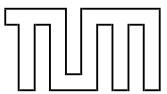
Fortgeschrittene Netzwerk- und Graph-Algorithmen

Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Wintersemester 2007/08



Übersicht

Maximum-Clique: besserer exponentieller Algorithmen

Satz

Eine Clique maximaler Kardinalität kann in Zeit $\mathcal{O}^*(1.3803^n)$ berechnet werden.

($\mathcal{O}^*$ ignoriert polynomielle Faktoren)

Maximum-Clique: besserer exponentieller Algorithmus

Beweis.

- Sei v ein Knoten mit minimalem Grad $\deg_G(v) = \delta(G)$
- Falls $\delta(G) \geq n - 3$, dann
  - fehlen in G nur einfache Pfade und Kreise im Vergleich zum vollständigen Graphen K_n

  $\Rightarrow$ Maximum Clique kann in $\mathcal{O}(m + n)$ berechnet werden:
  - In einem Pfad P kann man die Größe einer maximalen unabhängigen Menge (independent set) berechnen als $\lceil |V(P)|/2 \rceil$.
  - In einem Kreis C ist die Größe $\lfloor |V(C)|/2 \rfloor$.
  - Da die Pfade und Kreise paarweise disjunkt sind, kann man die einzelnen Werte einfach addieren.

Maximum-Clique: besserer exponentieller Algorithmus

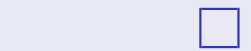
Beweis.

- Ansonsten sei v ein Knoten mit Grad $\deg_G(v) \leq n-4$
- Jede Maximum-Clique ist entweder
  - $\{v\}$ vereinigt mit einer Maximum-Clique von $G[N(v)]$ oder
  - eine Maximum-Clique von $G[V \setminus \{v\}]$.

$\Rightarrow$ Rekursive Berechnung mit worst-case-Zeit

$$T(n) \leq T(n-4) + T(n-1) + c \cdot (m+n)$$

$\Rightarrow$ $T(n) \in \mathcal{O}^*(\beta^n)$ mit $\beta \approx 1.3803$, wobei β die größte reelle Nullstelle des charakteristischen Polynoms $\beta^4 - \beta^3 - 1$ ist



Approximation von Maximum-Cliquen

Approximation der größten Clique mit Faktor $n/2$ ist einfach (wähle die Endknoten einer Kante, falls es eine gibt).

Satz

Es gibt einen Algorithmus, der bei Eingabe eines Graphen G mit n Knoten eine Clique ausgibt, deren Größe maximal um einen Faktor $\mathcal{O}\left(\frac{n}{(\log n)^2}\right)$ von der Maximum-Cliquengröße $\omega(G)$ abweicht.

Satz

Falls nicht $\mathcal{NP} = \mathcal{ZPP}$ gilt, dann existiert kein Polynomialzeitalgorithmus, der bei Eingabe eines Graphen G mit n Knoten eine Clique ausgibt, deren Größe maximal um einen Faktor $n^{1-\epsilon}$ von der Maximum-Cliquengröße $\omega(G)$ abweicht (für jedes $\epsilon > 0$).

Suche nach Cliquen fester Größe

- In einigen Fällen reicht es, nach Cliquen fester Größe zu suchen

$\Rightarrow$ Cliquengröße wird nicht als Teil der Eingabe betrachtet

- Vollständige Suche: $\mathcal{O}(k^2 \cdot n^k) = \mathcal{O}(n^k)$, wenn Cliquengröße k fest ist

Bessere Komplexität für Dreiecke

- $A(G)$: Adjazenzmatrix von Graph G
- $B(G) = A(G)^2 = A(G) \cdot A(G)$: Einträge b_{ij} sind die Wege der Länge 2 zwischen den Knoten v_i und v_j

$\Rightarrow$ läßt sich durch schnellere Matrixmultiplikation ausrechnen, z.B. in $\mathcal{O}(n^{2.376})$ (Coppersmith / Winograd, 1990)

- Existiert ein Eintrag $b_{ij} \geq 1$ mit $i \neq j$, dann gibt es einen Knoten $u \in V$, der zu v_i und zu v_j adjazent ist.
- Wenn nun auch eine Kante $\{v_i, v_j\}$ existiert, dann enthält der Graph ein Dreieck $\{v_i, v_j, u\}$

$\Rightarrow$ Checke für alle echt positiven Werte b_{ij}, ob es eine Kante $\{v_i, v_j\}$ gibt

$\Rightarrow$ Zeit-Komplexität: $\mathcal{O}(n^\alpha)$ mit $\alpha < 2.376$

Cliquen fester Größe $k \geq 3$

Sei $\alpha(r, s, t)$ so definiert, dass man die Multiplikation einer $n^r \times n^s$-Matrix mit einer $n^s \times n^t$-Matrix in Zeit $\mathcal{O}\left(n^{\alpha(r,s,t)}\right)$ berechnen kann.

Satz

Für jedes $k \geq 3$ existiert ein Algorithmus, der eine Clique der Größe k in einem Graphen mit n Knoten finden kann (falls eine solche existiert) und der in Zeit $\mathcal{O}\left(n^{\beta(k)}\right)$ läuft, wobei $\beta(k) = \alpha(\lfloor k/3 \rfloor, \lceil (k-1)/3 \rceil, \lceil k/3 \rceil)$.

Cliquen fester Größe $k \geq 3$

Beweis.

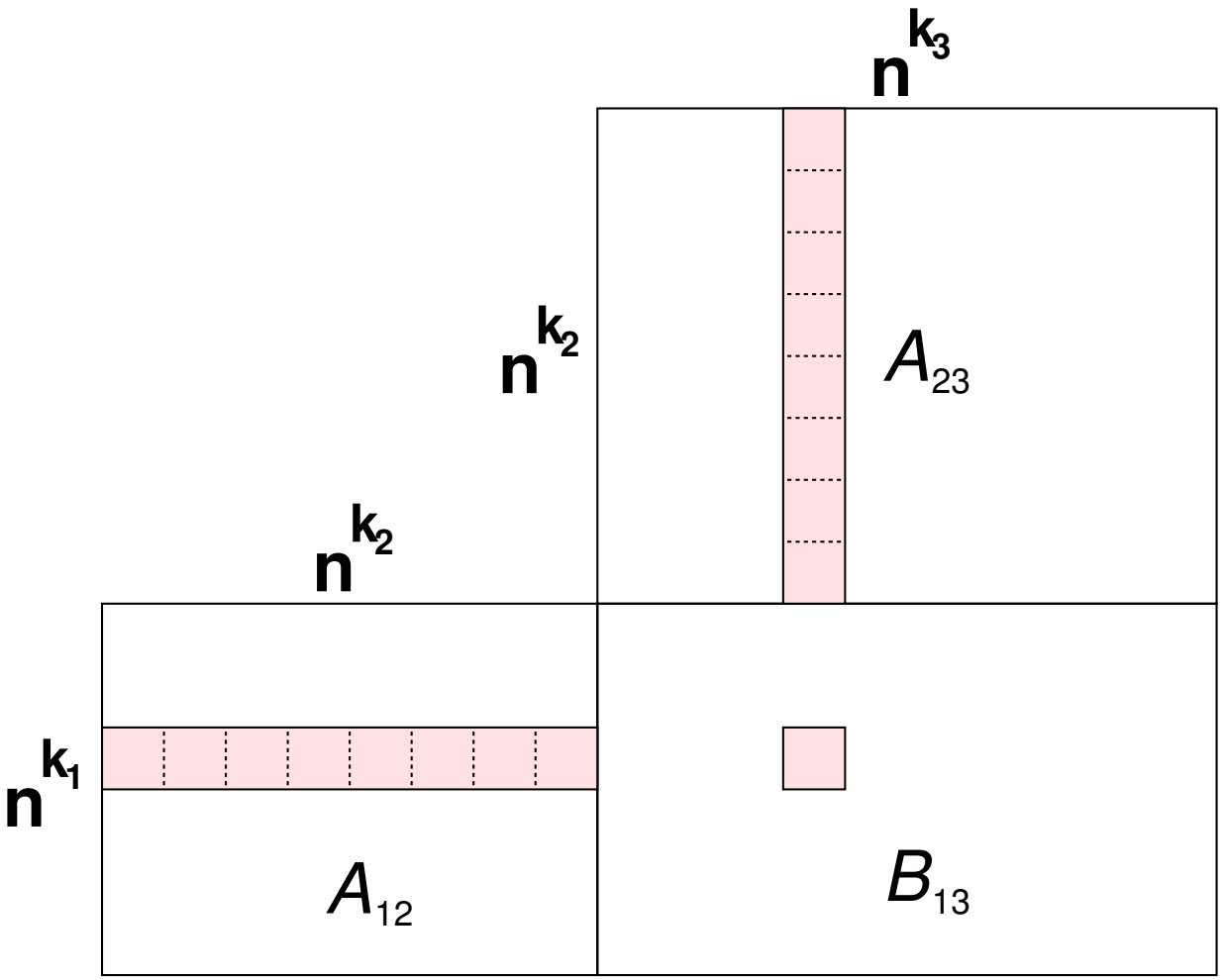
- Seien $k_1 = \lfloor k/3 \rfloor$, $k_2 = \lceil (k-1)/3 \rceil$ und $k_3 = \lceil k/3 \rceil$
- Es gilt $k = k_1 + k_2 + k_3$.
- Konstruiere tripartiten Hilfsgraph $\tilde{G}$
  - $\tilde{V} = \tilde{V}_1 \cup \tilde{V}_2 \cup \tilde{V}_3$, wobei V_i aus allen Cliquen der Größe k_i in G besteht
  - Knoten $U \in \tilde{V}_i$ und $U' \in \tilde{V}_j$ sind adjazent in $\tilde{G}$ g.d.w. $i \neq j$ und $U \cup U'$ eine Clique der Größe $k_i + k_j$ in G ist.
- Teste $\tilde{G}$ auf Dreiecke

Cliquen fester Größe $k \geq 3$

Beweis.

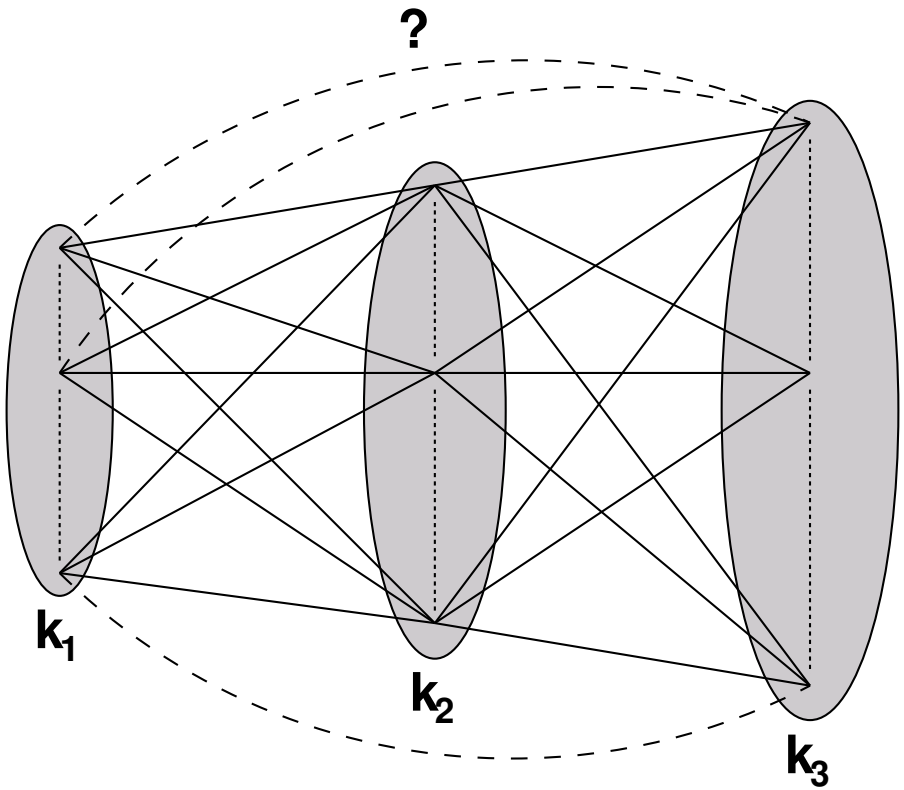
- Dreieck $\{U_1, U_2, U_3\}$ impliziert eine Clique der Größe k in G
- geht mit schneller Matrixmultiplikation, aber hier muss eine $n^{k_1} \times n^{k_2}$-Matrix (Kanten zwischen $\tilde{V}_1$ und $\tilde{V}_2$) mit einer $n^{k_2} \times n^{k_3}$-Matrix (Kanten zwischen $\tilde{V}_2$ und $\tilde{V}_3$) multipliziert werden, und zwar in Zeit $\mathcal{O}\left(n^{\beta(k)}\right)$
- Berechnung der drei Matrizen A_{12}, A_{23} und A_{13} in $\mathcal{O}\left(n^{\max\{k_1+k_2, k_1+k_3, k_2+k_3\}}\right) = \mathcal{O}\left(n^{\lceil \frac{2k}{3} \rceil}\right)$
  (wird dominiert durch die Zeit $\mathcal{O}(n^{\beta(k)})$ für die Multiplikation der rechteckigen Matrizen, also $B_{13} = A_{12} \cdot A_{23}$)

Cliquen fester Größe $k \geq 3$



B_{13} wird dann mit A_{13} verknüpft

Cliquen fester Größe $k \geq 3$



Cliquen fester Größe: Beispielkomplexitäten

Cliquengröße	Vollständige Suche	Matrixmultiplikation
3	$\mathcal{O}(n^3)$	$\mathcal{O}(n^{2.376})$
4	$\mathcal{O}(n^4)$	$\mathcal{O}(n^{3.376})$
5	$\mathcal{O}(n^5)$	$\mathcal{O}(n^{4.220})$
6	$\mathcal{O}(n^6)$	$\mathcal{O}(n^{4.751})$
7	$\mathcal{O}(n^7)$	$\mathcal{O}(n^{5.751})$
8	$\mathcal{O}(n^8)$	$\mathcal{O}(n^{6.595})$

Cliquen fester Größe: Mitgliedszahlen

Satz

Für jedes $k \geq 3$ existiert ein Algorithmus, der in Zeit $\mathcal{O}\left(n^{\beta(k)}\right)$ läuft und der für jeden Knoten zählt, an wieviel Cliquen der Größe k er beteiligt ist (in einem Graphen mit n Knoten), wobei $\beta(k) = \alpha(\lfloor k/3 \rfloor, \lceil (k-1)/3 \rceil, \lceil k/3 \rceil)$.

Cliquen fester Größe: Mitgliedszahlen

Beweis.

- Für $k = 3$ (Dreiecke) kann man nicht nur feststellen, *ob* zwei Knoten v_i und v_j zu einem Dreieck gehören, sondern auch *zu wievielen*.
- Wenn Kante $\{v_i, v_j\}$ in G existiert, dann ist diese Anzahl gleich dem Eintrag b_{ij} in der quadrierten Adjazenzmatrix $B(G) = A(G) \cdot A(G)$.
- Anwendung im allgemeinen Fall von $\tilde{G}$:
  für jeden Knoten $v \in V$ sei $C_k(v)$ die Anzahl verschiedener Cliquen der Größe k, in denen v enthalten ist.
- Entsprechend sei $\tilde{C}_3(U)$ die Anzahl der Dreiecke in $\tilde{G}$, zu denen Knoten U gehört.
  (U ist eine Clique der Größe kleiner als k)

Cliquen fester Größe: Mitgliedszahlen

Beweis.

- Cliquen der Größe k können viele verschiedene Repräsentationen in $\tilde{G}$ haben.
- Diese Anzahl ist die Anzahl der Partitionierungen von einer Menge der Kardinalität k in drei Mengen der Kardinalitäten k_1, k_2 und k_3, also der Multinomialkoeffizient $\binom{k}{k_1,k_2,k_3}$.
- o.B.d.A. sei k_1 der kleinste der drei Parameter.
  Sei $\mathcal{U}(v)$ die Menge aller Cliquen U der Größe k_1 in G, so dass $v \in U$. Dann gilt:

$$\sum_{U \in \mathcal{U}(v)} \tilde{C}_3(U) = \binom{(k-1)}{(k_1-1), k_2, k_3} \cdot C_k(v)$$

- Berechne linke Seite in $\mathcal{O}\left(n^{\beta(k)}\right)$ (berechne Matrizen; suche Einträge für alle U, die v enthalten); Berechne $C_k(v)$)

'Effiziente' Aufzählungsalgorithmen

- Ausgabe oft exponentiell in der Eingabelänge

$\Rightarrow$ etwas anderer Effizienzbegriff

- polynomielle Gesamtzeit:
  bei Ausgabe von C Konfigurationen Begrenzung der Zeit durch ein Polynom in C und in der Eingabegröße n
  (sozusagen output-sensitiv)
  - vollständige Suche ist nicht in polynomieller Gesamtzeit
  - Aufzählung aller maximalen Cliquen geht in pol. Gesamtzeit
  - Aufzählung aller Maximum-Cliquen geht nur in pol. Ges.zeit, falls $\mathcal{P} = \mathcal{NP}$
- polynomielle Verzögerung (polynomial delay):
  Zeit bis zur Ausgabe der ersten Konfiguration, zwischen zwei aufeinanderfolgnden Ausgaben von Konfigurationen und von der Ausgabe der letzten Konfiguration bis zum Stop ist polynomiell in der Eingabegröße

Aufzählung maximaler Cliquen

Satz

Es gibt einen Algorithmus, der alle maximalen Cliquen mit (polynomieller) Verzögerung $\mathcal{O}\left(n^3\right)$ und (linearem) Platzverbrauch $\mathcal{O}(m+n)$ aufzählt.

Algorithmus zur Aufzählung maximaler Cliquen

- Konstruiere einen Binärbaum mit n Leveln, dessen Blätter sich nur auf Level n befinden.
- Jedes Level ist einem Knoten des Eingabegraphen G zugeordnet. Im Level i betrachtet man Knoten v_i.
- Insbesondere entsprechen die Knoten von Level i des Baums den maximalen Cliquen des induzierten Teilgraphen $G[\{v_1, \ldots, v_i\}]$.

$\Rightarrow$ Die Blätter sind genau die maximalen Cliquen von G.

- Für ein gegebenes Level i und eine maximale Clique U in $G[\{v_1, \ldots, v_i\}]$ wollen wir die Kinder auf dem nächsten Level $i+1$ bestimmen:
  1. Alle Knoten von U sind adjazent zu v_{i+1} in G.
  2. Es existiert ein Knoten in U, der nicht zu v_{i+1} adjazent ist in G

Algorithmus zur Aufzählung maximaler Cliquen

Fallunterscheidung:

1. Alle Knoten von U sind adjazent zu v_{i+1} in G.
   $\Rightarrow$ $U \cup \{v_{i+1}\}$ ist maximale Clique in $G[\{v_1, \ldots, v_{i+1}\}]$
   - Das ist die einzige Möglichkeit, eine maximale Clique in $G[\{v_1, \ldots, v_{i+1}\}]$ zu bekommen, die U enthält
     In diesem Fall hat U nur ein einziges Kind im Baum.
2. Es existiert ein Knoten in U, der nicht zu v_{i+1} adjazent ist in G
   Man kann 2 maximale Cliquen in $G[\{v_1, \ldots, v_{i+1}\}]$ erhalten:
   1. U ist selbst eine maximale Clique
   2. $\left(U \setminus \bar{N}(v_{i+1})\right) \cup \{v_{i+1}\}$ ist eine Clique,
      wobei $\bar{N}(v_{i+1})$ alle nicht mit v_{i+1} adjazenten Knoten sind
      - Wenn die Menge eine *maximale* Clique ist, hätte U zwei Kinder im Baum
      - $\left(U \setminus \bar{N}(v_{i+1})\right) \cup \{v_{i+1}\}$ könnte aber Kind von mehreren sein

      $\Rightarrow$ Kind der lexikographisch kleinsten Menge U (falls maximal)

$\Rightarrow$ Binärbaum
(interne Knoten haben 1 oder 2 Kinder, Blätter nur in Level n)

Algorithmus zur Aufzählung maximaler Cliquen

- Traversiere den Binärbaum per Tiefensuche (DFS)
- Ausgabe aller Blätter
- Gegeben Knoten U auf Level i:
  - `Parent`(U,i):
    Vaterknoten von U ist die lexikographisch kleinste maximale Clique in $G[\{v_1,\ldots,v_{i-1}\}]$, die $U \setminus \{v_i\}$ enthält
    $\Rightarrow$ Grundfunktion, in $\mathcal{O}(m+n)$
  - `LeftChild`(U,i):
    - falls $U \subseteq N(v_{i+1})$ (1. Fall), dann $U \cup \{v_{i+1}\}$
    - falls $U \not\subseteq N(v_{i+1})$ (Teil des 2. Falls), dann U
    - Unterscheidung der Fälle kostet $\mathcal{O}(m+n)$ Zeit
  - `RightChild`(U,i):
    - falls $U \subseteq N(v_{i+1})$, dann existiert kein rechtes Kind
    - falls $U \not\subseteq N(v_{i+1})$, dann
      $\left(U \setminus \bar{N}(v_{i+1})\right) \cup \{v_{i+1}\}$ falls es maximale Clique ist und
      $U = \texttt{Parent}\left(\left(U \setminus \bar{N}(v_{i+1})\right) \cup \{v_{i+1}\}, i+1\right)$
      sonst keins
    - Kosten: $\mathcal{O}(m+n)$ Zeit

Algorithmus zur Aufzählung maximaler Cliquen

- Längster Pfad zwischen zwei Blättern im Baum ist $2n - 2$ und geht durch $2n - 1$ Knoten
- pro Knoten Zeitaufwand $\mathcal{O}(m + n)$
- Jeder Unterbaum hat ein Blatt auf Level n

$\Rightarrow$ Ausgabeverzögerung $\mathcal{O}(n^3)$

- Wenn ein Knoten bearbeitet wird, muss nur die Menge U, das Level i, sowie ein Label zur Unterscheidung von linkem/rechten Kind gespeichert werden

$\Rightarrow$ Speicheraufwand $\mathcal{O}(m + n)$

Aufzählung in lexikographischer Reihenfolge

- Es ist $\mathcal{NP}$-vollständig für einen Graphen G und eine maximale Clique U von G zu entscheiden, ob es eine maximale Clique U' gibt, die lexikographisch größer ist als U.
- Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es keinen Algorithmus, der in Polynomialzeit zu einem Graphen G und einer maximalen Clique U von G die lexikographisch nächstgrößere maximale Clique generiert.
- Überraschenderweise kann man trotzdem alle maximalen Cliquen in lexikographischer Ordnung mit polynomieller Verzögerung aufzählen.