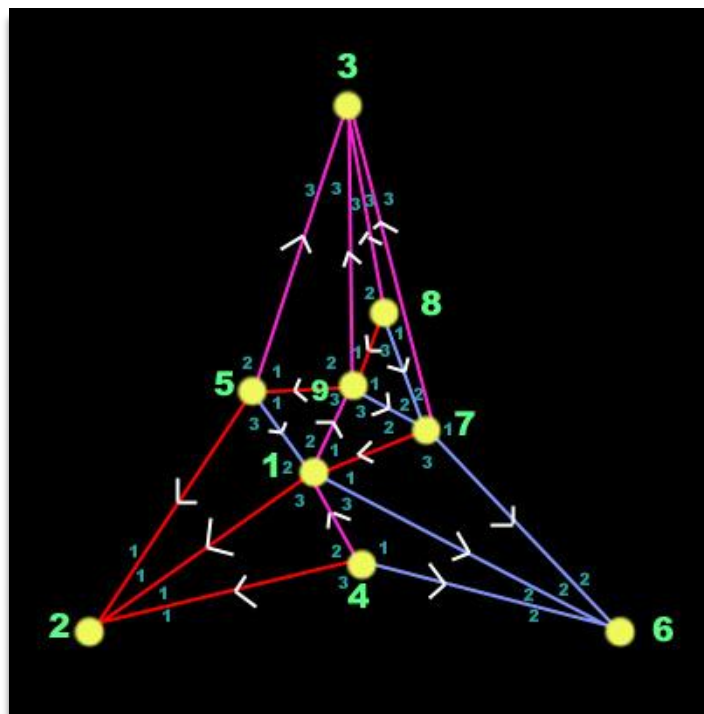


Angewandte Graphenalgorithmen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Gleichungssysteme, perfekte Eliminationsordnung,
Einführung in perfekte / chordale Graphen...



Seminarausarbeitung: Oren Avni (Halvani)

Betreuer: PD Dr. Elias Dahlhaus

Fachgebiet: Foundations of
Computing

Zeitraum: Sommersemester 2009



Gliederung...



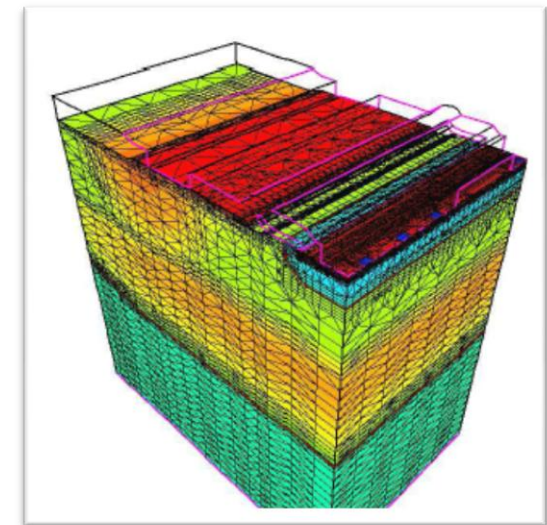
- Einleitung und Motivation zu Gauß Elimination & Cholesky Zerlegung
- Gauß Elimination in dünnbesetzten Matrizen als graphentheoretischer Vorgang
- Einführung: perfekte / chordale Graphen
- Definitionen und Sätze
- Perfekte Eliminationsordnung / Lexikographische Breitensuche
- Zusammenhang: chordale Graphen und Gauß Elimination
- Chordale Graphen als Durchschnittsgraphen von Bäumen



Motivation..

- Lineare Gleichungssysteme sind der Menschheit bereits seit der Antike bekannt und bilden auch heute noch eine wichtige Beschreibungsmöglichkeit für zahlreiche Probleme
- Beispiel: Eine Vielzahl von Optimierungsprobleme im Bereich des „Operations Research“...

Hier entstehen oft Gleichungssysteme mit mehreren Kilobytes (!!)



- Oft handelt es sich dabei um **dünnbesetzte Matrizen**...

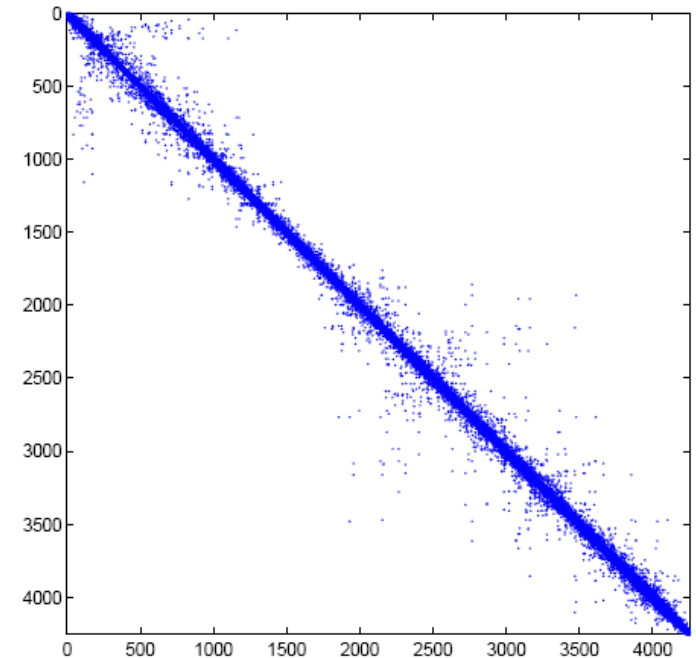
Einschub: dünnbesetzte Matrizen

- Was sind dünnbesetzte Matrizen ?

→ Matrizen mit (vielen) Null-Elementen...

- Welche Eigenschaften haben Sie ?

→ können effizient berechnet und abgespeichert werden...



- Im Allgemeinen heißt eine $n \times n$ Matrix A **dünnbesetzt**, wenn sie **$O(n)$** statt **$O(n^2)$** Einträgen hat



Einschub: Speicherungsschemata für dünnbesetzte Matrizen..

- **Compressed-Sparse-Row-Format**

Reduziert um bis zu 30% Platzbedarf, bietet schnellen Zugriff...

$$\begin{pmatrix} 0 & 0 & 3 & 0 \\ 0 & 0 & 4 & 1 \\ 7 & 3 & 1 & 0 \\ 1 & 2 & 0 & 8 \end{pmatrix} \quad \begin{array}{l} n = 4, \\ nnz = 9 \end{array}$$

→ $O(|\text{rowpoint}| + |\text{col}| + |\text{values}|)$ statt $O(n^2)$

rowpoint	0	1	3	6	9				
col	2	2	3	0	1	2	0	1	3
values	3	4	1	7	3	1	1	2	8

← (Zeilenpointer auf die Einträge in: col und values)

← (Spaltenindizes der Einträge a_{ij})

← (Alle vorkommenden Zahlen in A)

Bei einer $A^{50.000 \times 50.000}$ Matrix mit $2.5 \cdot 10^9$ Einträge werden insgesamt Gerade mal 7.8 Mbyte für Nicht-Null Elemente benötigt (!!)



Wie werden Gleichungssysteme gelöst ?

- Herkömmliche (iterative) Verfahren: **Gauß Elimination**
sowie aus der numerischen Mathematik: **Cholesky Zerlegung**

→ beide Verfahren laufen **leider** in $O(n^3)$

wobei letztere nur auf positiv definiten & symmetrischen Matrizen angewendet werden kann

→ treten zum Glück häufig in der Praxis auf...

- Die Verfahren haben Gemeinsamkeiten: Beide sind numerisch äquivalent und erzeugen eine eindeutige Zerlegung:

Bei Gauß-Verfahren: $\mathbf{A} = \mathbf{LR}$ bzw. bei Cholesky: $\mathbf{A} = \mathbf{LL}^T$



Existenz von Zerlegungen

- Es ist **leider nicht immer garantiert**, dass zu einer Matrix eine Zerlegung existiert
- Ein unmöglicher Fall tritt auf, falls für mindestens ein Diagonalelement gilt: $a_{ii} = 0$
 - **Lässt sich durch Pivotisierung vermeiden, aber kostet zusätzliche Laufzeit !**
- Für den weiteren Verlauf setzen wir voraus, dass die behandelten Matrizen eine eindeutige Zerlegung besitzen...



Beispiel: Cholesky Verfahren...



(Cholesky-Verfahren)

$$A = \begin{pmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -1 & -2 & 3 \end{pmatrix} \quad \begin{array}{l} (j=1): l_{11} = \sqrt{1} = 1 \\ (i=2): l_{21} = \frac{a_{21}}{l_{11}} = 1 \\ (i=3): l_{31} = \frac{a_{31}}{l_{11}} = -1 \end{array}$$

$$(j=2): l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{2 - 1} = 1$$

$$(i=3): l_{32} = \frac{a_{32} - l_{31}l_{21}}{l_{22}} = -1$$

$$(j=3): l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = 1$$

A besitzt also die Cholesky-Zerlegung $LL^T = A$, mit

(L^T ist die Transponierte zu L) \nearrow

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}$$



Beispiel: Gauß Verfahren...



(Gaußsches Eliminationsverfahren)

$$A = \begin{pmatrix} 1 & 4 & 6 \\ 2 & 6 & 4 \\ 0 & 2 & 4 \end{pmatrix} \quad \text{⚡} \quad R = \begin{pmatrix} 2 & 6 & 4 \\ 0 & 2 & 4 \\ 0 & 0 & 2 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix}$$

k	$A^{(k)}$	$\tilde{A}^{(k)}$	$L^{(k)}$	$\tilde{L}^{(k)}$
1	$\begin{pmatrix} 1 & 4 & 6 \\ \boxed{2} & 6 & 4 \\ 0 & 2 & 4 \end{pmatrix}$	$\begin{pmatrix} \boxed{2} & 6 & 4 \\ 1 & 4 & 6 \\ 0 & 2 & 4 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
2	$\begin{pmatrix} 2 & 6 & 4 \\ 0 & 1 & 4 \\ 0 & \boxed{2} & 4 \end{pmatrix}$	$\begin{pmatrix} 2 & 6 & 4 \\ 0 & \boxed{2} & 4 \\ 0 & 1 & 4 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}$
3	$\begin{pmatrix} 2 & 6 & 4 \\ 0 & 2 & 4 \\ 0 & 0 & 2 \end{pmatrix}$	$\begin{pmatrix} 2 & 6 & 4 \\ 0 & 2 & 4 \\ 0 & 0 & 2 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$



Gauß Verfahren Vorteile / Nachteile



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Vorteile:

- Vorhersagbarkeit der Laufzeit und des Speicherbedarfs
- Berechnung einer exakten Lösung (sofern vorhanden)
- Auf jedes LGS anwendbar

- Nachteile:


- Schlecht parallelisierbar → Besser: iterative Verfahren...
- Bei dünnbesetzten Matrizen entsteht leider oft ein ungewollter Nebeneffekt (Fill-In)



Eliminationsprozeß & Fill-In

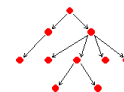
- Was passiert während des Eliminationsprozesses ?
- Die Gauß Elimination erzeugt (leider) oft weitere „Fill-in“ Elemente...

„Fill-In“ → Diejenigen Stellen in einer Matrix, an denen in der unteren /oberen Dreiecksmatrix Einträge $\neq 0$ entstehen, obwohl in der ursprünglichen Matrix A an dieser Stelle Einträge = 0 standen

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 4 & 2 & 0 & 1 \\ 6 & 0 & 3 & 0 \\ 0 & 0 & 4 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 6 & 0 & 1 & 0 \\ 0 & 0 & -\frac{4}{3} & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & -4 & 1 \\ 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$




- Die Fill-In Stellen, die bei der Zerlegung entstehen müssen, zwingend berücksichtigt werden...Warum ?
- Ihnen muss Speicherplatz zugewiesen werden, ansonsten:
 - keine **100% korrekte** Faktorisierung der Matrix mehr möglich...
- Um den entstehenden Fill-In über die Struktur der Graphen darstellen und berechnen zu lassen, wählten viele Forscher einen graphentheoretischen Ansatz → **dazu gleich mehr...**
- Grundlegende und vor allem bemerkenswerte Erkenntnisse lieferten die ersten Forscher: **Tarjan & Rose**



Fill-In



Die Fragen:

(1) In wie weit ist Fill-In vermeidbar ?

(2) Wie erhält man einem minimalen entstehenden Fill-In ?

werden wir im weiteren Verlauf dieser Präsentation als auch in den anderen Seminarthemen klären...

→ **Themen:** „Minimum Fill-In“, „Minimum Degree / Nested Dissection Heuristik“



Vorab, die schlechten Nachrichten:

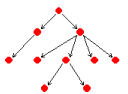
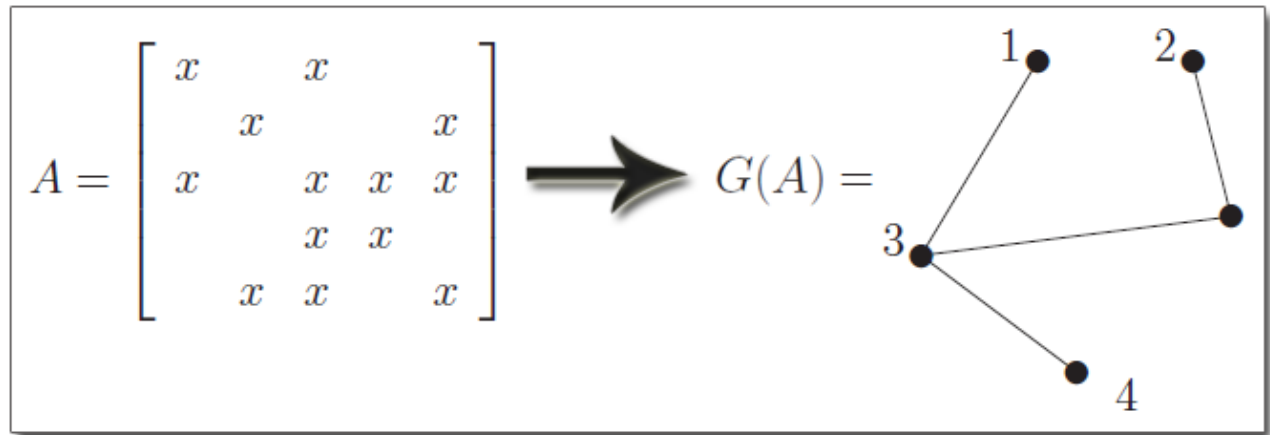
- (1) Eine generelle Vermeidung von entstehenden Fill-In ist leider **unumgänglich**, jedoch in manchen Fällen vorhersagbar
- (2) Die Berechnung eines minimalen Fill-In's ist **NP-vollständig**
Es existieren jedoch heuristische Algorithmen die brauchbare Approximationen liefern...

→ mehr über (2) in den anderen Seminarthemen...



Fill-In

- In wie weit Fill-In entsteht, hängt sehr stark von der **Struktur** der Matrix **A** ab...
- Wir verwenden von nun an die Bezeichnung: $G(\mathbf{A})$ die den Adjazensgraphen der Matrix **A** darstellt
- Beispiel:



Fill-In

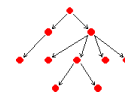
- Betrachten wir nun eine Matrix bei der durch zwei verschiedene Pivotisierungen Fill-In **entsteht**, bzw. **vermieden** wird...

Schlechte Wahl... $\begin{pmatrix} \textcircled{4} & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \textcircled{3} & -1 & -1 \\ 0 & -1 & 3 & -1 \\ 0 & -1 & -1 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \textcircled{8} & -4 \\ 0 & 0 & -4 & 8 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \textcircled{12} \end{pmatrix}$

Gute Wahl... $\begin{pmatrix} 4 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & \textcircled{1} \end{pmatrix} \rightarrow \begin{pmatrix} 3 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & \textcircled{1} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & \textcircled{1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

↓

- Wie kann die Struktur der Matrix verbessert werden um so, den nicht vermeidbaren Fill-In zu reduzieren ?



Vorbereitung für den Eliminationsprozeß

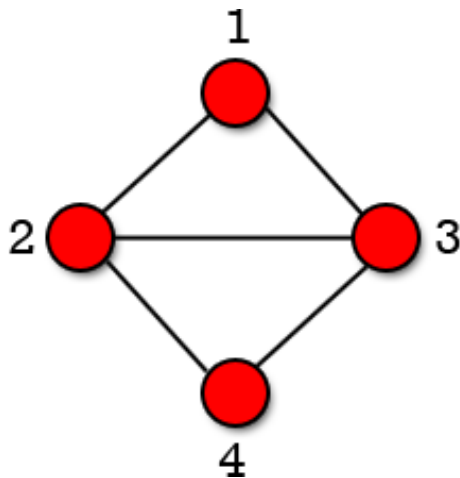
- Die Struktur kann zunächst durch eine Permutation der Zeilen und Spalten von \mathbf{A} geändert werden, dies entspricht also der Pivotisierung... (d.h. das Gleichungssystem bleibt nach der Permutation gleich)
- Man transformiert: $Ax = b$ in $PAP^T Px = Pb$
- Dann sucht man eine Permutations-Matrix \mathbf{P} so dass $A_p = PAP^T$ eine bessere Gestalt erhält...

$$(P P^T = E) \begin{matrix} \nearrow \\ \searrow \end{matrix} PAP^T (Px) = Pb \iff A_p x_p = b_p$$




Vorbereitung für den Eliminationsprozeß

Beispiel für die Permutation P : Betrachten wir den linken Graph G zu einer entsprechenden dünnbesetzten Matrix...




Adj_1



	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

Adj_2



	2	1	4	3
2	0	1	1	1
1	1	0	0	1
4	1	0	0	1
3	1	1	1	0

Konkret: Der Graph repräsentiert die Struktur des Problems !
Er bleibt also erhalten, während sich die Matrix ändern kann...



Symmetrische Gauß Elimination und Knoten Elimination

Das Finden einer permutationsbedingten „Fill-in“-Reduktion:

$$A_p = PAP^T$$

lässt sich somit durch das Finden eines bestimmten Knoten-Eliminierungsmodells des Graphen **G(A)** ersetzen...



Vorbereitung für den Eliminationsprozeß

- Es kommen also bei der Elimination für dünnbesetzte Matrizen zwei weitere Schritte hinzu:

1.) Reordering (Fill-In Reduktion): $\mathbf{A}_p = \mathbf{PAP}^T$, $\mathbf{b}_p = \mathbf{Pb}$

2.) Faktorisierung: $\mathbf{A}_p = \mathbf{LL}^T$

3.) Substitution: $\mathbf{Ly} = \mathbf{b}_p$, $\mathbf{L}^T \mathbf{x}_p = \mathbf{y}$

4.) Rücktransformation: $\mathbf{x} = \mathbf{P}^T \mathbf{x}_p$

(die Matrix A permutiert)

- Wie interpretiert man die Elimination graphentheoretisch ?



Graph Modell für die **symmetrische** Gauss-Elimination

- Matrix **A** wird durch einen ungerichteten Graph **G** = (**X**, **E**) repräsentiert mit **X** = { x_1, x_2, \dots, x_n }
- **X** repräsentiert also die Unbekannten x in **Ax** = **b**
- $\{x_i, x_j\} \in \mathbf{V}$ falls **A**_{ij} ≠ 0 und $i \neq j$
- Wir beginnen die Gauß Elimination

→ (aus graphentheoretischer Sicht)



Symmetrische Gauß Elimination und Knoten Elimination



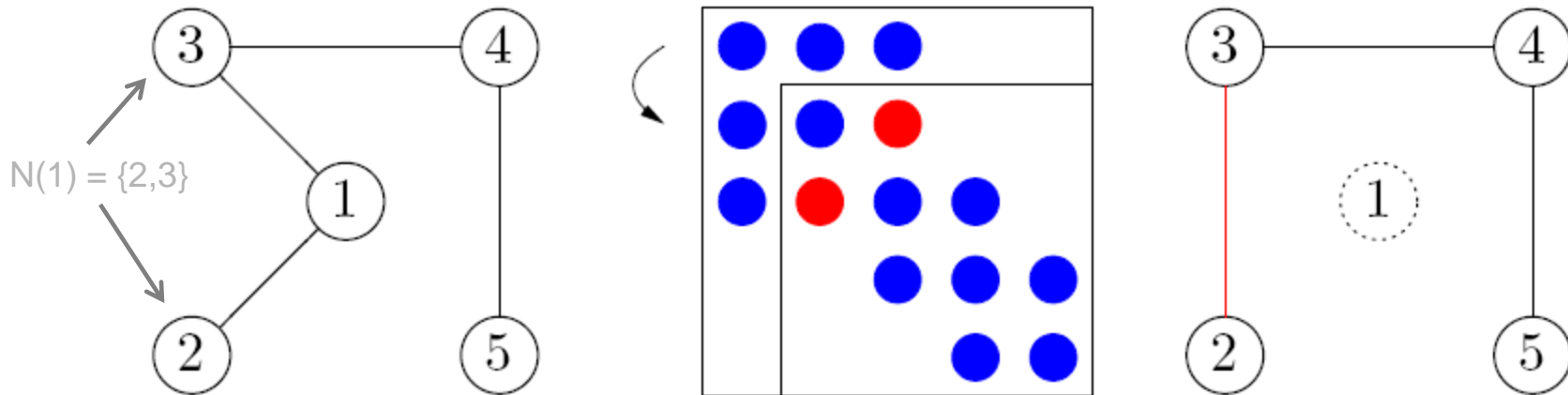
- Eine dünnbesetzte, symmetrische Gauß-Elimination kann durch ein Knoten-Eliminierungsmodell auf den Graphen \mathbf{G} reduziert werden...

Regeln des Knoten-Eliminierungsmodelles:

- Wähle einen Knoten \mathbf{x}_i , welcher eliminiert werden soll
- Entferne alle zu \mathbf{x}_i inzidenten Kanten aus dem Graphen $\mathbf{G}(\mathbf{A}) = (\mathbf{X}, \mathbf{E})$
- Bilde den vollständigen Graphen aller zu \mathbf{x}_i benachbarten Knoten und füge diesen in \mathbf{G} ein



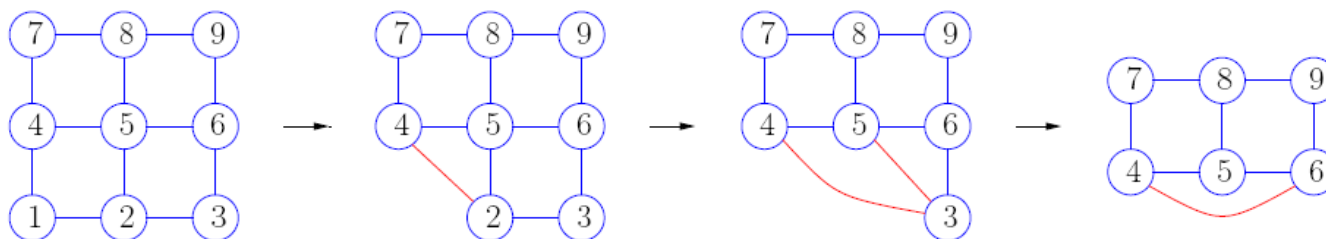
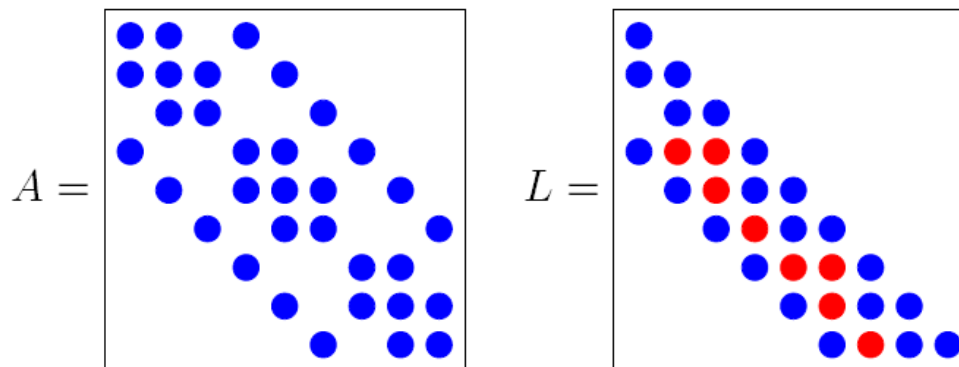
Graph Modell für die **symmetrische** Gauss-Elimination



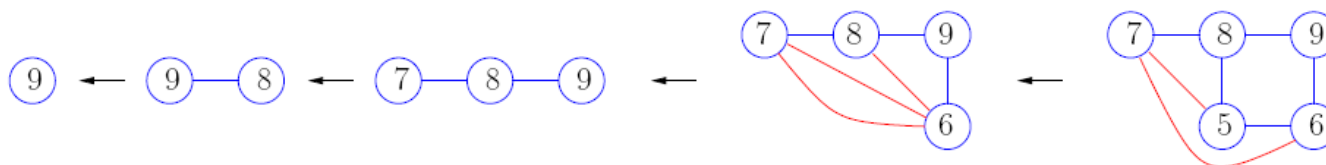
- Wir betrachten den ersten Schritt bei der Elimination...
- Der erste Knoten ($i = 1$) in der zweiten Zeile ($j = 2$) wird eliminiert: $a_{2,k} = a_{2,k} - (a_{2,1} = a_{1,1}) * a_{1,k}$ ($k = 2, \dots, 5$)
- **Annahme:** $a_{2,k}$ ist ein Null-Element in der Matrix A (z.B. $k = 3$)
→ Fill-in entsteht, falls $a_{2,1} \neq 0$ und $a_{1,k} \neq 0$ (z.B. $k = 3$)



Graph Modell für die **symmetrische** Gauss-Elimination



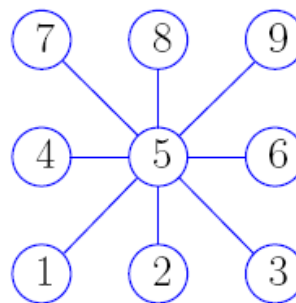
Erkenntnis: Neue Kanten **entsprechen Fill-In** während der Elimination...



Graph Modell für die **symmetrische** Gauss-Elimination

- Wir haben gesehen: Je nach Wahl der Pivotisierung während des Eliminierung variiert die Anzahl des Fill-Inn Stellen...

- **Beispiel:** Extrem Fill-In:



- Wird Knoten 5 zuerst eliminiert, erzeugt dies einen vollständigen Graphen mit $O(n^2)$ Kanten = Fill-In

→ Bei vollständige Graphen gilt: $|E| = \binom{|V|}{2}$

- Wird Knoten 5 zuletzt eliminiert, erzeugt dies keine einzige neue Kante



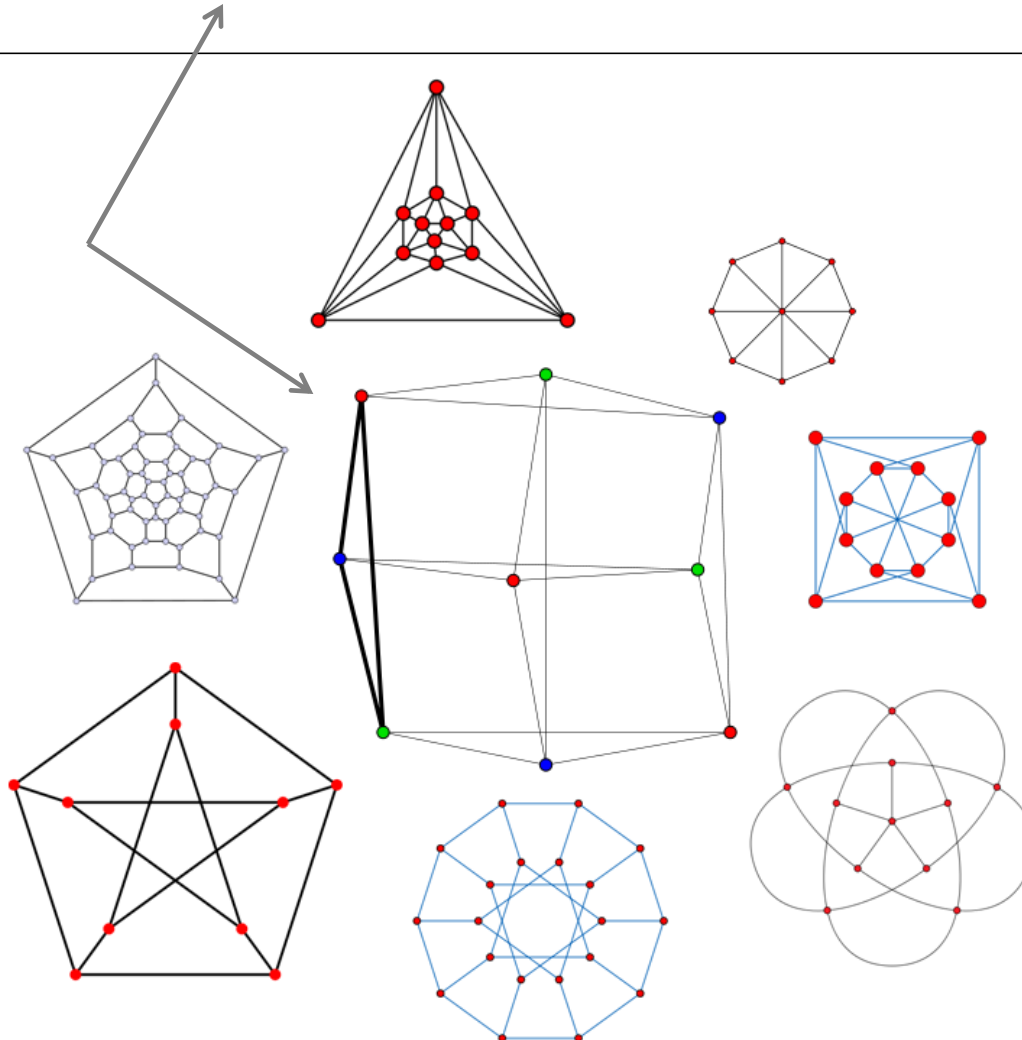
Überblick bisher...

Frage: Was genau haben wir bis hierhin erreicht ?

- Wir erhielten zunächst eine einfache Charakterisierung der Struktur einer Matrix → d.h. eine kompakte Beschreibung an welchen Positionen, Einträge $\neq 0$ eingetragen sind...
- Anschließend haben wir gesehen, dass es eben von dieser Struktur abhängt, in wie weit Fill-In entsteht
- Was also, wollen wir weiterhin erreichen ? → Wir suchen eine geeignete Ordnung um Fill-In so klein wie möglich zu halten, bzw. eine Ordnung die überhaupt kein Fill-In erzeugt...



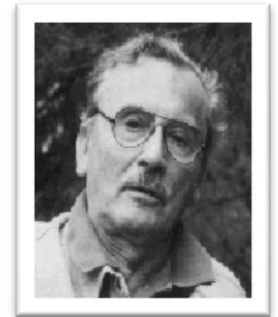
Einführung: **perfekte** Graphen



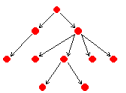
Perfekte Graphen

Bevor wir perfekte Graphen definieren, schauen wir uns einige Hintergründe an...

In den frühen 60er Jahren des letzten Jahrhunderts entdeckte **Claude Berge** diese ganz spezielle Graphenklasse. Er stellte u.A. Vermutungen die erst viele Jahre später bewiesen wurden → **Perfect Graph Conjecture**



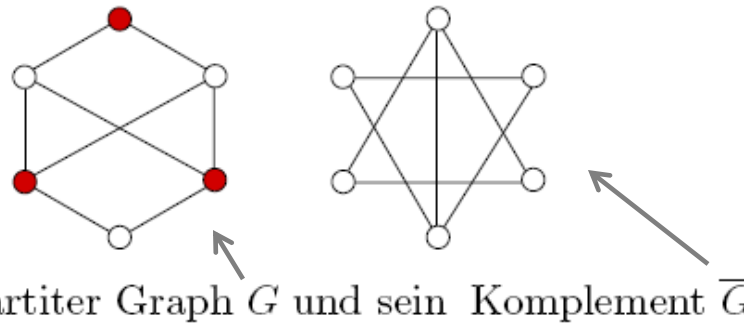
Aus algorithmischer Sicht sind perfekte Graphen äußerst interessant. Es existieren polynomielle Verfahren um die minimale Färbung / Knotenüberdeckung / maximale Clique oder stabile Menge (Farbklasse) zu bestimmen
→ **Bei anderen Graphen ein NP-vollständiges Problem...**



Definition: perfekte Graphen

(schwaches Theorem) Ein Graph \mathbf{G} heißt perfekt, gdw. sein Komplement \mathbf{G}' perfekt ist

Berge (1960), Lovasz (1971)



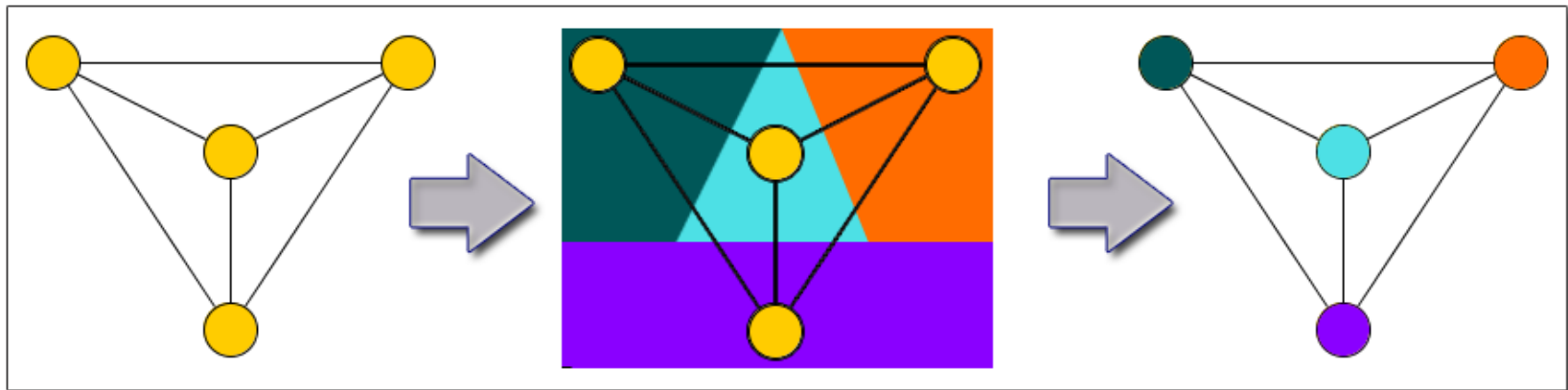
(strenges Theorem) Ein Graph \mathbf{G} heißt perfekt, gdw. weder \mathbf{G} noch sein Komplement \mathbf{G}' einen induzierten ungeraden Kreis ≥ 5 enthält

Chudnovsky, Robertson, Seymour und Thomas (2003)

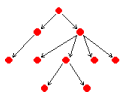


Definition: perfekte Graphen

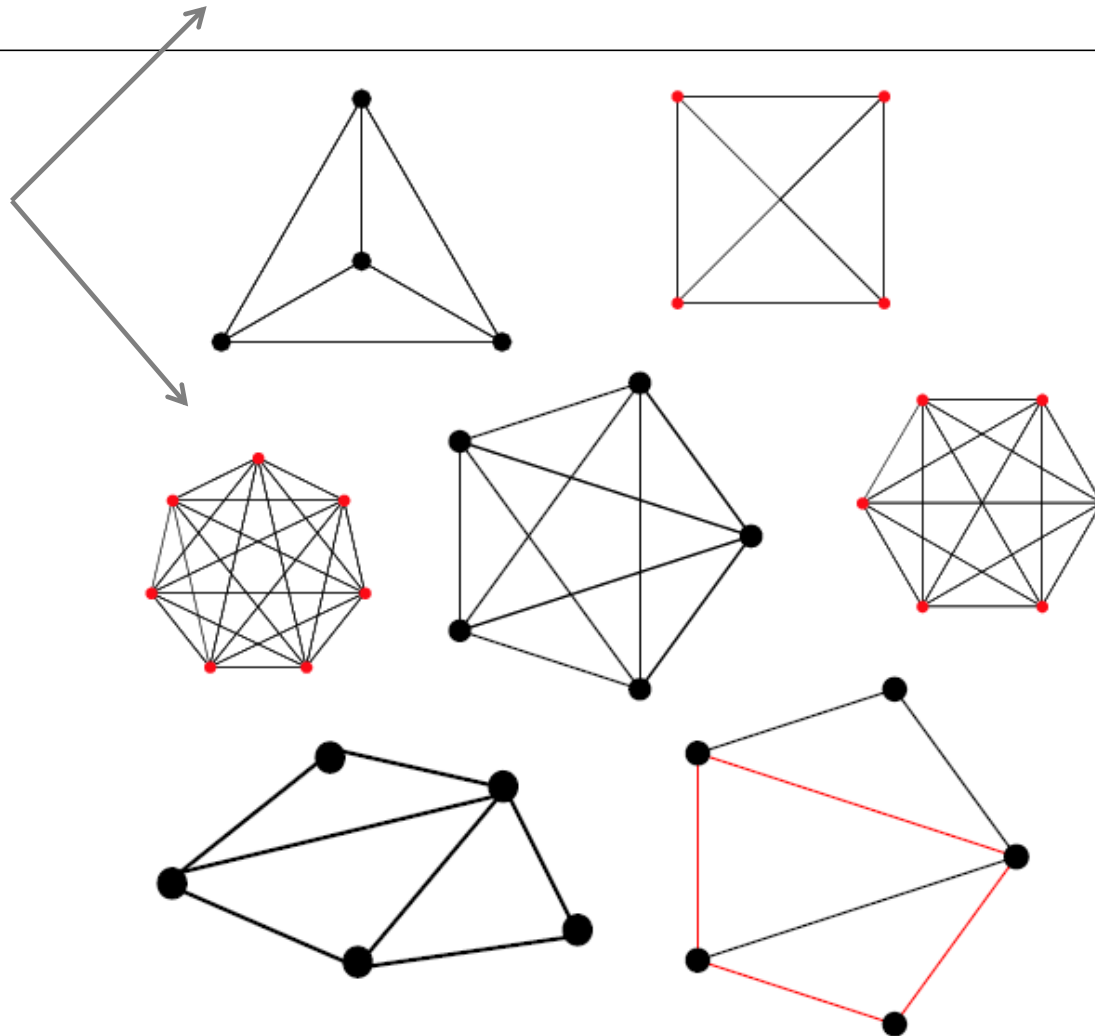
- Ein Graph G ist perfekt gdw. für jeden induzierten Teilgraph gilt, dass seine Cliquenzahl mit seiner chromatische Zahl übereinstimmt
- Die chromatische Zahl ist die kleinste Zahl, für die ein Graph G k -färbbar ist, Beispiel...



Vier Cliquen = Vier Farben \rightarrow perfekter Graph



Einführung: **chordale** Graphen



Einführung: **chordale** Graphen



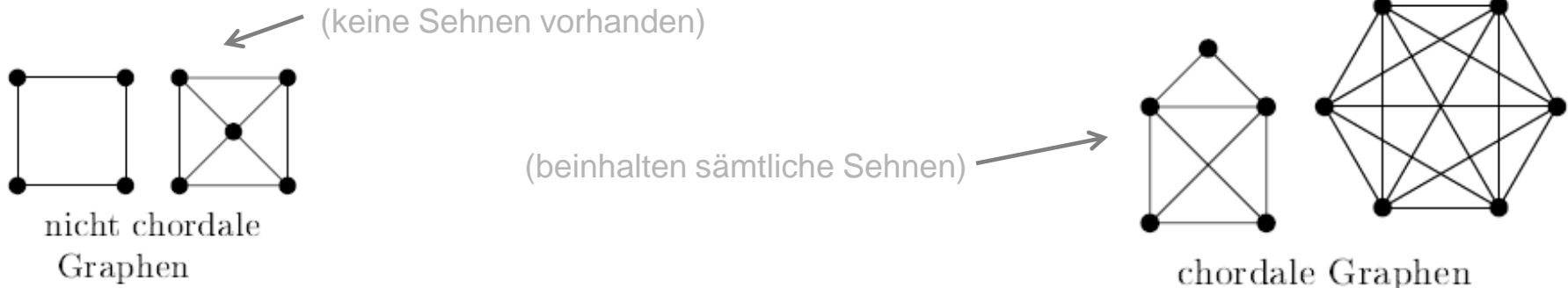
Eine der ältesten Klassen perfekter Graphen, ist die der chordalen Graphen...

- In der Literatur bekannt unter vielen Bezeichnungen: triangulierte Graphen, perfekte Eliminationsgraphen, Rigid-Circuit Graphen, ..., Dreiecksgraphen
 - insbesondere letzter Begriff ist irreführend, da Dreiecksgraphen planar sind, chordale Graphen jedoch **NICHT** zwingend (z.B. K_5)
- Einfache Beispiele für chordale Graphen sind vollständige Graphen, sowie Graphen ohne jegliche Kreise
 - Zusammenhängende Graphen ohne jegliche Kreise werden Bäume genannt



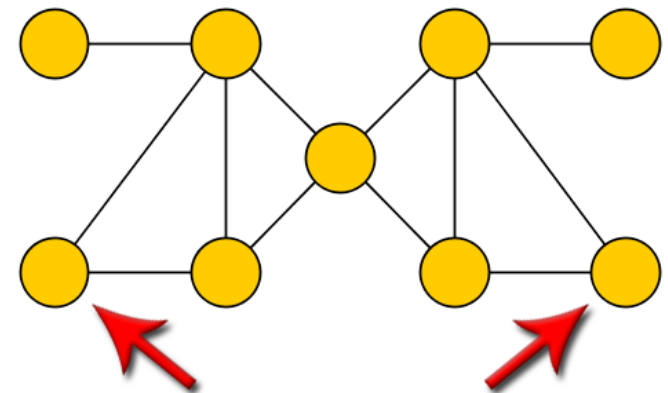
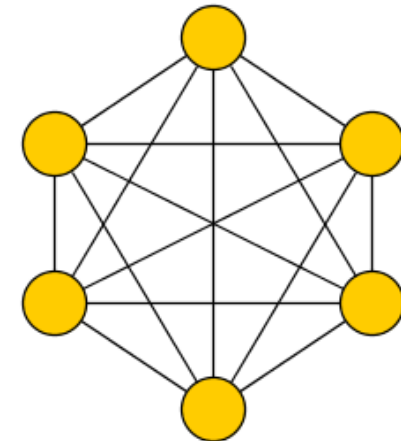
Definition: chordale Graphen

- Chordale Graphen und deren Komplemente sind perfekt
 - **Vorsicht: perfekt ist nicht chordal !!**
- Ein Graph **G** heißt chordal, falls alle induzierten Kreise aus genau 3 Knoten bestehen...
- Äquivalent dazu ist die Aussage, dass alle Kreise mit ≥ 4 Knoten eine Sehne besitzen. Eine Sehne (= chord) ist eine Kante zwischen zwei nicht aufeinander folgenden Knoten eines Kreises



Definition: chordale Graphen

- Ein Knoten \mathbf{v} dessen Nachbarschaft $N(\mathbf{v})$ eine Clique bildet, nennt man einen simplizialen Knoten
- Vollständige chordale Graphen haben **einen** simplizialen Knoten
- Nicht-Vollständige chordale Graphen mit $|V| \geq 2$ haben dagegen **mindestens zwei** simpliziale Knoten



Simpliziale Knoten sind die Grundlage von perfekten Eliminationsordnungen, dazu später...



Zusammenhang: chordale Graphen / Matrix Zerlegung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Wie am Anfang erwähnt modellierten Tarjan & Rose die Struktur dünnbesetzter Matrizen hinsichtlich der Zerlegung...
 - Dabei wurde die Struktur durch einen chordalen Graphen repräsentiert
- Warum ausgerechnet mit chordalen Graphen?
- Das Gauß-Eliminationsverfahren funktioniert damit, ohne das Auffüllen der Matrix **A** gdw. der durch die Matrix **A** definierter Graph: $G(\mathbf{A})$ **chordal** ist...
 - es entsteht also **kein** Fill-In !



Perfekte Eliminationsordnung (PEO)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Wir halten also nochmal fest:

- Falls $G(\mathbf{A})$ **chordal** ist, dann erzeugt eine Knoten-Pivotisierung hinsichtlich einer perfekten Eliminationsordnung keinen Fill-In
- Falls $G(\mathbf{A})$ **nicht chordal** ist, dann liegt das sogenannte Fill-In Problem vor, wo es darum geht für $G(\mathbf{A})$ eine Vervollständigung zu einem chordalen Graphen zu finden

→ (mit der kleinsten Anzahl an Kanten)



Perfekte Eliminationsordnung (PEO)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Ein Graph G ist chordal gdw. G eine sogenannte:
„perfekte Eliminationsordnung“ kurz: (PEO) besitzt
 - Insbesondere besitzt G für jedes seiner Cliques
eine perfekte Eliminationsordnung !
- Eine PEO lässt sich i.A. in polynomieller Laufzeit bestimmen...
 - bei geschickter Implementierung mittels Queues in **linearer Laufzeit**
(Rose, Tarjan & Lueker, 1976)

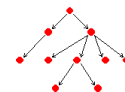


Abstrakter Algorithmus: „Chordalitätstest“

Input: Ein Graph $\mathbf{G} := (V, E)$

Output: Eine Aussage ob \mathbf{G} chordal ist oder nicht

- (1) Anwenden einer einfachen lexikographische Breitensuche (kurz: Lex-BFS) die eine Ordnung $\sigma := \{v_0 > v_1 > \dots > v_n\}$ zurückliefert
- (2) Falls es sich bei: σ^{-1} (also die Umkehrung der Ordnung σ) um eine perfekte Eliminationsordnung handelt, dann folgern wir hieraus: \mathbf{G} ist chordal, ansonsten: \mathbf{G} ist nicht chordal



Lexikographische Breitensuche & perfekte Eliminationsordnung

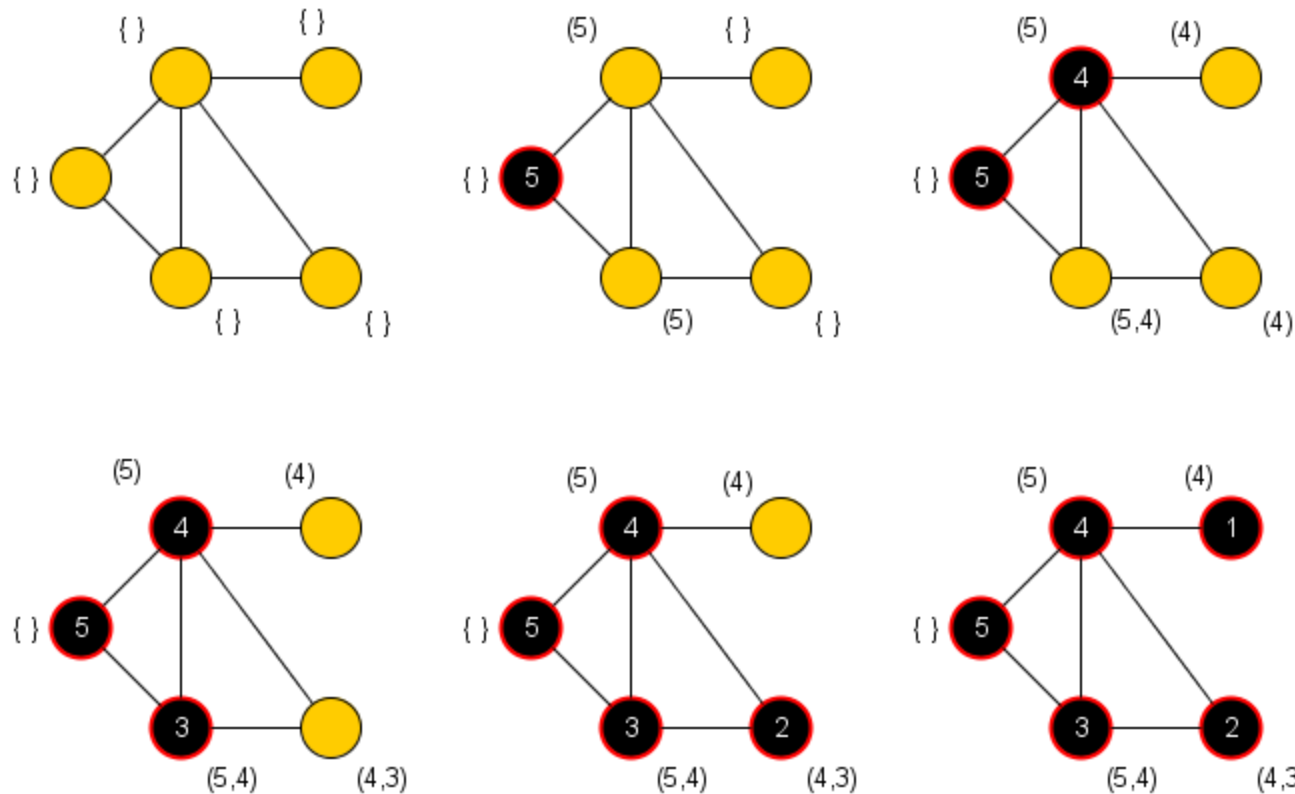


- Wir zeigen hier, wie mittels Lex.-BFS die PEO zu einem chordalen Graphen bestimmt werden kann...

```
LEX-BFS( $G$ )
Input: Ein zusammenhängender ungerichteter Graph  $G$  in
Adjazenzlistendarstellung; eine Ecke  $v \in V(G)$ .
Output: Ein perfektes Eliminationsschema, falls  $G$  chordal ist
1 for all  $v \in V$  do
2   Setze  $\text{label}[v] := \emptyset$ . { Alle Ecken sind unentdeckt }
3 for  $i := n, n-1, \dots, 1$  do
4   Wähle eine unnummerierte Ecke  $u$  mit lexikographisch größter Marke
    $\text{label}[u]$ 
5    $\sigma(u) := i$ 
6   for all  $v \in \text{ADJ}[u]$  do
7     if  $v$  hat noch keine Nummer  $\sigma(v)$  then
8       Setze  $\text{label}[v] := (\text{label}[v], i)$  { Füge  $i$  an die Marke von  $v$  an }
9 return  $\sigma$ 
```

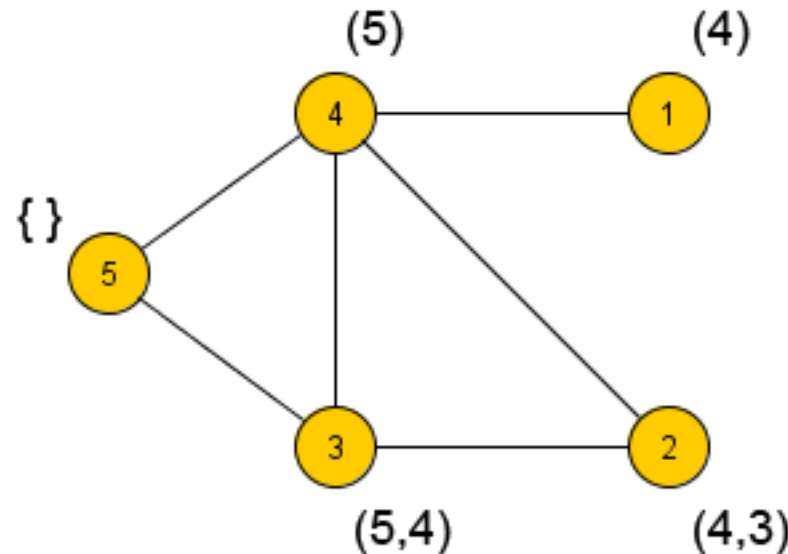


Lexikographische Breitensuche & perfekte Eliminationsordnung



Lexikographische Breitensuche & perfekte Eliminationsordnung

- Wir erhalten die Knotenmarkierung $\sigma^{-1} := (5, 4, 3, 2, 1)$

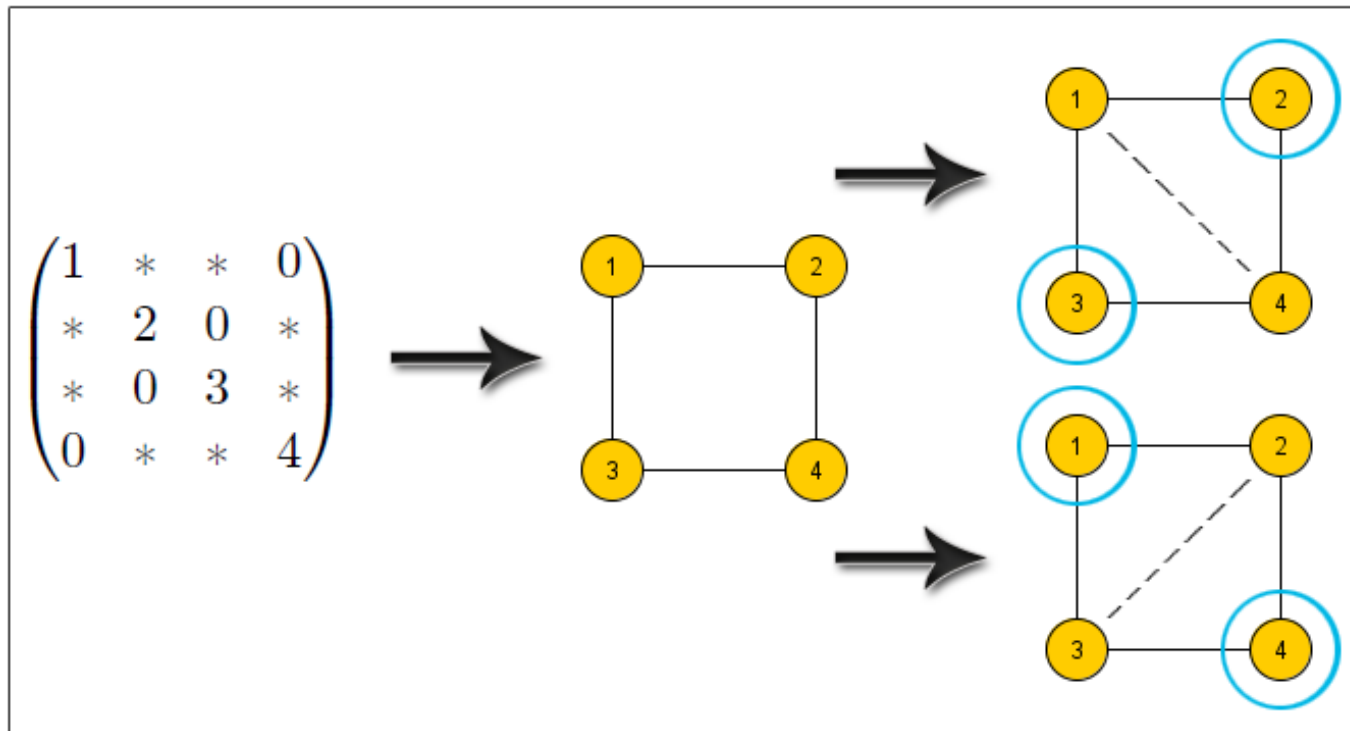


- Damit ergibt sich folgende Reihenfolge für die Elimination:
(4), (4,3), (5,4), (5) und schließlich: {}



Lexikographische Breitensuche & perfekte Eliminationsordnung

- Was passiert, wenn die Lex.-BFS auf einen nicht chordalen Graphen angewendet wird...?



Effizienz: chordale Graphen

- Es wurde erwähnt das es mittels lex. Breitensuche möglich ist, ein Chordalitätstest in linearer Zeit $O(|V| + |E|)$ durchzuführen

→ bei perfekten Graphen bisher nur in polynom. Laufzeit...

- In chordalen Graphen lassen sich darüber hinaus andere wichtige Parameter wie z.B.:

- minimale Färbungen
- maximale Cliques

ebenfalls in linearer Zeit berechnen !

→ Bei anderen Graphen stellt dies wieder ein **NP**-vollständiges Problem



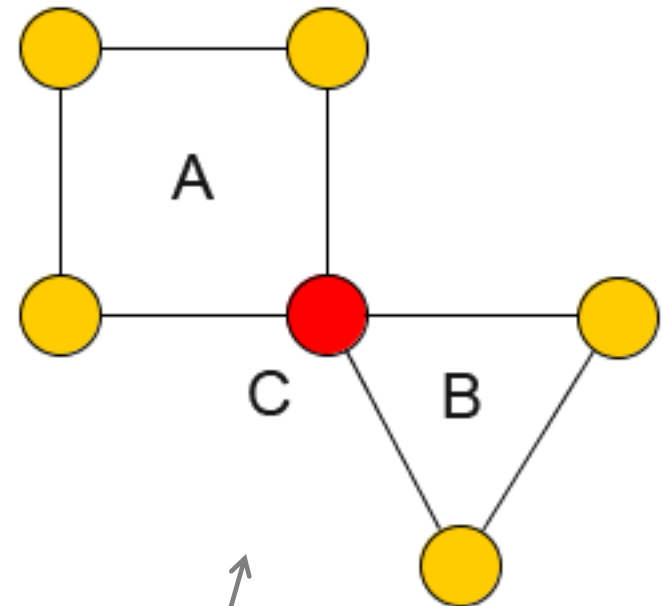
Chordale und zerlegbare Graphen



Seien A, B, C nichtleere, disjunkte Teilmengen von V eines Graphen $\mathbf{G} = (V, E)$ und sei $V = A \cup B \cup C$

Dann ist (A, B, C) eine exakte Zerlegung von \mathbf{G} falls folgende Bedingungen gelten:

- ✓ A und B sind nicht leer
- ✓ C ist vollständige Untermenge von V
- ✓ C separiert A und B in \mathbf{G}
- ✓ $A \cup C$ und $B \cup C$ sind zerlegbar

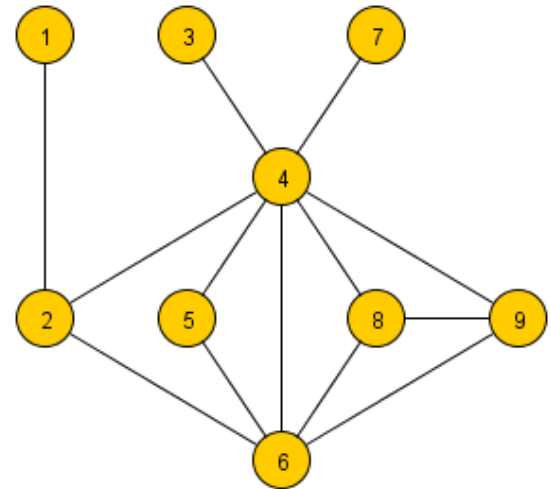


→ wir betrachten hier ein generellen Fall,
der abgebildete Graph ist **nicht chordal** !



Chordale und zerlegbare Graphen

- Für einen Graphen $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ bezeichnet man einen Separator \mathbf{S} in \mathbf{V} als minimal, wenn keine echte Teilmenge von \mathbf{S} ein Separator für \mathbf{G} darstellt
- Zudem bezeichnet man \mathbf{S} als $\mathbf{u-v}$ Separator für zwei Knoten \mathbf{u}, \mathbf{v} in \mathbf{V} , wenn \mathbf{u} und \mathbf{v} in verschiedenen Zusammenhangskomponenten von $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ wobei $\mathbf{V} \setminus \mathbf{S}$ liegen...
- Beispiel: Die minimalen Separatoren lauten hierfür: $\{2\}$, $\{4\}$ und $\{4,6\}$



Chordale und zerlegbare Graphen



Sei \mathbf{G} ein ungerichteter Graph, dann sind folgende Aussagen äquivalent...

- (1) \mathbf{G} ist chordal
- (2) \mathbf{G} ist rekursiv simplizial & zerlegbar
- (3) \mathbf{G} hat höchstens eine PEO
- (4) \mathbf{G} besitzt ein Durchschnittsgraph
- (5) Jeder minimale Knotenseparator $\mathbf{u-v}$ induziert ein vollständigen Untergraph von \mathbf{G}

→ Bei (5) anders ausgedrückt: **Jede minimale $u-v$ trennende Menge ist eine Clique**



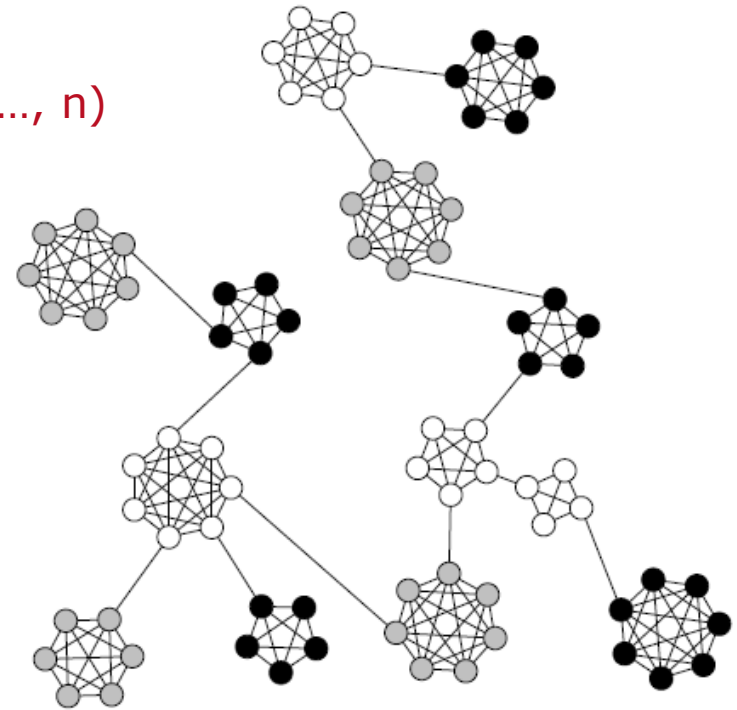
Chordale Graphen vs. Bäume

- Ein Graph \mathbf{G} ist genau dann chordal, wenn \mathbf{G} ein Baumgraph ist
 - Baumgraphen sind definiert als Durchschnittsgraphen von Bäumen in einem Baum
- Chordale Graphen sind genau die Durchschnittsgraphen von Unterbäumen eines Baumes
- Unterbäume eines Baumes $\mathbf{T} = (\mathbf{V}, \mathbf{E})$ sind dabei zusammenhängende Untergraphen $\mathbf{T}_i = (\mathbf{V}_i, \mathbf{E}_i)$ ($i = 1, \dots, n$) von \mathbf{T}
 - Diese sind daher selbst wieder Bäume



Chordale Graphen als Durchschnittsgraphen von Bäumen

- Ein Durchschnittsgraph $\mathbf{G}_T = (\mathbf{V}_T, \mathbf{E}_T)$ besteht aus der Knotenmenge $\mathbf{V}_T = \{v_i^T \mid i = 1, \dots, n\}$
 - wobei jedes Element den Unterbaum: T_i zugeordnet wird (für $i = 1, \dots, n$)
- sowie der Kantenmenge $\mathbf{E}_T = \{ \{v_i, v_j\} \mid \mathbf{V}_i \cap \mathbf{V}_j = \emptyset \}$
- Durchschnittsgraphen finden in der Literatur häufig auch andere Bezeichnungen, z.B.: Cliquenbaum, Intersection Graph, etc...

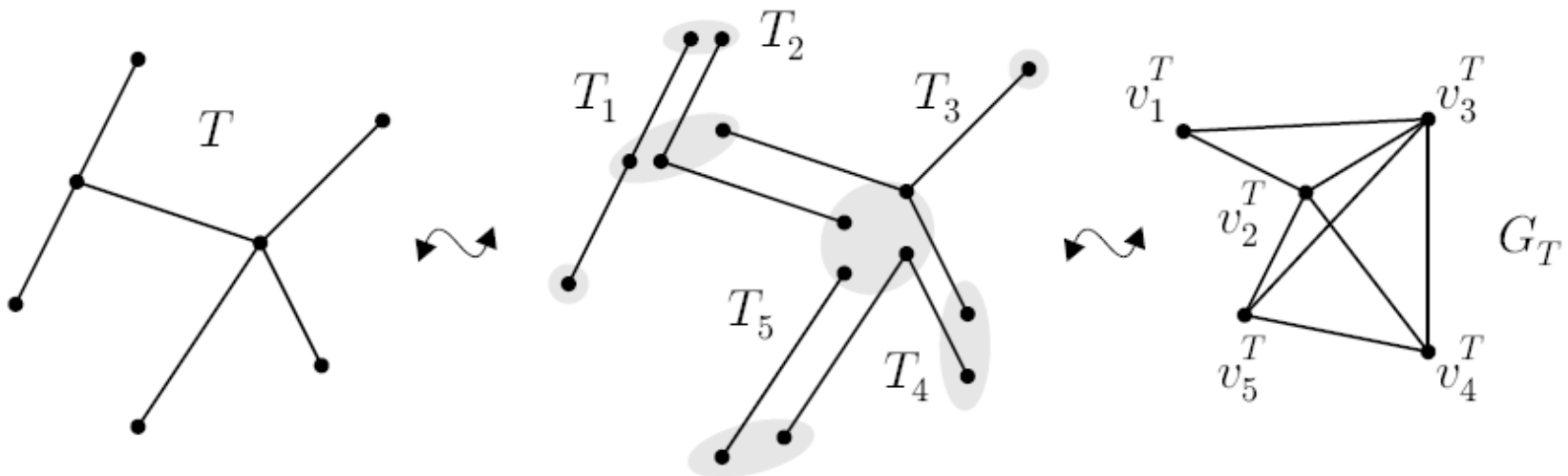


• ...



Chordale Graphen als Durchschnittsgraphen von Bäumen

- Ein Durchschnichtsbaum/Cliquenbaum wird durch eine Wurzel ausgezeichnet, wobei diese den größten Knotenmarkierung in der PEO darstellt...



Chordale Graphen...Schluß für heute



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Chordale Graphen beinhalten viele weitere Eigenschaften, die sie zu einer mächtigen Graphenklasse machen
- Alles abzudecken würde den Rahmen dieser Arbeit sprengen, aus diesem Grunde seien die Interessierten auf die genannte Literatur verwiesen

Vielen Dank für eure Aufmerksamkeit



Quellen:

- [**Algorithmic Graph Theory and Perfect Graphs**]
Prof. Martin Charles Golumbic, Department Computer at the
University of Haifa, Israel
- [**Perfekte Graphen, Vorlesung → Kapitel 10**]
Professor R. Schrader, Lehrveranstaltung: Graphentheorie,
Zentrum für Angewandte Informatik Köln, Germany
- [**Direkte Verfahren für dünnbesetzte Matrizen**]
Olaf Schenk, Departement Informatik,
Universität Basel, Schweiz



Quellen:

- [**Sparse LU Factorization**]
Mit freundlicher Genehmigung von: Prof. Uwe Naumann,
LuFG Software & Tools for Computational Engineering
RWTH Aachen University, Germany
- [**Erkennen von Graphenklassen mittels lexikographischer Breitensuche**] Mathias Biermann, Diplomarbeit,
Fern Universität Hagen, Germany
- [**Minimum Fill-In and Treewidth for Graphs Modularly Decomposable into Chordal Graphs**] PD Dr. Elias Dahlhaus,
Department of Computer Science, University of Bonn, Germany



Quellen:

- [**Efficient Graph Representations**] Jeremy P. Spinrad,
American Mathematical Society, Providence- Rhode Island, USA
- [**Introduction To Parallel Algorithms**]
C. Xavier, Sundararaja S. Iyengar, USA

