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

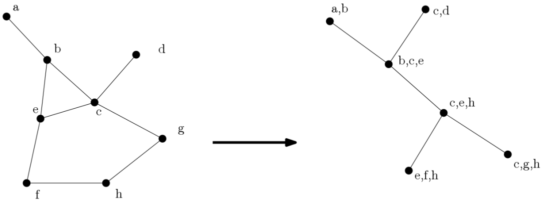

Figure: 1

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

- Eine Baumzerteilung der Weite k heißt auch k-Baumzerteilung oder k-Zerteilung

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 $x \rightarrow y$
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
   - $G \setminus v$ 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 $H = (V, E)$ mit $n - 1$ Knoten wie folgt erstellt werden:

- Zu H einen Knoten u hinzufügen ($|V| = (n - 1) \rightarrow |V| = n$)
- Knoten u mit bel. 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 oder
- 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.: tw(G)$\leq k \Leftrightarrow$ 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.

Algorithmus
Allgemeines

Eingabe: Graph $G = (V, E)$ mit $|V| = n$ und Konstante $k \in \mathbb{N}$.
Der Graph wird als Adjazenzliste übergeben.
Ausgabe in $O(n)$:

- "Baumweite von G ist größer als k"
- "Baumweite von G ist maximal k"
  - Baumzerteilung von G mit Baumweite k

Für "sehr kleine" Graphen werden andere bekannte Algorithmen genutzt, ansonsten wird wie folgt vorgegangen:

Anzahl an Friendly-Knoten in G

LEMMA 4.2.: G hat Baumweite max. $k \Leftarrow$ 1 von 2 gilt mindestens:

- 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.
Die Anzahl Friendly-Knoten in G wird mit n_f notiert.

Algorithmus

Fall: Min. λ Friendly-Knoten

1. Maximum-Matching $M \subseteq E$ finden
2. Jede Kante in M kontrahieren um Graphen $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ zu erhalten
3. Kompletten Algorithmus auf $\widetilde{G}$ ausführen um Baumzerteilung (Y, T) von $\widetilde{G}$ auszugeben $\rightarrow$ Wenn Baumweite von $\widetilde{G} > k \Rightarrow$ STOP (LEMMA 3.4.)
4. Mit LEMMA 3.3. Zerteilung (X, T) aus (Y, T) erstellen
5. Mit THEOREM 2.4. prüfen ob Weite von $G > k$ ist $\Rightarrow$ STOP
   $\rightarrow$ Zerteilung von G errechnen und ausgeben

Algorithmus

Fall: Max. $\lambda - 1$ Friendly-Knoten

Karlsruher Institut für Technologie

1. Improved-Graph G' berechnen
   $\rightarrow \exists$ I.simp.-Knoten v mit deg(v) = $k + 1 \Rightarrow$ STOP
2. Alle I.simp.-Knoten in Menge SL und von G entfernen $\Rightarrow \widehat{G}$ entsteht
   $\rightarrow |SL| < c_2 \cdot |V| \Rightarrow$ STOP (THEOREM 4.2.)
3. Algorithmus rekursiv auf $\widehat{G}$ ausführen $\Rightarrow$ Ausgabe von Zerteilung (Y, T) von $\widehat{G}$
   $\rightarrow$ tw($\widehat{G}$) $> k \Rightarrow$ STOP ($\widehat{G}$ Teilgraph von $G \Rightarrow$ tw(G) $> k$)
4. $\forall v \in SL$: Finde ein $Y_{i_v} \in Y$ in dem alle Nachbarn von v sind $(N_G(v) \subseteq Y_{i_v})$
5. Füge $Y_{j_v} = \{v\} \cup N_G(v)$ zu Y hinzu und mache es adjazent zu Y_{i_v}
   $\rightarrow$ Baumzerteilung von G mit Baumweite max. k

Algorithmus

Fall: Max. $\lambda - 1$ Friendly-Knoten: Letzter Schritt

1. $\forall v \in SL$: Finde ein $Y_{i_v} \in Y$ in dem alle Nachbarn von v sind ($N_G(v) \subseteq Y_{i_v}$)
2. Füge $Y_{j_v} = \{\{v\} \cup N_G(v)\}$ zu Y hinzu und mache es adjazent zu Y_{i_v}
   $\Rightarrow$ Baumzerteilung von G mit Baumweite max. k

Y_{i_v} existiert für jedes v, da I.simp.-Knoten in G nicht adjazent sind und $N_G(v)$ eine Clique formt.
LEMMA 2.1.i): "Ist (X, T) Zerteilung von G und formt $W \subseteq V$ eine Clique in G, $\exists i \in IW \subseteq X_i$"

Laufzeitanalyse

Fall 1

1. Maximum Matching $M \subseteq E$ finden

Ein Maximum Matching kann greedy in $O(|V| + |E|)$ gefunden werden.

LEMMA 2.3.: "tw(G) $\leq k \Rightarrow |E| \leq k|V| - \frac{1}{2}k(k+1)$"
$\Rightarrow |E| \in O(n)$
$\Rightarrow$ Max. Matching kann in $O(n)$ gefunden werden

Laufzeitanalyse

Fall 1

2. Jede Kante in M kontrahieren um Graphen $\widetilde{G}$ zu erhalten

Eine Kante kann in $O(1)$ kontrahiert werden, liegt der Graph als Adjazenzliste vor.
LEMMA 2.3.: "tw(G) $\leq k \Rightarrow |E| \leq k|V| - \frac{1}{2}k(k+1)$"
$\Rightarrow |M| \in O(|E|)$
$\Rightarrow$ Alle Kanten können in $O(n)$ kontrahiert werden.

Laufzeitanalyse
Fall 1

3. Kompletten Algorithmus auf $\widetilde{G}$ ausführen um Baumzerteilung (Y, T) von $\widetilde{G}$ auszugeben

Ein Maximum Matching hat min. $\frac{n_f}{2\cdot(2k^3(k+1)(4k^2+12k+16))}$ Kanten.

Für jeden Friendly-Knoten gilt:

- Er ist Endpunkt von einem $m \in M$ oder
- Er ist adjazent zu einem Friendly-Knoten, der Endpunkt ist

$\Rightarrow \forall m \in M$ werden max. $2d$ Friendly-Knoten assoziiert, die Endpunkt sind oder adjazent zu einem Friendly-Endpunkt sind. $\Rightarrow$: Ist ein Friendly-Knoten nicht assoziiert so ist M nicht maximal $\Rightarrow |M| \geq \frac{n_f}{2d}$

Laufzeitanalyse
Fall 1

3. Kompletten Algorithmus auf $\widetilde{G}$ ausführen um Baumzerteilung (Y, T) von $\widetilde{G}$ auszugeben

Ein Maximum Matching hat min. $\frac{n_f}{2\cdot(2k^3(k+1)(4k^2+12k+16))}$ Kanten.
$\Rightarrow |\widetilde{V}| = (1 - \frac{1}{2d(4k^2+12k+16)}) \cdot |V|$

Laufzeitanalyse

Fall 1

5. Mit LEMMA 3.3. Zerteilung (X, T) aus (Y, T) erstellen

$$f_M : V \mapsto \widetilde{V} = \begin{cases} f_M(v) = v & \text{Wenn } v \text{ nicht Endpunkt in } M \text{ ist} \\ f_M(v) = f_M(w) & \text{Der Knoten der bei der Kontraktion } (v, w) \in M \text{ bleibt} \end{cases}$$

(Y, T) Zerteilung von $\widetilde{G}$, so ist (X, T) mit $X_i = \{v \in V | f_M(v) \in Y_i\}$ Zerteilung von G mit Weite max. $2k + 1$

Laufzeitanalyse

Fall 1

6. prüfen ob Weite von $G > k$ ist $\Rightarrow$ STOP

6.1. Zerteilung von G errechnen und ausgeben

THEOREM 2.4.: "$\forall k, l \in \mathbb{N} \exists$ Linearzeitalgorithmus, welcher aus einem Graph $G = (V, E)$ und einer l-Zerteilung prüft ob die Baumweite von G max. k ist und eine k-Zerteilung errechnet"

Laufzeit: $O(l^{l-2} \cdot ((2l+3)^{2l+3} \cdot (\frac{8}{3}2^{2k+2})^{2l+3})^{2l-1})) \in O(k^3)$ bei $l \in O(k)$

Laufzeitanalyse
Fall 2

1.