Karlsruher Institut für Technologie

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

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

Institut für Theoretische Informatik - Proseminar Algorithmen für NP-schwere Probleme

Tree-decomposition

Definition

Eine Baumzerteilung eines Graphen $G = (V, E)$ ist ein Tupel (X, T) wo $T = (I, F)$ ein Baum ist und $X = \{X_i | i \in I\}$ eine Familie von Teilmengen von V wobei jedes X_i einen Knoten in T darstellt.

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

Tree-decomposition

Veranschaulichung

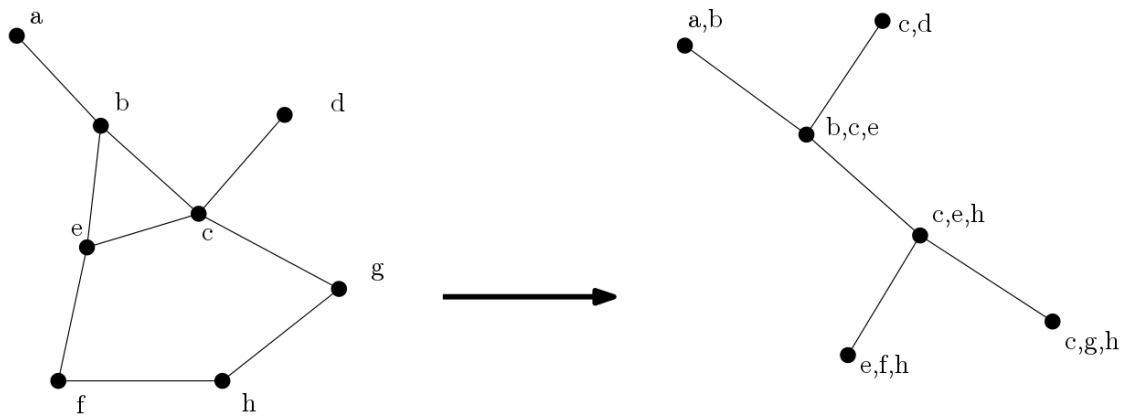

Figure: 1

Treewidth

Definition

Jede Baumzerteilung hat eine "Baumweite" (treewidth).

- Baumweite einer Zerteilung: $\max(|X_i|_{i \in I} - 1)$ ("Zerteilungsweite")
- Baumweite eines Graphen: Minimale Zerteilungsweite aller Zerteilungen

k-Trees

Definition

Die folgenden Aussagen zu k-Bäumen sind äquivalent:

1. $G = (V, E)$ ist ein k-Baum
2. G ist verbunden und hat eine k-Clique, aber keine $(k+2)$-Clique und
   - Jeder minimale Seperator von G ist eine k-Clique oder
   - $\forall$ nicht-adjazenten Knotenpaare $x, y \in V \; \exists \; k$ Wege $x \to y$
3. G ist verbunden, $|E| = k|V| - \frac{1}{2}k(k+1)$ und jeder minimale Seperator von G ist eine k-Clique
4. G hat Knoten v mit 3 Eigenschaften:
   - $\deg(v) = k$ und
   - Nachbarknoten von v formen eine k-Clique und
   - $G\ v$ ist k-Baum

Jeder komplette Graph mit k Knoten ist damit auch ein k-Baum.

k-Trees

Erstellung

Andersherum: Ein k-Baum-Graph G mit $n \geq k$ Knoten kann aus einem k-Baum-Graph H mit $n-1$ Knoten wie folgt erstellt werden:

- Zu H einen Knoten u hinzufügen ($|V| = (n-1) \rightarrow |V| = n$)
- Knoten u mit Knoten $v_1, ..., v_k$ verbinden

Damit wird Aussage 4 erfüllt.

Partial k-Trees

Definition

Graph $G = (V, E)$ ist partieller k-Baum $\Leftrightarrow$

- G ist Teilgraph eines k-Baumes
- G hat Baumweite max. k

Baumzerteilung

Anwendungen

- Maximum-Weight Independent Set in Linearzeit lösbar
- Hohe Baumweite $\Leftrightarrow$ Hohe Komplexität in der Systemanalyse
-
-

Knotentypen

Simplizial, freundlich, low- und highdegree

Ein Knoten v ist ...

- ... "von niedrigem Grad" wenn $\deg(v) \leq d$
  - $d = 2k^3 \cdot (k+1) \cdot (4k^2 + 12k + 16)$
  - Analog: Hoher Grad $\Leftrightarrow \deg(v) > d$
  - Auch "low-deg.-" und "high-deg.-Knoten" genannt
- ... "Freundlich" wenn er low-deg. und adjazent zu einem weiteren low-deg.-Knoten ist
- ... "Simplizial" wenn alle Nachbarn in einer Clique sind
- ... "I-Simplizial" wenn simp. in G' und $\deg(v) \leq k$ in G

Verbesserter Graph G'

Erstellung und Eigenschaften

$G' = (V, E')$ ist $G = (V, E)$ mit Kanten $(v, w) \in E' \forall v, w \in V$ sodass v, w min. $k + 1$ gem. Nachbarn mit Grad max. k haben.

LEMMA 4.1.: $\text{tw}(G) \leq k \Leftrightarrow \text{tw}(G') \leq k$.
Außerdem ist jede k-Zerteilung von G auch eine k-Zerteilung von G' und umgekehrt.

Maximum Matching $M \subseteq E$

$M \subseteq E$ ist Maxmimum Matching in $G = (V, E) \Leftrightarrow$ Keine 2 Kanten aus M haben gemeinsamen Endknoten und $|M|$ maximal

Ein Maximal Matching kann in $O(|V|)$ gefunden werden, wenn die Baumweite durch ein k gebunden ist.

Anzahl an Friendly-Knoten in G

LEMMA 4.2.: G hat Baumweite max. $k \Rightarrow$ 1 von 2 muss min. gelten:

- G hat min. $\frac{|V|}{4k^2+12k+16} =: \lambda$ Friendly-Knoten
- G' hat min. $\frac{1}{8k^2+24k+32} \cdot |V|$ I-simp.-Knoten

Algorithmus hat eine Fallunterscheidung ab λ Friendly-Knoten.

Algorithmus

Fall: Min. λ Friendly-Knoten

1. Maximum-Matching $M \subseteq E$ finden
2. Jede Kante in M kontrahieren um Graphen $\widetilde{G}$ zu erhalten
3. Wenn Baumweite von $\widetilde{G} > k \Rightarrow$ STOP
4. Kompletten Algorithmus auf $\widetilde{G}$ ausführen um Baumzerteilung (Y, T) von $\widetilde{G}$ auszugeben
5. Mit LEMMA 3.3. Zerteilung (X, T) aus (Y, T) erstellen
6. Mit THEOREM 2.4. prüfen ob Weite von $G > k$ ist $\Rightarrow$ STOP
7. Zerteilung von G errechnen und ausgeben

Hauptidee

Viele NP-schwere Probleme sind in Linearzeit lösbar, wenn die Baumweite des Graphen konstant ist. $\rightarrow$ Kann man die Baumweite (für bel., festes $k \in \mathbb{N}$) in Linearzeit errechnen?

~~Z~~Z: $\forall k \in \mathbb{N} : \exists$ Linearzeitalgorithmus welcher für $G = (V, E)$ prüft ob die Baumweite max. k ist und eine Zerteilung ausgibt.

Für $k = 1, 2, 3, 4$ existieren schon Linearzeitalgorithmen.

Algorithmus

2 Schritte:

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

Problem *"Für einen Graph $G = (V, E)$ und ein $k \in \mathbb{N}$: Ist die Baumweite von G maximal k?"* ist NP-Vollständig für bel. k