

A Linear-Time Algorithm For Finding Tree-Decompositions Of Small Treewidth

Maximilian F. Göckel – *uzkns@student.kit.edu*

Institut für Theoretische Informatik - Proseminar NP-schwere Probleme

Viele (NP-schwere) Probleme laufen auf Bäumen besser als auf Graphen.


Problem: Nicht jeder Graph ist ein Baum.

Jeder Graph kann zu einem Baum umgewandelt ("zerteilt") werden.

- Der Graph ist bereits ein Baum \Rightarrow fertig
- Der Graph ist kein Baum \Rightarrow Algorithmus anwenden

Der Algorithmus teilt die Knoten in Mengen ("Nodes") auf, welche die Knoten des neuen Baumes darstellen, und soll dabei drei Vorgaben einhalten:

Sei X_i der Node mit dem Index i .

1. $\bigcup_{i \in \mathbb{N}} X_i = V$ 
2. $\forall (v, w) \in E : \exists X_i \in V_{Treedecomposition} : v, w \in X_i$
3. $w \in X_i, X_j \Rightarrow$ Jedes X_k im Pfad zwischen X_i, X_j enthält w

Tree-decomposition: Beispiel

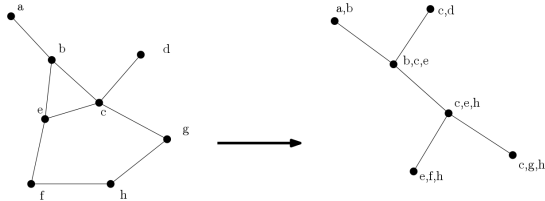


Figure: 1



Tree-decomposition: Beispiel

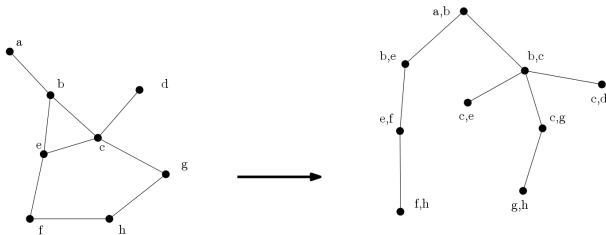


Figure: 2



Jede Baumzerteilung hat eine "Baumweite" (treewidth).

- Baumweite einer Zerteilung: $\max(|X_i|_{i \in V})$ "Zerteilungsweite"
- Baumweite eines Graphen: Minimale Zerteilungsweite aller Zerteilungen

"Normaler" Baum hat Baumweite 0.

2 Schritte:

1. Für gegebenen Graph $G = (V, E)$ und geg. $k \in \mathbb{N}$ eine Zerteilung mit max. Baumweite k finden
2. Graph-Zugehörigkeit zur Klasse "Graphen mit Baumweite k " prüfen




- Zeiteffizient [$O(n \cdot \log(n))$] von Reed]
- Frage "Hat G maximal Baumweite k ?" ist NP-vollständig
- Nutzt "balanced separators"



Algorithmus: Schritt 2

- Relativ effizient (Linearzeit)
- Vergleicht Graph mit "obstruction set"



- Ersten Schritt verbessern
 - Lineare Laufzeit
 - Keine "balanced separators" nutzen
- $\forall k \in \mathbb{N} : \exists$ Linearzeitalgorithmus welcher für $G = (V, E)$ prüft ob die Baumweite max. k ist und eine Zerteilung ausgibt 
- Zweiter Schritt ist "gut genug"
 - Keine Obstruction Sets nutzen 