Einführungsvortrag zum

Hauptseminar im WS 1997/98 am

Lehrstuhl für Effiziente Algorithmen

Schnelle Algorithmen für baumartige Graphen

8. Januar 1998

Thomas Erlebach

1. Polynomielle Algorithmen und NP
2. Baumzerlegung und Baumweite
3. k-Bäume und partielle k-Bäume
4. Äquivalenz der Konzepte
5. Fakten zu baumartigen Graphen
6. Ausblick auf weitere Seminar-Vorträge

Polynomielle Algorithmen und NP

- Kriterium für **Effizienz**: Laufzeit eines Algorithmus in Abhängigkeit von der Größe der Eingabe im Worst-Case
- Falls Eingabe ein **Graph** mit n Knoten und m Kanten ist, ist die Eingabegröße $O(n+m)$.
- Ist die Worst-Case-Laufzeit eines Algorithmus durch ein Polynom in n und m beschränkt, so hat der Algorithmus **polynomielle Laufzeit**.

Beispiele:

- Zusammenhangskomponenten:
  Laufzeit $O(n+m)$
- Minimale Spannbäume nach Kruskal:
  Laufzeit $O(m \log m)$
- Matching in Graphen:
  Laufzeit $O(\sqrt{n} \cdot m)$

NP (= **N**on-deterministic **P**olynomial-time) ist die Menge aller Sprachen (Entscheidungsprobleme), die von einer nichtdeterministischen Turing-Maschine in polynomieller Zeit akzeptiert werden.

Beispiel für ein Problem in NP: “Gibt es in G eine Clique der Größe k?”

Ein Problem p heißt **NP-hart**, wenn sich jedes Problem in NP in polynomieller Zeit auf p reduzieren läßt.

Beispiel: “Finde die größte Clique in G!”

Ein Entscheidungsproblem p heißt **NP-vollständig**, wenn es sowohl in NP als auch NP-hart ist. Beispiel: “Entscheide für eine gegebene aussagenlogische Formel, ob sie erfüllbar ist!”

NP, Bäume, baumartige Graphen

- Viele Graphenprobleme sind **NP-hart**, wenn als Eingabe ein beliebiger Graph zugelassen wird:
  CLIQUE, INDEPENDENT SET, COLORABILITY, DOMINATING SET, VERTEX COVER, . . .

- In **Bäumen** (zusammenhängende Graphen ohne Kreise) sind viele dieser Probleme dagegen effizient, d.h. in polynomieller oder sogar in linearer Zeit, lösbar.

- **Idee:** Konzept der “Baumartigkeit” von Graphen einführen, um:
  - Kluft zwischen Bäumen und beliebigen Graphen zu überbrücken
  - effiziente Algorithmen für Bäume auf baumartige Graphen zu verallgemeinern

Baumzerlegung

Definition: Sei $G = (V, E)$ ein Graph. Eine **Baumzerlegung** von G ist ein Baum $T = (I, F)$, wobei jedem Knoten $i \in I$ des Baums eine Teilmenge X_i von V zugeordnet ist und die folgenden Bedingungen erfüllt sind:

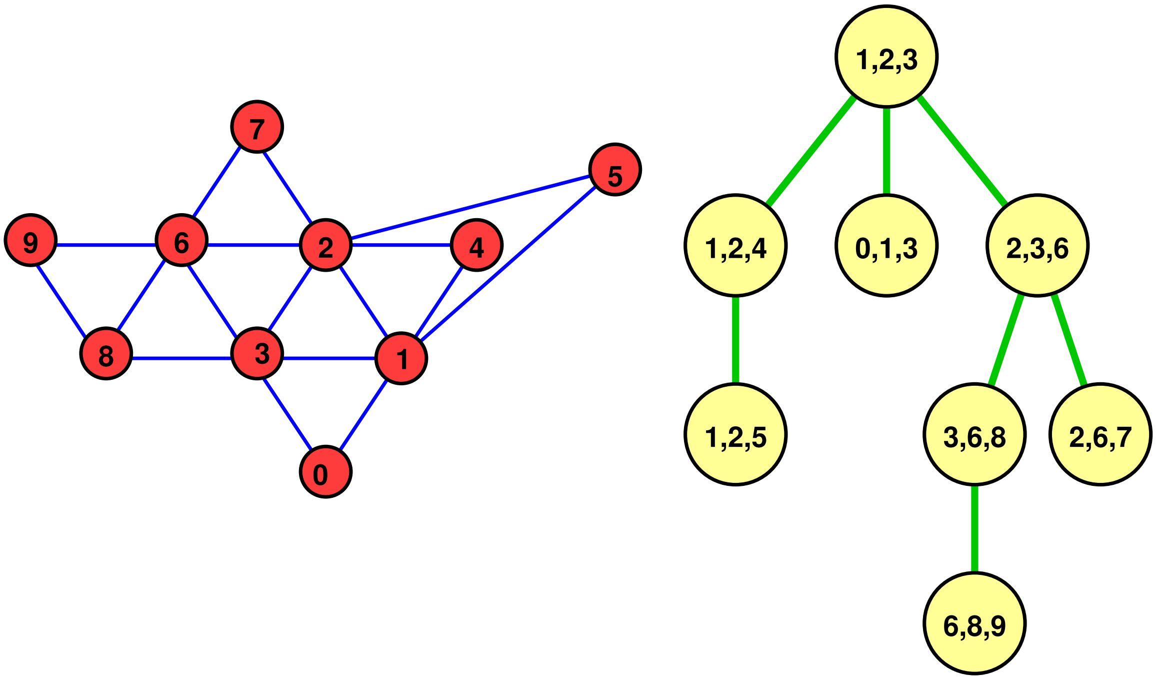
(1) V wird von den X_i überdeckt: $V = \bigcup_{i \in I} X_i$

(2) $\forall \; \{u, v\} \in E \quad \exists \; i \in I \;:\; u, v \in X_i$

(3) für jeden Knoten $v \in V$ induziert die Menge $\{i \in I \mid v \in X_i\}$ einen Teilbaum von T

Die **Weite** der Baumzerlegung ist $\max_{i \in I} |X_i| - 1$. Die **Baumweite** eines Graphen G ist das minimale $k \geq 0$, so daß G eine Baumzerlegung mit Weite k besitzt.

Beispiel: Graph mit Baumzerlegung der Weite 2



Eine Klasse von Graphen hat **beschränkte Baumweite**, falls es eine Konstante k gibt, so daß jeder Graph der Klasse Baumweite höchstens k hat.

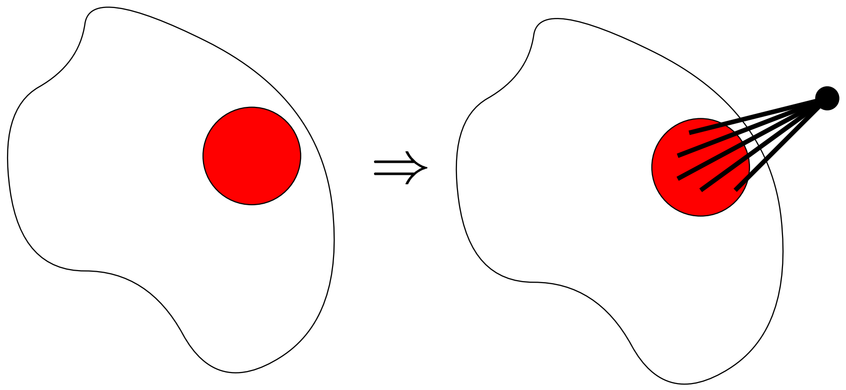
Graphklasse	Baumweite
Bäume und Wälder	1
serienparallele Graphen	2
Halin-Graphen	3

k-Bäume

Definition: Sei $k \in \mathbb{N}$. Die k-Bäume sind alle Graphen, die sich wie folgt konstruieren lassen:

(i) der vollständige Graph auf k Knoten ist ein k-Baum

(ii) sei $G = (V, E)$ ein k-Baum, in dem die Knoten $v_1, \ldots, v_k$ eine k-Clique bilden; dann ist auch G' ein k-Baum, wobei G' aus G entsteht, indem ein neuer Knoten eingefügt und genau mit $v_1, \ldots, v_k$ verbunden wird.



Bemerkungen zu k-Bäumen

Sei G ein k-Baum, der aus der k-Clique $v_1, \ldots, v_k$ (der **Basis** von G) durch Hinzufügen der Knoten $v_{k+1}, v_{k+2}, \ldots, v_n$ entstanden ist.

- Ein Knoten, dessen Nachbarschaft eine Clique bildet, heißt auch **simplexartig** (engl. *simplicial*).
- Eliminiert man die Knoten in der Reihenfolge v_n, $v_{n-1}, \ldots, v_1$ aus G, so ist die Nachbarschaft jedes Knotens v_i zum Zeitpunkt der Elimination eine Clique. Also ist die Reihenfolge v_n,. . . ,v_1 eine **perfekte Eliminationsordnung**.
- In dieser perfekten Eliminationsordnung bildet die Nachbarschaft von Knoten v_i für $n \geq i \geq k+1$ zum Zeitpunkt der Elimination von v_i eine Clique mit genau k Knoten.

Partielle k-Bäume

Alle Graphen G, die durch Weglassen von Kanten aus einem k-Baum entstehen können, sind **partielle k-Bäume**.

Bemerkung: Definition mit Weglassen von Knoten macht nur für Graphen mit $< k$ Knoten einen Unterschied.

Äquivalent für $|V| \geq k$: $G = (V, E)$ ist ***k*-zerlegbar**, d.h.

- G besteht aus höchstens $k + 1$ Knoten, **oder**
- Es gibt einen Separator S der Größe $\leq k$, dessen Entfernung V in $S_1, \ldots, S_t$ zerlegt, so daß alle $G[S_i \cup S]$ mit Clique S auch k-zerlegbar sind.

Lemma. Sei $G = (V, E)$ ein Graph mit $|V| \geq k$. Dann gilt:

$$\boxed{G \text{ ist partieller } k\text{-Baum} \Leftrightarrow G \text{ hat Baumweite} \leq k.}$$

Beweis:

(a) z.z.: G partieller k-Baum $\Rightarrow$ G hat Baumzerlegung mit Weite $\leq k$

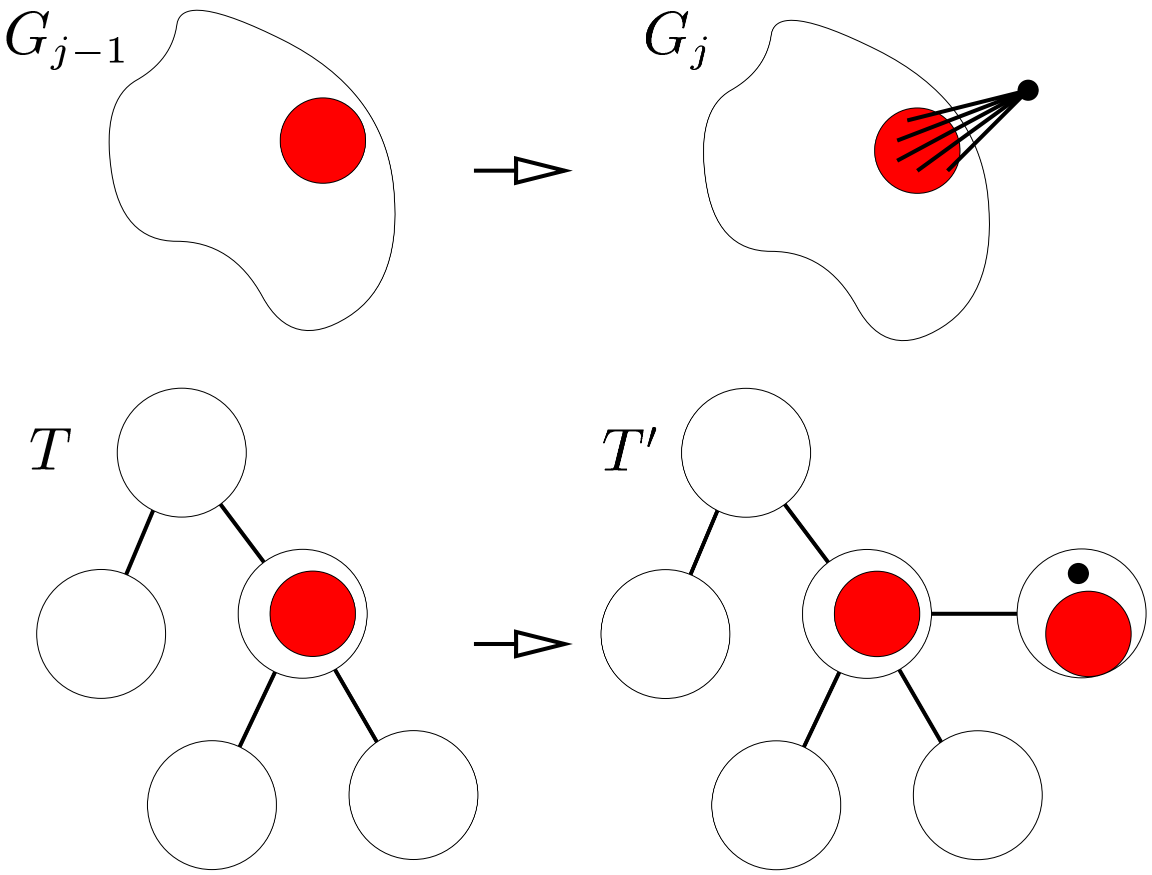
G sei ein k-Baum (durch Einfügen von Kanten wird die Baumweite höchstens größer), der aus der Basis v_1,. . . ,v_k durch Hinzufügen von v_{k+1},. . . ,v_n entstanden ist.

Wir konstruieren für $j = k, k + 1, \ldots, n$ eine Baumzerlegung der Weite k für den von $v_1, \ldots, v_j$ induzierten Teilgraphen G_j von G, in der jede k-Clique von G_j in einem der X_i enthalten ist.

Für $j = k$ nehmen wir einfach eine Baumzerlegung mit nur einem Knoten i und $X_i = \{v_1, \ldots, v_k\}$.

Für $j > k$ sei $T = (I, F)$ eine Baumzerlegung für G_{j-1}. Der Knoten v_j wurde beim Einfügen in G_{j-1} zu einer k-Clique C von G_{j-1} adjazent gemacht. Sei i ein Knoten von T, in dem diese k-Clique C enthalten ist.

Dann erhalten wir eine Baumzerlegung T' für G_j, indem wir einen neuen Knoten i' einfügen, der nur zu i benachbart ist und dem die Menge $X_{i'} = C \cup \{v_j\}$ entspricht.



(b) z.z.: G hat Baumzerlegung mit Weite $\leq k \Rightarrow G$ partieller k-Baum

G habe eine Baumzerlegung T der Weite k mit einem Knoten i_0, so daß $|X_{i_0}| = k + 1$.

Wir konstruieren ausgehend von $I' = \{i_0\}$ k-Bäume $G'_{I'}$, die alle G-Knoten enthalten, die zu $i \in I'$ gehören, und folgende Eigenschaft erfüllen:

Jede Teilmenge von $\leq k$ Knoten der Knotenmenge, die einem $i \in I'$ zugeordnet ist, ist Teil einer k-Clique in $G'_{I'}$.

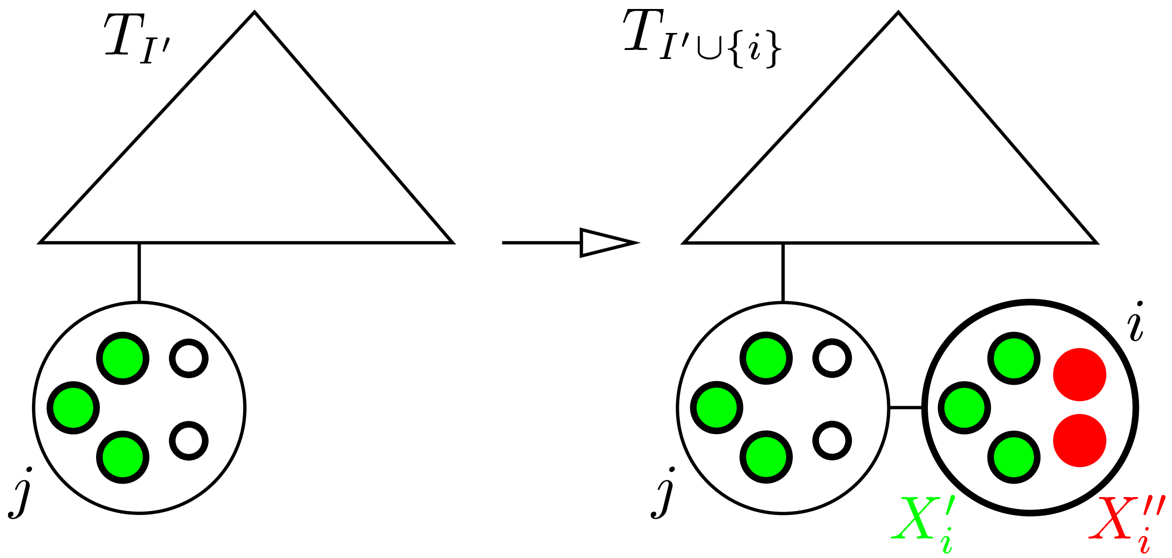
Für $I' = \{i_0\}$ nehmen wir als $G'_{I'}$ einfach eine $(k+1)$-Clique mit Knotenmenge X_{i_0}.

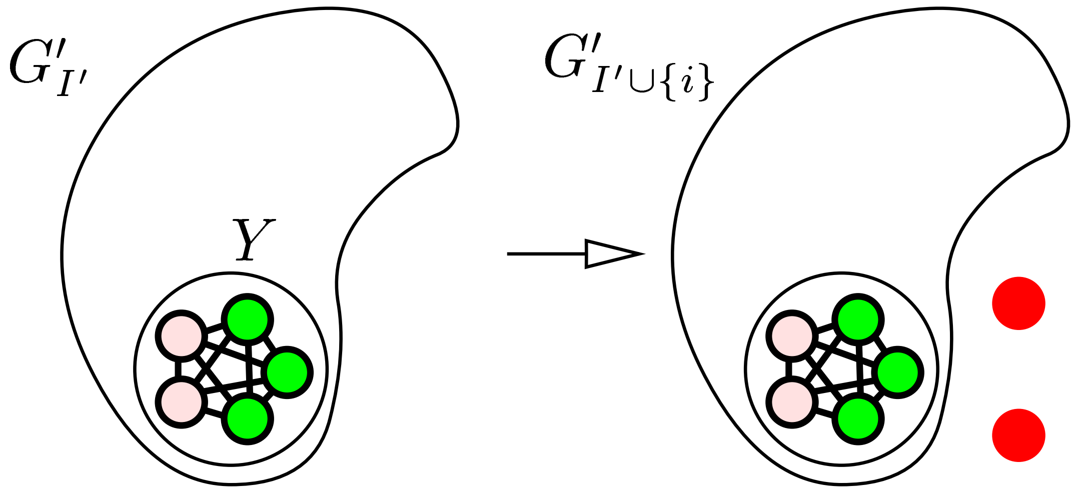
Erweiterung von I' um ein zu $j \in I'$ adjazentes i:

Teile X_i in $X_i' = X_i \cap X_j$ und $X_i'' = X_i \setminus X_i'$.

- falls $X_i'' = \emptyset$: o.k.
- sonst: $\exists$ k-Clique $Y \supseteq X_i'$ in $G'_{I'}$

  füge Knoten aus X_i'' mittels Y in $G'_{I'}$ ein





Fakten zu baumartigen Graphen

- Berechnung der Baumweite ist NP-hart.
- Für festes k ist Test auf Baumweite $\leq k$ und Finden einer entsprechenden Baumzerlegung mit Laufzeit $O(n)$ möglich.
- Für $k = 1, 2, 3, 4$ praktikable Verfahren durch Graph-Reduktionen.
- Viele Probleme in Graphen mit beschränkter Baumweite sind durch Bottom-Up-Berechnungen lösbar.
- Alle Probleme, die sich mit CMSO-Formeln (Counting Monadic Second-Order Logic) ausdrücken lassen, sind in linearer Zeit lösbar.

Überblick über weitere Vorträge

- **Lösung NP-harter Probleme in linearer Zeit**
  INDEPENDENT SET, CHROMATIC NUMBER,...
  (Bottom-Up-Berechnung, Tabellen mit Einträgen für äquivalente Lösungen)
- **Erkennung partieller k-Bäume**
  Laufzeit $O(n^{k+2})$ für Finden der Einbettung in einen k-Baum
- **Anwendung auf ein Scheduling-Problem**
  χ_t-COLORING: nur bis zu t Knoten pro Farbklasse zugelassen; ausdrückbar als CMSO-Formel
- **Dominierungsprobleme**
  praktikable Algorithmen mit Laufzeit $O(2^{4k}n)$ für eine Familie von Problemen, die INDEPENDENT SET und DOMINATING SET umschließt