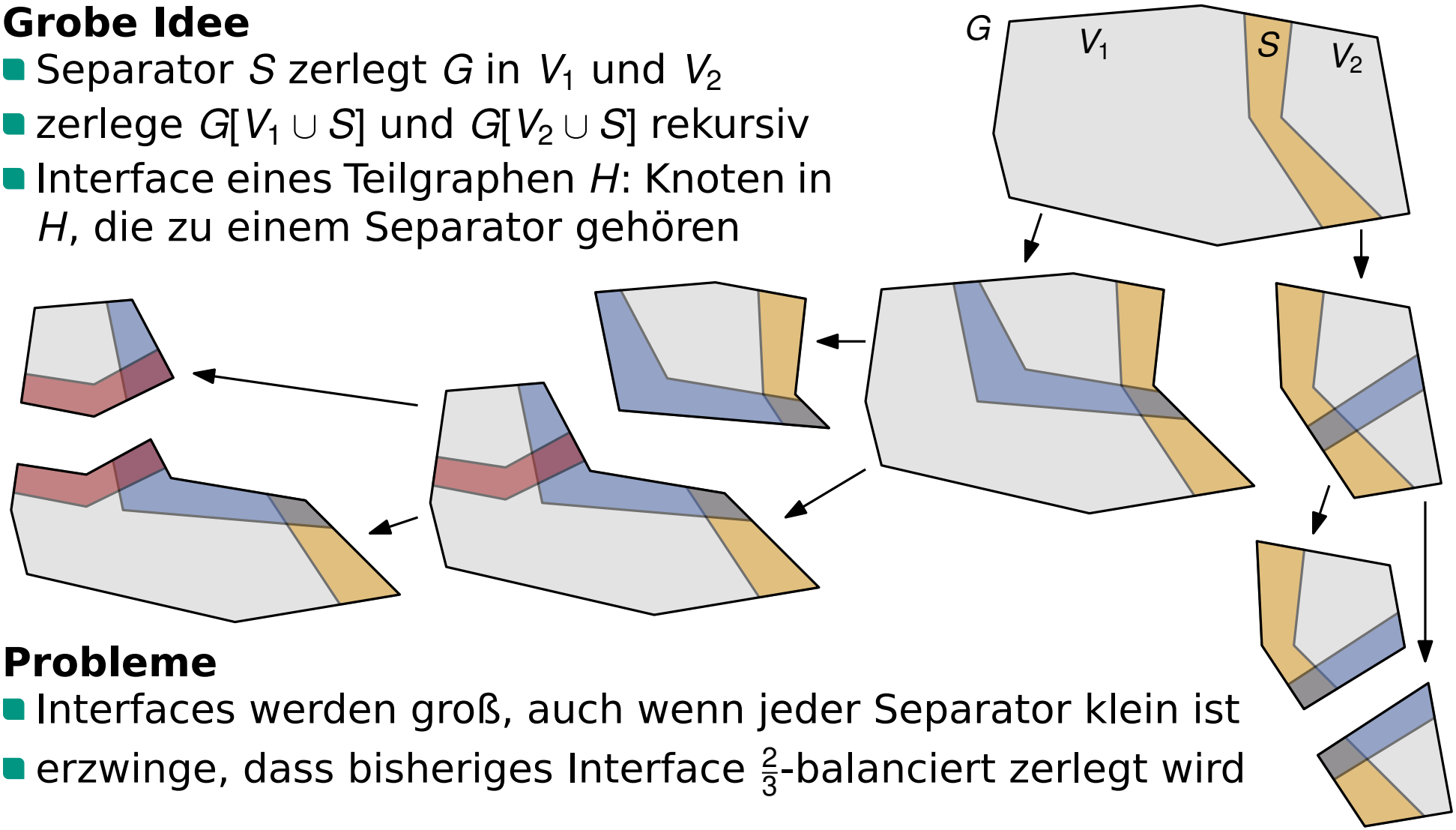
Probleme

- Interfaces werden groß, auch wenn jeder Separator klein ist

Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



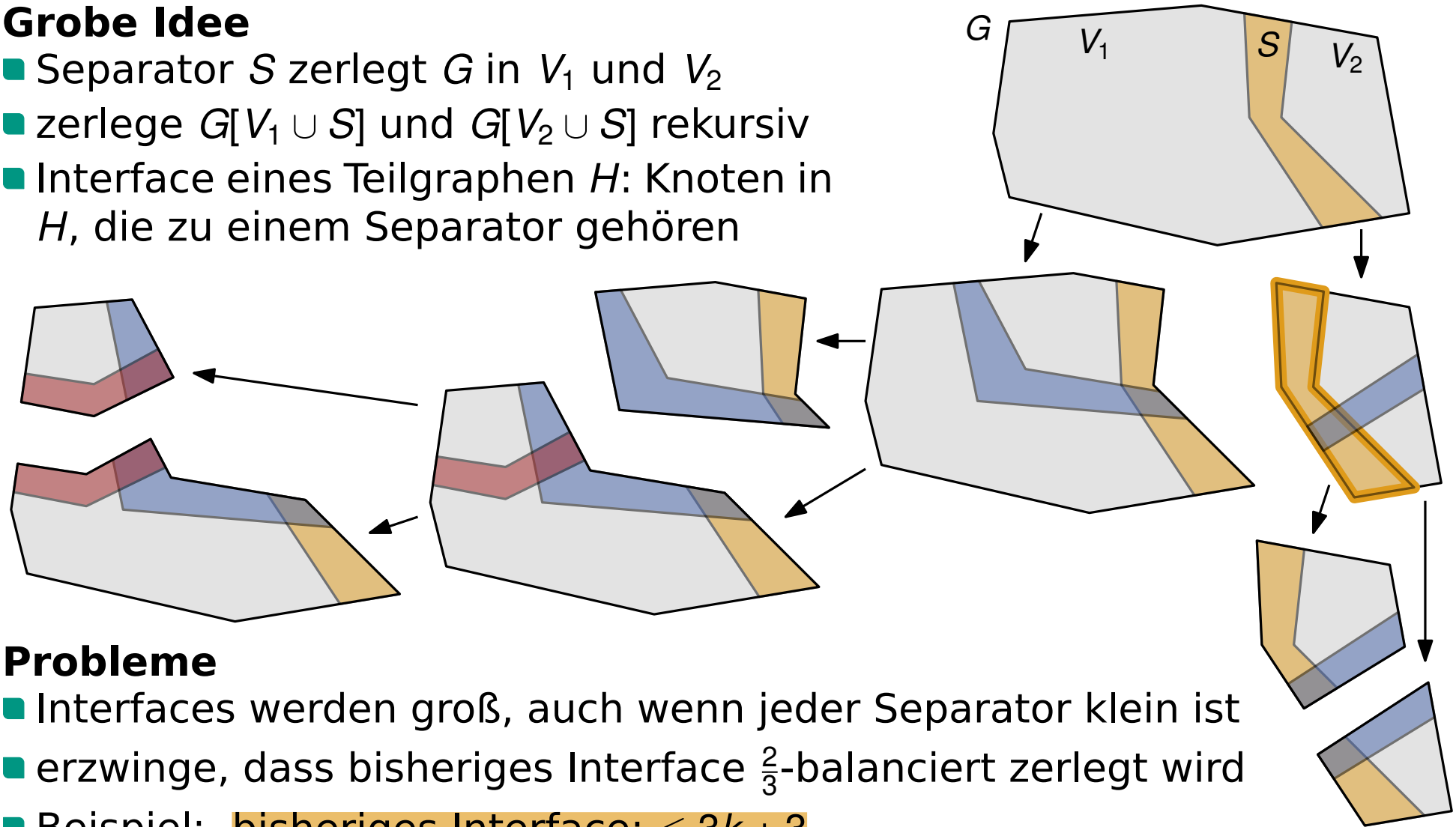
Probleme

- Interfaces werden groß, auch wenn jeder Separator klein ist
- erzwinge, dass bisheriges Interface $\frac{2}{3}$-balanciert zerlegt wird

Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



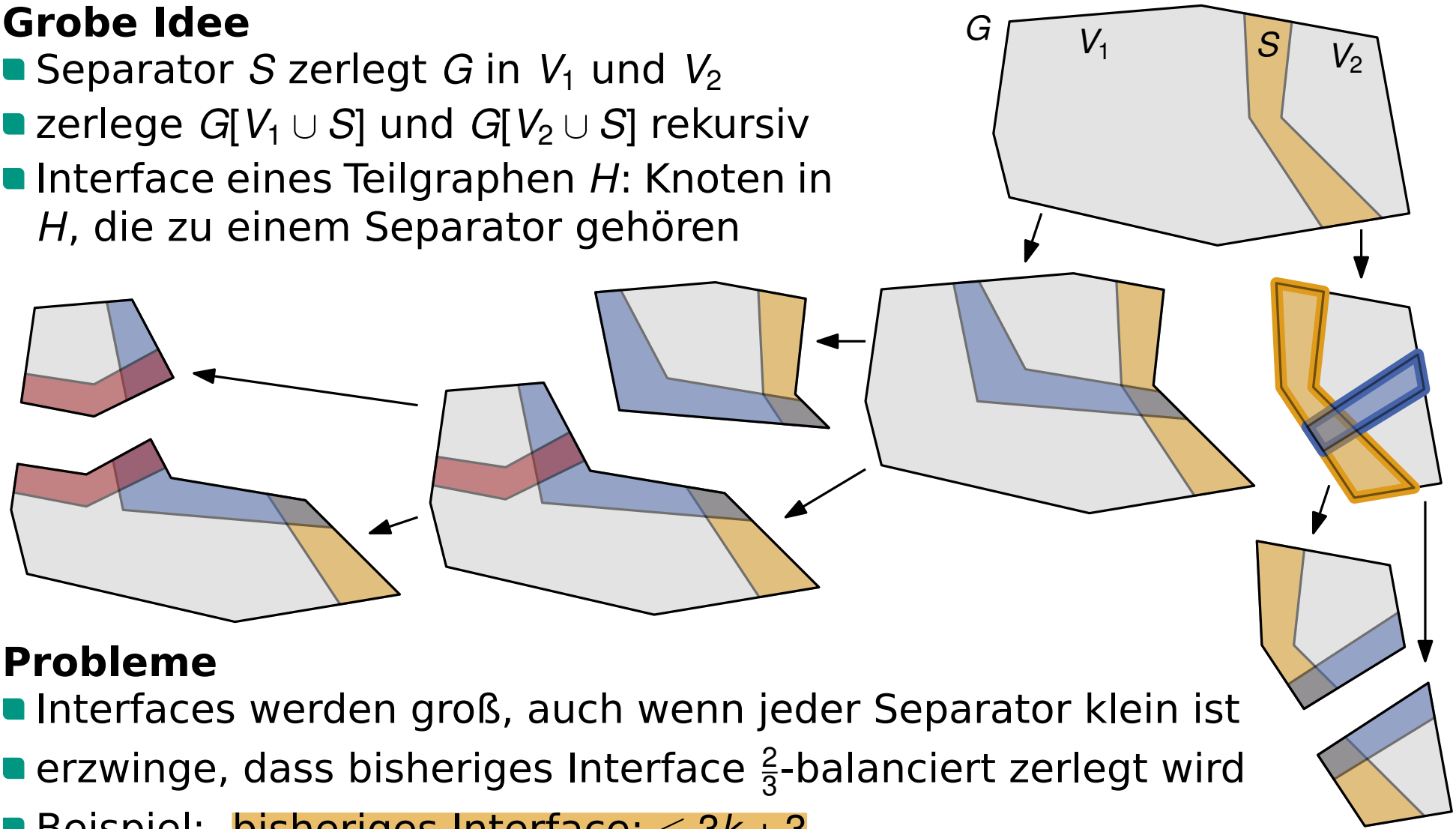
Probleme

- Interfaces werden groß, auch wenn jeder Separator klein ist
- erzwinge, dass bisheriges Interface $\frac{2}{3}$-balanciert zerlegt wird
- Beispiel: bisheriges Interface: $\leq 3k+3$

Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



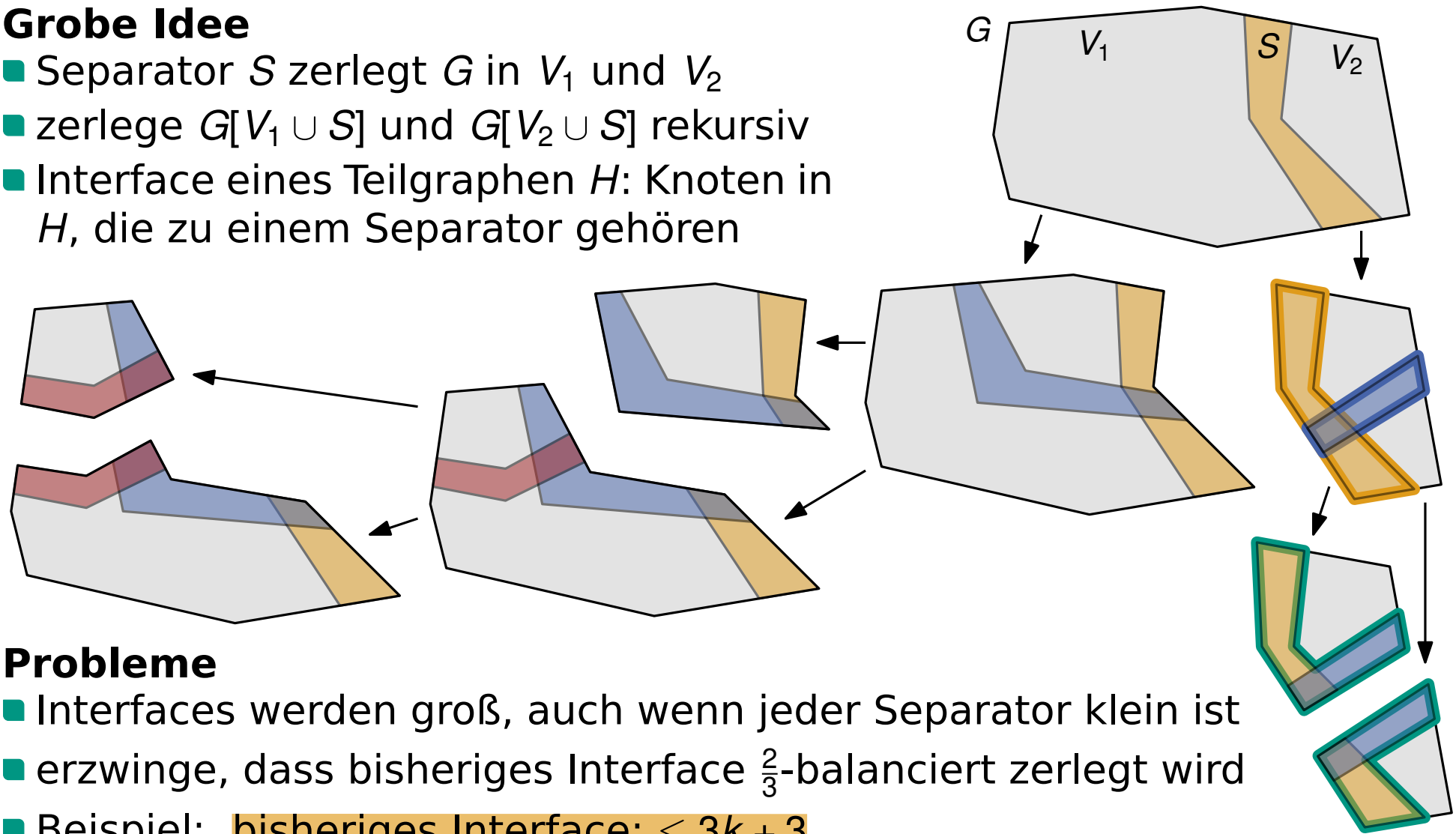
Probleme

- Interfaces werden groß, auch wenn jeder Separator klein ist
- erzwinge, dass bisheriges Interface $\frac{2}{3}$-balanciert zerlegt wird
- Beispiel: bisheriges Interface: $\leq 3k+3$
  aktueller Separator: $\leq k+1$

Berechnung einer Baumzerlegung

Grobe Idee

- Separator S zerlegt G in V_1 und V_2
- zerlege $G[V_1 \cup S]$ und $G[V_2 \cup S]$ rekursiv
- Interface eines Teilgraphen H: Knoten in H, die zu einem Separator gehören



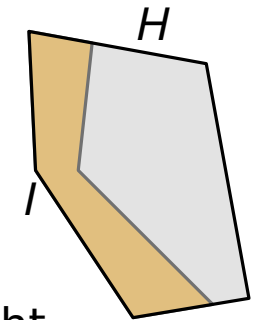
Probleme

- Interfaces werden groß, auch wenn jeder Separator klein ist
- erzwinge, dass bisheriges Interface $\frac{2}{3}$-balanciert zerlegt wird
- Beispiel: bisheriges Interface: $\leq 3k+3$
  aktueller Separator: $\leq k+1$
  $\Rightarrow$ neues Interface: $\leq 3k+3$

Baumweite $\Rightarrow$ Separator

Wir wünschen uns einen Separator S

- sei H der aktuelle Teilgraph mit Interface I ($|I| \leq 3k+4$)

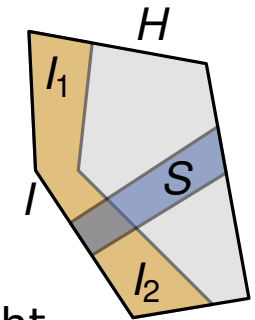


wir nehmen $3k+4$ statt $3k+3$, da es nicht wirklich einen Unterschied macht, die Laufzeitanalyse aber vereinfacht

Baumweite $\Rightarrow$ Separator

Wir wünschen uns einen Separator S

- sei H der aktuelle Teilgraph mit Interface I ($|I| \leq 3k+4$)
- S soll I in I_1, I_2 zerlegen, sodass $|I_1|, |I_2| \leq 2k+2$

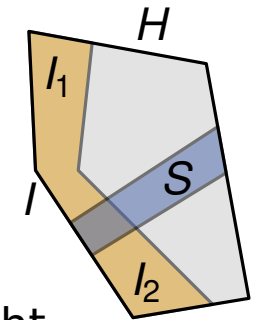


wir nehmen $3k+4$ statt $3k+3$, da es nicht wirklich einen Unterschied macht, die Laufzeitanalyse aber vereinfacht

Baumweite $\Rightarrow$ Separator

Wir wünschen uns einen Separator S

- sei H der aktuelle Teilgraph mit Interface I ($|I| \leq 3k+4$)
- S soll I in I_1, I_2 zerlegen, sodass $|I_1|, |I_2| \leq 2k+2$
- $|S| \leq k+1$ soll gelten

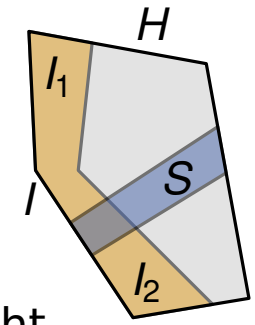


wir nehmen $3k{+}4$ statt $3k{+}3$, da es nicht wirklich einen Unterschied macht, die Laufzeitanalyse aber vereinfacht

Baumweite $\Rightarrow$ Separator

Wir wünschen uns einen Separator S

- sei H der aktuelle Teilgraph mit Interface I ($|I| \leq 3k+4$)
- S soll I in I_1, I_2 zerlegen, sodass $|I_1|, |I_2| \leq 2k+2$
- $|S| \leq k+1$ soll gelten



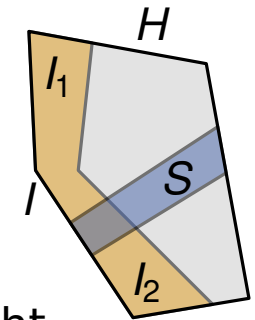
wir nehmen $3k+4$ statt $3k+3$, da es nicht wirklich einen Unterschied macht, die Laufzeitanalyse aber vereinfacht

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$.

Baumweite $\Rightarrow$ Separator

Wir wünschen uns einen Separator S

- sei H der aktuelle Teilgraph mit Interface I ($|I| \leq 3k+4$)
- S soll I in I_1, I_2 zerlegen, sodass $|I_1|, |I_2| \leq 2k+2$
- $|S| \leq k+1$ soll gelten



wir nehmen $3k+4$ statt $3k+3$, da es nicht wirklich einen Unterschied macht, die Laufzeitanalyse aber vereinfacht

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$.

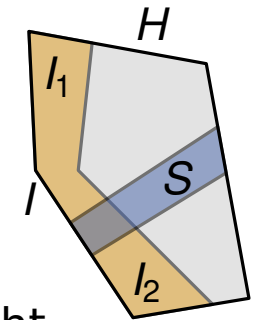
Plan im Folgenden

- zeige: kleine Baumweite $\Rightarrow$ kleine balancierte Separatoren

Baumweite $\Rightarrow$ Separator

Wir wünschen uns einen Separator S

- sei H der aktuelle Teilgraph mit Interface I ($|I| \leq 3k+4$)
- S soll I in I_1, I_2 zerlegen, sodass $|I_1|, |I_2| \leq 2k+2$
- $|S| \leq k+1$ soll gelten



wir nehmen $3k+4$ statt $3k+3$, da es nicht wirklich einen Unterschied macht, die Laufzeitanalyse aber vereinfacht

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$.

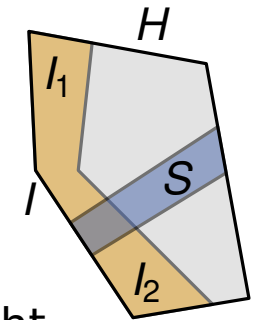
Plan im Folgenden

- zeige: kleine Baumweite $\Rightarrow$ kleine balancierte Separatoren
- folgere daraus das Lemma

Baumweite $\Rightarrow$ Separator

Wir wünschen uns einen Separator S

- sei H der aktuelle Teilgraph mit Interface I ($|I| \leq 3k+4$)
- S soll I in I_1, I_2 zerlegen, sodass $|I_1|, |I_2| \leq 2k+2$
- $|S| \leq k+1$ soll gelten



wir nehmen $3k+4$ statt $3k+3$, da es nicht wirklich einen Unterschied macht, die Laufzeitanalyse aber vereinfacht

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$.

Plan im Folgenden

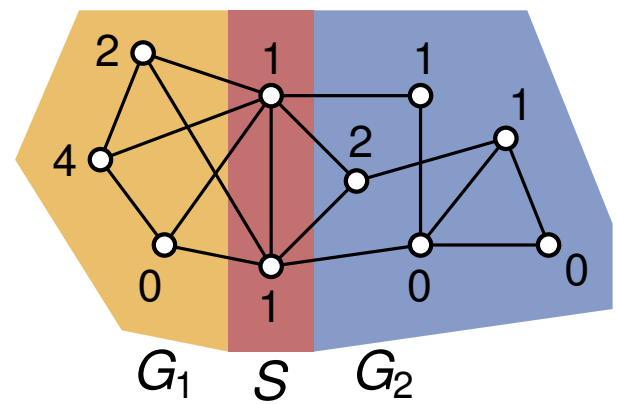
- zeige: kleine Baumweite $\Rightarrow$ kleine balancierte Separatoren
- folgere daraus das Lemma
- formalisiere eben gesehene rekursive Strategie
- Laufzeitanalyse

Balancierte Separatoren in Bäumen

Definition
Sei $G = (V, E)$ ein Graph mit Gewichten $w\colon V \to \mathbb{N}$. Ein Separator S, der G in G_1 und G_2 zerlegt, ist α-**balanciert**, wenn $w(G_1), w(G_2) \leq \alpha w(G)$.

($w(H)$ = Gesamtgewicht der Knoten im Teilgraph H)

Beispiel



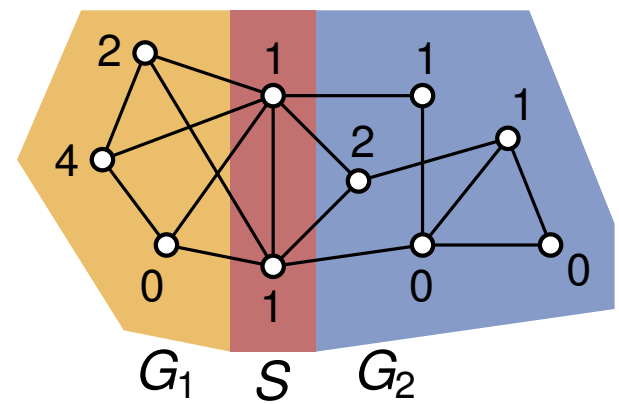
Balancierte Separatoren in Bäumen

Definition
Sei $G = (V, E)$ ein Graph mit Gewichten $w\colon V \to \mathbb{N}$. Ein Separator S, der G in G_1 und G_2 zerlegt, ist α-**balanciert**, wenn $w(G_1), w(G_2) \leq \alpha w(G)$.

($w(H)$ = Gesamtgewicht der Knoten im Teilgraph H)

Beispiel

- $w(G) = 12$
- $w(G_1) = 6$
- $w(G_2) = 4$



Balancierte Separatoren in Bäumen

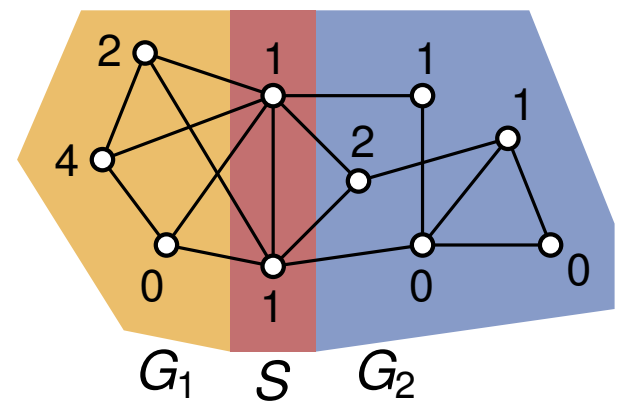
Definition
Sei $G = (V, E)$ ein Graph mit Gewichten $w\colon V \to \mathbb{N}$. Ein Separator S, der G in G_1 und G_2 zerlegt, ist α-**balanciert**, wenn $w(G_1), w(G_2) \leq \alpha w(G)$.

($w(H)$ = Gesamtgewicht der Knoten im Teilgraph H)

Beispiel

- $w(G) = 12$
- $w(G_1) = 6$
- $w(G_2) = 4$

$\Rightarrow \frac{1}{2}$-balanciert



Balancierte Separatoren in Bäumen

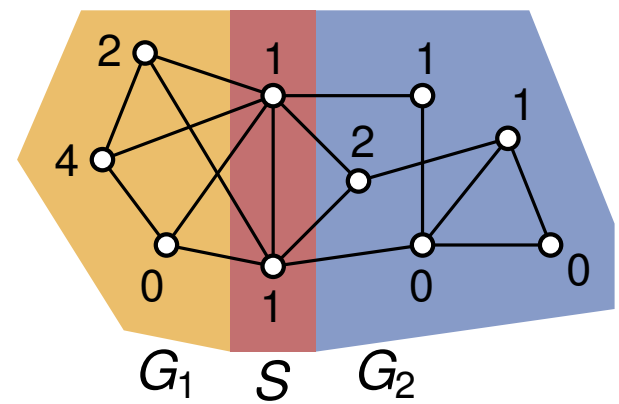
Definition
Sei $G = (V, E)$ ein Graph mit Gewichten $w\colon V \to \mathbb{N}$. Ein Separator S, der G in G_1 und G_2 zerlegt, ist α-**balanciert**, wenn $w(G_1), w(G_2) \leq \alpha w(G)$.

($w(H)$ = Gesamtgewicht der Knoten im Teilgraph H)

Beispiel

- $w(G) = 12$
- $w(G_1) = 6$
- $w(G_2) = 4$

$\Rightarrow \frac{1}{2}$-balanciert



Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Balancierte Separatoren in Bäumen

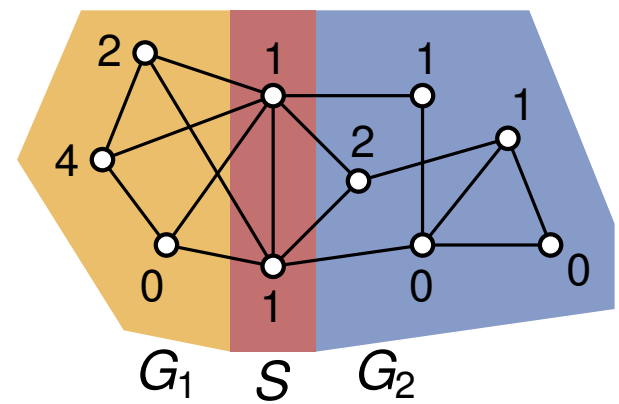
Definition
Sei $G = (V, E)$ ein Graph mit Gewichten $w\colon V \to \mathbb{N}$. Ein Separator S, der G in G_1 und G_2 zerlegt, ist α-**balanciert**, wenn $w(G_1), w(G_2) \leq \alpha w(G)$.

($w(H)$ = Gesamtgewicht der Knoten im Teilgraph H)

Beispiel

- $w(G) = 12$
- $w(G_1) = 6$
- $w(G_2) = 4$

$\Rightarrow \frac{1}{2}$-balanciert



Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Folgerung
In jedem (gew.) Baum gibt einen $\frac{2}{3}$-balancierten Separator der Größe 1.

Balancierte Separatoren in Bäumen

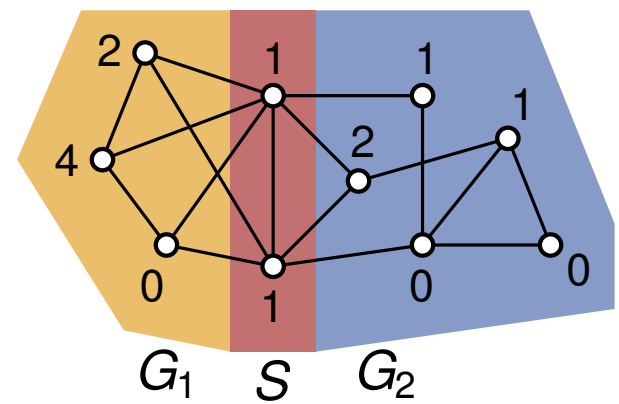
Definition
Sei $G = (V, E)$ ein Graph mit Gewichten $w\colon V \to \mathbb{N}$. Ein Separator S, der G in G_1 und G_2 zerlegt, ist α-**balanciert**, wenn $w(G_1), w(G_2) \leq \alpha w(G)$.

($w(H)$ = Gesamtgewicht der Knoten im Teilgraph H)

Beispiel

- $w(G) = 12$
- $w(G_1) = 6$
- $w(G_2) = 4$

$\Rightarrow \frac{1}{2}$-balanciert



Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Folgerung
In jedem (gew.) Baum gibt einen $\frac{2}{3}$-balancierten Separator der Größe 1.

Warum folgt das?

Warum $\frac{2}{3}$ und nicht $\frac{1}{2}$?

Balancierte Separatoren in Bäumen

Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Beweis

Balancierte Separatoren in Bäumen

Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Beweis

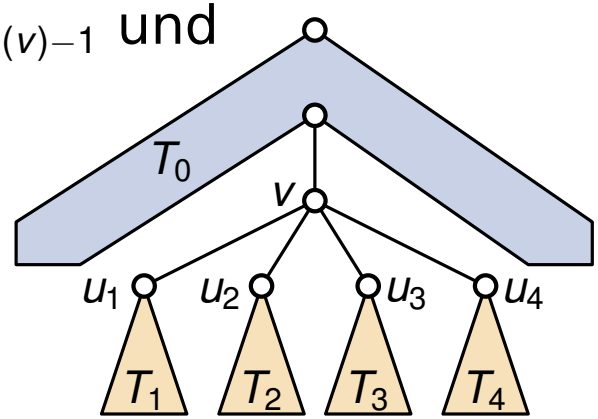
- wähle $v \in V$, sodass $w(T_v) \geq \frac{1}{2} w(T)$ und v hat maximale Distanz zur Wurzel r
  (T_v ist der Teilbaum unter v)

Balancierte Separatoren in Bäumen

Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Beweis

- wähle $v \in V$, sodass $w(T_v) \geq \frac{1}{2} w(T)$ und v hat maximale Distanz zur Wurzel r (T_v ist der Teilbaum unter v)
- $T - v$ hat $\deg(v) - 1$ *Kindkomponenten* $T_1, \ldots, T_{\deg(v)-1}$ und eine *Elternkomponente* T_0

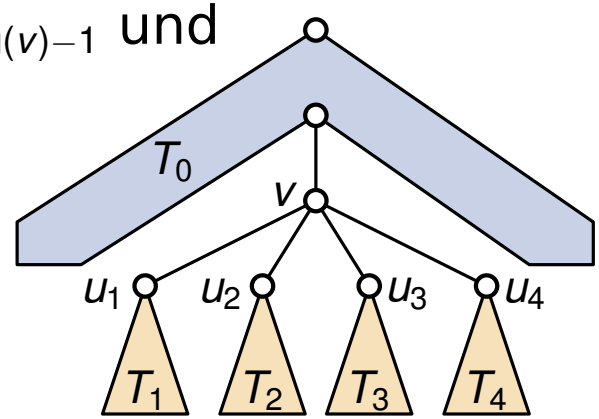


Balancierte Separatoren in Bäumen

Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Beweis

- wähle $v \in V$, sodass $w(T_v) \geq \frac{1}{2} w(T)$ und v hat maximale Distanz zur Wurzel r
  (T_v ist der Teilbaum unter v)
- $T - v$ hat $\deg(v) - 1$ *Kindkomponenten* $T_1, \ldots, T_{\deg(v)-1}$ und eine *Elternkomponente* T_0
- für $1 \leq i < \deg(v)$: $w(T_i) \leq \frac{1}{2} w(T)$
  (sonst wäre u_i statt v gewählt worden)

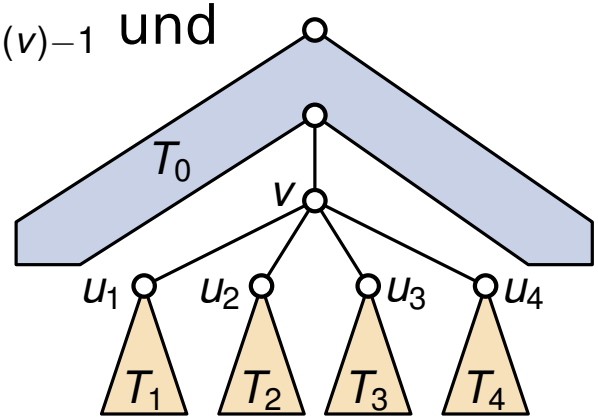


Balancierte Separatoren in Bäumen

Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Beweis

- wähle $v \in V$, sodass $w(T_v) \geq \frac{1}{2} w(T)$ und v hat maximale Distanz zur Wurzel r
  (T_v ist der Teilbaum unter v)
- $T - v$ hat $\deg(v) - 1$ *Kindkomponenten* $T_1, \ldots, T_{\deg(v)-1}$ und eine *Elternkomponente* T_0
- für $1 \leq i < \deg(v)$: $w(T_i) \leq \frac{1}{2} w(T)$
  (sonst wäre u_i statt v gewählt worden)
- außerdem: $w(T_0) = w(T) - w(T_v) \leq \frac{1}{2} w(T)$

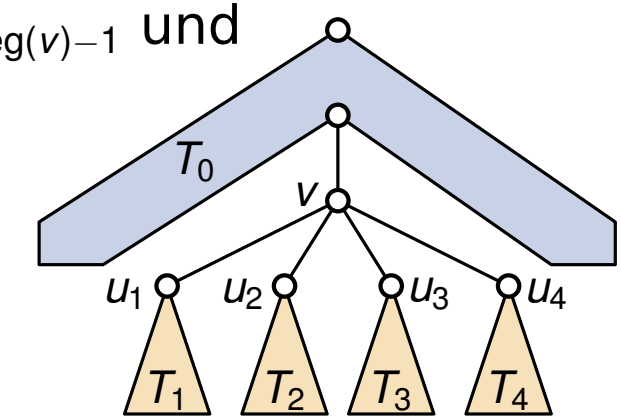


Balancierte Separatoren in Bäumen

Lemma
In jedem (gew.) Baum $T = (V, E)$ gibt es eine Menge $S \subseteq V$ mit $|S| = 1$, sodass $w(T_i) \leq \frac{1}{2} w(T)$ für jede Komponente T_i von $T - S$.

Beweis

- wähle $v \in V$, sodass $w(T_v) \geq \frac{1}{2} w(T)$ und v hat maximale Distanz zur Wurzel r
  (T_v ist der Teilbaum unter v)
- $T - v$ hat $\deg(v) - 1$ *Kindkomponenten* $T_1, \ldots, T_{\deg(v)-1}$ und eine *Elternkomponente* T_0
- für $1 \leq i < \deg(v)$: $w(T_i) \leq \frac{1}{2} w(T)$
  (sonst wäre u_i statt v gewählt worden)
- außerdem: $w(T_0) = w(T) - w(T_v) \leq \frac{1}{2} w(T)$



Lemma
In jedem (gew.) Graph $G = (V, E)$ mit Baumweite k gibt es eine Menge $S \subseteq V$ mit $|S| \leq k + 1$, sodass $w(G_i) \leq \frac{1}{2} w(G)$ für jede Komp. G_i von $G - S$.

Beweis: analog

Balancierte Separatoren & Baumweite

Lemma
In jedem (gew.) Graph $G = (V, E)$ mit Baumweite k gibt es eine Menge $S \subseteq V$ mit $|S| \leq k+1$, sodass $w(G_i) \leq \frac{1}{2} w(G)$ für jede Komp. G_i von $G - S$.

Folgerung
In jedem (gew.) Graphen mit Baumweite k gibt einen $\frac{2}{3}$-balancierten Separator mit maximal $k+1$ Knoten.

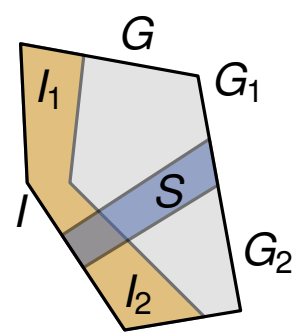
Balancierte Separatoren & Baumweite

Lemma
In jedem (gew.) Graph $G = (V, E)$ mit Baumweite k gibt es eine Menge $S \subseteq V$ mit $|S| \leq k+1$, sodass $w(G_i) \leq \frac{1}{2} w(G)$ für jede Komp. G_i von $G - S$.

Folgerung
In jedem (gew.) Graphen mit Baumweite k gibt einen $\frac{2}{3}$-balancierten Separator mit maximal $k+1$ Knoten.

Lemma
Sei $G = (V, E)$ ein Graph mit $\mathrm{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$.

Beweis



Balancierte Separatoren & Baumweite

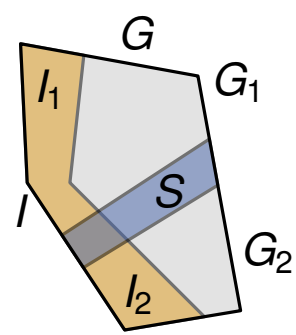
Lemma
In jedem (gew.) Graph $G = (V, E)$ mit Baumweite k gibt es eine Menge $S \subseteq V$ mit $|S| \leq k+1$, sodass $w(G_i) \leq \frac{1}{2} w(G)$ für jede Komp. G_i von $G - S$.

Folgerung
In jedem (gew.) Graphen mit Baumweite k gibt einen $\frac{2}{3}$-balancierten Separator mit maximal $k+1$ Knoten.

Lemma
Sei $G = (V, E)$ ein Graph mit $\mathrm{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$.

Beweis

- setze $w(v) = 1$ für $v \in I$ und $w(v) = 0$ sonst



Balancierte Separatoren & Baumweite

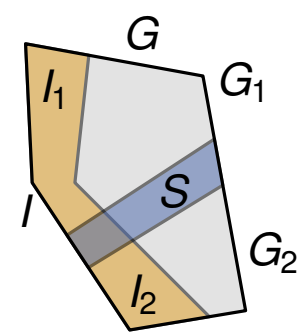
Lemma
In jedem (gew.) Graph $G = (V, E)$ mit Baumweite k gibt es eine Menge $S \subseteq V$ mit $|S| \leq k+1$, sodass $w(G_i) \leq \frac{1}{2} w(G)$ für jede Komp. G_i von $G - S$.

Folgerung
In jedem (gew.) Graphen mit Baumweite k gibt einen $\frac{2}{3}$-balancierten Separator mit maximal $k+1$ Knoten.

Lemma
Sei $G = (V, E)$ ein Graph mit $\mathrm{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$.

Beweis

- setze $w(v) = 1$ für $v \in I$ und $w(v) = 0$ sonst
- sei S ein $\frac{2}{3}$-balancierter Separator mit $|S| \leq k+1$, der G in G_1 und G_2, sowie I in I_1 und I_2 zerlegt



Balancierte Separatoren & Baumweite

Lemma
In jedem (gew.) Graph $G = (V, E)$ mit Baumweite k gibt es eine Menge $S \subseteq V$ mit $|S| \leq k+1$, sodass $w(G_i) \leq \frac{1}{2} w(G)$ für jede Komp. G_i von $G - S$.

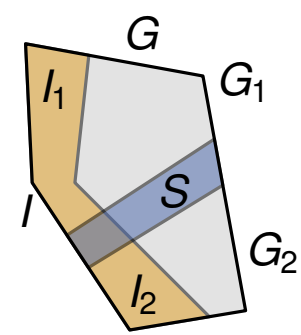
Folgerung
In jedem (gew.) Graphen mit Baumweite k gibt einen $\frac{2}{3}$-balancierten Separator mit maximal $k+1$ Knoten.

Lemma
Sei $G = (V, E)$ ein Graph mit $\mathrm{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$.

Beweis

- setze $w(v) = 1$ für $v \in I$ und $w(v) = 0$ sonst
- sei S ein $\frac{2}{3}$-balancierter Separator mit $|S| \leq k+1$, der G in G_1 und G_2, sowie I in I_1 und I_2 zerlegt

$\Rightarrow |I_1| = w(G_1), |I_2| = w(G_2) \leq \frac{2}{3} w(G) = \frac{2}{3}|I| \leq \frac{2}{3}(3k+4) \leq 2k+2$



Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

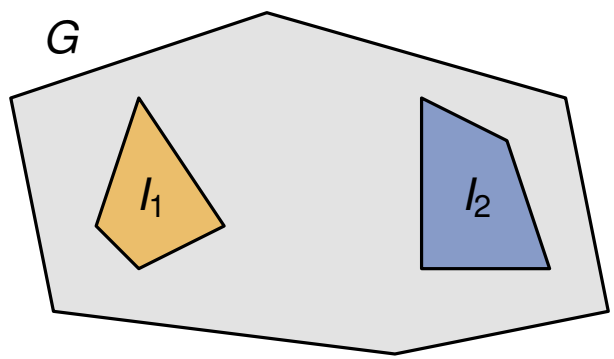
Beweis

Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.

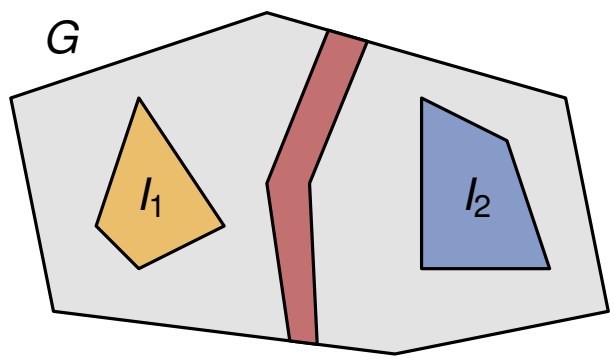


Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt

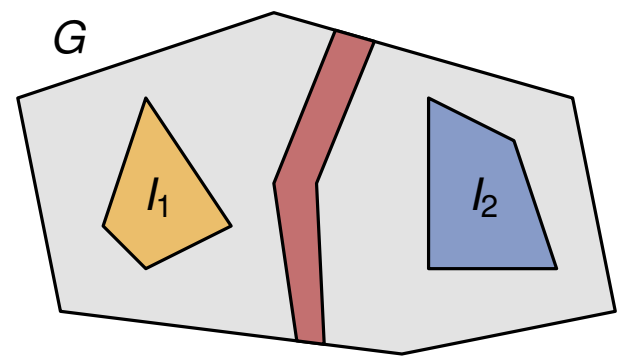


Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\mathrm{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

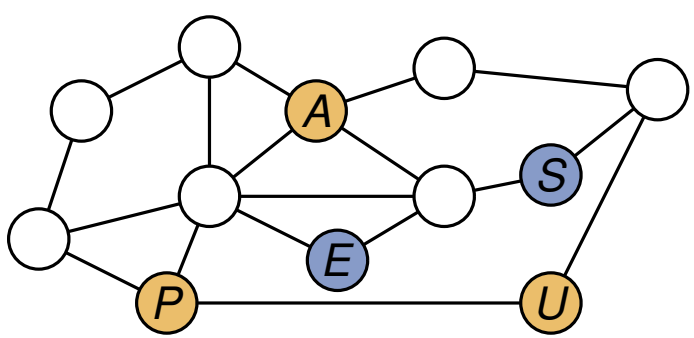
Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt



Beispiel

- $I_1 = \{P, A, U\}$ und $I_2 = \{S, E\}$

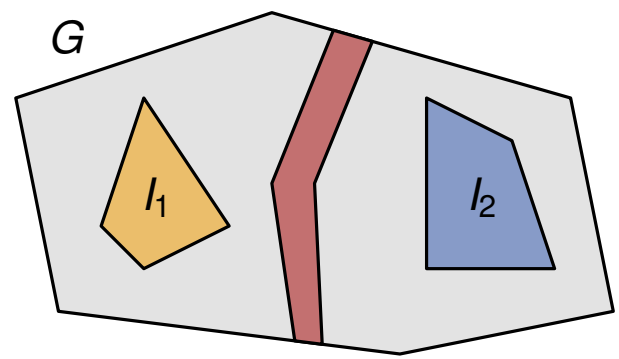


Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\mathrm{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

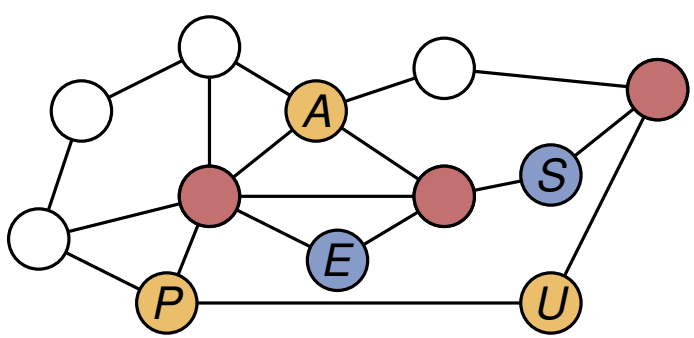
Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt



Beispiel

- $I_1 = \{P, A, U\}$ und $I_2 = \{S, E\}$

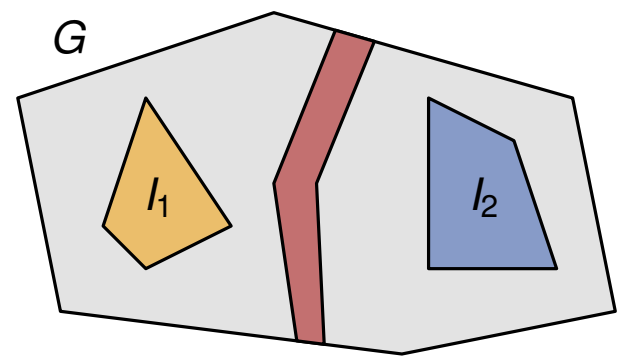


Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\mathrm{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

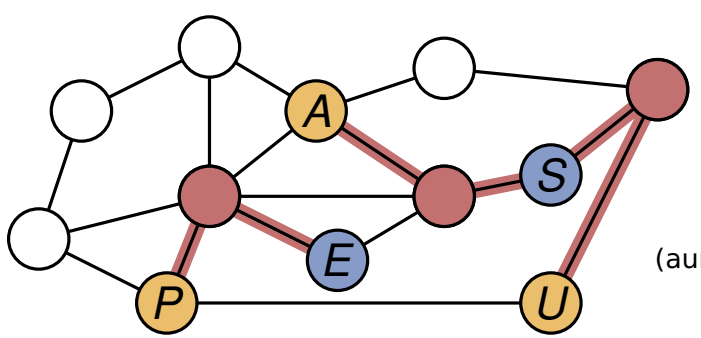
Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt



Beispiel

- $I_1 = \{P, A, U\}$ und $I_2 = \{S, E\}$



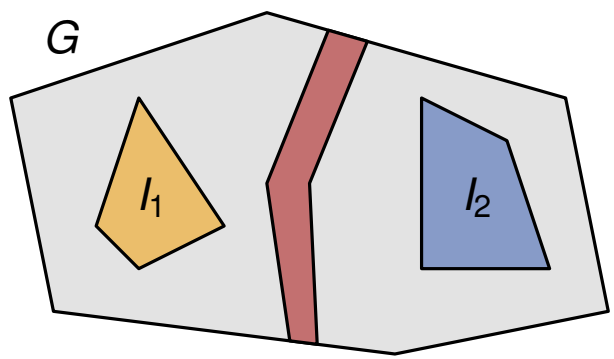
besser geht nicht: es gibt drei Knotendisjunkte Pfade
(außer, wenn man erlaubt Knoten aus I_1 oder I_2 zu löschen)

Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt
  - Flussberechnung mittels Ford-Fulkerson

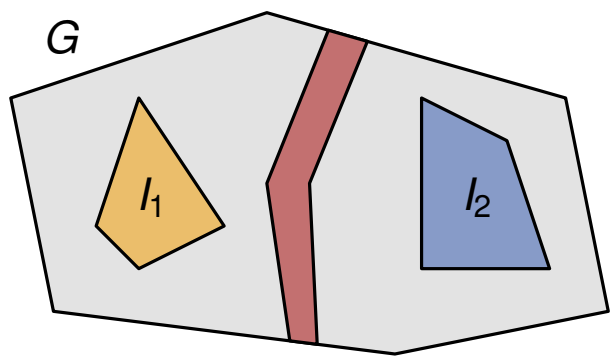


Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt
  - Flussberechnung mittels Ford-Fulkerson
  - finde knotendisjunkte erhöhende Wege ($\equiv$ Knotenkapazitäten 1)

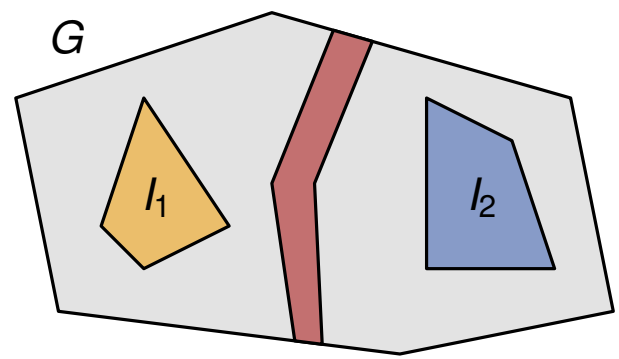


Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt
  - Flussberechnung mittels Ford-Fulkerson
  - finde knotendisjunkte erhöhende Wege ($\equiv$ Knotenkapazitäten 1)
  - maximal $k + 1$ erhöhende Wege nötig (sonst wird $|S| > k + 1$)

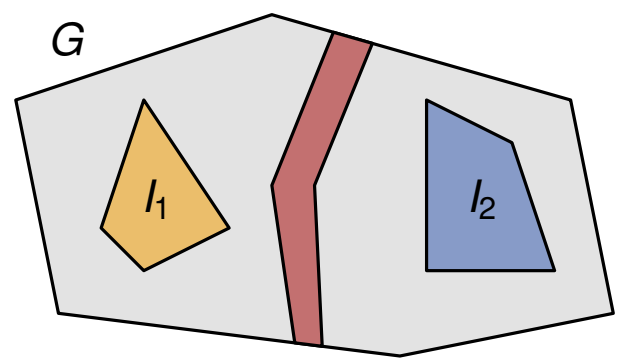


Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4} k^2 n)$ berechnet werden.

Beweis

- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt
  - Flussberechnung mittels Ford-Fulkerson
  - finde knotendisjunkte erhöhende Wege ($\equiv$ Knotenkapazitäten 1)
  - maximal $k + 1$ erhöhende Wege nötig (sonst wird $|S| > k + 1$)
  - $O(n + m)$ pro erhöhenden Weg (BFS), wobei $m \leq kn$

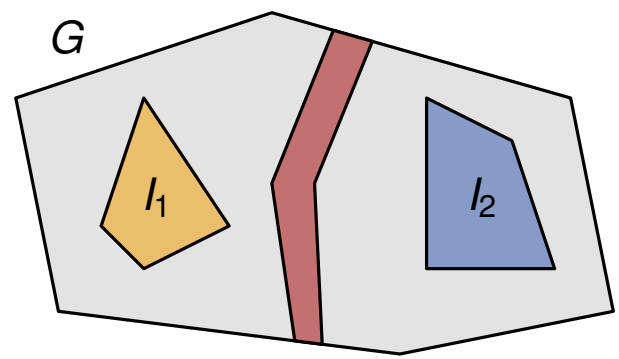


Berechnung eines Separators

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k + 4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4}k^2n)$ berechnet werden.

Beweis

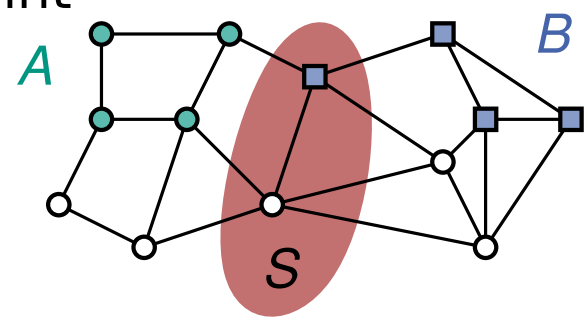
- probiere jede mögliche Aufteilung von I in I_1 und $I_2 \rightarrow 2^{3k+4}$ Mögl.
- berechne min. Separator ($\equiv$ Knotenschnitt), der I_1 und I_2 trennt
  - Flussberechnung mittels Ford-Fulkerson
  - finde knotendisjunkte erhöhende Wege ($\equiv$ Knotenkapazitäten 1)
  - maximal $k + 1$ erhöhende Wege nötig (sonst wird $|S| > k + 1$)
  - $O(n + m)$ pro erhöhenden Weg (BFS), wobei $m \leq kn$
  - $\Rightarrow O(k^2n)$ pro Aufteilung



Einschub: Separator mittels Fluss

Situation

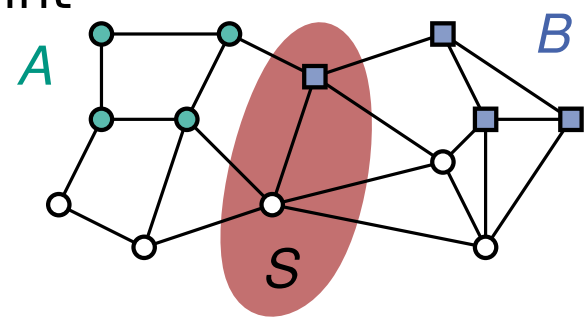
- gegeben: Graph $G = (V, E)$ und zwei Knotenmengen $A, B \subseteq V$ $(A \cap B = \emptyset)$
- finde kleinen Separator S in G, der A und B trennt
- S darf Knoten aus A und B enthalten



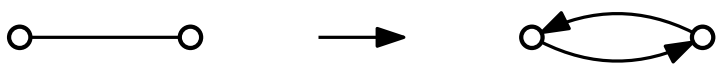
Einschub: Separator mittels Fluss

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



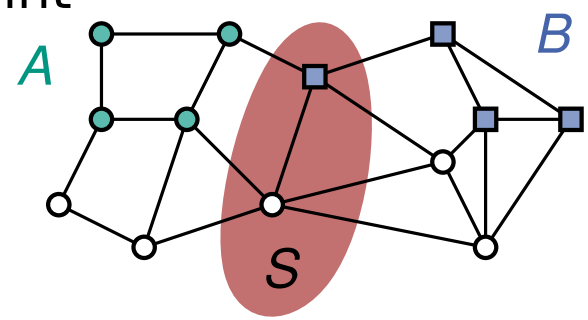
Schritt 1: betrachte gerichteten Graphen



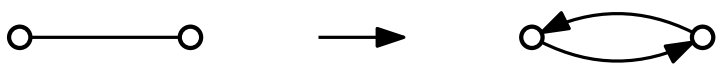
Einschub: Separator mittels Fluss

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

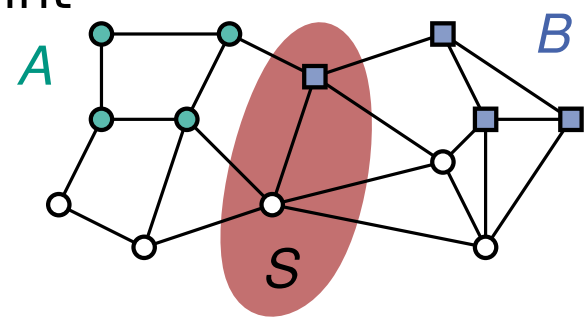


Schritt 2: Superquelle und -senke für A und B

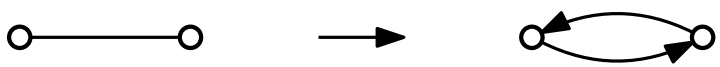
Einschub: Separator mittels Fluss

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

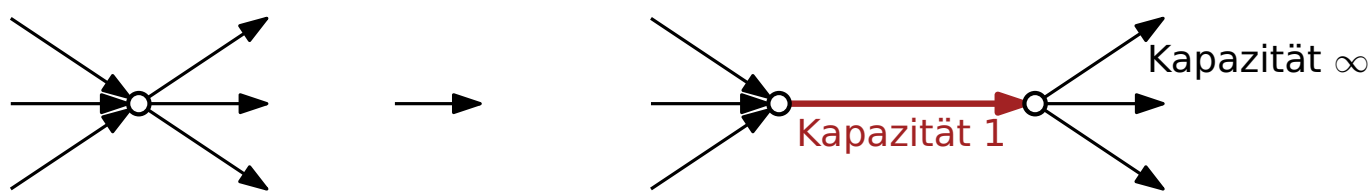


Schritt 1: betrachte gerichteten Graphen



Schritt 2: Superquelle und -senke für A und B

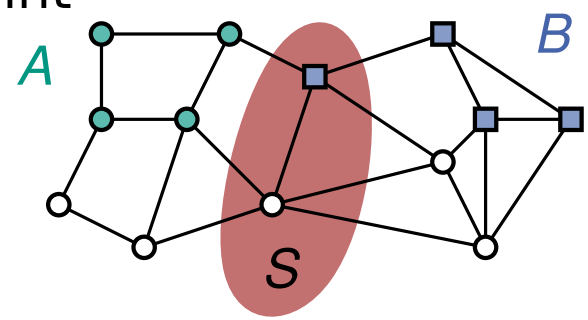
Schritt 3: Knotenkapazitäten durch Aufspaltung jedes Knotens in zwei



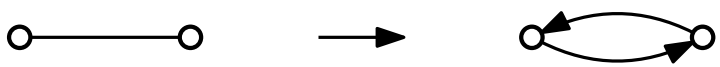
Einschub: Separator mittels Fluss

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

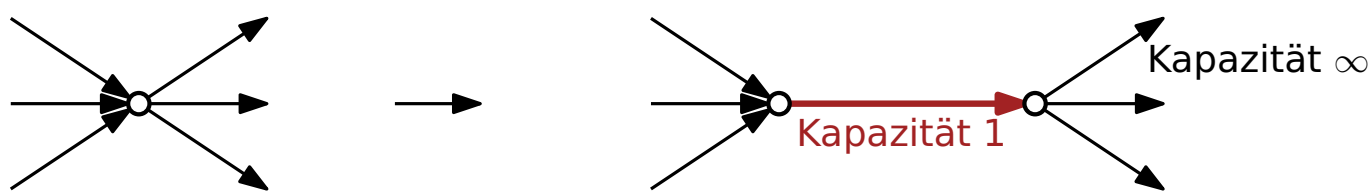


Schritt 1: betrachte gerichteten Graphen



Schritt 2: Superquelle und -senke für A und B

Schritt 3: Knotenkapazitäten durch Aufspaltung jedes Knotens in zwei

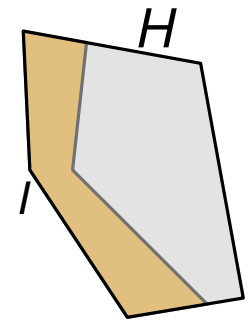


Schritt 4: maximaler *ab*-Fluss (minimaler *ab*-Schnitt)

Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

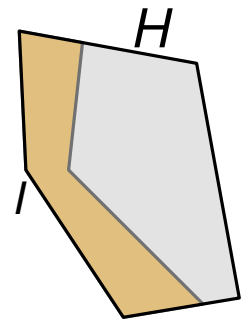
- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k + 4$



Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

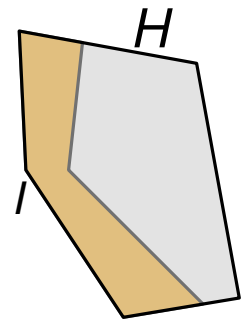
- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k + 4$
- oder entscheide, dass $\mathrm{tw}(H) > k$



Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k + 4$
- oder entscheide, dass $\mathrm{tw}(H) > k$

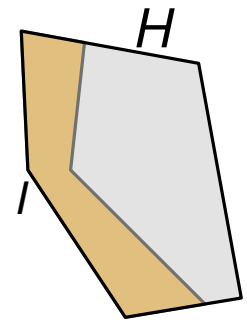


Invariante: $|I| \leq 3k + 3$

Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k + 4$
- oder entscheide, dass $\mathrm{tw}(H) > k$

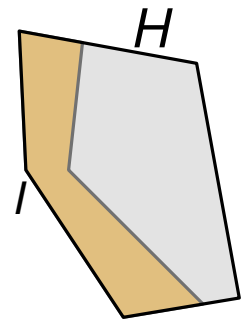


Invariante: $|I| \leq 3k + 3$

Umsetzung decomp(H, I)

- fertig, falls $|V(H)| \leq 3k + 3$

Berechnung einer Baumzerlegung



Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k + 4$
- oder entscheide, dass $\text{tw}(H) > k$

Invariante: $|I| \leq 3k + 3$

Umsetzung decomp(H, I)

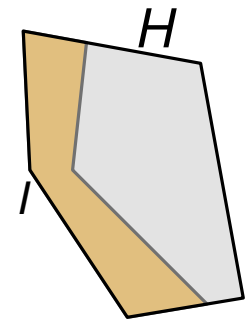
- fertig, falls $|V(H)| \leq 3k + 3$
- füge beliebigen Knoten aus H zu I hinzu $\Rightarrow |I| \leq 3k + 4$

(vereinfacht die Laufzeitanalyse)

Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k + 4$
- oder entscheide, dass $\text{tw}(H) > k$

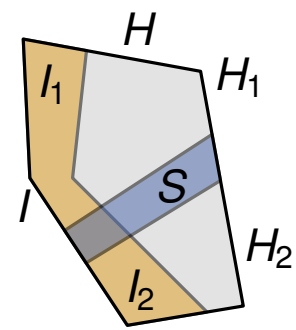


Invariante: $|I| \leq 3k + 3$

Umsetzung decomp(H, I)

- fertig, falls $|V(H)| \leq 3k + 3$
- füge beliebigen Knoten aus H zu I hinzu $\Rightarrow |I| \leq 3k + 4$
- berechne S, wie im Lemma ($\text{tw}(H) > k$, falls das scheitert)

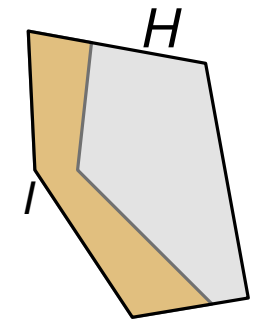
Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4}k^2n)$ berechnet werden.



Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k+4$
- oder entscheide, dass $\text{tw}(H) > k$

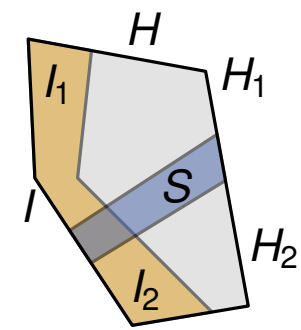


Invariante: $|I| \leq 3k+3$

Lemma
Sei $G=(V,E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$. Er kann in $O(2^{3k+4}k^2n)$ berechnet werden.

Umsetzung decomp(H, I)

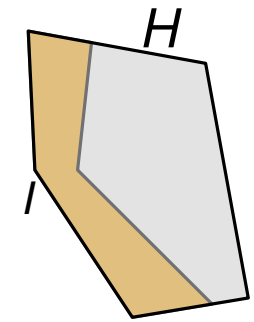
- fertig, falls $|V(H)| \leq 3k+3$
- füge beliebigen Knoten aus H zu I hinzu $\Rightarrow |I| \leq 3k+4$
- berechne S, wie im Lemma ($\text{tw}(H) > k$, falls das scheitert)
- erstelle Wurzel mit Bag $B = S \cup I$; $|B| \leq 4k+5$



Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k+4$
- oder entscheide, dass $\text{tw}(H) > k$

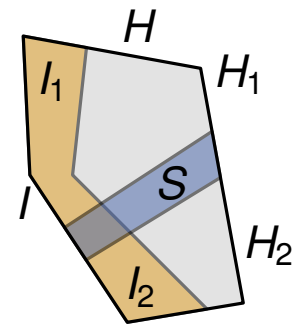


Invariante: $|I| \leq 3k+3$

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$. Er kann in $O(2^{3k+4}k^2n)$ berechnet werden.

Umsetzung decomp(H, I)

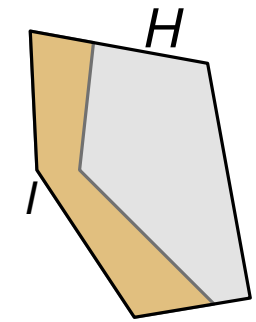
- fertig, falls $|V(H)| \leq 3k+3$
- füge beliebigen Knoten aus H zu I hinzu $\Rightarrow |I| \leq 3k+4$
- berechne S, wie im Lemma ($\text{tw}(H) > k$, falls das scheitert)
- erstelle Wurzel mit Bag $B = S \cup I$; $|B| \leq 4k+5$
- zwei Kinder: decomp$(H_1, I_1 \cup S)$ und decomp$(H_2, I_2 \cup S)$



Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k + 4$
- oder entscheide, dass $\text{tw}(H) > k$

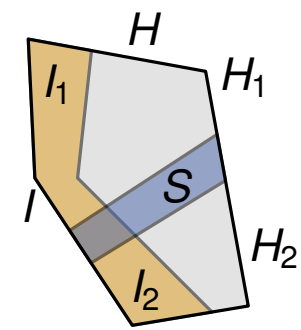


Invariante: $|I| \leq 3k + 3$

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4}k^2n)$ berechnet werden.

Umsetzung decomp(H, I)

- fertig, falls $|V(H)| \leq 3k + 3$
- füge beliebigen Knoten aus H zu I hinzu $\Rightarrow |I| \leq 3k + 4$
- berechne S, wie im Lemma ($\text{tw}(H) > k$, falls das scheitert)
- erstelle Wurzel mit Bag $B = S \cup I$; $|B| \leq 4k + 5$
- zwei Kinder: decomp$(H_1, I_1 \cup S)$ und decomp$(H_2, I_2 \cup S)$



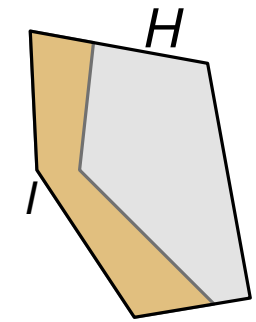
Laufzeit

- pro Aufruf wird mindestens ein Knoten zum Interface hinzugefügt

Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k+4$
- oder entscheide, dass $\text{tw}(H) > k$

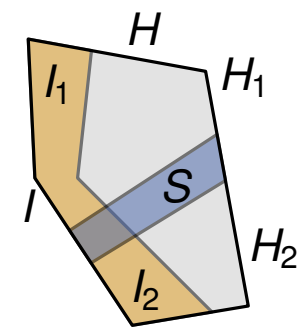


Invariante: $|I| \leq 3k+3$

Lemma
Sei $G=(V,E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k+1$, $|I_1|, |I_2| \leq 2k+2$. Er kann in $O(2^{3k+4}k^2n)$ berechnet werden.

Umsetzung decomp(H, I)

- fertig, falls $|V(H)| \leq 3k+3$
- füge beliebigen Knoten aus H zu I hinzu $\Rightarrow |I| \leq 3k+4$
- berechne S, wie im Lemma ($\text{tw}(H) > k$, falls das scheitert)
- erstelle Wurzel mit Bag $B = S \cup I$; $|B| \leq 4k+5$
- zwei Kinder: decomp$(H_1, I_1 \cup S)$ und decomp$(H_2, I_2 \cup S)$



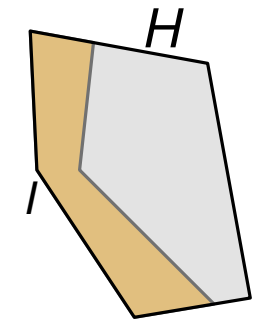
Laufzeit

- pro Aufruf wird mindestens ein Knoten zum Interface hinzugefügt
- das passiert jedem Knoten maximal ein Mal $\Rightarrow O(n)$ Aufrufe

Berechnung einer Baumzerlegung

Ziel der Methode decomp(H, I)

- berechnet Baumzerlegung von H, sodass
  - Wurzelbag enthält I
  - Weite der Zerlegung $\leq 4k + 4$
- oder entscheide, dass $\text{tw}(H) > k$

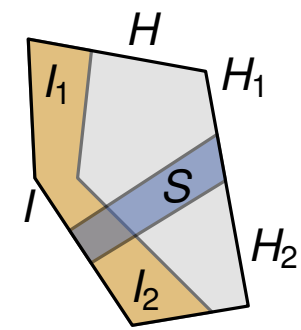


Invariante: $|I| \leq 3k + 3$

Lemma
Sei $G = (V, E)$ ein Graph mit $\text{tw}(G) \leq k$ und sei $I \subseteq V$ mit $|I| \leq 3k+4$. Dann gibt es einen Separator S in G, der I in I_1 und I_2 zerlegt, sodass $|S| \leq k + 1$, $|I_1|, |I_2| \leq 2k + 2$. Er kann in $O(2^{3k+4}k^2n)$ berechnet werden.

Umsetzung decomp(H, I)

- fertig, falls $|V(H)| \leq 3k + 3$
- füge beliebigen Knoten aus H zu I hinzu $\Rightarrow |I| \leq 3k + 4$
- berechne S, wie im Lemma ($\text{tw}(H) > k$, falls das scheitert)
- erstelle Wurzel mit Bag $B = S \cup I$; $|B| \leq 4k + 5$
- zwei Kinder: decomp$(H_1, I_1 \cup S)$ und decomp$(H_2, I_2 \cup S)$

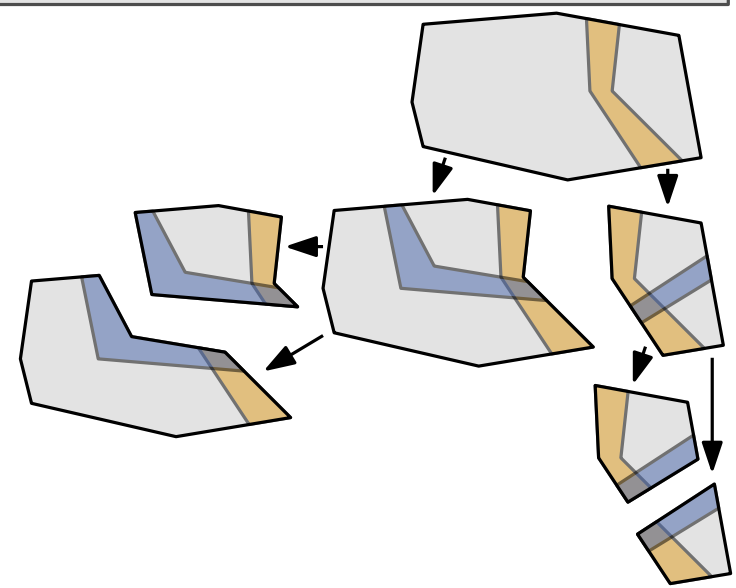


Laufzeit

- pro Aufruf wird mindestens ein Knoten zum Interface hinzugefügt
- das passiert jedem Knoten maximal ein Mal $\Rightarrow O(n)$ Aufrufe
- Laufzeit: $O(2^{3k+4}k^2n^2) = O(8^k k^2 n^2)$

Berechnung einer Baumzerlegung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

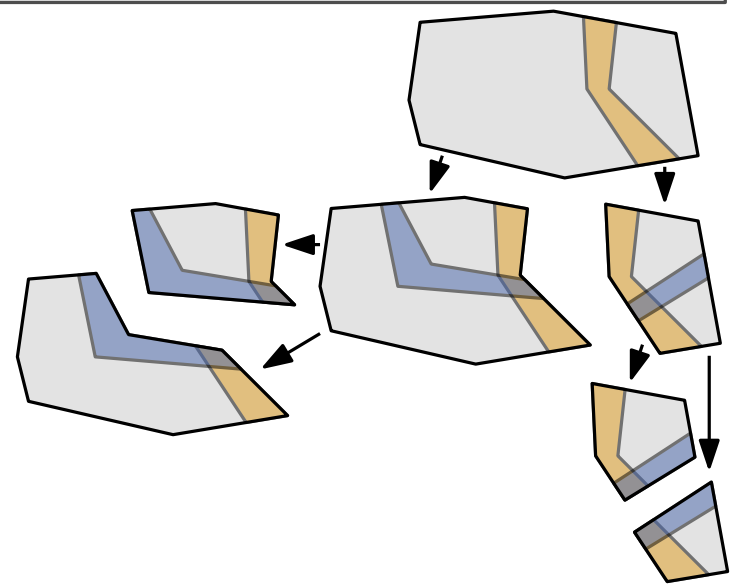


Berechnung einer Baumzerlegung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\text{tw}(G) > k$.

Bisher bewiesen

- Weite ist maximal $4k + 4$
- Laufzeit



Berechnung einer Baumzerlegung

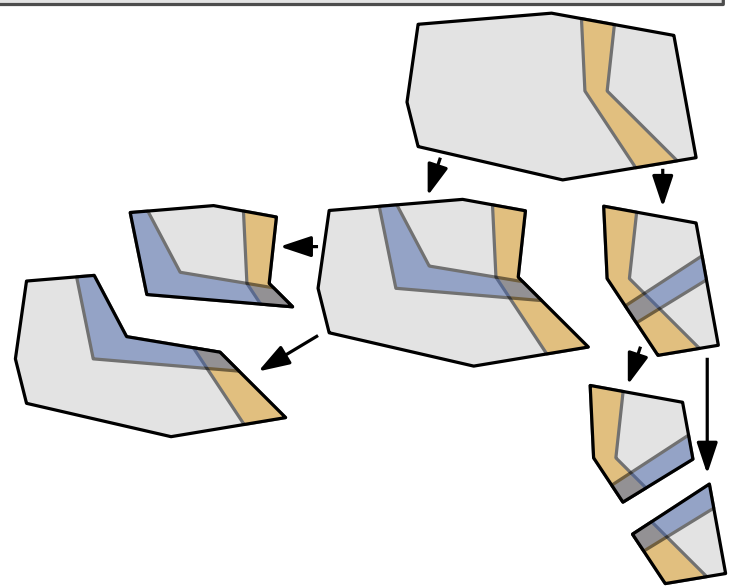
Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k+4$ berechnet oder entscheidet, dass $\text{tw}(G) > k$.

Bisher bewiesen

- Weite ist maximal $4k+4$
- Laufzeit

Ergebnis ist Baumzerlegung

- jeder Knoten in einem Bag enthalten



Berechnung einer Baumzerlegung

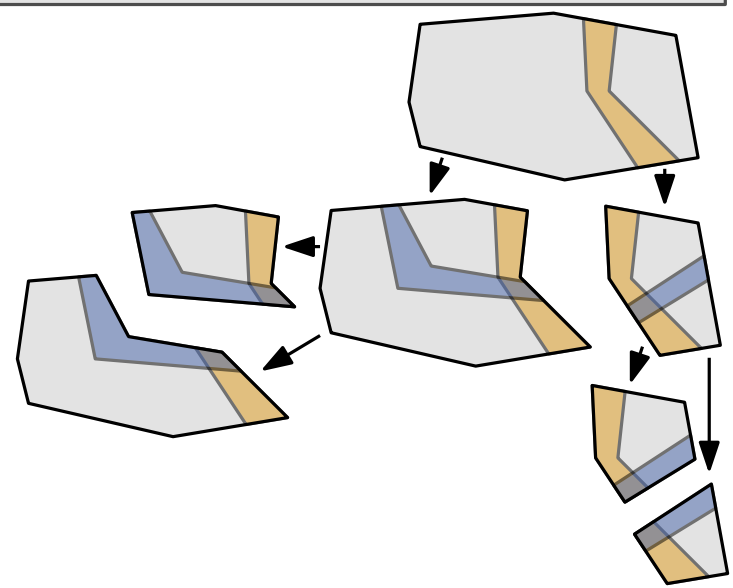
Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k+4$ berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

Bisher bewiesen

- Weite ist maximal $4k+4$
- Laufzeit

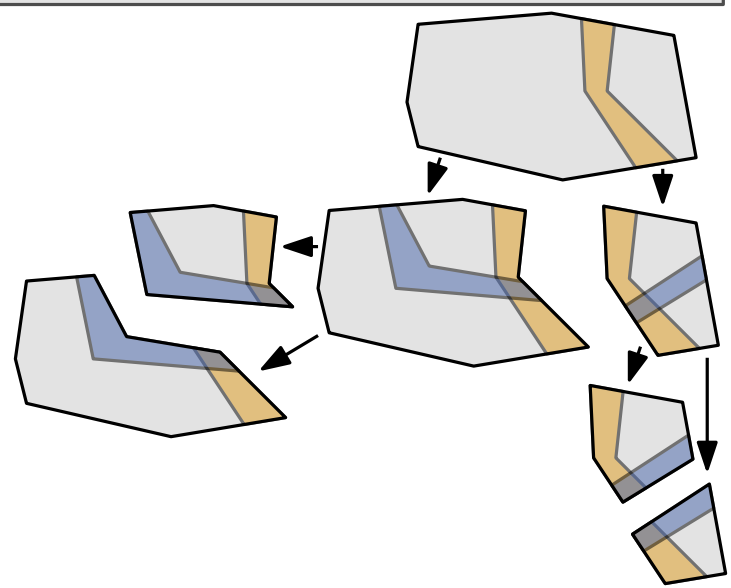
Ergebnis ist Baumzerlegung

- jeder Knoten in einem Bag enthalten
- jede Kante wird repräsentiert



Berechnung einer Baumzerlegung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k+4$ berechnet oder entscheidet, dass $\text{tw}(G) > k$.



Bisher bewiesen

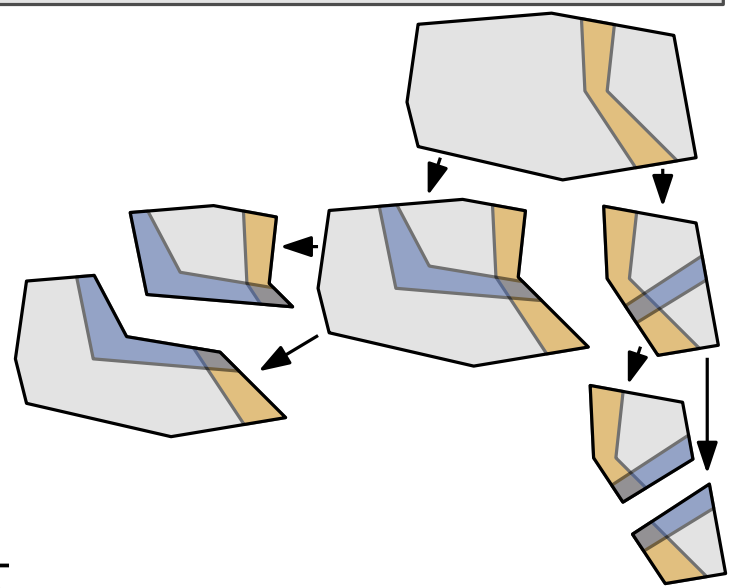
- Weite ist maximal $4k+4$
- Laufzeit

Ergebnis ist Baumzerlegung

- jeder Knoten in einem Bag enthalten
- jede Kante wird repräsentiert
  - $uv \in E \Rightarrow u$ und v werden nie separiert

Berechnung einer Baumzerlegung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k+4$ berechnet oder entscheidet, dass $\text{tw}(G) > k$.



Bisher bewiesen

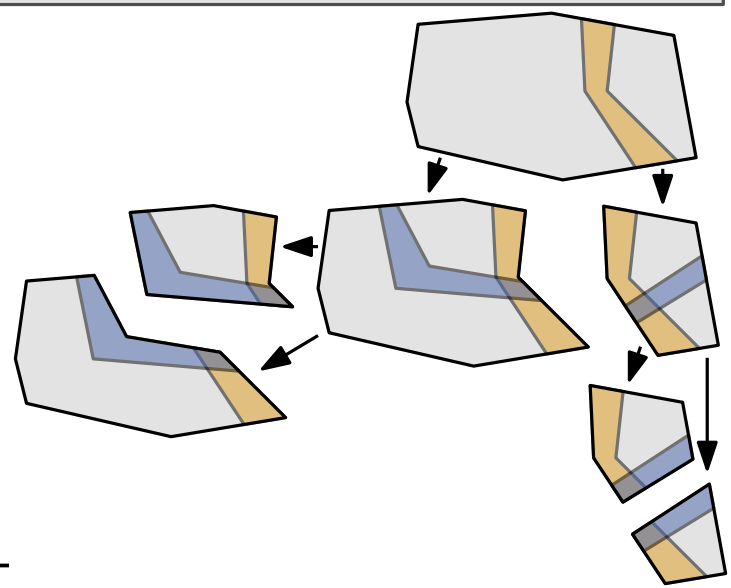
- Weite ist maximal $4k+4$
- Laufzeit

Ergebnis ist Baumzerlegung

- jeder Knoten in einem Bag enthalten
- jede Kante wird repräsentiert
  - $uv \in E \Rightarrow u$ und v werden nie separiert
  - u und v teilen sich ein Bag in einem Blatt

Berechnung einer Baumzerlegung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k+4$ berechnet oder entscheidet, dass $\text{tw}(G) > k$.



Bisher bewiesen

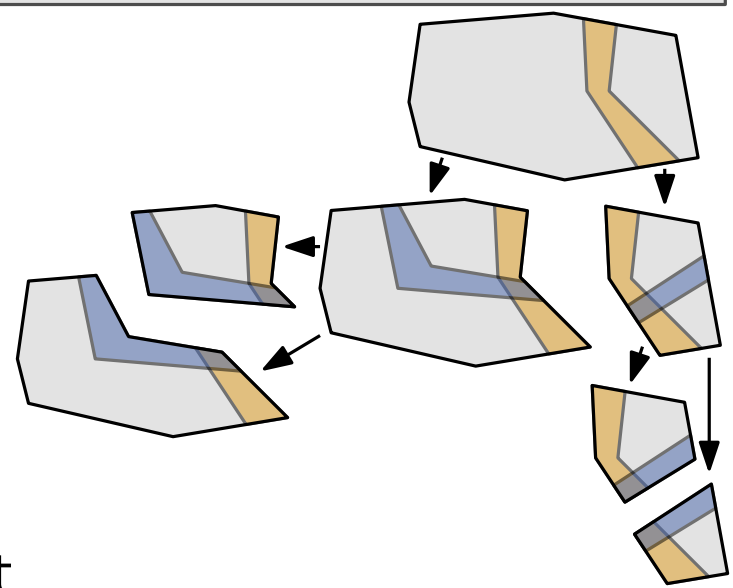
- Weite ist maximal $4k+4$
- Laufzeit

Ergebnis ist Baumzerlegung

- jeder Knoten in einem Bag enthalten
- jede Kante wird repräsentiert
  - $uv \in E \Rightarrow u$ und v werden nie separiert
  - u und v teilen sich ein Bag in einem Blatt
- Bags jedes Knotens induzieren Teilbaum

Berechnung einer Baumzerlegung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k+4$ berechnet oder entscheidet, dass $\text{tw}(G) > k$.



Bisher bewiesen

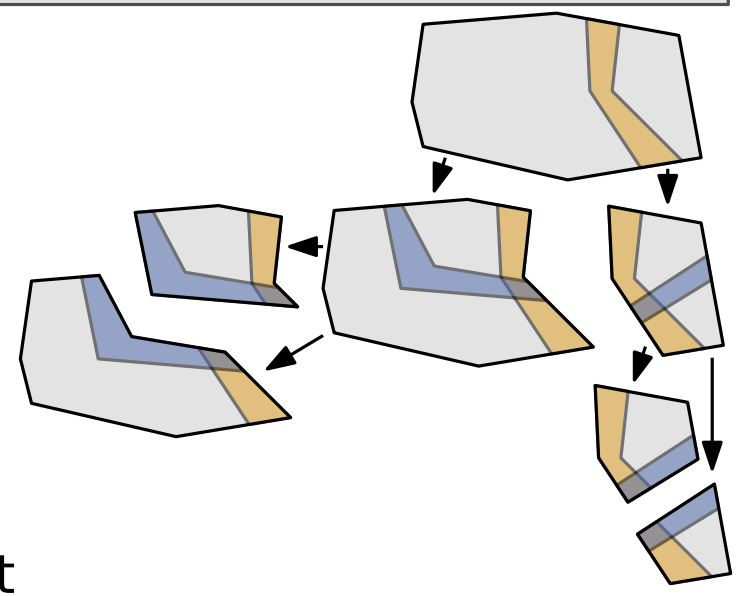
- Weite ist maximal $4k+4$
- Laufzeit

Ergebnis ist Baumzerlegung

- jeder Knoten in einem Bag enthalten
- jede Kante wird repräsentiert
  - $uv \in E \Rightarrow u$ und v werden nie separiert
  - u und v teilen sich ein Bag in einem Blatt
- Bags jedes Knotens induzieren Teilbaum
  - betrachte Bag B mit $v \in B$, sodass B möglichst nah an der Wurzel

Berechnung einer Baumzerlegung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k+4$ berechnet oder entscheidet, dass $\text{tw}(G) > k$.



Bisher bewiesen

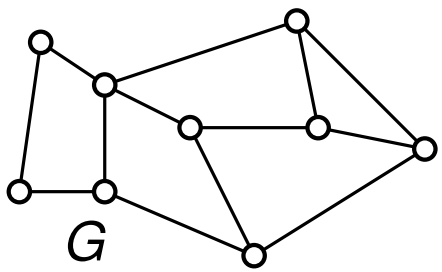
- Weite ist maximal $4k+4$
- Laufzeit

Ergebnis ist Baumzerlegung

- jeder Knoten in einem Bag enthalten
- jede Kante wird repräsentiert
  - $uv \in E \Rightarrow u$ und v werden nie separiert
  - u und v teilen sich ein Bag in einem Blatt
- Bags jedes Knotens induzieren Teilbaum
  - betrachte Bag B mit $v \in B$, sodass B möglichst nah an der Wurzel
  - für Nachfolger B' von B gilt: entweder $v \in B'$ oder $v \notin B''$ für jeden Nachfolger B'' von B'

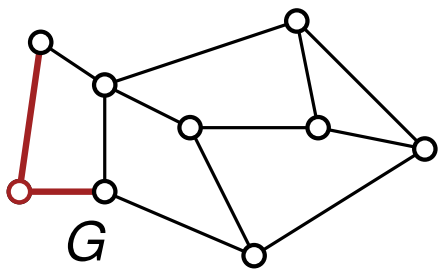
Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



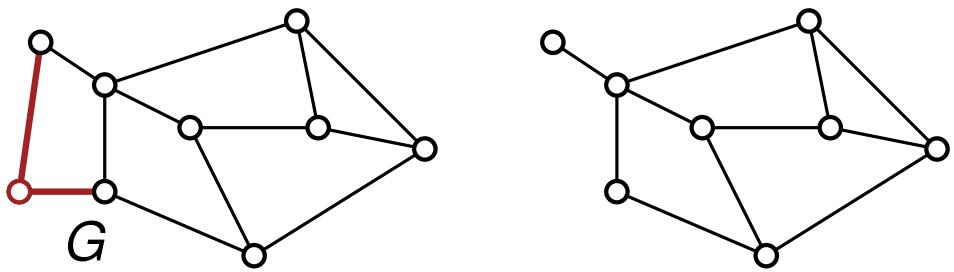
Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



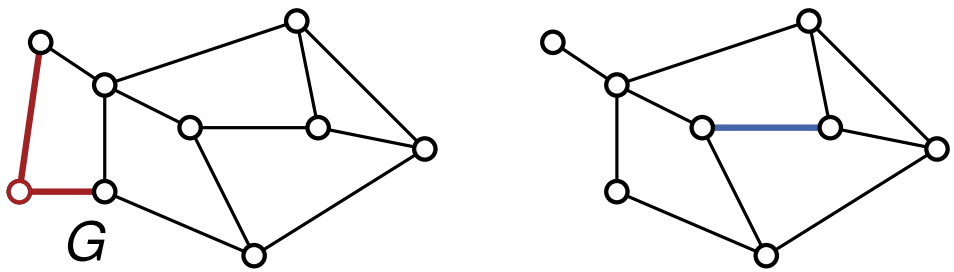
Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



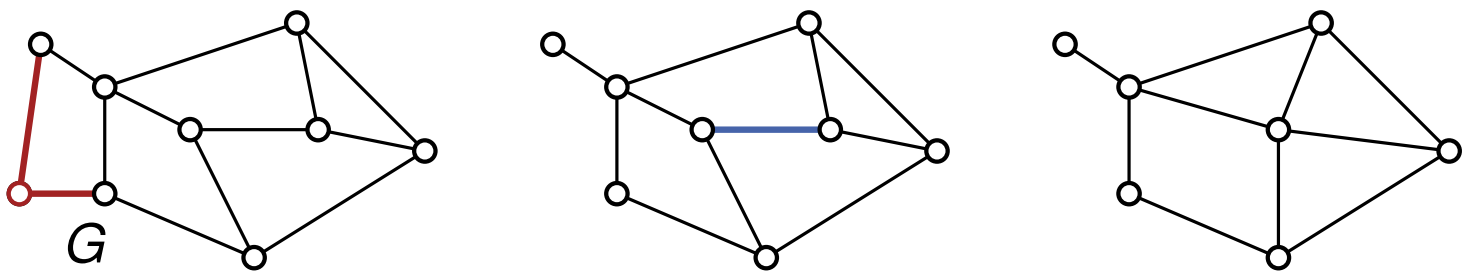
Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



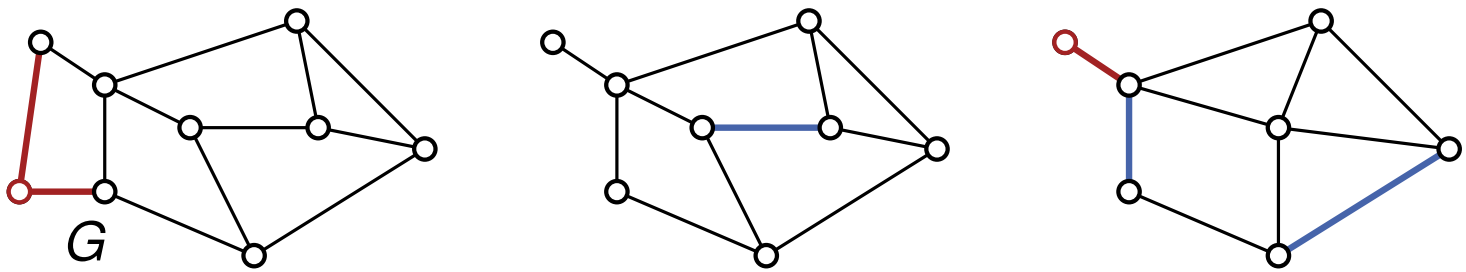
Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



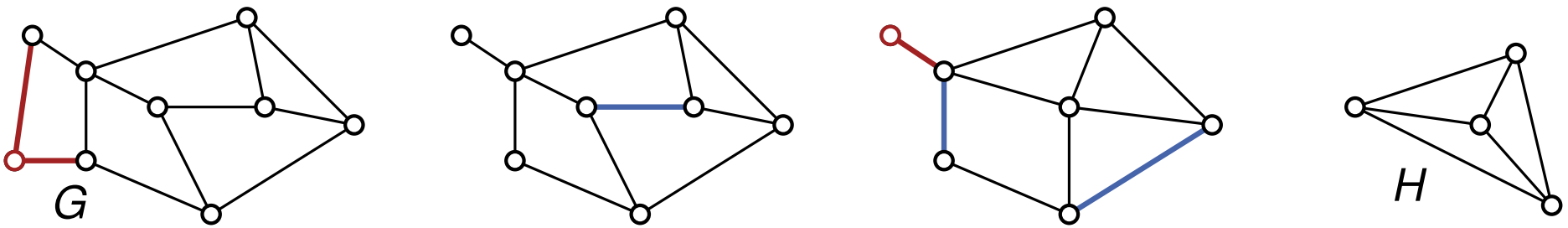
Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



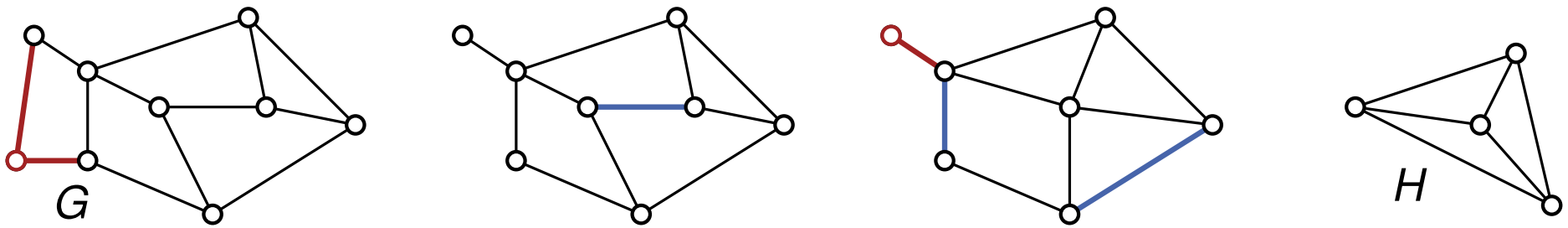
Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.

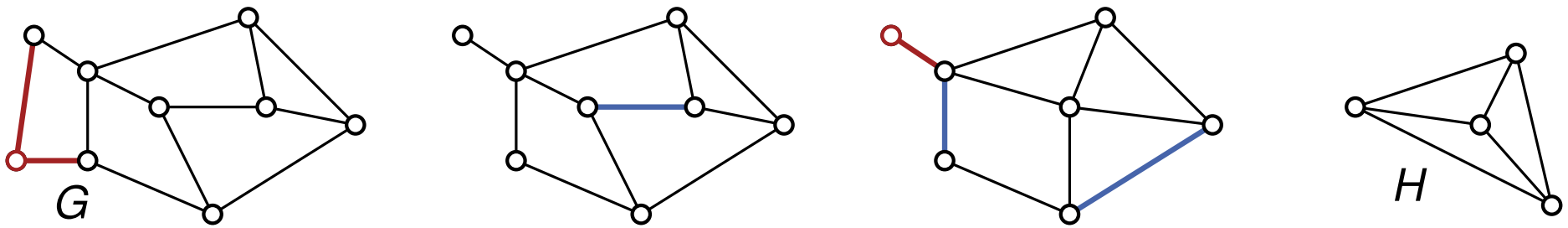


Was hat das mit Baumweite zu tun?

- Baumweite ist monoton bezüglich Minorenbildung

Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.

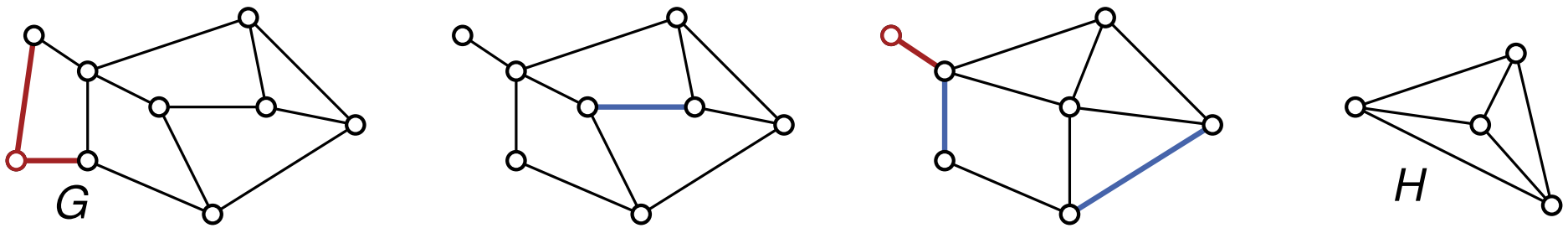


Was hat das mit Baumweite zu tun?

- Baumweite ist monoton bezüglich Minorenbildung
- große Clique als Minor $\Rightarrow$ große Baumweite

Baumweite und verbotene Minoren

Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.

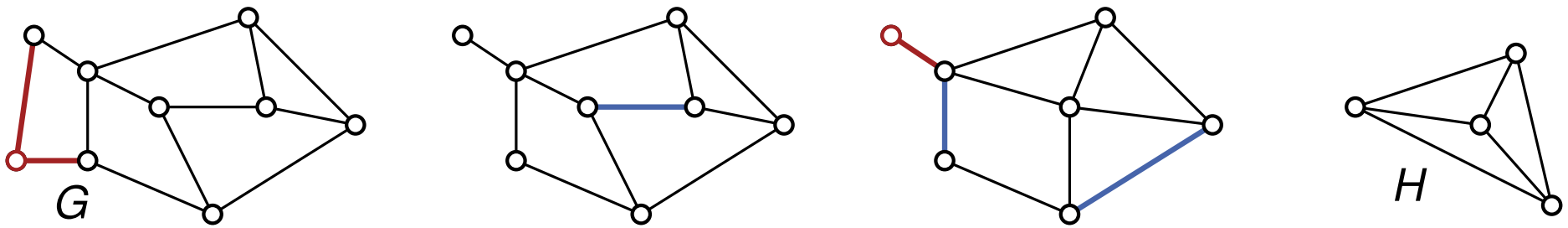


Was hat das mit Baumweite zu tun?

- Baumweite ist monoton bezüglich Minorenbildung
- große Clique als Minor $\Rightarrow$ große Baumweite
- Umkehrung gilt leider nicht

Baumweite und verbotene Minoren

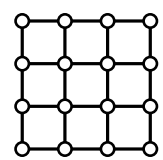
Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



Was hat das mit Baumweite zu tun?

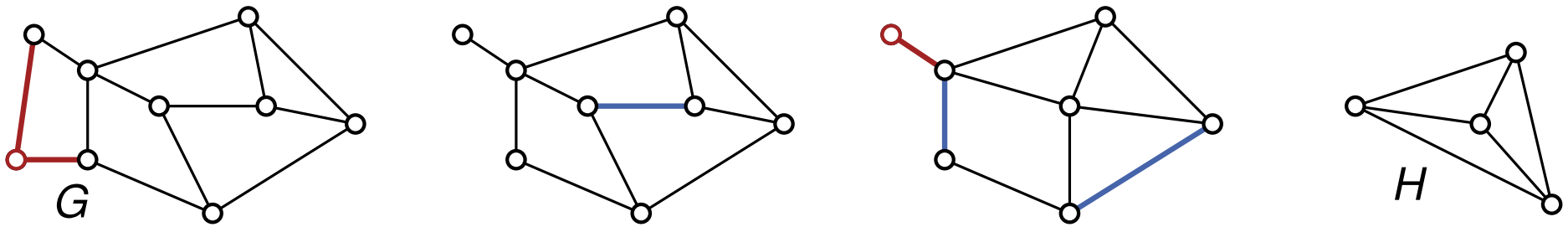
- Baumweite ist monoton bezüglich Minorenbildung
- große Clique als Minor $\Rightarrow$ große Baumweite
- Umkehrung gilt leider nicht
- aber: große Baumweite $\Rightarrow$ großes Gitter als Minor

Gitter Γ_4



Baumweite und verbotene Minoren

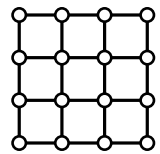
Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



Was hat das mit Baumweite zu tun?

- Baumweite ist monoton bezüglich Minorenbildung
- große Clique als Minor $\Rightarrow$ große Baumweite
- Umkehrung gilt leider nicht
- aber: große Baumweite $\Rightarrow$ großes Gitter als Minor

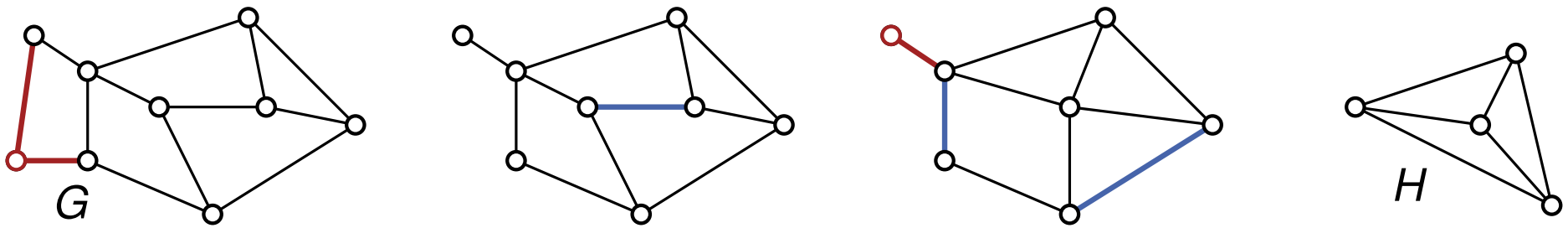
Gitter Γ_4



Theorem
Es gibt eine Funktion $g(t) = O(t^{98+o(1)})$, sodass jeder Graph mit Baumweite über $g(t)$ das Gitter Γ_t als Minor enthält.

Baumweite und verbotene Minoren

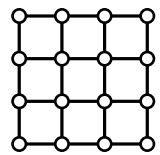
Definition
Ein Graph H ist ein **Minor** von G, wenn man H durch Subgraphbildung und Kantenkontraktion aus G erhalten kann.



Was hat das mit Baumweite zu tun?

- Baumweite ist monoton bezüglich Minorenbildung
- große Clique als Minor $\Rightarrow$ große Baumweite
- Umkehrung gilt leider nicht
- aber: große Baumweite $\Rightarrow$ großes Gitter als Minor

Gitter Γ_4



Theorem
Es gibt eine Funktion $g(t) = O(t^{98+o(1)})$, sodass jeder Graph mit Baumweite über $g(t)$ das Gitter Γ_t als Minor enthält.

Baumweite muss **sehr** groß sein, um ein großes Gitter zu garantieren

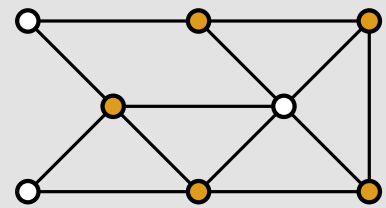
Gitter, Planarität und Vertex Cover

Theorem
Jeder planare Graph G mit Baumweite $\geq \frac{9}{2}t$ enthält Γ_t als Minor. Für jedes $\varepsilon > 0$ gibt es einen $O(n^2)$ Algorithmus, der entweder eine Baumzerlegung der Weite $(\frac{9}{2} + \varepsilon)t$ oder einen Γ_t Minor von G konstruiert.

Gitter, Planarität und Vertex Cover

Theorem
Jeder planare Graph G mit Baumweite $\geq \frac{9}{2}t$ enthält Γ_t als Minor. Für jedes $\varepsilon > 0$ gibt es einen $O(n^2)$ Algorithmus, der entweder eine Baumzerlegung der Weite $(\frac{9}{2} + \varepsilon)t$ oder einen Γ_t Minor von G konstruiert.

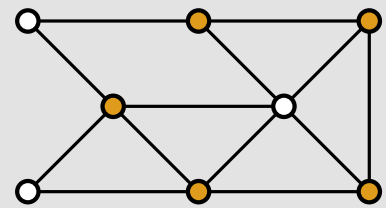
Problem: Vertex Cover
Gegeben sind ein Graph $G = (V, E)$ und ein Parameter k.
Gibt es ein Vertex Cover der Größe k?
(Knotenmenge $V' \subseteq V$ mit $e \cap V' \neq \emptyset$ für alle $e \in E$)



Gitter, Planarität und Vertex Cover

Theorem
Jeder planare Graph G mit Baumweite $\geq \frac{9}{2}t$ enthält Γ_t als Minor. Für jedes $\varepsilon > 0$ gibt es einen $O(n^2)$ Algorithmus, der entweder eine Baumzerlegung der Weite $(\frac{9}{2}+\varepsilon)t$ oder einen Γ_t Minor von G konstruiert.

Problem: Vertex Cover
Gegeben sind ein Graph $G = (V, E)$ und ein Parameter k.
Gibt es ein Vertex Cover der Größe k?
(Knotenmenge $V' \subseteq V$ mit $e \cap V' \neq \emptyset$ für alle $e \in E$)



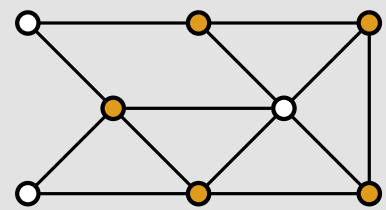
Beobachtungen

- monoton bezüglich Minorenbildung (min. VC wird nur kleiner)

Gitter, Planarität und Vertex Cover

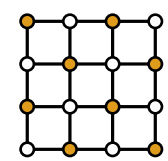
Theorem
Jeder planare Graph G mit Baumweite $\geq \frac{9}{2}t$ enthält Γ_t als Minor. Für jedes $\varepsilon > 0$ gibt es einen $O(n^2)$ Algorithmus, der entweder eine Baumzerlegung der Weite $(\frac{9}{2}+\varepsilon)t$ oder einen Γ_t Minor von G konstruiert.

Problem: Vertex Cover
Gegeben sind ein Graph $G = (V, E)$ und ein Parameter k.
Gibt es ein Vertex Cover der Größe k?
(Knotenmenge $V' \subseteq V$ mit $e \cap V' \neq \emptyset$ für alle $e \in E$)



Beobachtungen

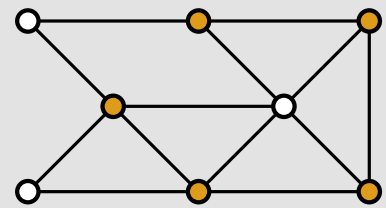
- monoton bezüglich Minorenbildung (min. VC wird nur kleiner)
- jedes VC in Γ_t hat Größe mindestens $\frac{1}{2}t^2$



Gitter, Planarität und Vertex Cover

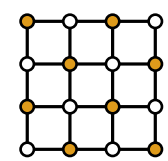
Theorem
Jeder planare Graph G mit Baumweite $\geq \frac{9}{2}t$ enthält Γ_t als Minor. Für jedes $\varepsilon > 0$ gibt es einen $O(n^2)$ Algorithmus, der entweder eine Baumzerlegung der Weite $(\frac{9}{2}+\varepsilon)t$ oder einen Γ_t Minor von G konstruiert.

Problem: Vertex Cover
Gegeben sind ein Graph $G = (V, E)$ und ein Parameter k.
Gibt es ein Vertex Cover der Größe k?
(Knotenmenge $V' \subseteq V$ mit $e \cap V' \neq \emptyset$ für alle $e \in E$)



Beobachtungen

- monoton bezüglich Minorenbildung (min. VC wird nur kleiner)
- jedes VC in Γ_t hat Größe mindestens $\frac{1}{2}t^2$



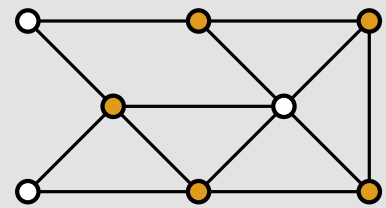
Algorithmus

- benutze Theorem mit $t = \sqrt{2k+2}$

Gitter, Planarität und Vertex Cover

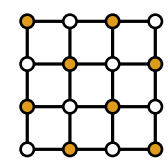
Theorem
Jeder planare Graph G mit Baumweite $\geq \frac{9}{2}t$ enthält Γ_t als Minor. Für jedes $\varepsilon > 0$ gibt es einen $O(n^2)$ Algorithmus, der entweder eine Baumzerlegung der Weite $(\frac{9}{2}+\varepsilon)t$ oder einen Γ_t Minor von G konstruiert.

Problem: Vertex Cover
Gegeben sind ein Graph $G = (V, E)$ und ein Parameter k.
Gibt es ein Vertex Cover der Größe k?
(Knotenmenge $V' \subseteq V$ mit $e \cap V' \neq \emptyset$ für alle $e \in E$)



Beobachtungen

- monoton bezüglich Minorenbildung (min. VC wird nur kleiner)
- jedes VC in Γ_t hat Größe mindestens $\frac{1}{2}t^2$



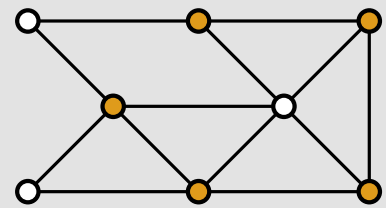
Algorithmus

- benutze Theorem mit $t = \sqrt{2k+2}$
- Fall 1: $\Gamma_{\sqrt{2k+2}}$ ist Minor von $G \Rightarrow$ es gibt kein VC der Größe k

Gitter, Planarität und Vertex Cover

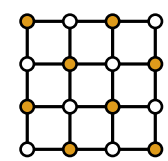
Theorem
Jeder planare Graph G mit Baumweite $\geq \frac{9}{2}t$ enthält Γ_t als Minor. Für jedes $\varepsilon > 0$ gibt es einen $O(n^2)$ Algorithmus, der entweder eine Baumzerlegung der Weite $(\frac{9}{2}+\varepsilon)t$ oder einen Γ_t Minor von G konstruiert.

Problem: Vertex Cover
Gegeben sind ein Graph $G = (V, E)$ und ein Parameter k.
Gibt es ein Vertex Cover der Größe k?
(Knotenmenge $V' \subseteq V$ mit $e \cap V' \neq \emptyset$ für alle $e \in E$)



Beobachtungen

- monoton bezüglich Minorenbildung (min. VC wird nur kleiner)
- jedes VC in Γ_t hat Größe mindestens $\frac{1}{2}t^2$



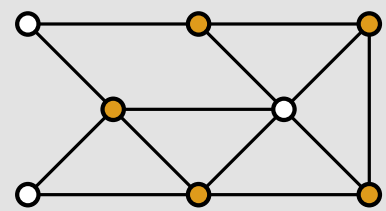
Algorithmus

- benutze Theorem mit $t = \sqrt{2k+2}$
- Fall 1: $\Gamma_{\sqrt{2k+2}}$ ist Minor von $G \Rightarrow$ es gibt kein VC der Größe k
- Fall 2: Baumzerl. der Weite $O(\sqrt{k}) \Rightarrow$ DP mit Laufzeit $2^{O(\sqrt{k})}n^{O(1)}$

Gitter, Planarität und Vertex Cover

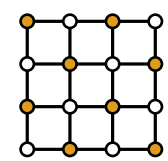
Theorem
Jeder planare Graph G mit Baumweite $\geq \frac{9}{2}t$ enthält Γ_t als Minor. Für jedes $\varepsilon > 0$ gibt es einen $O(n^2)$ Algorithmus, der entweder eine Baumzerlegung der Weite $(\frac{9}{2}+\varepsilon)t$ oder einen Γ_t Minor von G konstruiert.

Problem: Vertex Cover
Gegeben sind ein Graph $G = (V, E)$ und ein Parameter k.
Gibt es ein Vertex Cover der Größe k?
(Knotenmenge $V' \subseteq V$ mit $e \cap V' \neq \emptyset$ für alle $e \in E$)



Beobachtungen

- monoton bezüglich Minorenbildung (min. VC wird nur kleiner)
- jedes VC in Γ_t hat Größe mindestens $\frac{1}{2}t^2$



Algorithmus

- benutze Theorem mit $t = \sqrt{2k+2}$
- Fall 1: $\Gamma_{\sqrt{2k+2}}$ ist Minor von $G \Rightarrow$ es gibt kein VC der Größe k
- Fall 2: Baumzerl. der Weite $O(\sqrt{k}) \Rightarrow$ DP mit Laufzeit $2^{O(\sqrt{k})}n^{O(1)}$

$\Rightarrow$ Win-Win

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter
- funktioniert ähnlich mit z.B. Independent Set oder Dominating Set

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter
- funktioniert ähnlich mit z.B. Independent Set oder Dominating Set
- diese sind $W[1]$- bzw. $W[2]$-schwer auf allgemeinen Graphen

(d.h. es gibt vermutlich keinen FPT-Algorithmus)

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter
- funktioniert ähnlich mit z.B. Independent Set oder Dominating Set
- diese sind $W[1]$- bzw. $W[2]$-schwer auf allgemeinen Graphen

(d.h. es gibt vermutlich keinen FPT-Algorithmus)

Erweiterung auf andere Probleme

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter
- funktioniert ähnlich mit z.B. Independent Set oder Dominating Set
- diese sind $W[1]$- bzw. $W[2]$-schwer auf allgemeinen Graphen

(d.h. es gibt vermutlich keinen FPT-Algorithmus)

Erweiterung auf andere Probleme

- entscheidende Eigenschaften:

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter
- funktioniert ähnlich mit z.B. Independent Set oder Dominating Set
- diese sind $W[1]$- bzw. $W[2]$-schwer auf allgemeinen Graphen

(d.h. es gibt vermutlich keinen FPT-Algorithmus)

Erweiterung auf andere Probleme

- entscheidende Eigenschaften:
  - monoton bezüglich Minorenbildung

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter
- funktioniert ähnlich mit z.B. Independent Set oder Dominating Set
- diese sind $W[1]$- bzw. $W[2]$-schwer auf allgemeinen Graphen

(d.h. es gibt vermutlich keinen FPT-Algorithmus)

Erweiterung auf andere Probleme

- entscheidende Eigenschaften:
  - monoton bezüglich Minorenbildung
  - optimale Lösung in Γ_t ist $\Omega(t^2)$ groß

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter
- funktioniert ähnlich mit z.B. Independent Set oder Dominating Set
- diese sind $W[1]$- bzw. $W[2]$-schwer auf allgemeinen Graphen

(d.h. es gibt vermutlich keinen FPT-Algorithmus)

Erweiterung auf andere Probleme

- entscheidende Eigenschaften:
  - monoton bezüglich Minorenbildung
  - optimale Lösung in Γ_t ist $\Omega(t^2)$ groß

] bidimensionales Problem

Planarität und Vertex Cover

Theorem
Auf planaren Graphen kann Vertex Cover mit der Ergebnisgröße k als Parameter in $2^{O(\sqrt{k})} \cdot n^{O(1)}$ gelöst werden.

Bemerkenswert aus mehreren Gründen

- Baumzerlegung hilft hier, obwohl die Baumweite kein Parameter ist
- Laufzeit subexponentiell im Parameter
- funktioniert ähnlich mit z.B. Independent Set oder Dominating Set
- diese sind $W[1]$- bzw. $W[2]$-schwer auf allgemeinen Graphen

(d.h. es gibt vermutlich keinen FPT-Algorithmus)

Erweiterung auf andere Probleme

- entscheidende Eigenschaften:
  - monoton bezüglich Minorenbildung
  - optimale Lösung in Γ_t ist $\Omega(t^2)$ groß

  ] bidimensionales Problem
  - effizienter Algorithmus auf einer Baumzerlegung vorhanden

Zusammenfassung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

Zusammenfassung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

Einsichten

- rechtfertigt die Annahme, dass wir eine Baumzerlegung kennen
  (Laufzeiten bleiben zumindest FPT)

Zusammenfassung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

Einsichten

- rechtfertigt die Annahme, dass wir eine Baumzerlegung kennen
  (Laufzeiten bleiben zumindest FPT)
- Baumzerlegungen und Knotenseparatoren sind sehr verwandt

Zusammenfassung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

Einsichten

- rechtfertigt die Annahme, dass wir eine Baumzerlegung kennen
  (Laufzeiten bleiben zumindest FPT)
- Baumzerlegungen und Knotenseparatoren sind sehr verwandt

Teil 2: Win-Win auf planaren Graphen

- die meisten NP-schweren Probleme bleiben auf planaren Graphen NP-schwer
  (Max Cut ist mehr oder weniger die einzige bekannte Ausnahme)
- aber: viele $W[1]$-schwere Probleme sind auf planaren Graphen FPT

Zusammenfassung

Theorem
Es gibt einen Algorithmus, der in $O(8^k k^2 \cdot n^2)$ Zeit eine Baumzerlegung der Weite $4k + 4$ berechnet oder entscheidet, dass $\mathrm{tw}(G) > k$.

Einsichten

- rechtfertigt die Annahme, dass wir eine Baumzerlegung kennen
  (Laufzeiten bleiben zumindest FPT)
- Baumzerlegungen und Knotenseparatoren sind sehr verwandt

Teil 2: Win-Win auf planaren Graphen

- die meisten NP-schweren Probleme bleiben auf planaren Graphen NP-schwer
  (Max Cut ist mehr oder weniger die einzige bekannte Ausnahme)
- aber: viele $W[1]$-schwere Probleme sind auf planaren Graphen FPT
- Grund: große Baumweite $\Rightarrow$ großes Gitter (als Minor)