Abschlussvortrag

Effiziente Extraktion von Kuratowski-Teilgraphen

Jens Schmidt

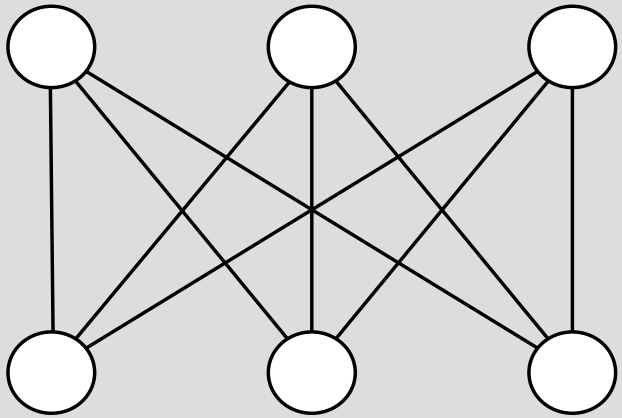
Übersicht

1. **Einleitung**
2. Der Boyer-Myrvold Planaritätstest
3. Erweiterungen
4. Experimentelle Ergebnisse

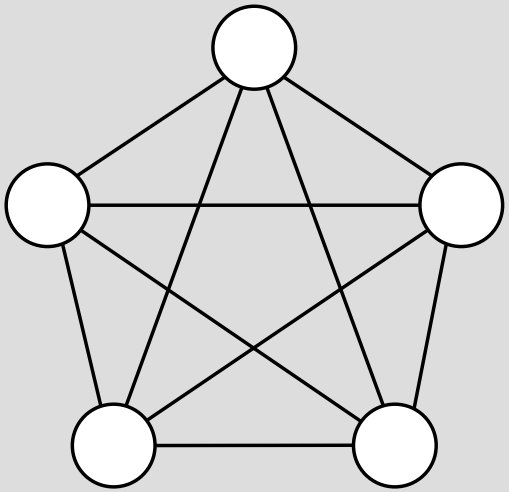
Planarität

- **Definition:**
  Ein Graph G=(V,E) ist genau dann planar, wenn er sich überkreuzungsfrei in die Ebene einbetten lässt.

- **Nicht planare Graphen**:



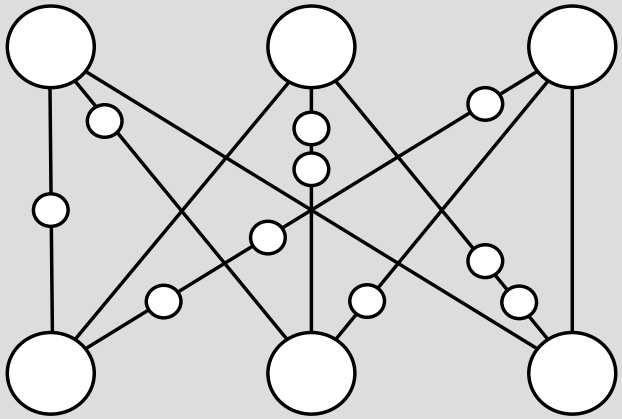
$K_{3,3}$



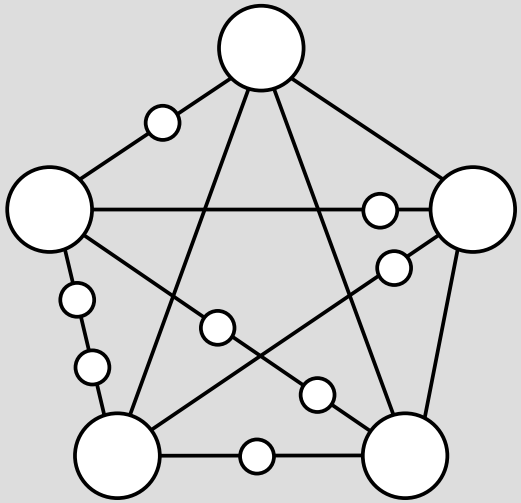
K_5

(Beweis: Eulerscher Polyedersatz)

Kuratowski-Subdivisions



$K_{3,3}$-Subdivision



K_5-Subdivision

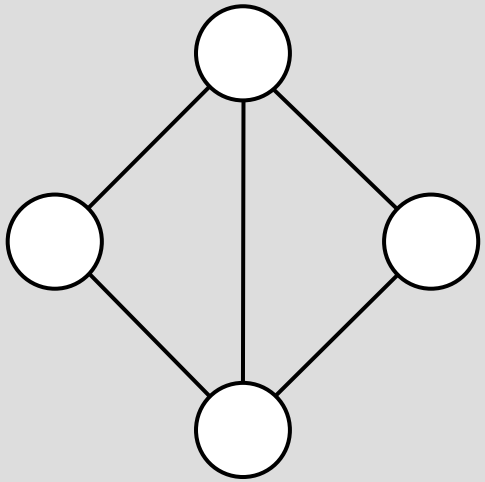
Satz von Kuratowski (1930):

Ein Graph ist genau dann planar, wenn er weder eine **$K_{3,3}$-** noch **K_5**-Subdivision enthält.

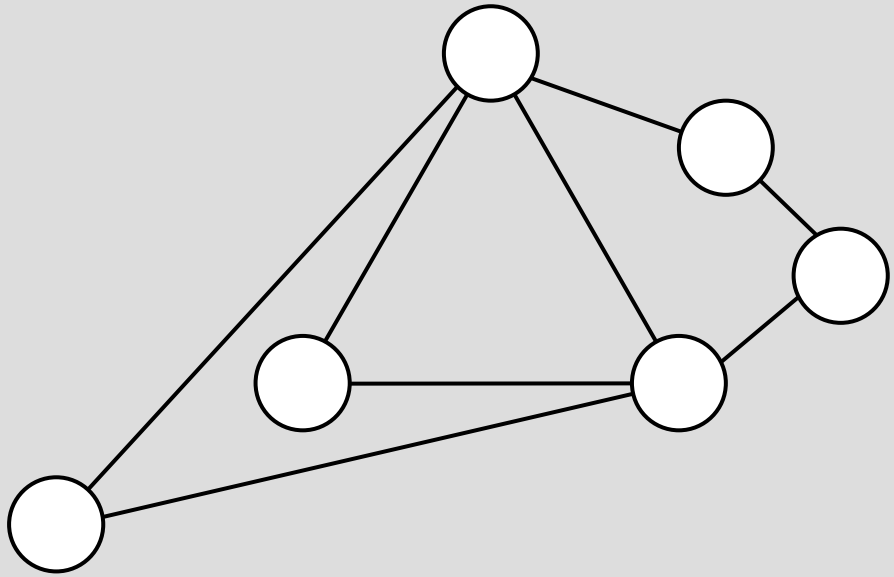
Minoren

Ähnlich: Minoren

- G,H sind Graphen
- G' entsteht durch eine Folge von Kantenkontraktionen aus G
- H ist Minor von G, falls H Subgraph von G' ist.



H



G

Übersicht

1. Einleitung
2. **Der Boyer-Myrvold Planaritätstest**
3. Erweiterungen
4. Experimentelle Ergebnisse

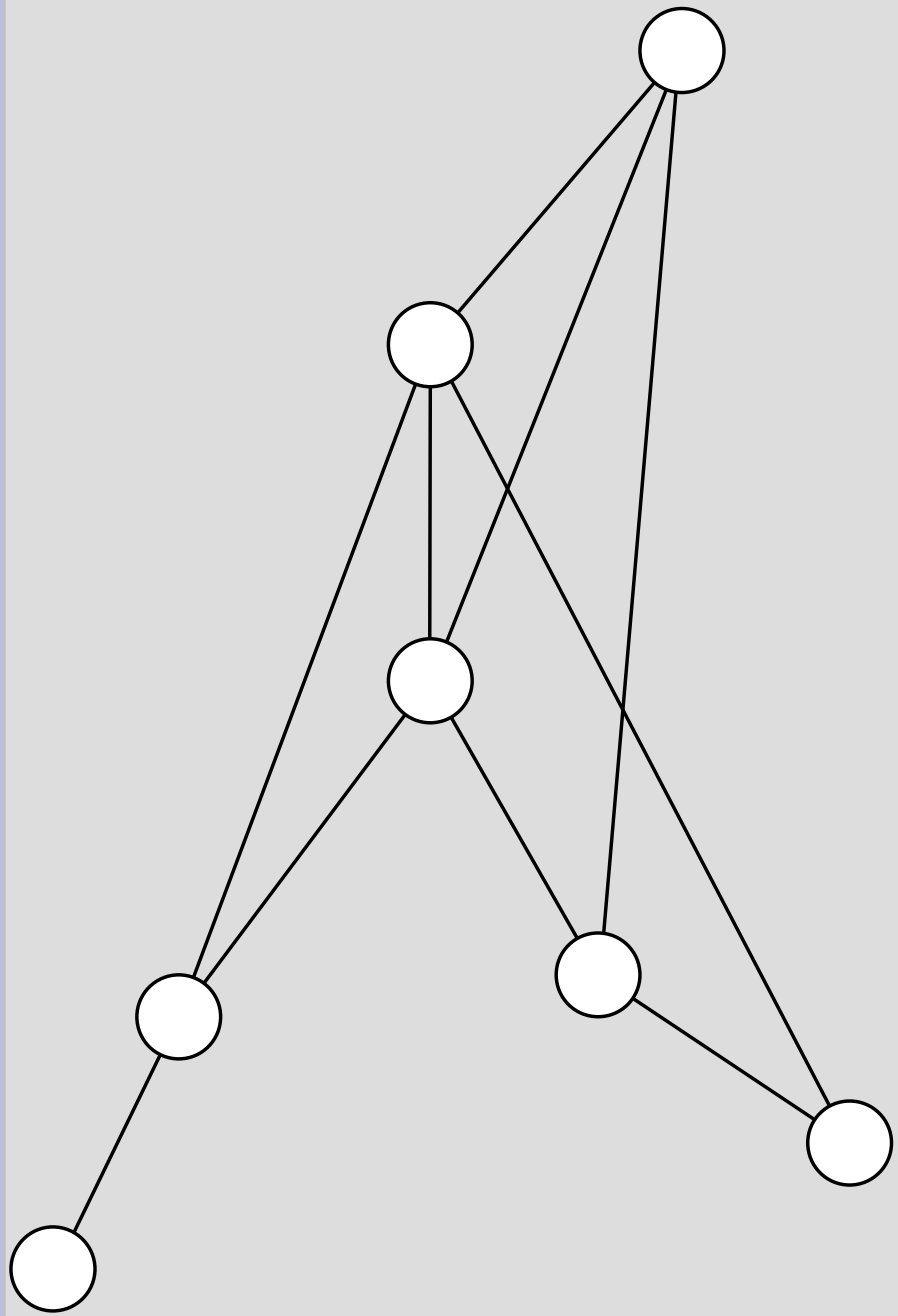
Boyer-Myrvold Planaritätstest

- Basiert auf
  „On the Cutting Edge: Simplified O(n) planarity by edge addition“ (2004)

- Vorteile:
  - Liefert planare Einbettung oder Kuratowski-Subdivison
  - Linear in O(n), sehr kleiner konstanter Faktor
  - Eingabegraph muss nicht zusammenhängend sein

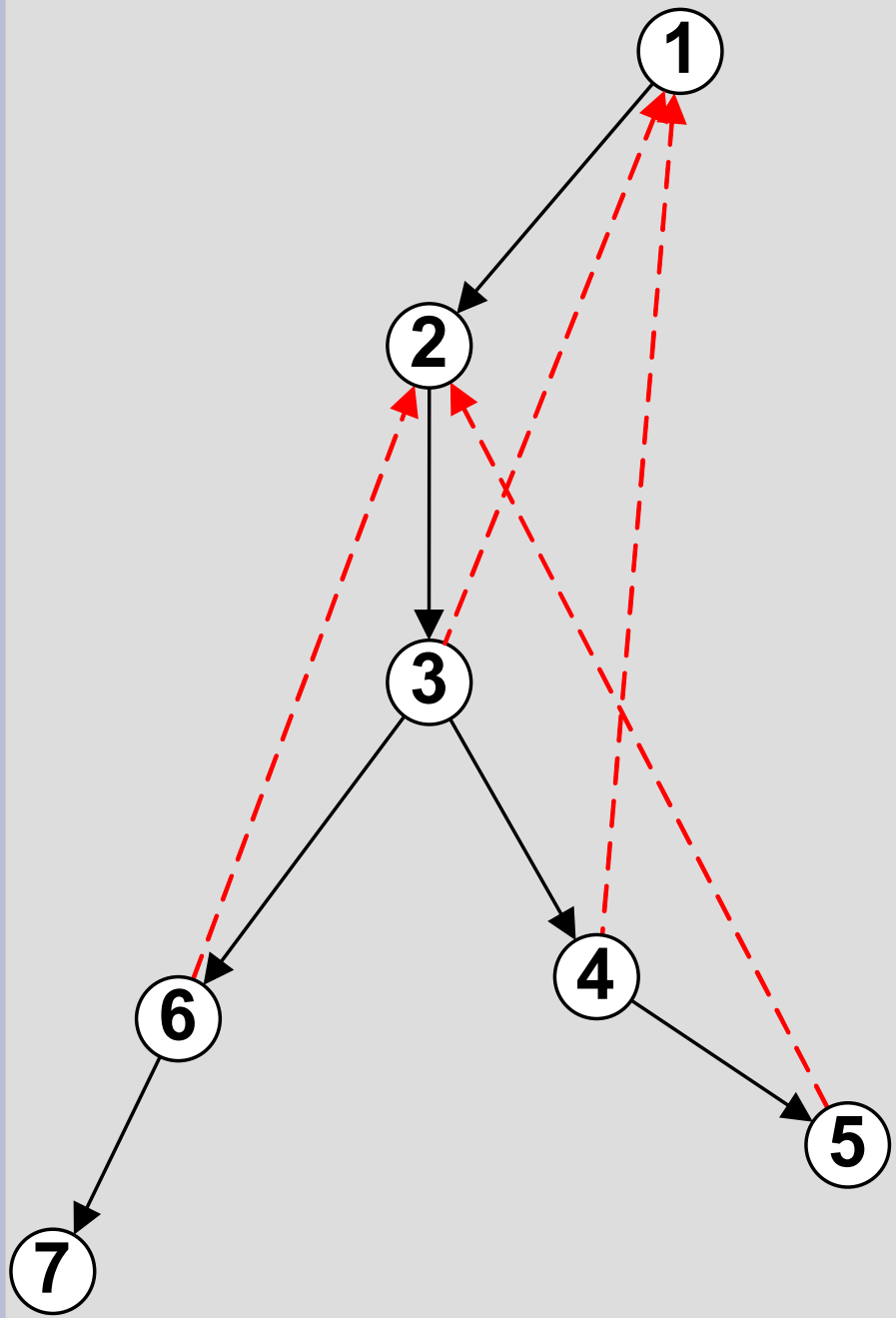
- Nachteile:
  - Trotz aller Bemühungen noch immer komplex

Ein Beispiel



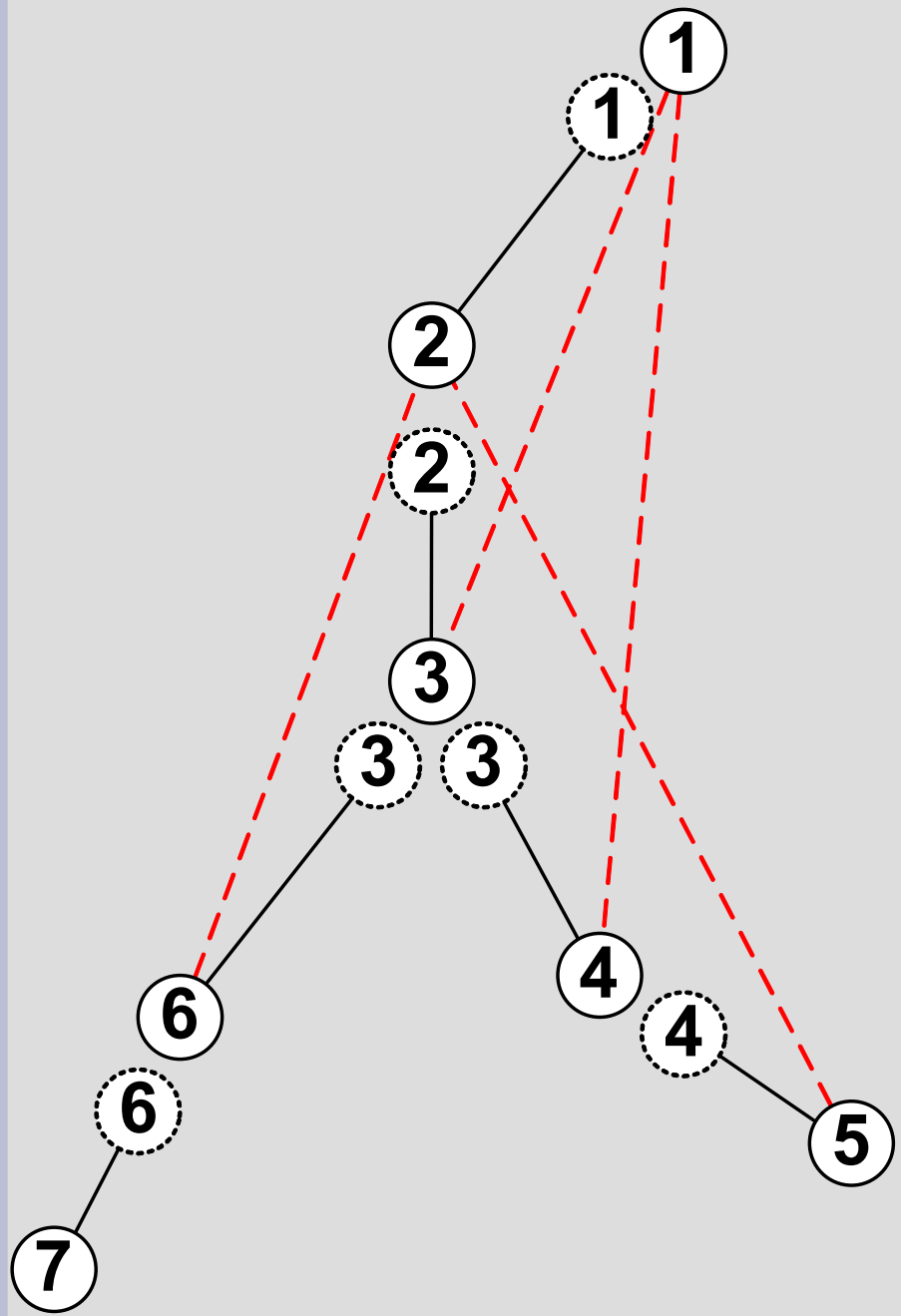
Der Eingabegraph

DFS-Baum



- Grundlage DFS-Baum (allg. Wald)
  - DFS-Kanten
  - Noch nicht eingebettete Backedges
- Multikanten und Selfloops werden entfernt
- Ersetze gerichtete Kanten durch ungerichtete

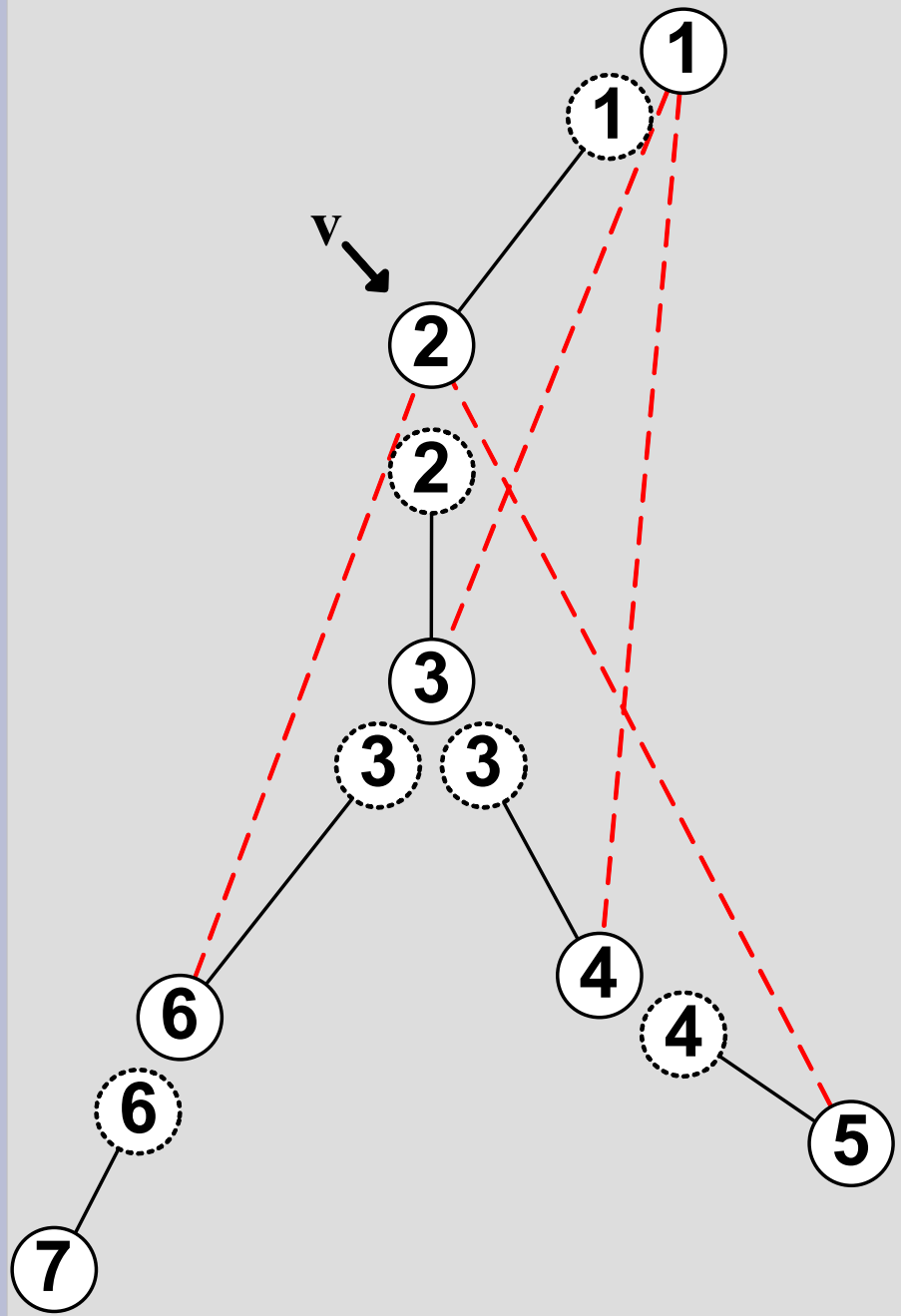
Virtuelle Knoten



Für jede DFS-Kante wird **virtueller Knoten** eingefügt:

- Baum von Bicomps
- Backedges werden dabei ignoriert

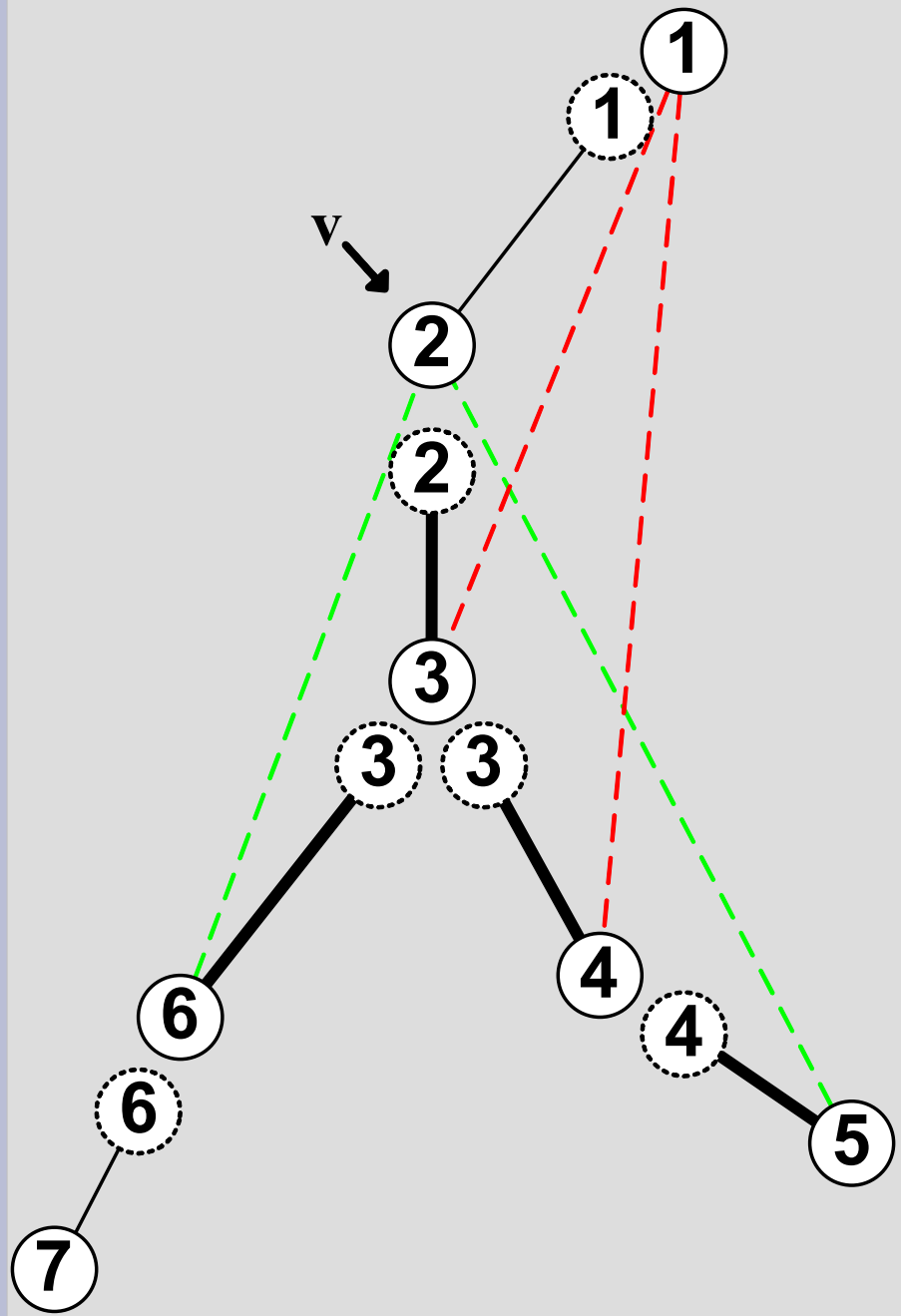
Äußere Schleife



Iteration über alle Knoten v in absteigender DFS-Reihenfolge:

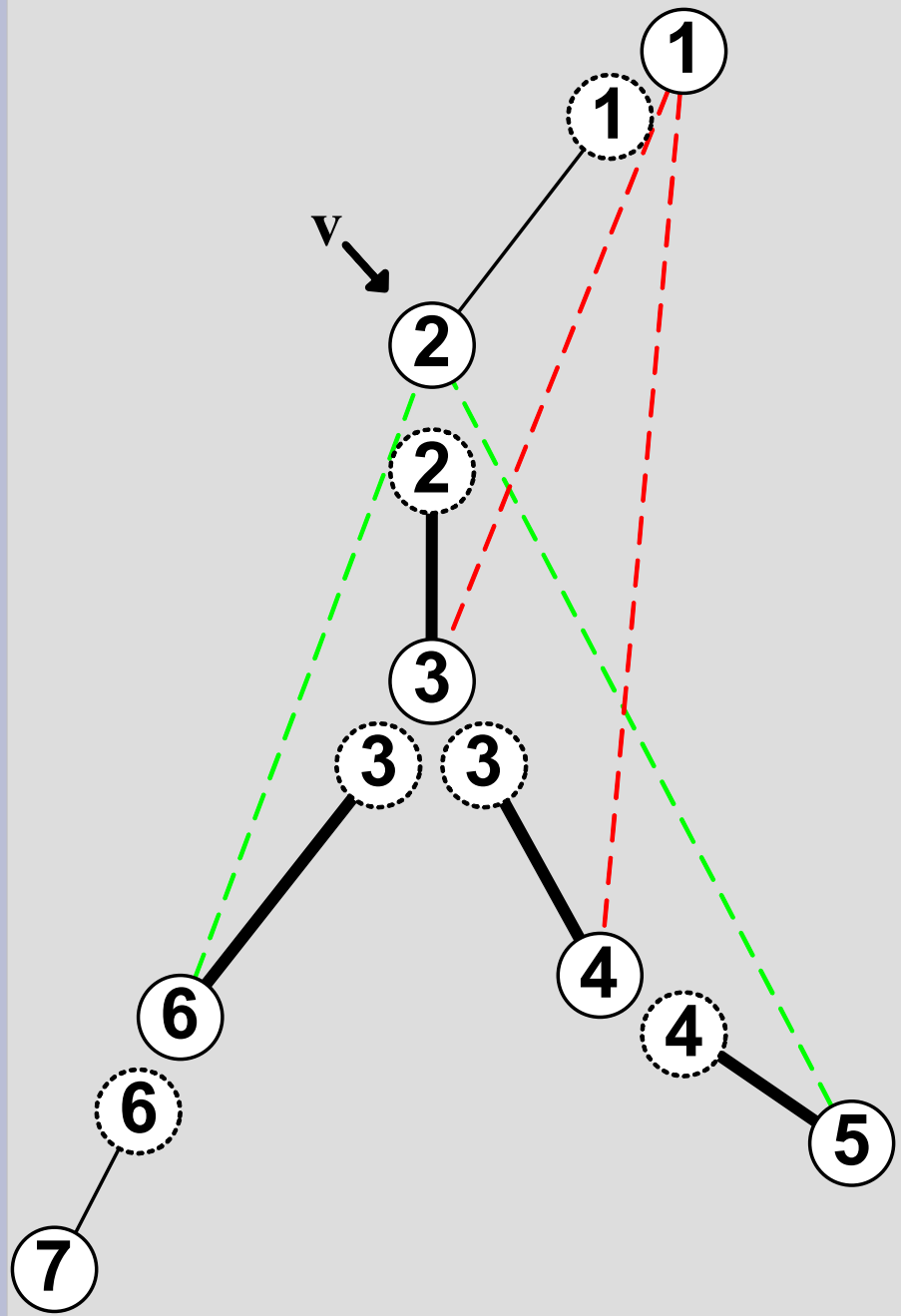
- Falls v Endpunkt einer Backedge: Bette diese ein
- Knoten 6,5,4,3: Keine Endpunkte von Backedges

Walkup



- Knoten 2:
  Starte „**Walkup**“ für jede Backedge, um den involvierten Teilgraphen für die Einbettung abzugrenzen
- Traversiere dazu die Backedges und dann alle Vaterbicomps bis zum Knoten 2 zurück

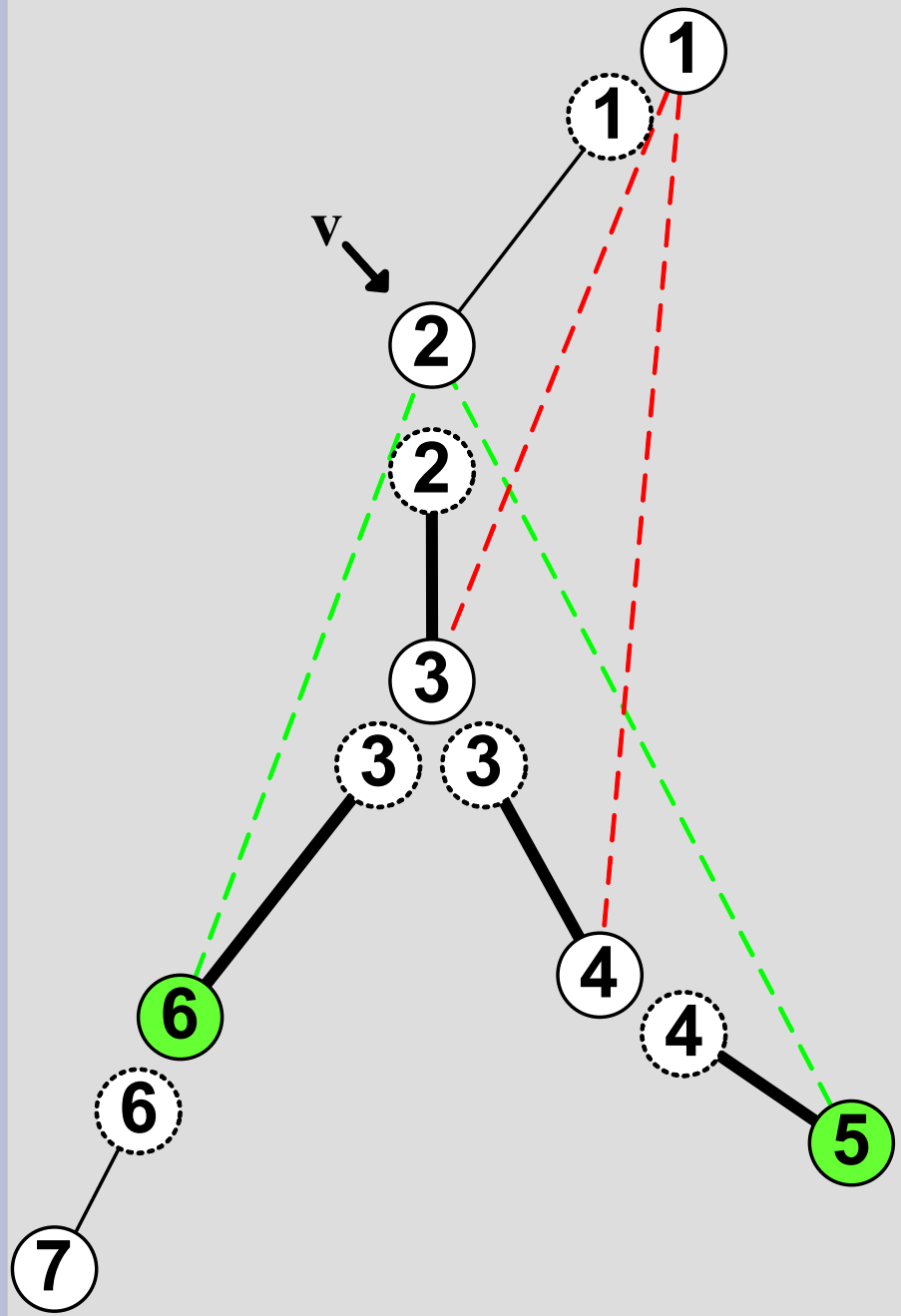
pertinent vs. extern



Ein Knoten **w** ist

- pertinent, wenn er oder ein Knoten aus einer anliegenden Kinderbicomp mit **v** verbunden ist.

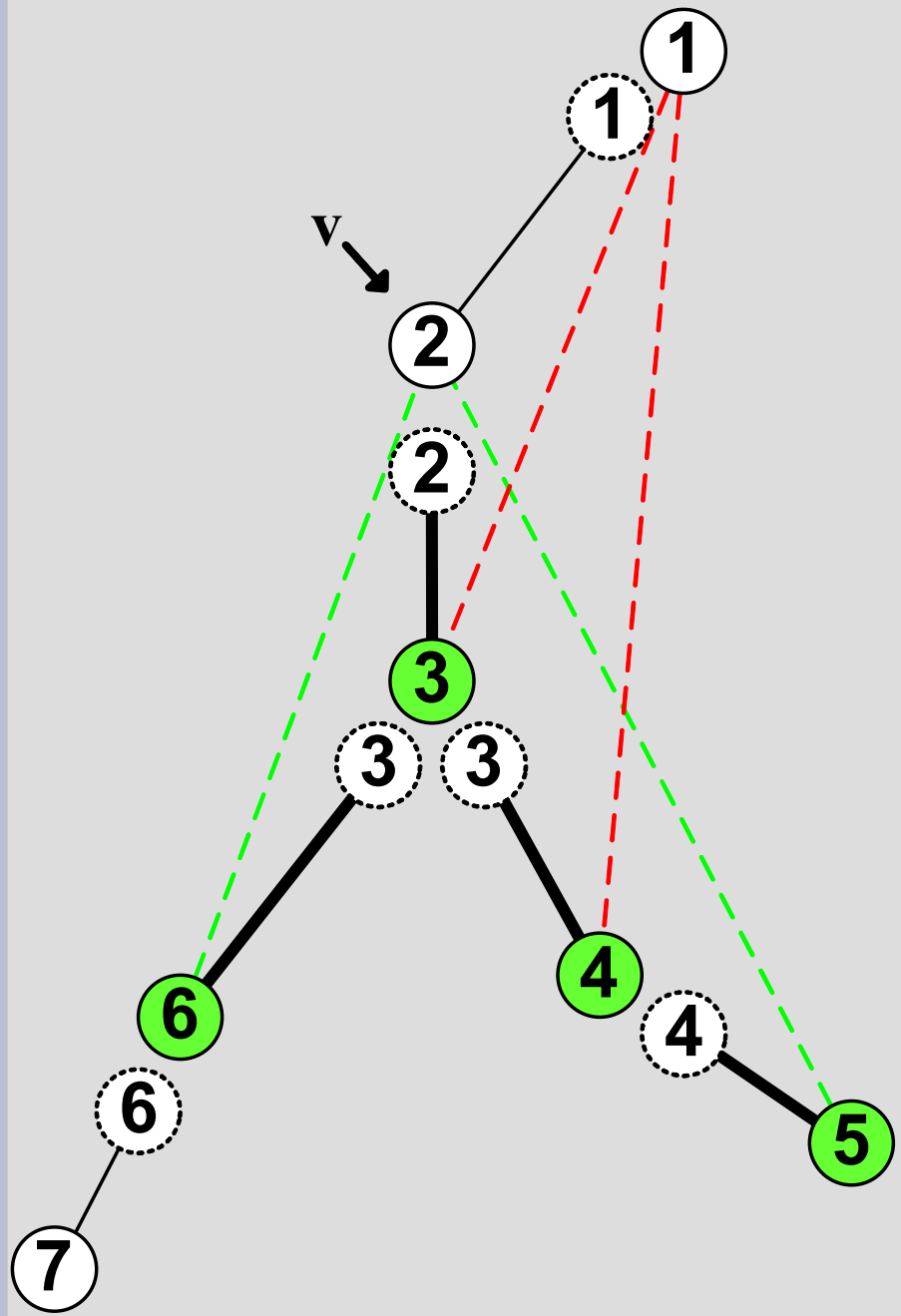
pertinent vs. extern



Ein Knoten **w** ist

- pertinent, wenn er oder ein Knoten aus einer anliegenden Kinderbicomp mit **v** verbunden ist.

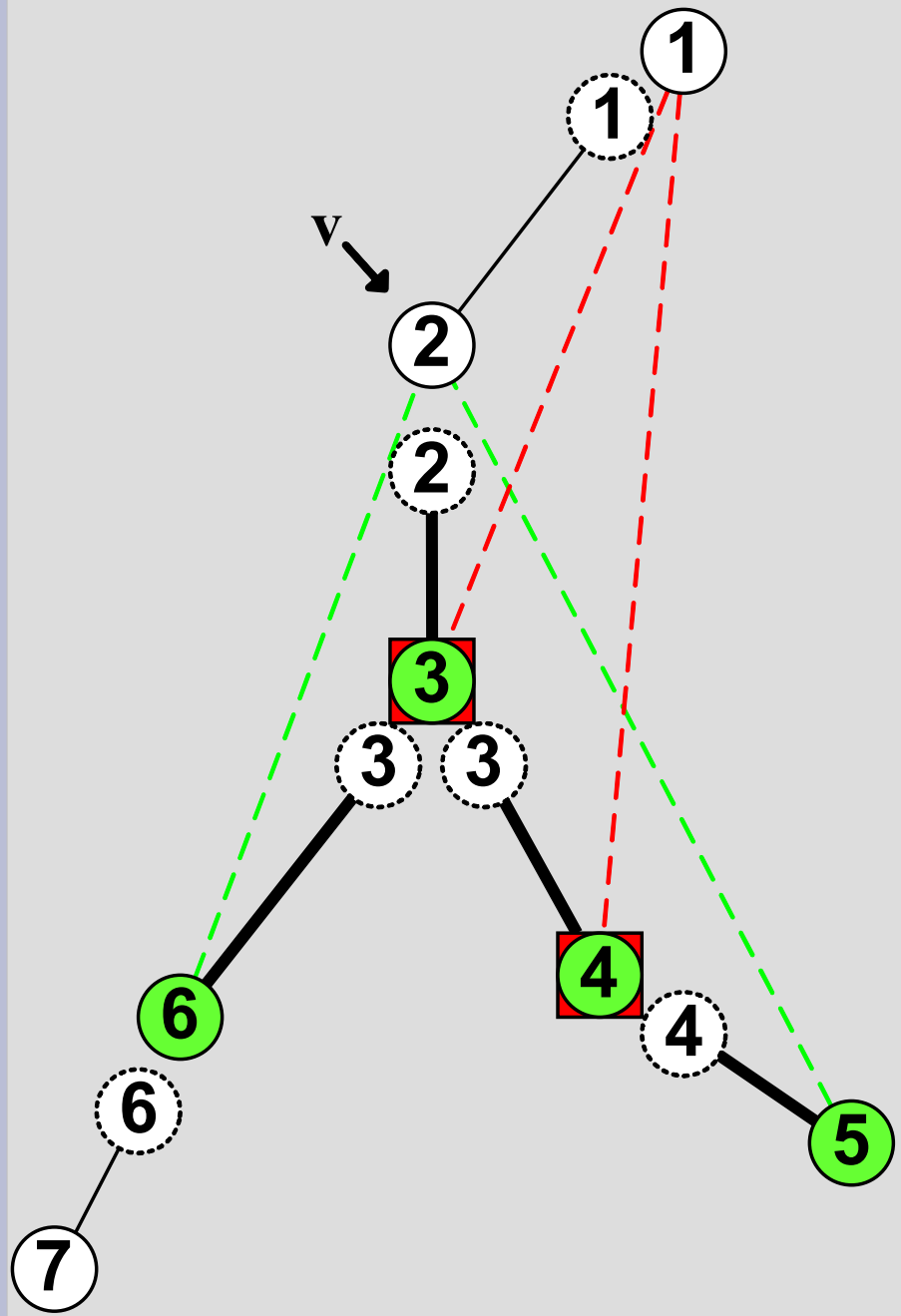
pertinent vs. extern



Ein Knoten **w** ist

- pertinent, wenn er oder ein Knoten aus einer anliegenden Kinderbicomp mit v verbunden ist.
- extern, wenn er oder ein Knoten aus einer anliegenden Kinderbicomp mit einem Knoten mit DFI < v verbunden ist.

pertinent vs. extern

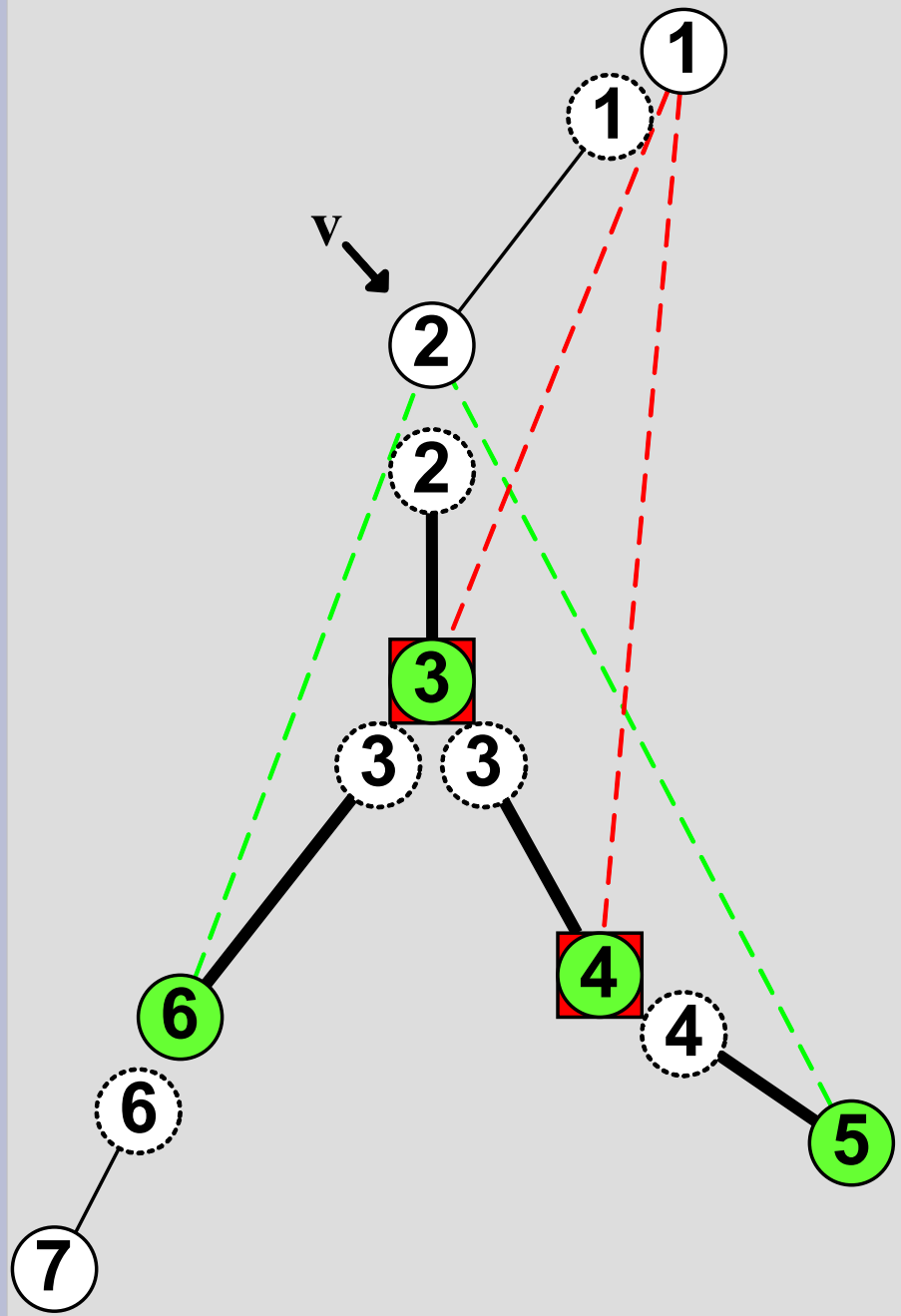


Ein Knoten **w** ist

- pertinent, wenn er oder ein Knoten aus einer anliegenden Kinderbicomp mit v verbunden ist.
- extern, wenn er oder ein Knoten aus einer anliegenden Kinderbicomp mit einem Knoten mit DFI < v verbunden ist.

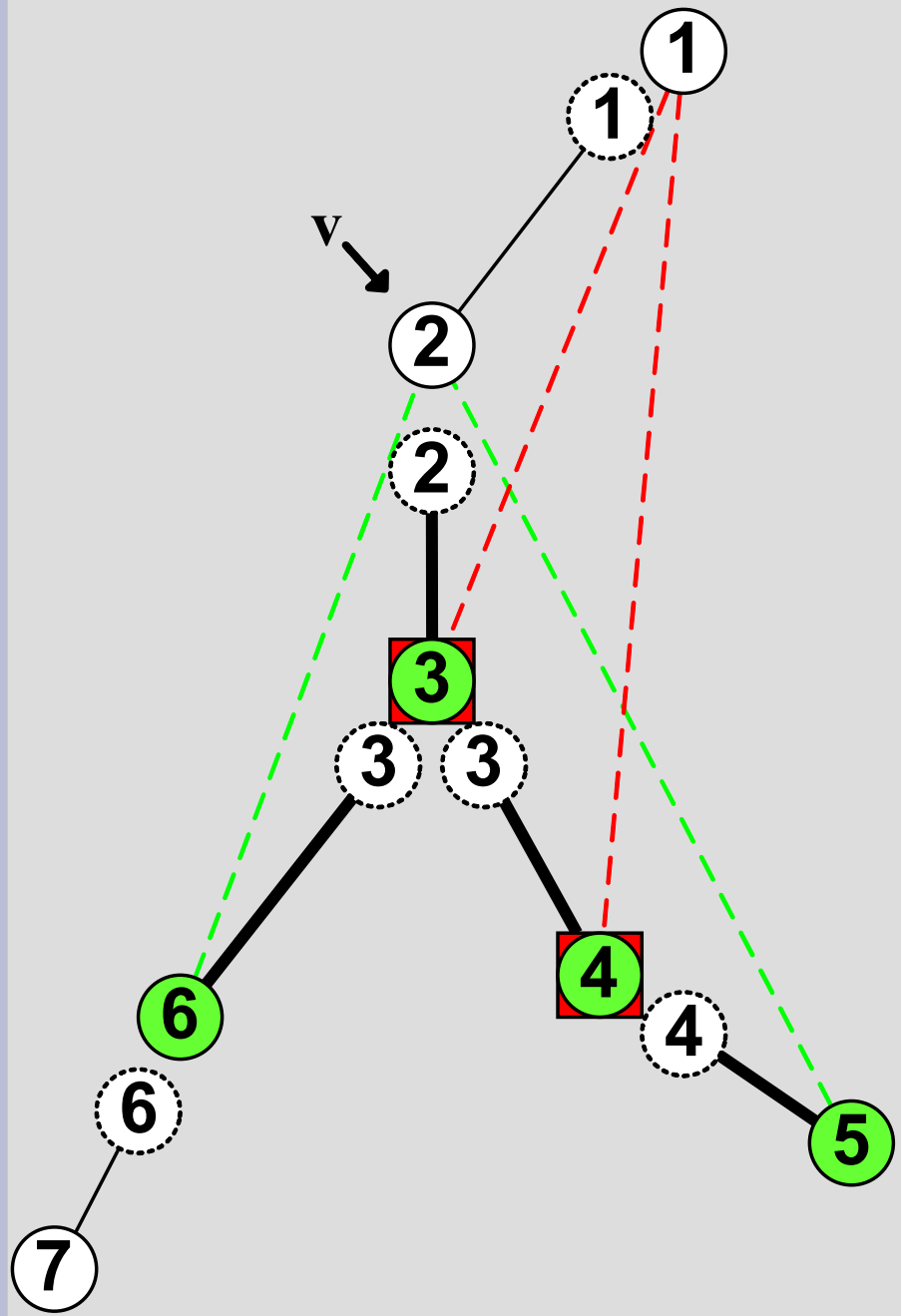
- Eine Bicomp heißt pertinent bzw. extern, wenn sie mindestens einen pertinenten bzw. externen Knoten enthält.
- Klassifikation eines Knotens in O(1) mit dynamischen Datenstrukturen

Walkdown



- Zwei Walkdowns für jede Kinderbicomp von v: **CCW** und **CW**
- Jede Bicomp wird auf ihrer **external Face** in aktueller Richtung traversiert

Walkdown

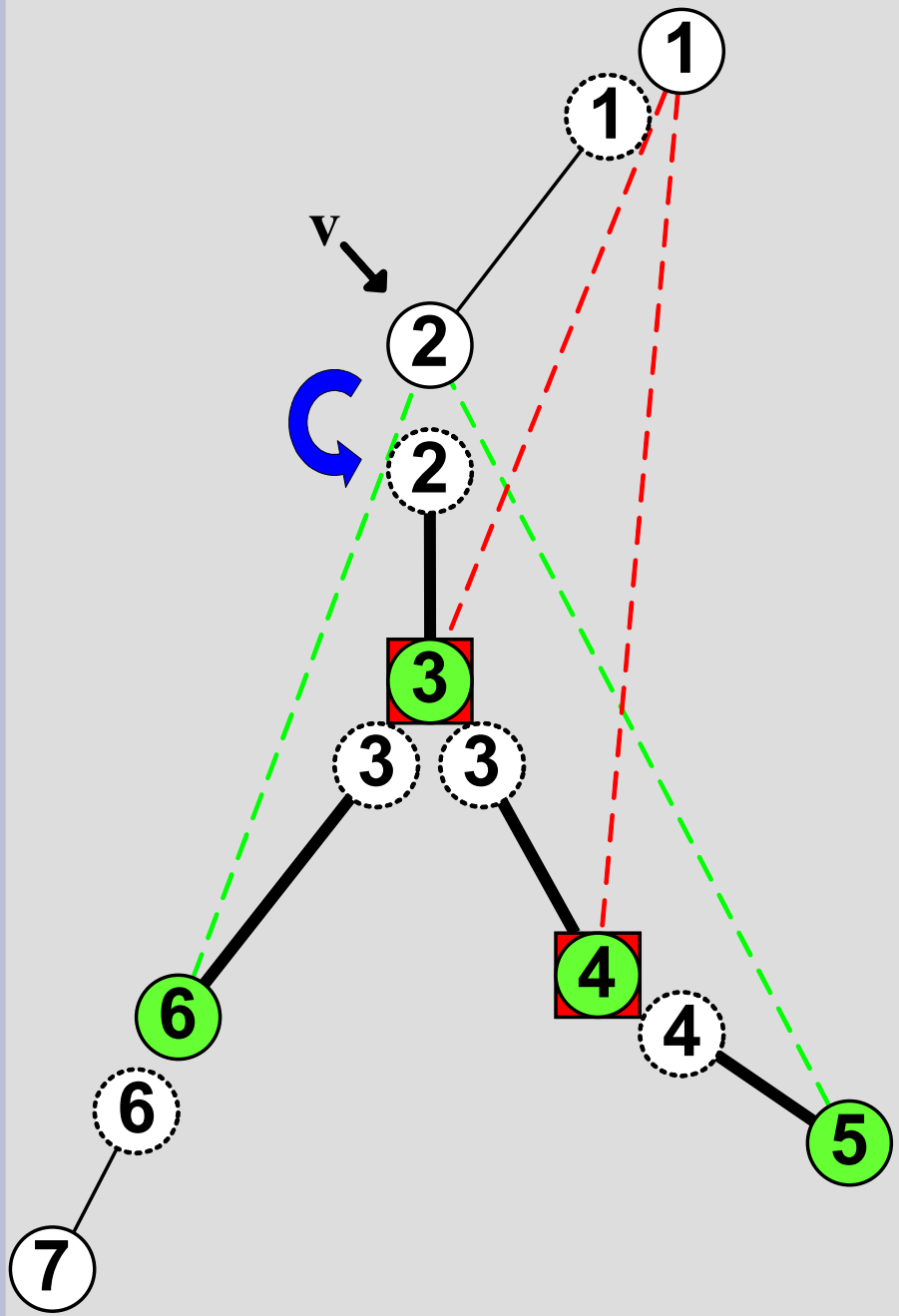


- Zwei Walkdowns für jede Kinderbicomp von v: **CCW** und **CW**
- Jede Bicomp wird auf ihrer **external Face** in aktueller Richtung traversiert

- Regel für jeden erreichten Knoten **w**:
  - Bette alle Backedges zu v ein
  - Steige zu Bicomp-Kind ab (wenn vorhanden zu nicht-externem)
  - Stoppe, falls **w** extern und nicht pertinent ist.
  - Traversiere weiter bis **w**=v.

- Steige ab zu 2, traversiere zu **3**. Wähle linke 3, (nicht rechte, da extern), traversiere zu **6**.
- Backedge gefunden.

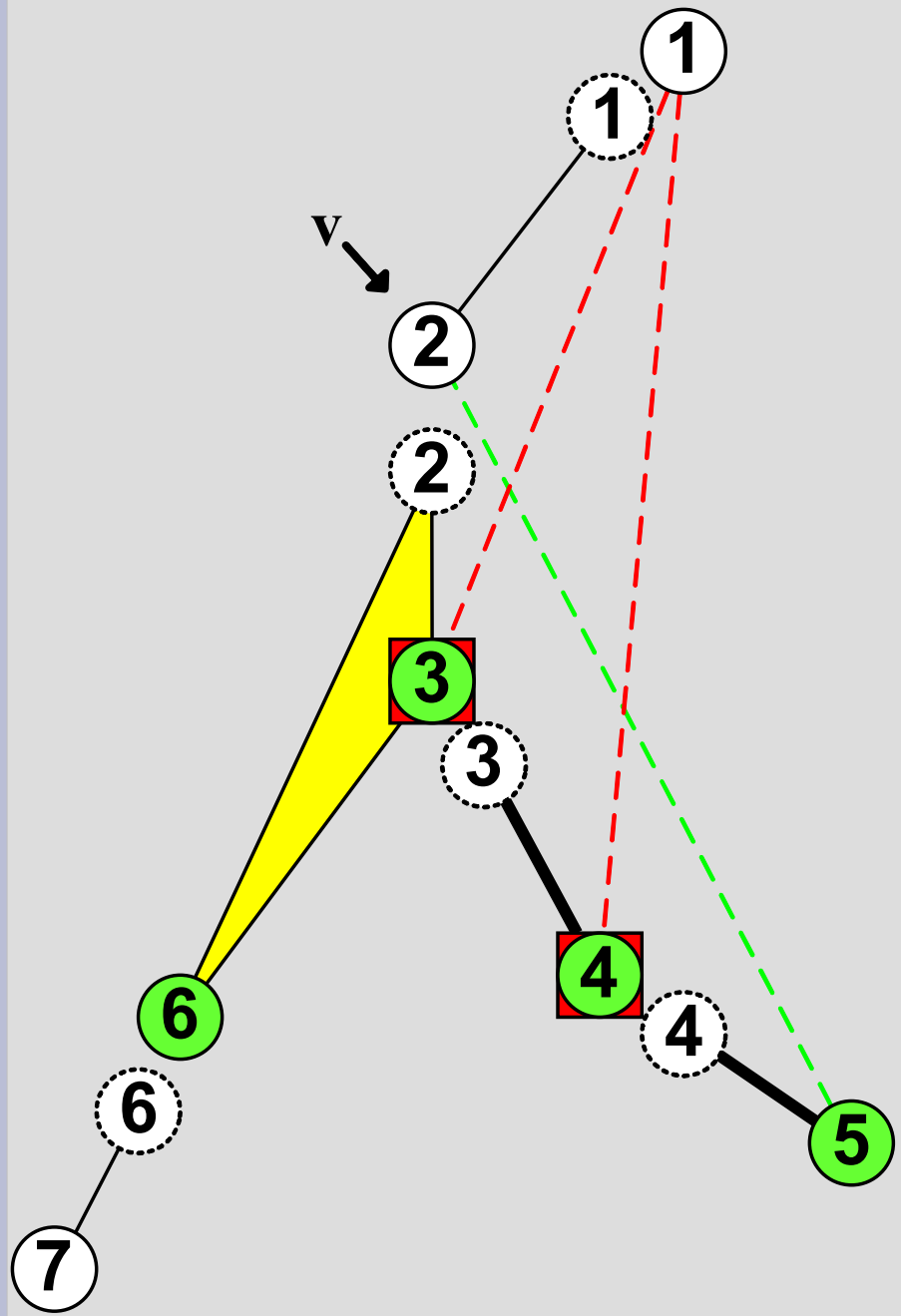
Walkdown: Einbettungen



Die Backedge wird jetzt eingebettet:

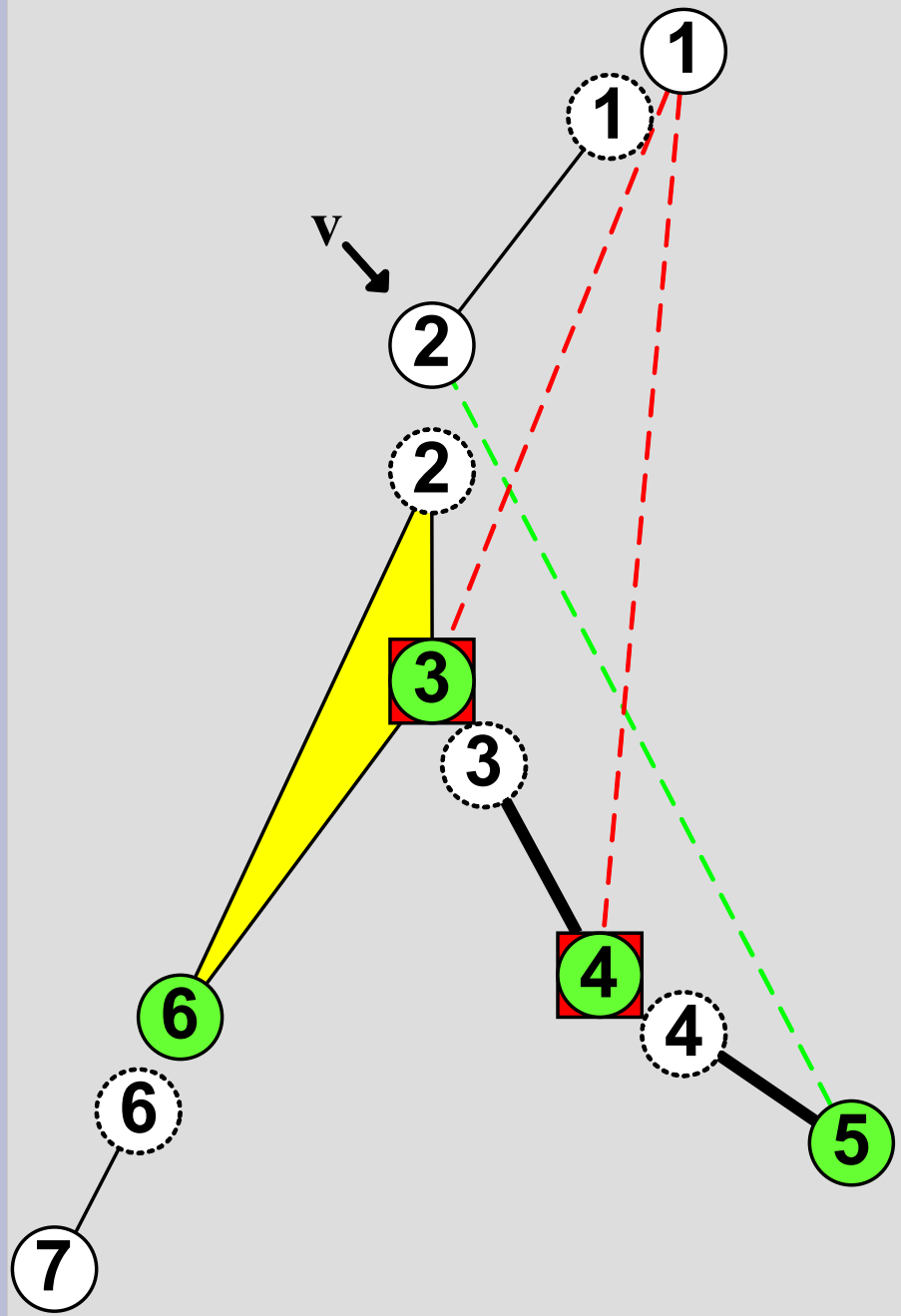
- Traversierungsrichtung ist momentan **CCW**, die Backedge wird immer auf derselben Seite eingebettet:
  - Im Knoten **6** wird sie also gegen den Uhrzeigersinn eingebettet.
  - Im Knoten 2 umgekehrt.
- Alle traversierten Bicomps werden nun verschmolzen

Walkdown: Einbettungen



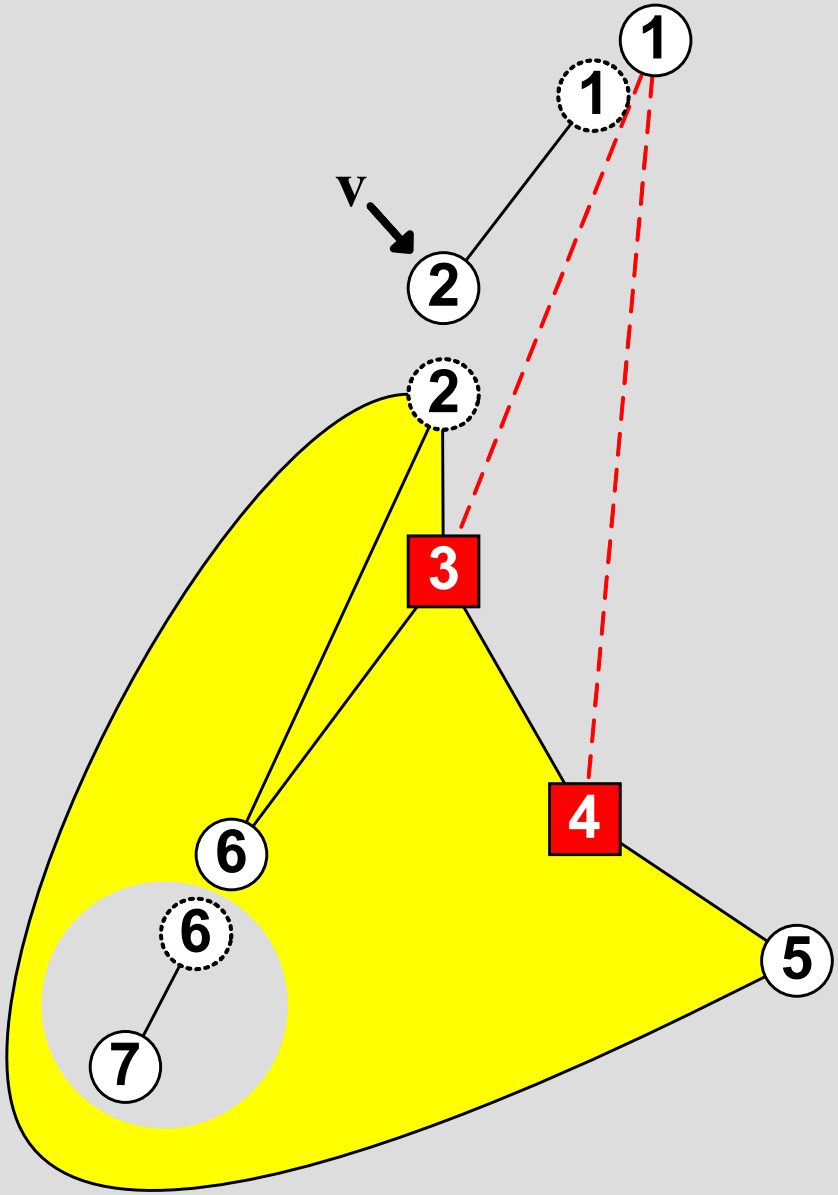
- Bei jeder Einbettung entsteht eine neue, größere Bicomp.

Walkdown



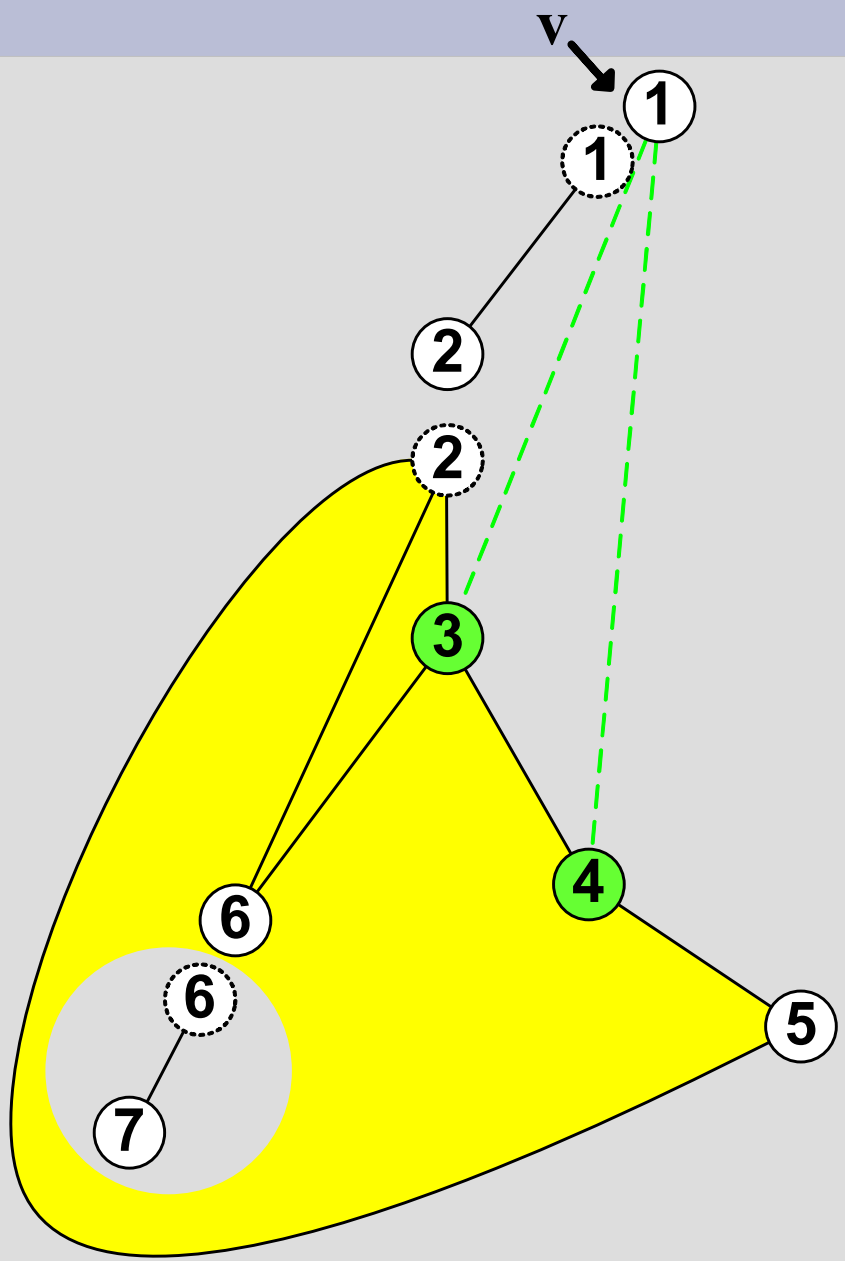
- Wir sind noch immer am Knoten **6** in Richtung CCW.
- Der nächste Knoten an der **external Face** ist **3** und extern.
- An externen Knoten wird gestoppt. Ein weiterer Abstieg zu einer pertinenten Kinderbicomp ist aber möglich.
- Traversiere zu 3, **4**, 4, **5.**

Walkdown



- Bette Backedge an **5** ein. Verschmelze dabei involvierte Bicomps.
- Zweiter Walkdown im Uhrzeigersinn stoppt sofort an **3**. Ende für **v=2**.

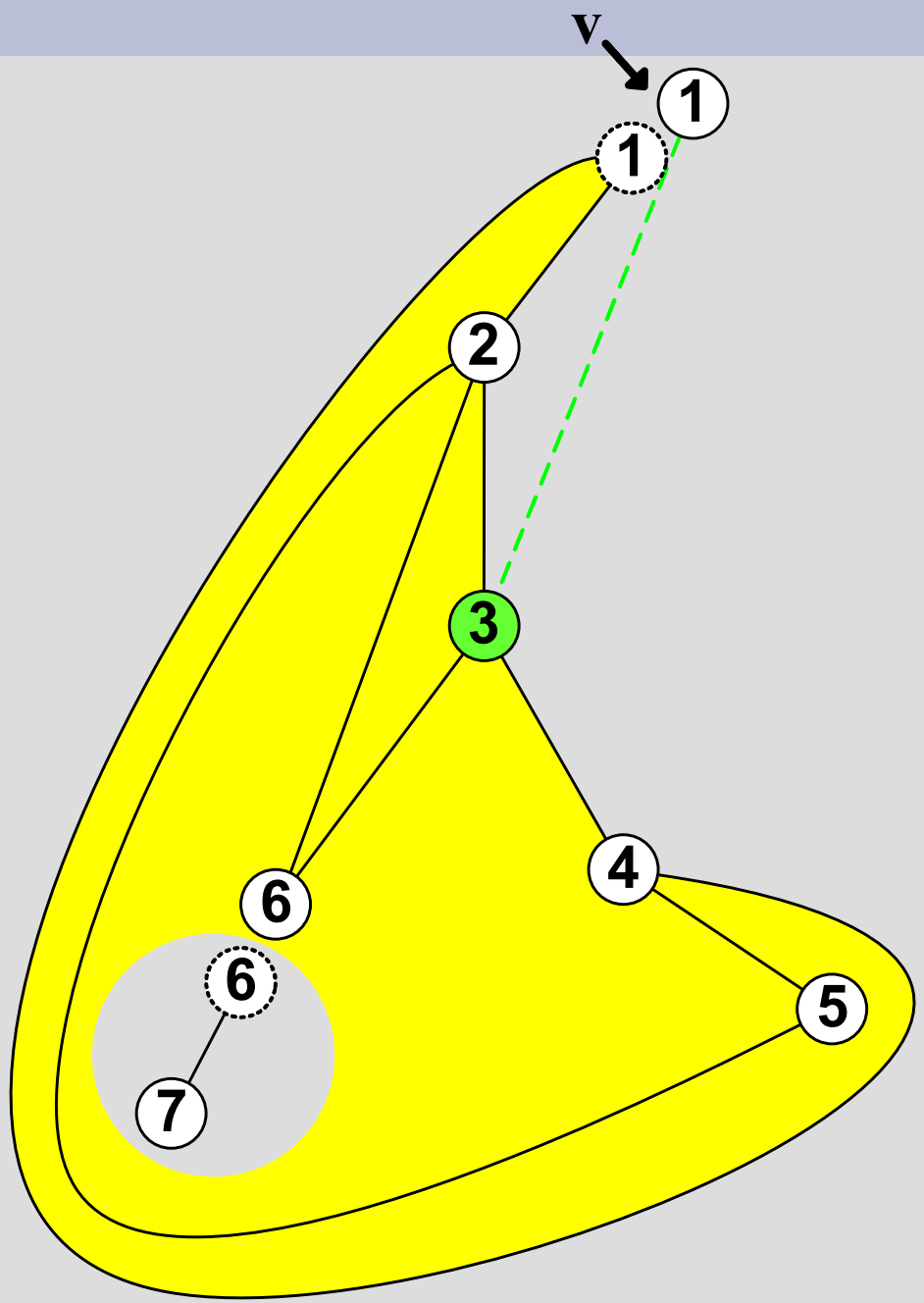
Knoten 1 einbetten



- Knoten 2: hat keine Backedges mehr
- Knoten 1: zunächst Walkup, berechne pertinente/externe Knoten

- Walkdown in Richtung CCW:
  - 1, 2, 2, 5, **4**
  - **4** einbetten, Bicomps verschmelzen

Knoten 1 einbetten

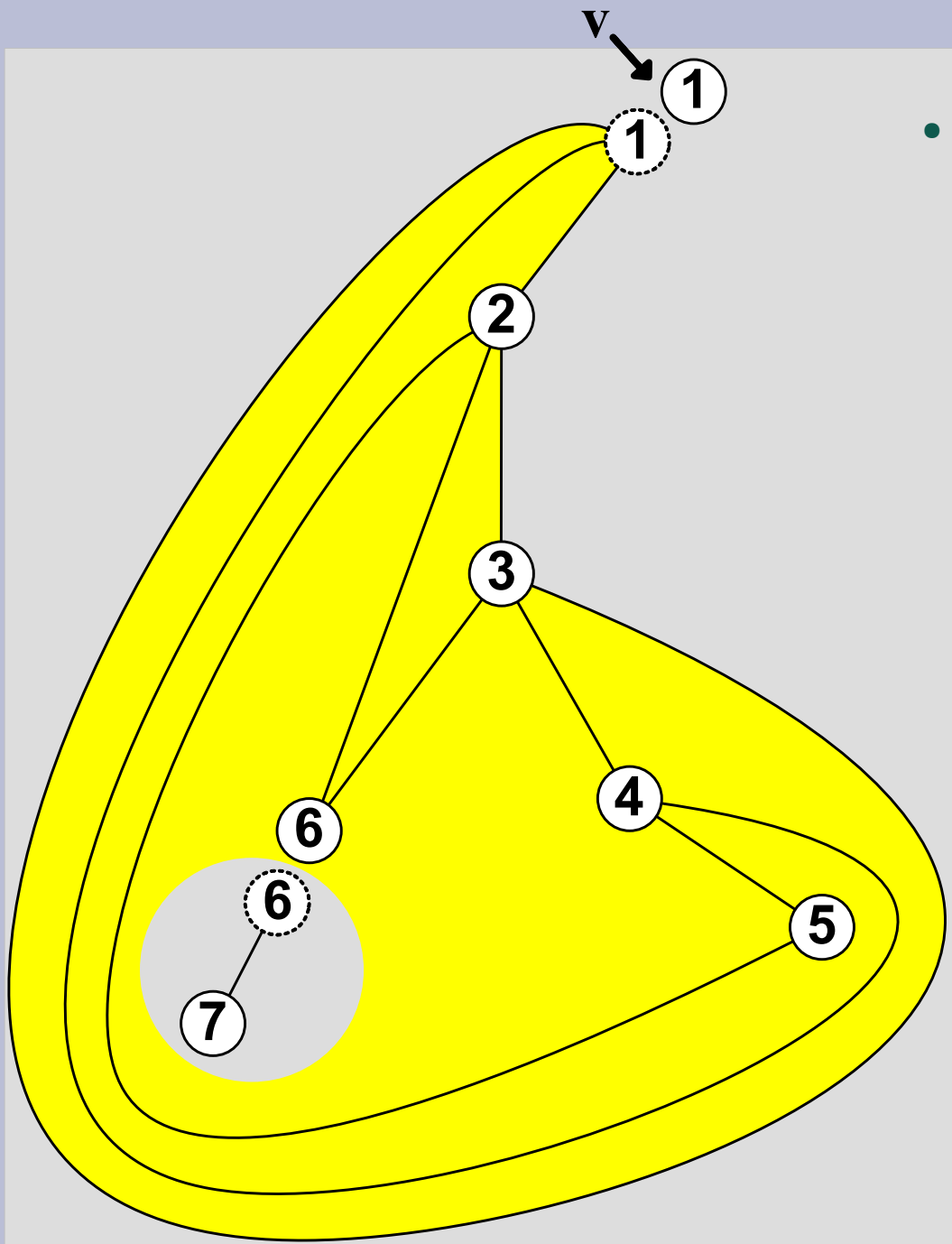


- Knoten 2: hat keine Backedges mehr
- Knoten 1: zunächst Walkup, berechne pertinente/externe Knoten

- Walkdown in Richtung CCW:
  - 1, 2, 2, 5, **4**
  - **4** einbetten, Bicomps verschmelzen
  - **3** einbetten

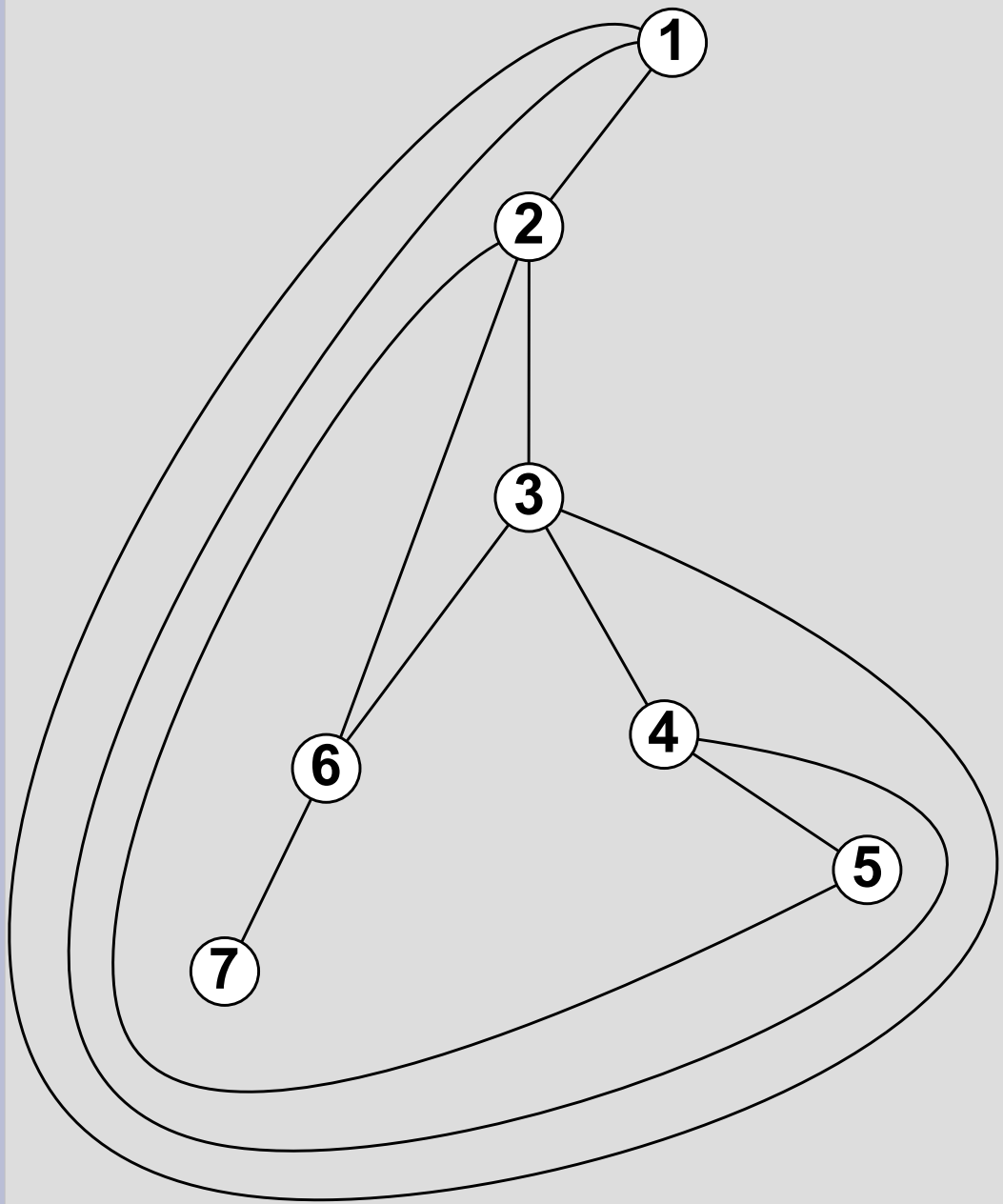
- Fertig!

Postprocessing



- Zum Abschluss:
  Verschmelze alle virtuellen Knoten mit ihren realen Knoten

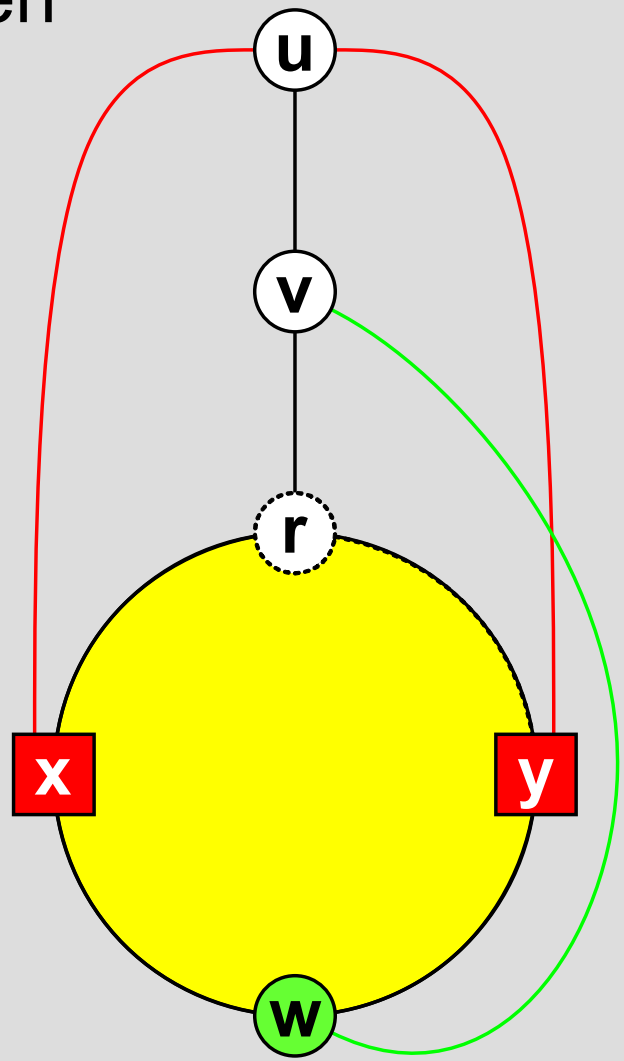
Graph ist planar!



Nicht planare Graphen

- Wie werden nicht planare Graphen erkannt?

- **Stoppkonfiguration:**
  - Während des Walkdowns wird in beiden Richtungen an externen Stoppknoten gehalten
  - Zusätzlich existieren noch pertinente Backedges

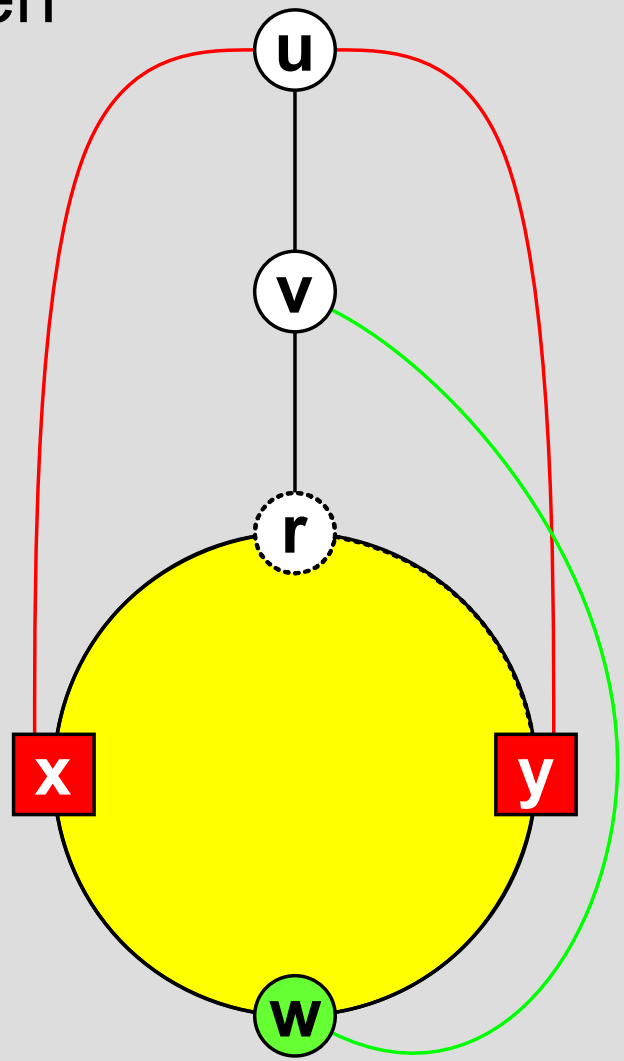


Nicht planare Graphen

- Wie werden nicht planare Graphen erkannt?

- **Stoppkonfiguration:**
  - Während des Walkdowns wird in beiden Richtungen an externen Stoppknoten gehalten
  - Zusätzlich existieren noch pertinente Backedges

- Hier $K_{3,3}$
- Viele verschiedene Minortypen: A,B,C,D,E_1-E_4



Minortyp A

Hier nicht behandelt

- „**Flipping**“ von Bicomps in konstanter Zeit
- Dynamische Datenstrukturen in O(1)
- Short Circuit Edges

Damit O(n) möglich!

Grundidee:

- Jeder Walkdown erzeugt neue Face.
- Summe aller Facekanten linear, also auch Walkdown linear.

Übersicht

1. Einleitung
2. Der Boyer-Myrvold Planaritätstest
3. **Erweiterungen**
4. Experimentelle Ergebnisse

Erweiterungen

- **Aufgabe:**
  Effiziente Extraktion von vielen Kuratowski-Subdivisions

- **Motivation:**
  - Generierung von Constraints beim Branch&Cut
    - Berechnung kreuzungsminimaler Einbettungen (NP-schwer)
    - Maximum Planar Subgraph Problem (NP-schwer)
  - Möglichst ähnliche bzw. verschiedene Kuratowski-Subdivisions wünschenswert
  - Heuristik für Maximal Planar Subgraph Problem

Effiziente Kuratowski-Extraktion

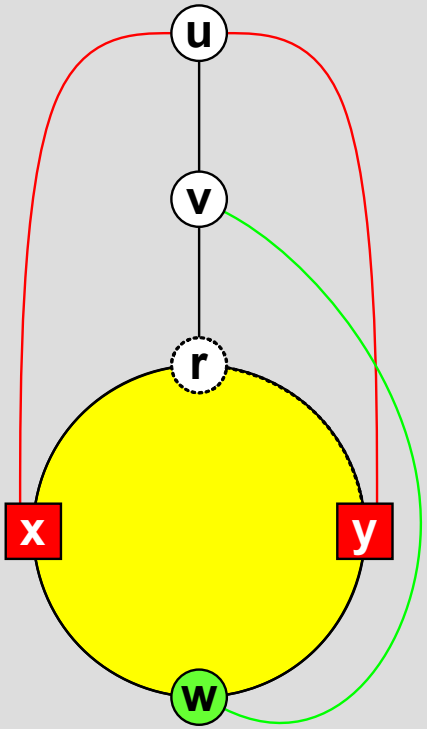
Probleme:

- Es kann $O(2^m)$ viele Kuratowki-Subdivisions geben
- Wie arbeitet der Planaritätstest nach einer gefundenen Kuratowski-Subdivision weiter?
- Algorithmus ändert sich dadurch vollständig, neue Laufzeitanalyse nötig.

Untere, lineare Schranke:

$$\Omega\left(n+m+\sum_{K\in\text{Kuratowskis}}|K|\right)$$

Wie nah kommt man theoretisch an die untere Schranke dran?



Effiziente Kuratowski-Extraktion

Boyer-Myrvold:

- ≤ 1 Kuratowski-Subdivisions, danach Abbruch
- Hier: Möglichst viele

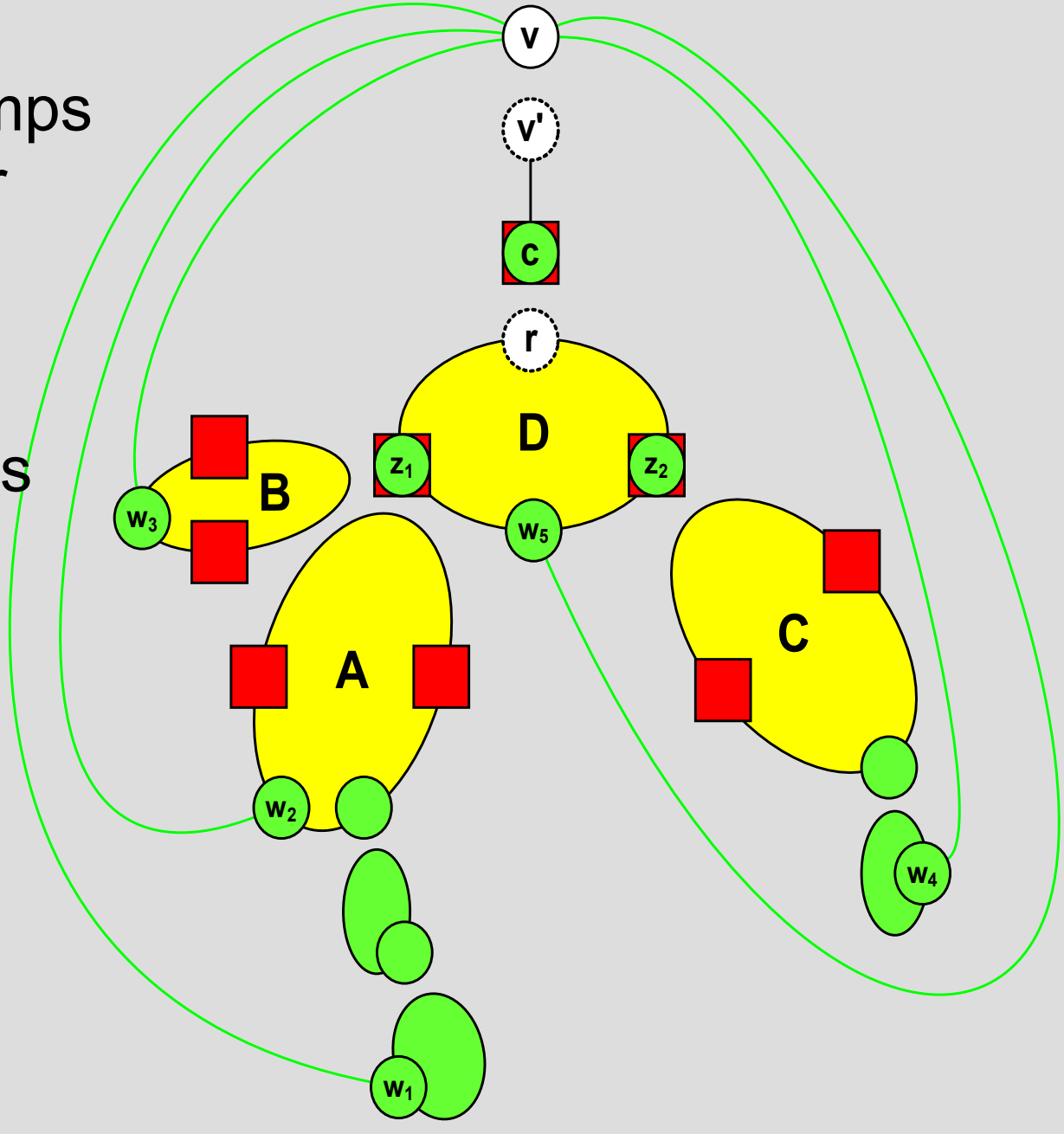
Erweiterungen:

- Global (Anzahl Stoppkonfigurationen erhöhen)
- Lokal (innerhalb einer Stoppkonfiguration)

Globale Erweiterungen

- **T:** Baum von Bicomps
- **Idee**: Bei erreichter Stoppkonfiguration alle Subdivisions extrahieren und kritische Backedges löschen.

- Wie gehts weiter?

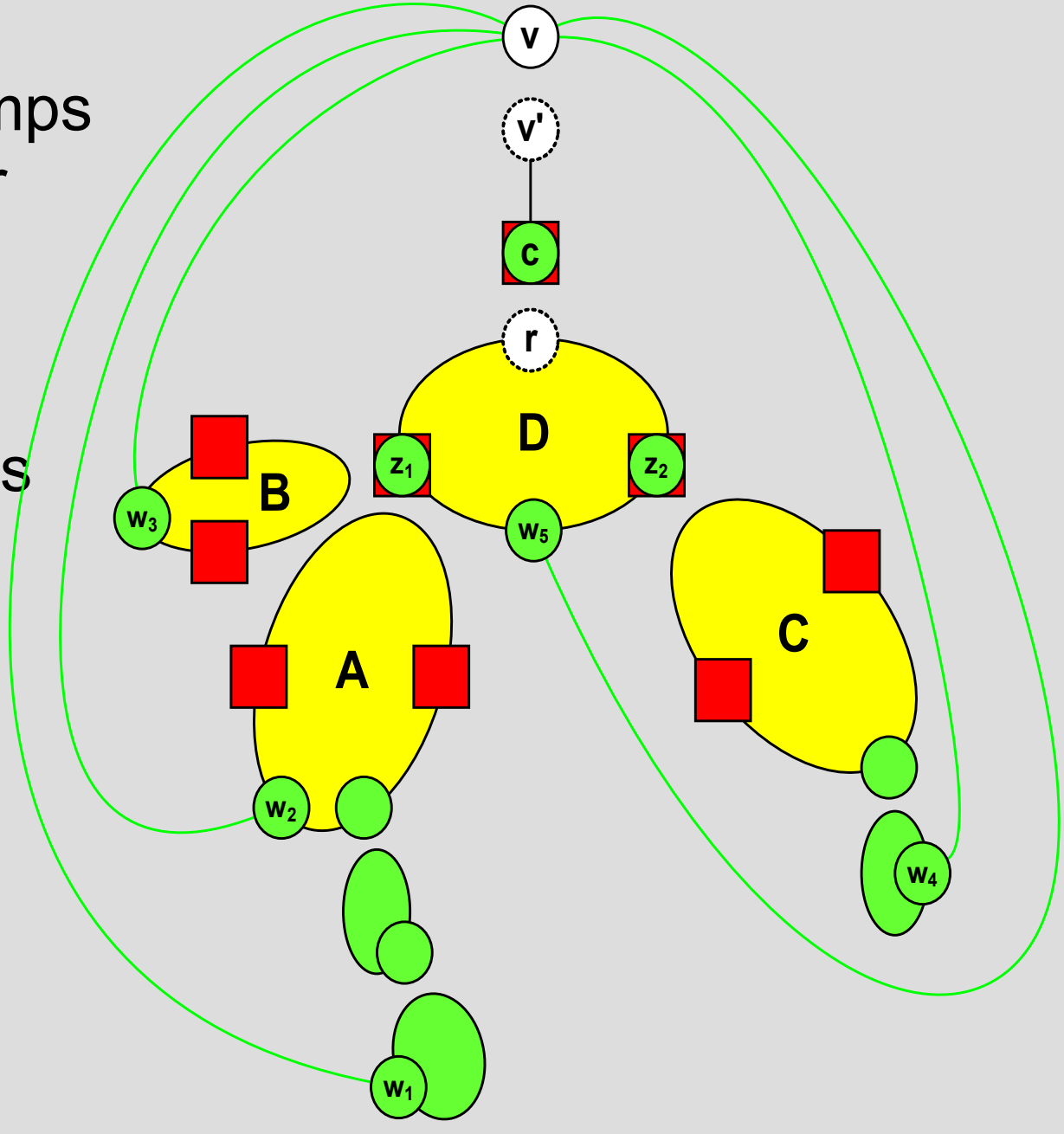


Globale Erweiterungen

- **T:** Baum von Bicomps
- **Idee**: Bei erreichter Stoppkonfiguration alle Subdivisions extrahieren und kritische Backedges löschen.

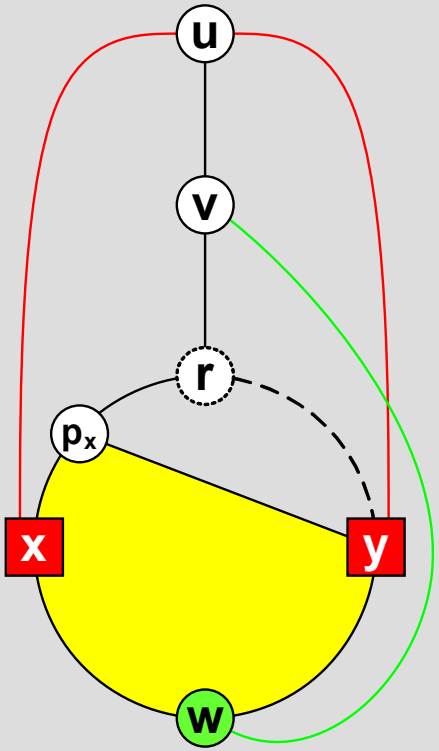
- Wie gehts weiter?

- Zwischenzeitliche Einbettungen sind möglich

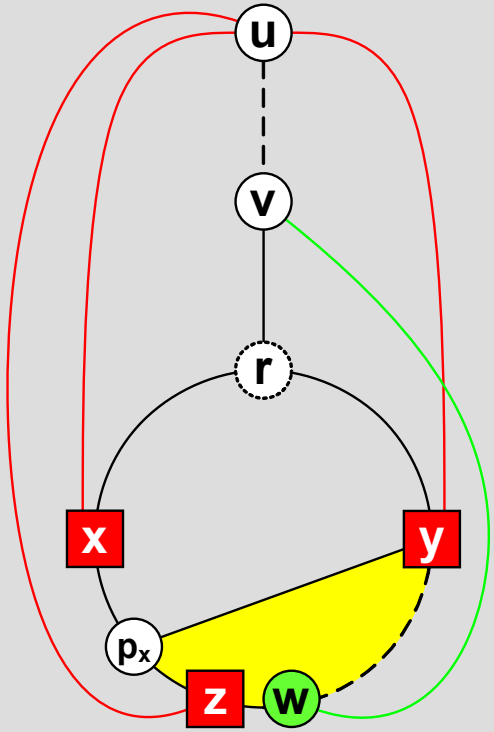


HighestXYPath

- „Oberster Pfad innerhalb der Bicomp“, der r und w trennt
- Wichtig für die Erkennung des zu extrahierenden Minortyps

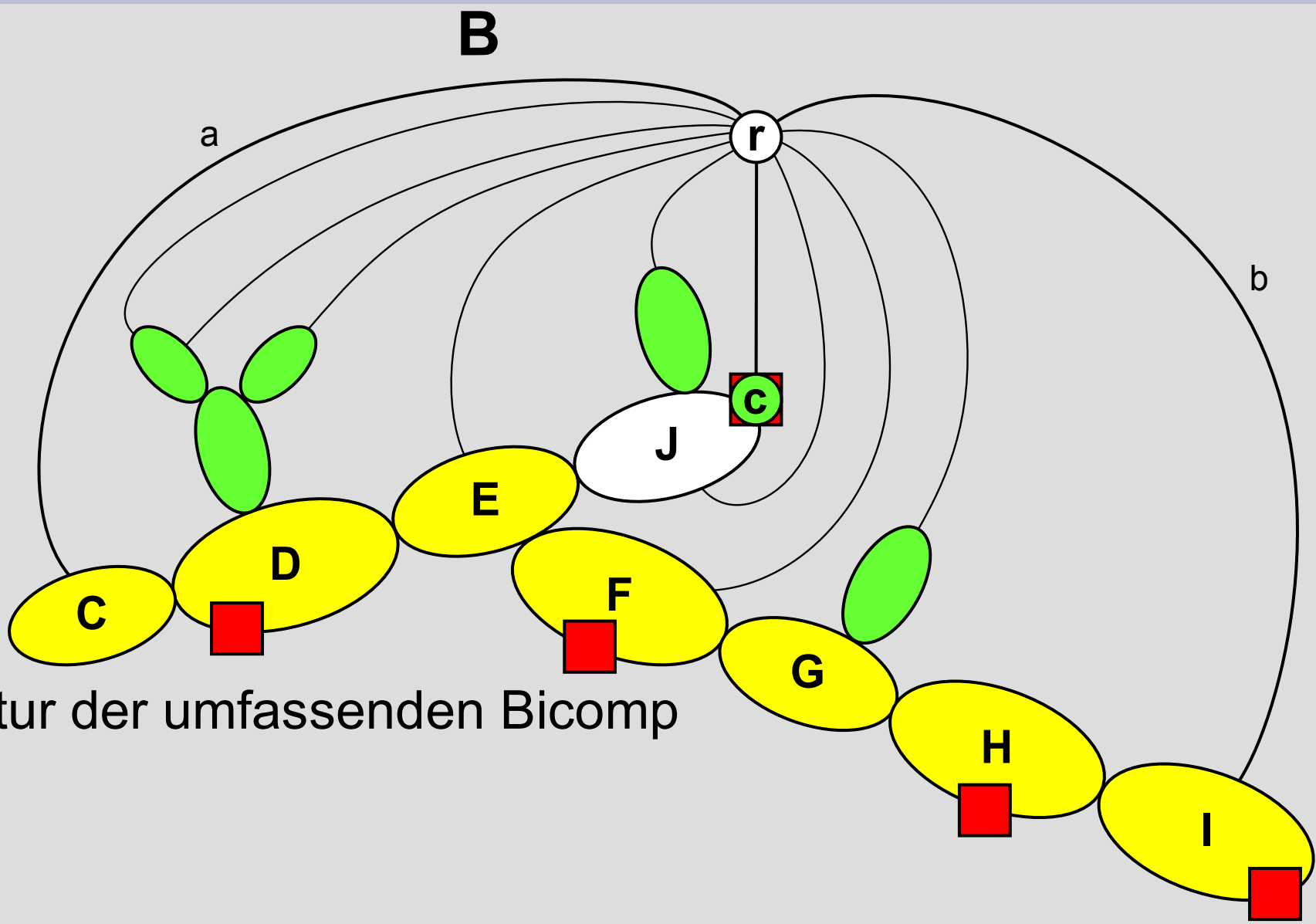


Minor AD



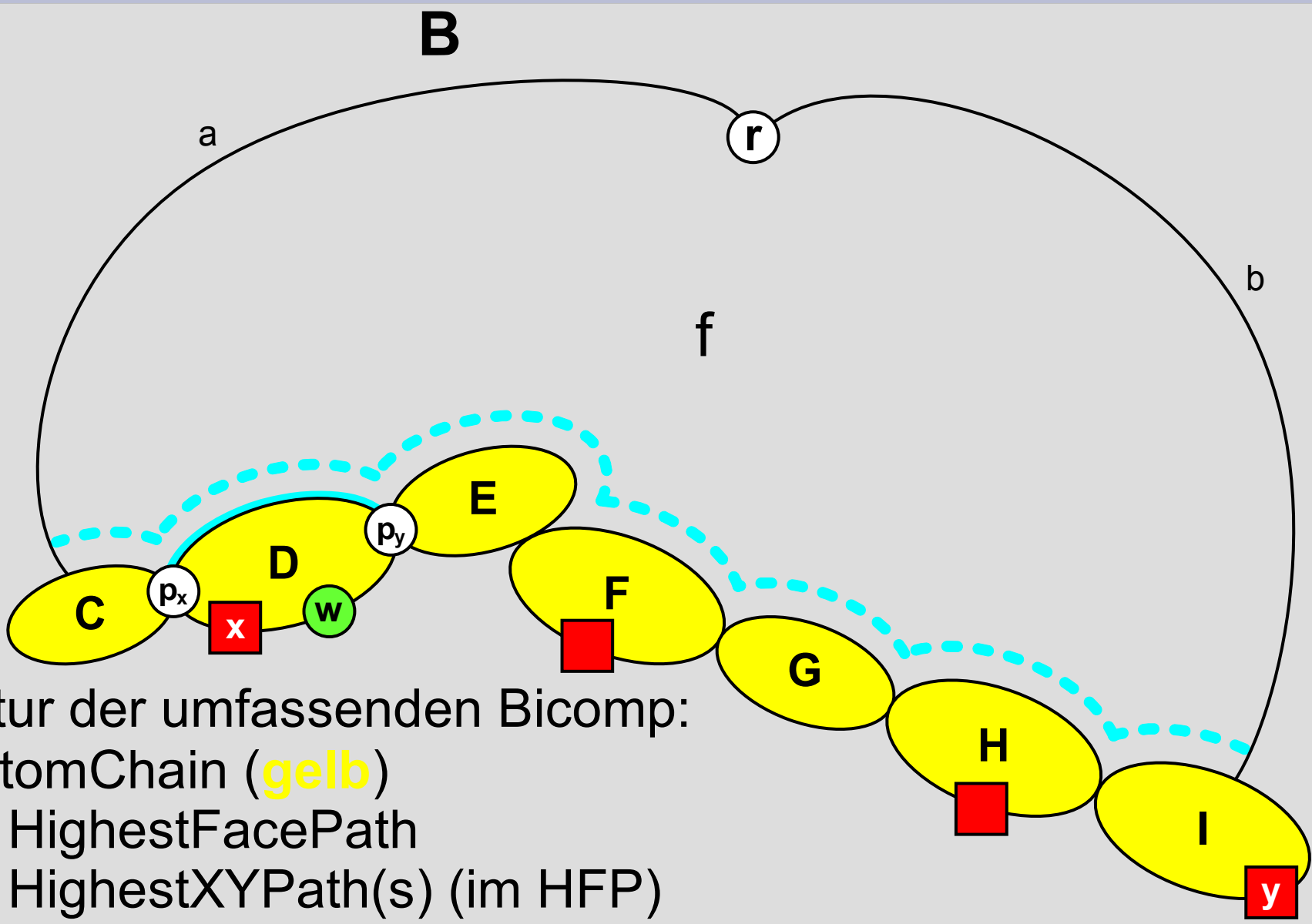
Minor AE_4

HighestXYPath



Struktur der umfassenden Bicomp

HighestXYPath



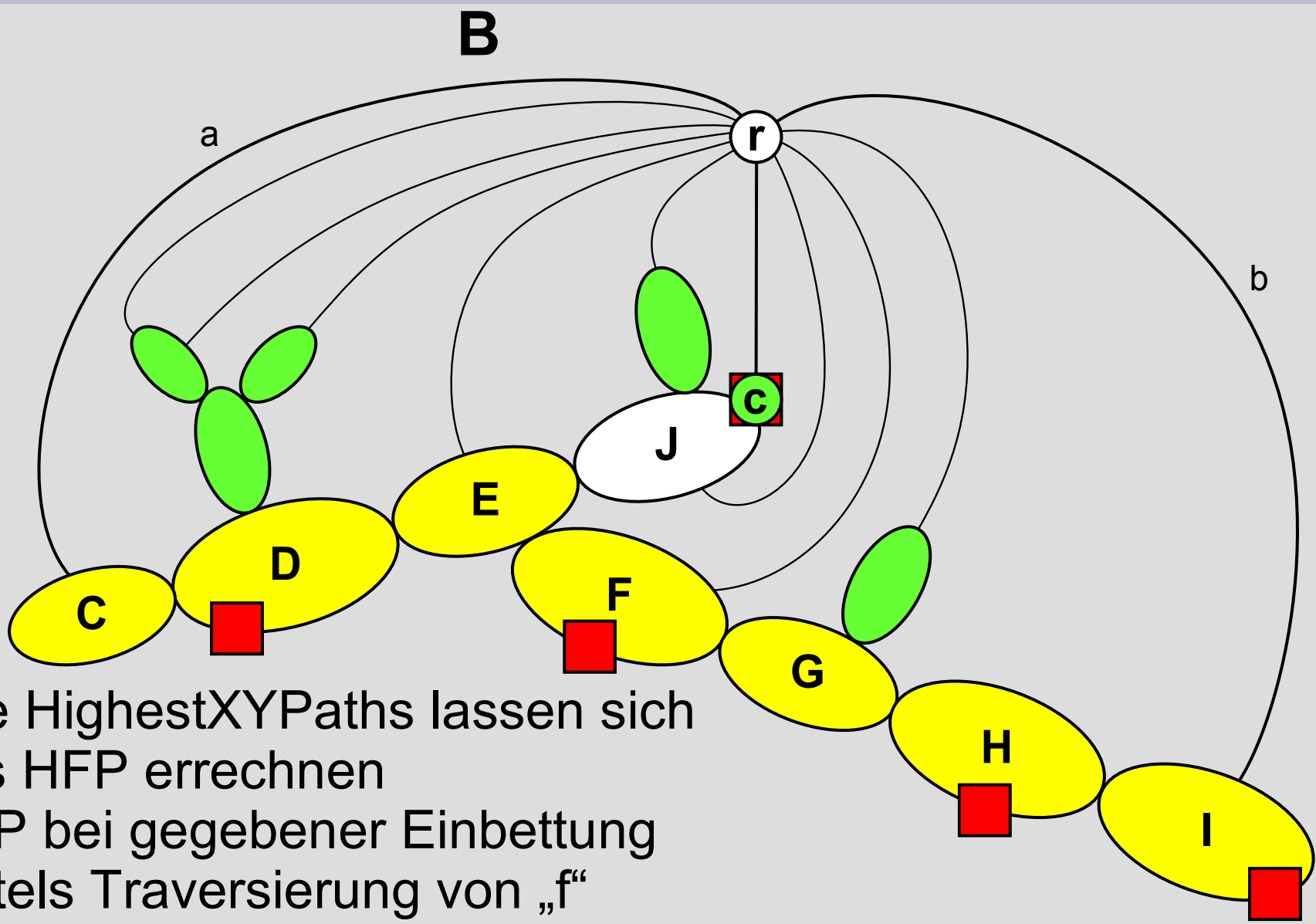
Struktur der umfassenden Bicomp:

- BottomChain (gelb)

HighestFacePath

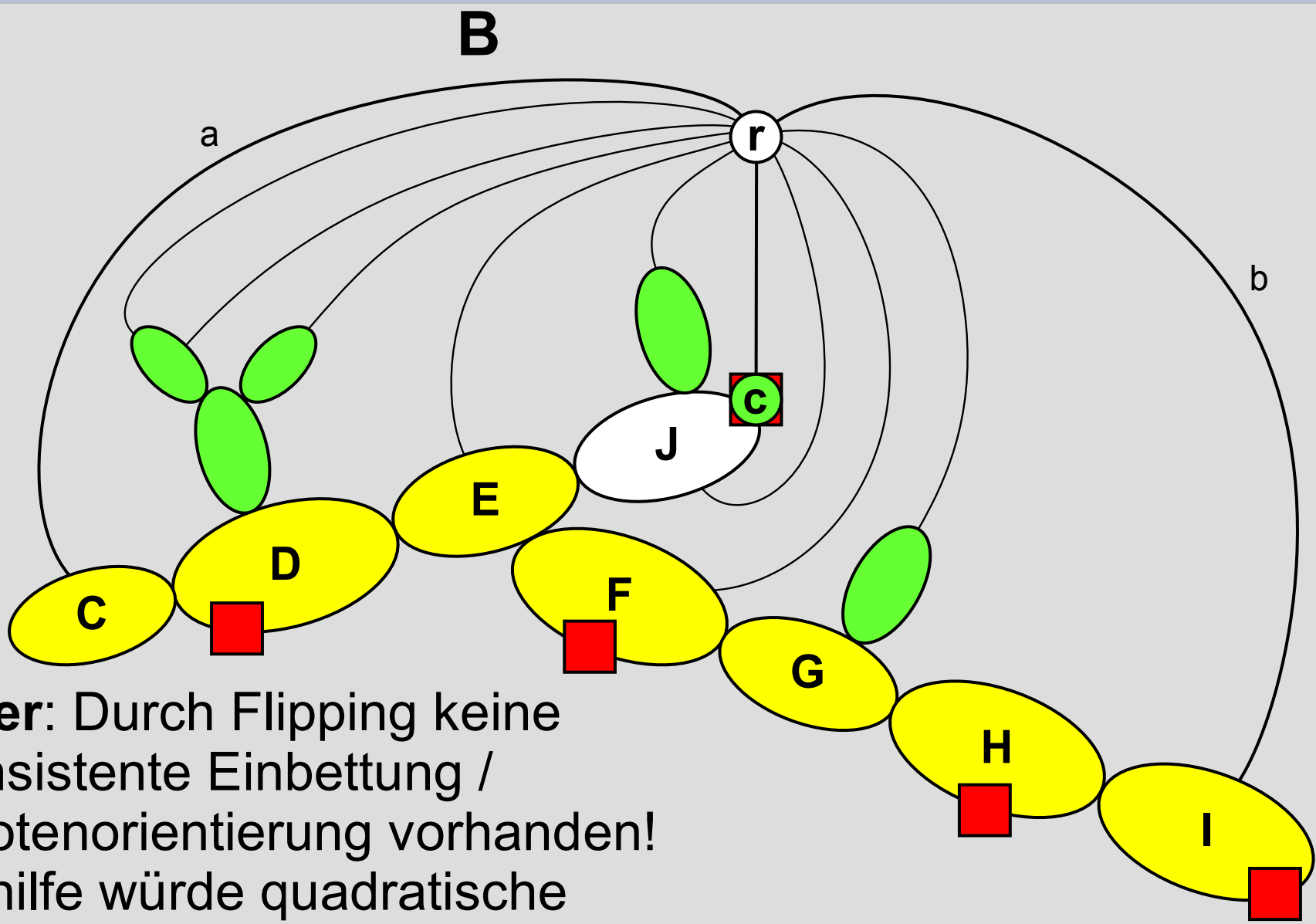
HighestXYPath(s) (im HFP)

HighestFacePath



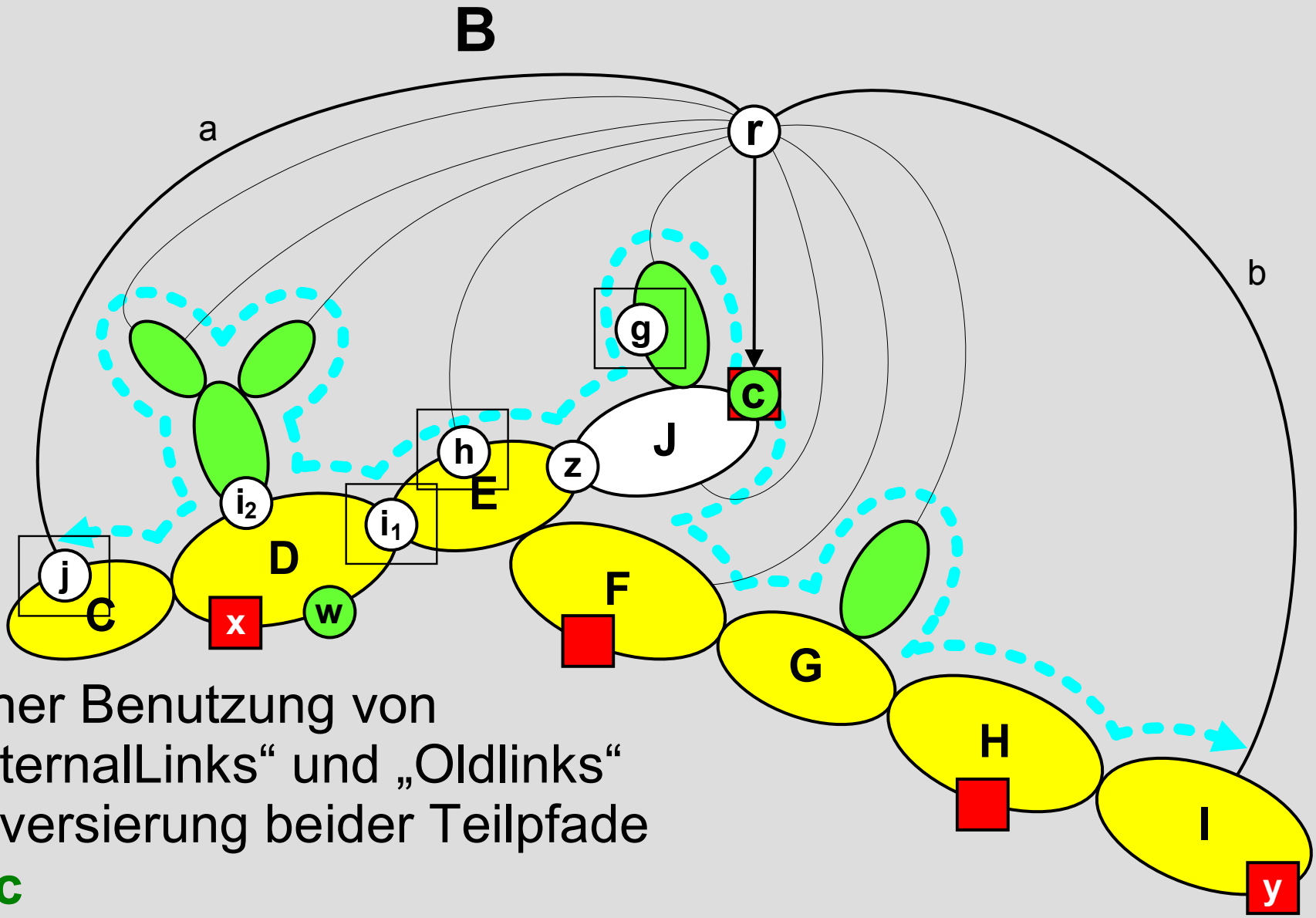
- Alle HighestXYPaths lassen sich aus HFP errechnen
- HFP bei gegebener Einbettung mittels Traversierung von „f“

HighestFacePath



- **Aber**: Durch Flipping keine konsistente Einbettung / Knotenorientierung vorhanden!
- Abhilfe würde quadratische Laufzeit verursachen

HighestFacePath

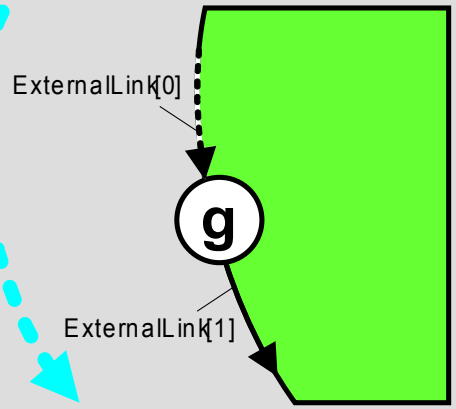


- Daher Benutzung von „ExternalLinks“ und „Oldlinks“
- Traversierung beider Teilpfade ab **c**

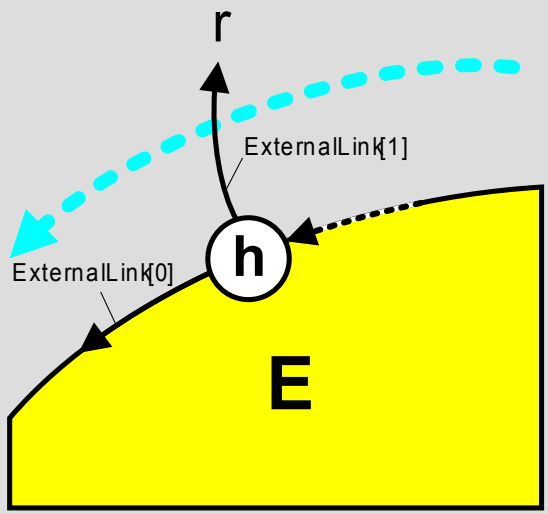
HighestFacePath

- Damit navigieren trotz inkonsistenter Orientierung möglich
- 4 Knotentypen mit Traversierungsregel für jeden einzelnen
- Insg. lineare Laufzeit O(n+m)

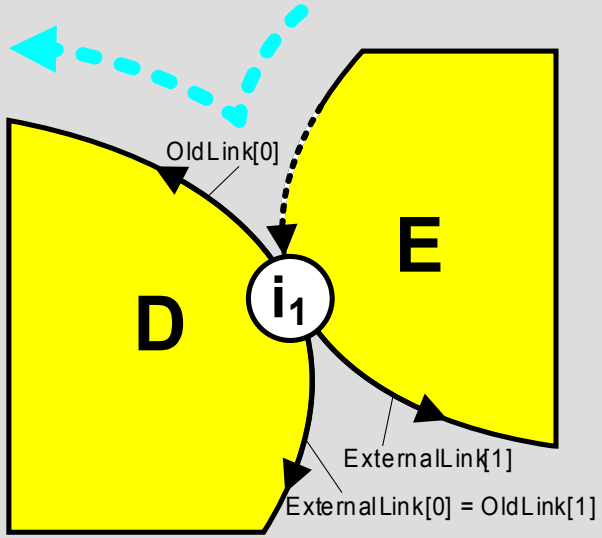
g)



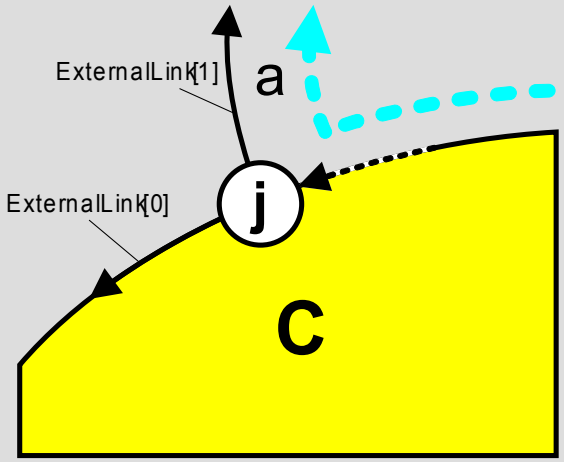
h)



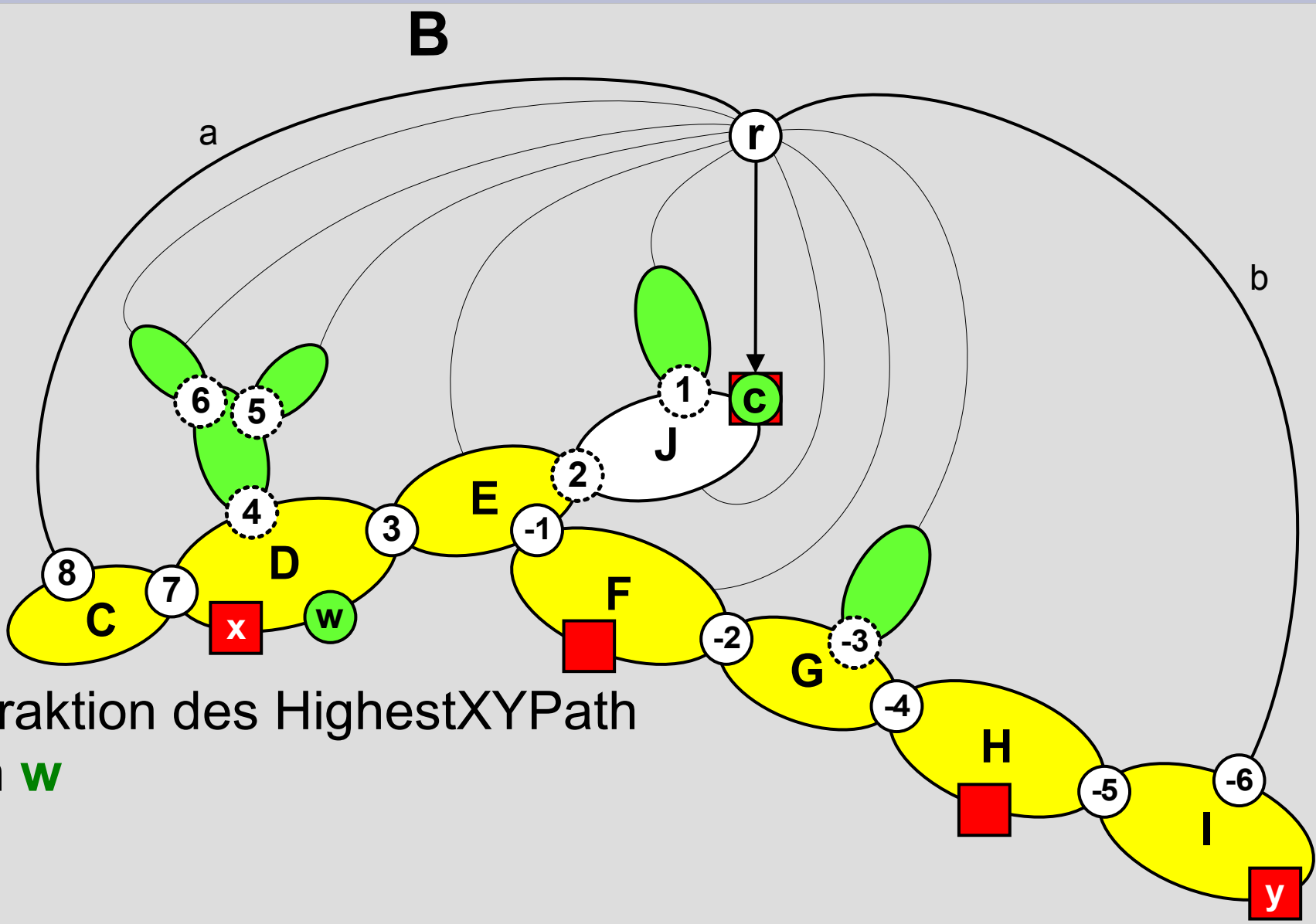
i)



j)



HighestXYPath



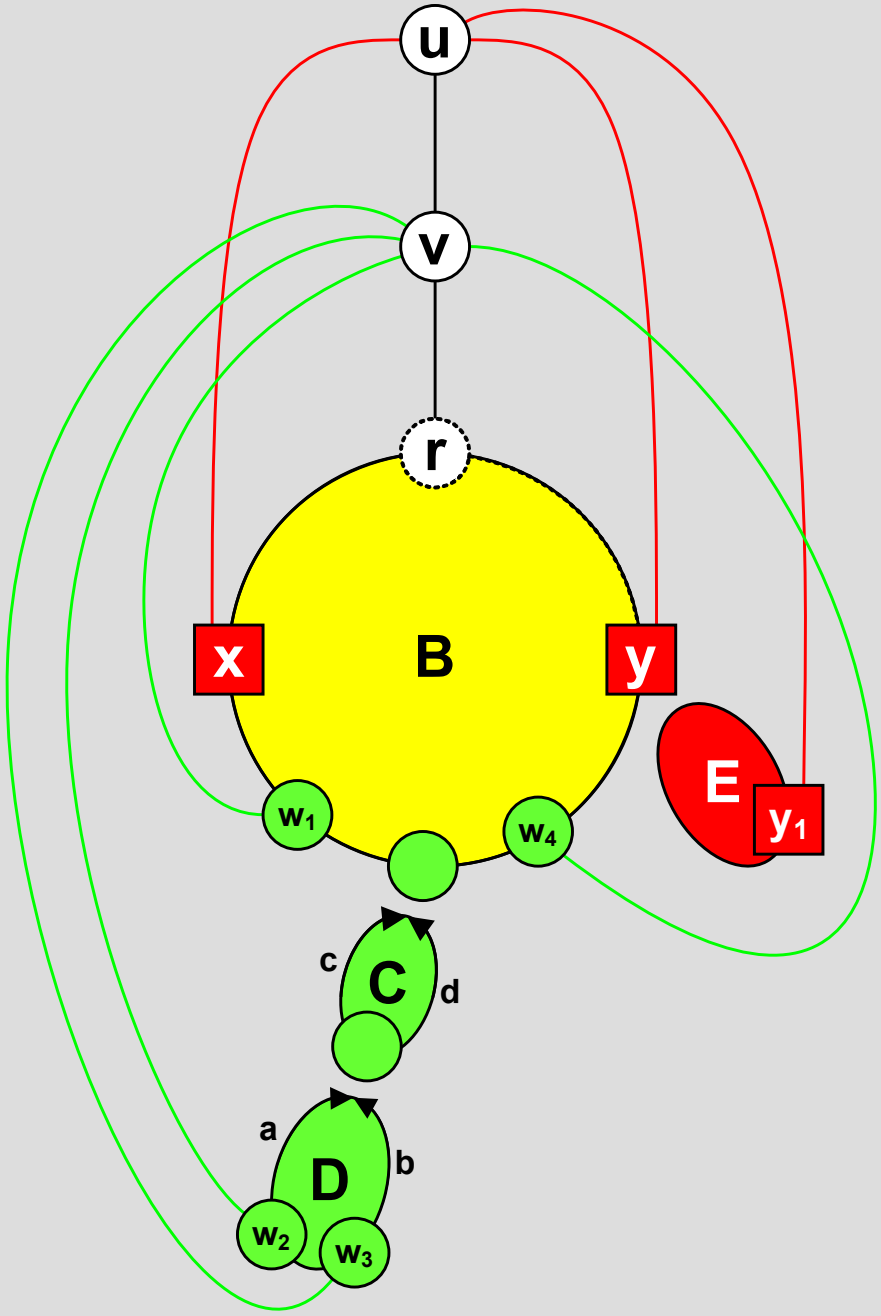
- Extraktion des HighestXYPath von **w**

Lokale Erweiterungen

Mehrere Kuratowskis durch:

- Verschiedene „kritische“ Backedges / - pfade
- Verschiedene externe Backedges / -pfade

- Hier: (**2**+**8**)*2 = **20** Kuratowski-Subdivisions aus einer Stoppkonfiguration.



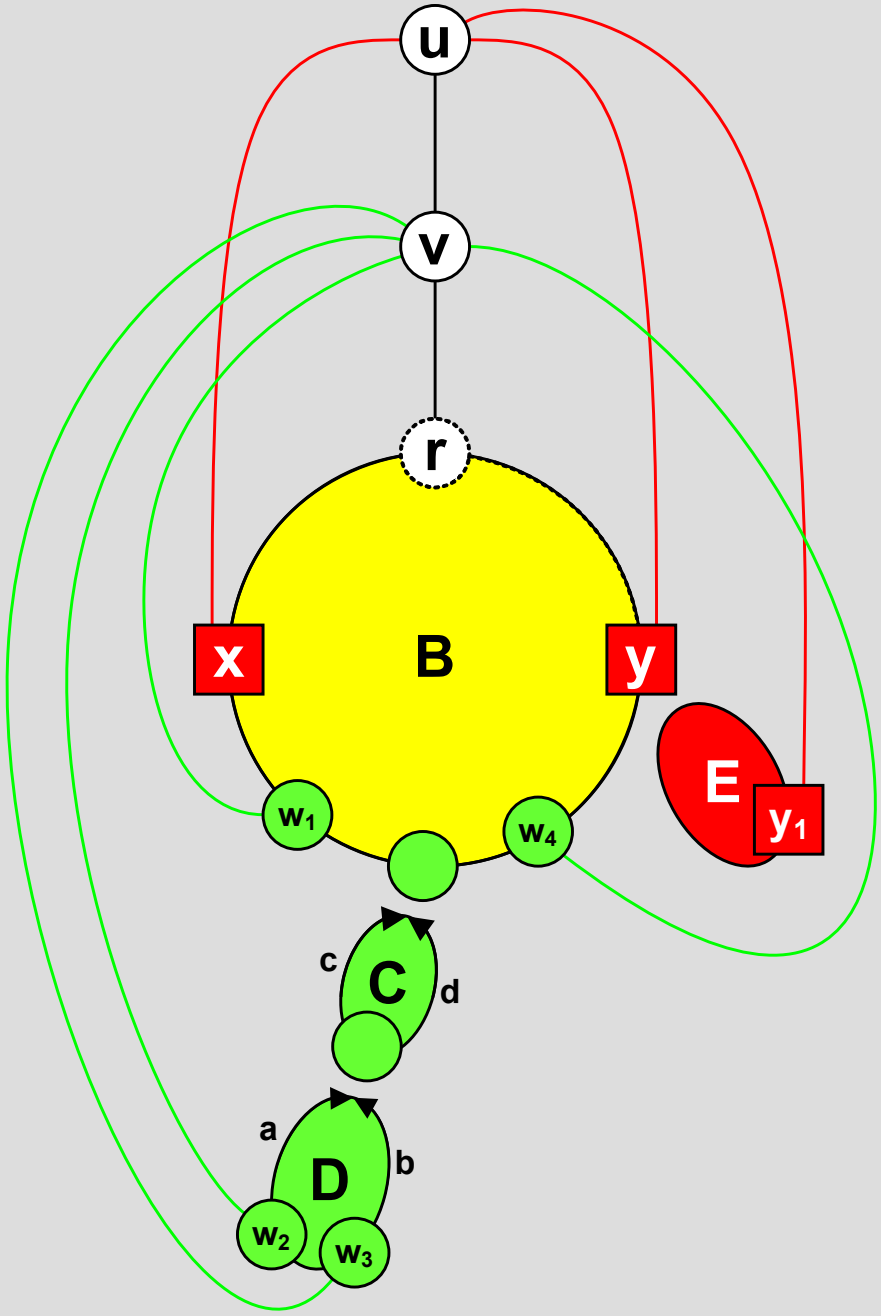
Lokale Erweiterungen

Mehrere Kuratowskis durch:

- Verschiedene „kritische“ Backedges / - pfade
- Verschiedene externe Backedges / -pfade

- Hier: (**2**+**8**)*2 = **20** Kuratowski-Subdivisions aus einer Stoppkonfiguration.

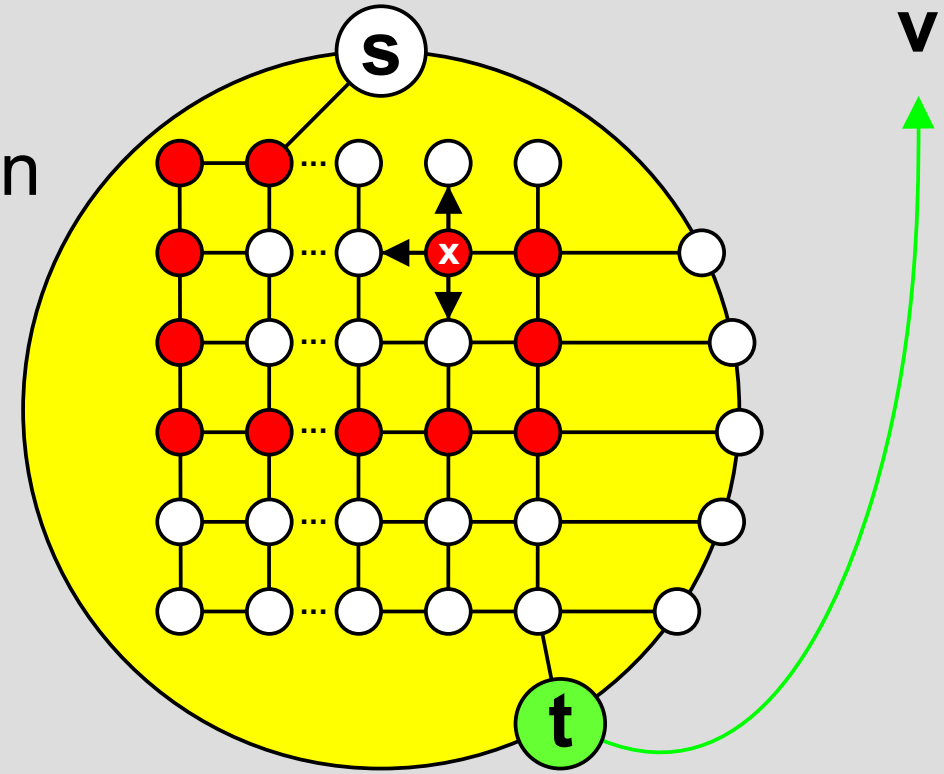
- Bundle-Variante
- Zusätzliche Minortypen



Bundle-Variante

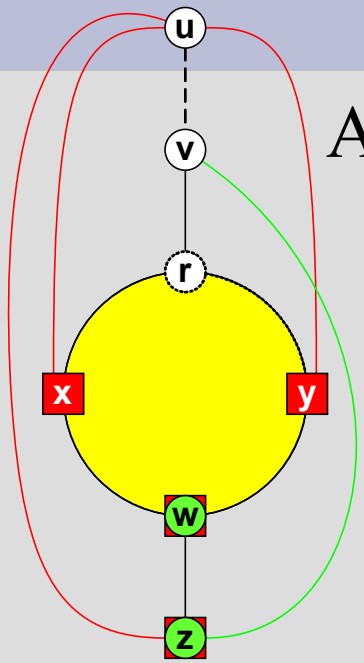
„Bündel“ von Kuratowski-Pfaden:

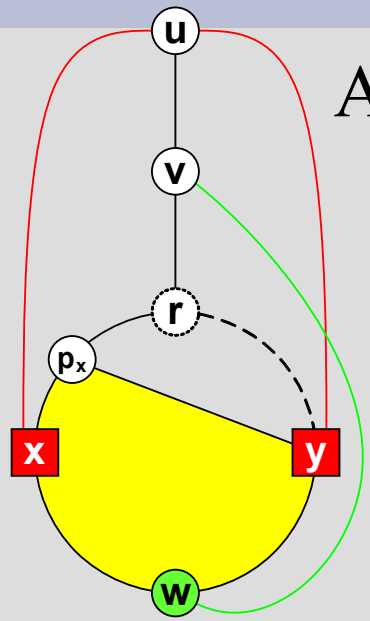
- Alle s-t-Pfade jeder Bicomp werden genutzt (i.A. exponentiell viele).
- Durch Sackgassen keine lineare Laufzeit bei Extraktion mehr möglich (untere Schranke von Patrascu & Demaine, 2004)
- Zusätzlicher Faktor „log(n)“ für jede extrahierte Subdivision

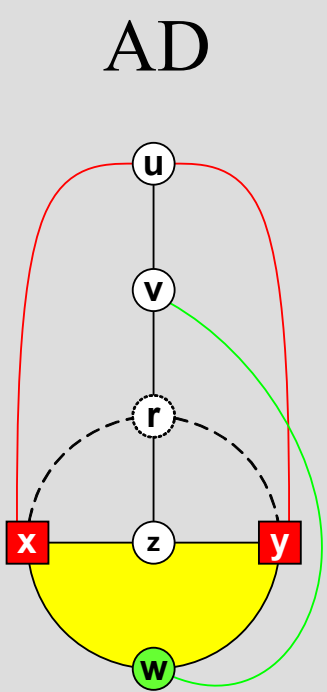


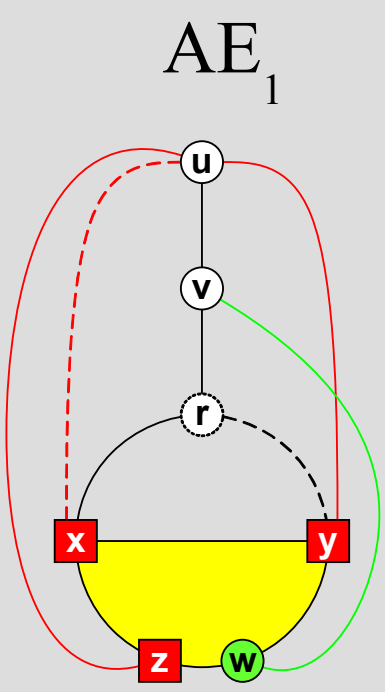
Zusätzliche Minortypen

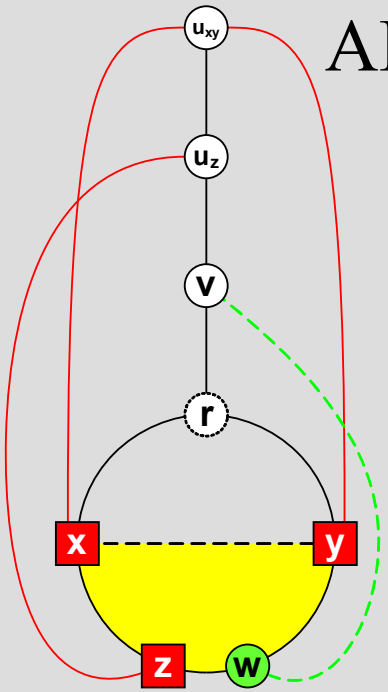
- Statt 9 verschiedener Minortypen nun 16!
- Kuratowski-Subdivisions ergeben sich durch Löschen der gestrichelten Pfade.

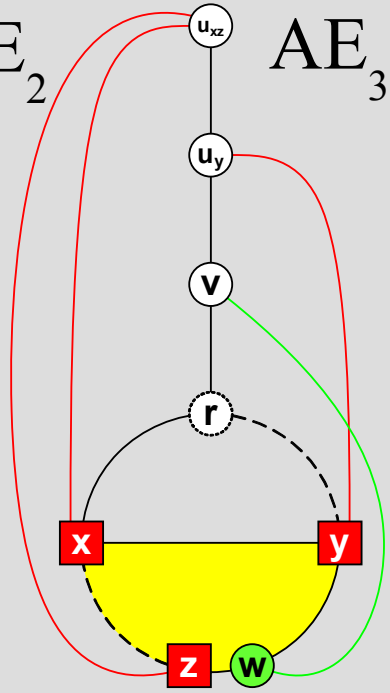


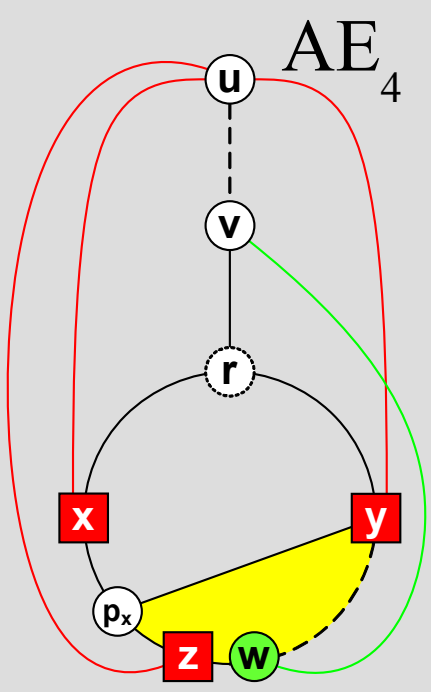












Gesamtlaufzeit

S = Menge der extrahierten Kuratowski-Subdivisions
Ziel: Kompensiere Mehraufwand durch Subdivision-Edges

Gesamtlaufzeit ist

$$O(n+m+\sum_{K\in S}|K|)$$

und damit linear

Berechnungsschritt	Erw.	Gesamtlaufzeit
Erweiterter Walkup	global	$O(n+m+\sum_{K\in S}\|E(K)\|)$
Erweiterter Walkdown	global	$O(n+m+\sum_{K\in S}\|E(K)\|)$
– ShortCircuit-Kanten	global	$O(n+m)$
Zusätzliche Backedgepfade	lokal	$O(\sum_{K\in S}\|E(K)\|)$
Klassifikation der Minortypen	lokal	$O(n+m+\sum_{K\in S}\|E(K)\|)$
Extraktion...		
– ...von v,r,$stopX$,$stopY$	global	$O(m)$
– ...kritischer Backedges	global	$O(n+m+\sum_{K\in S}\|E(K)\|)$
– ...externer Backedgepfade	lokal	$O(\sum_{K\in S}\|E(K)\|)$
– ...der *HighestFacePaths*	global	$O(n+m)$
– ...und Lage der *HighestXYPaths*	global	$O(\sum_{K\in S}\|E(K)\|)$
– ...externer z-Knoten für die Minortypen E/AE	lokal	$O(\sum_{K\in S}\|E(K)\|)$
Extraktion aller Minortypen	lokal	$O(\sum_{K\in S}\|E(K)\|)$

Übersicht

1. Einleitung
2. Der Boyer-Myrvold Planaritätstest
3. Erweiterungen
4. **Experimentelle Ergebnisse**

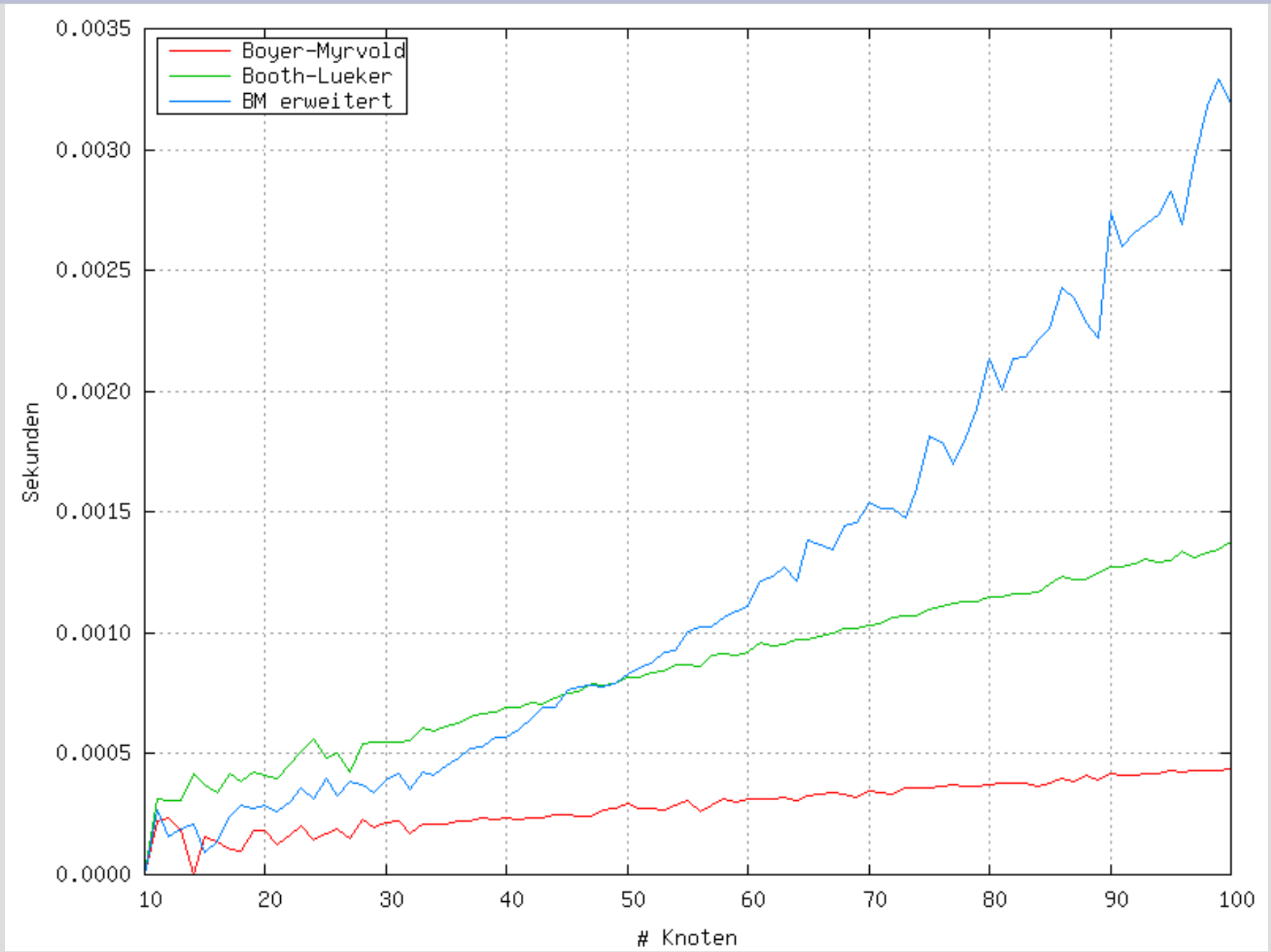
Experimentelle Ergebnisse

Testplattform

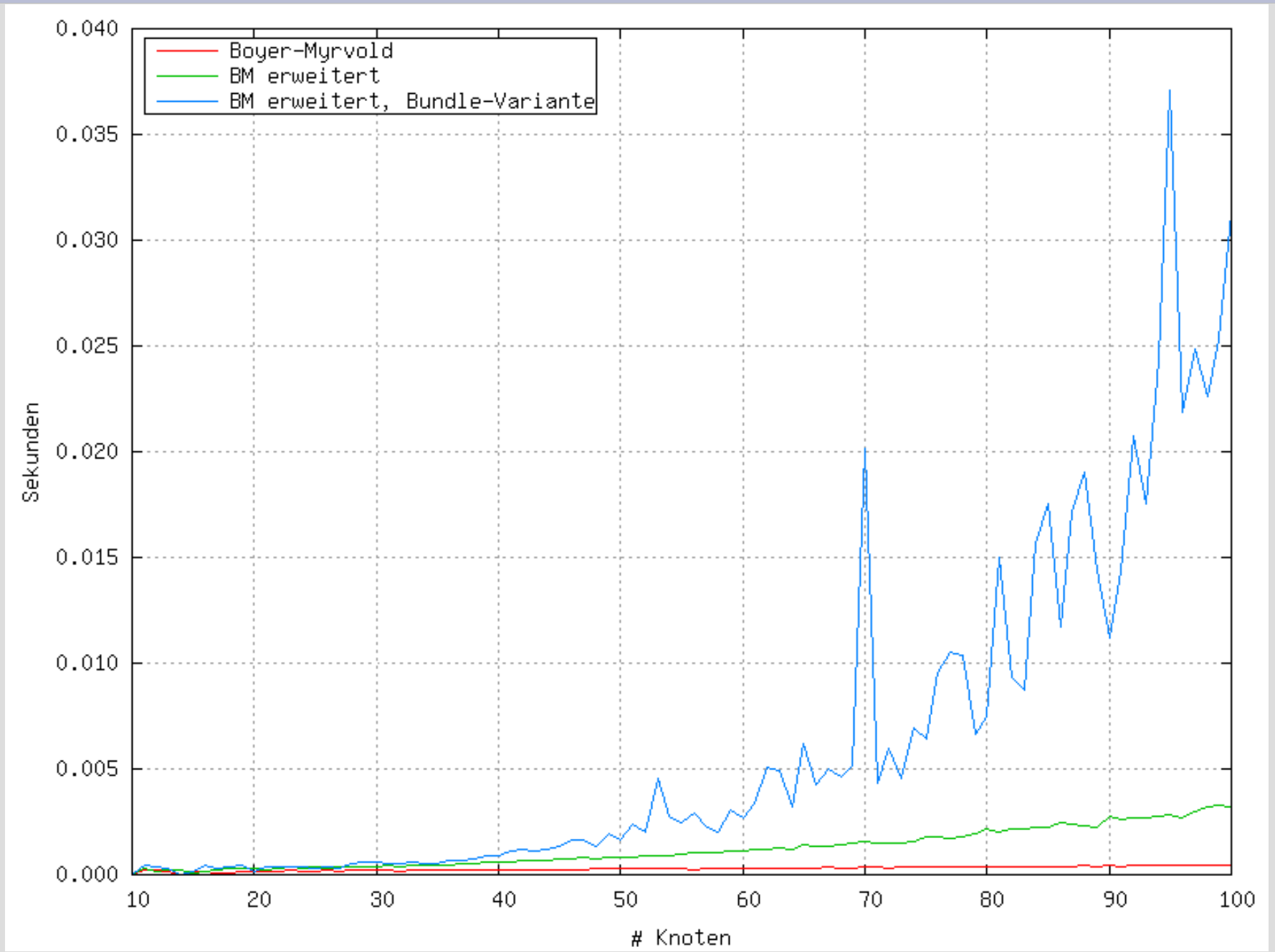
- Implementierung im Open Graph Drawing Framework (OGDF)
- Intel CoreDuo 6300
- 2GB Ram
- GCC 3.4.4, -O1

- Graphen:
  - Rome-Library (50 Samples)
  - Random Graphs (über OGDF generiert, 20 Samples)

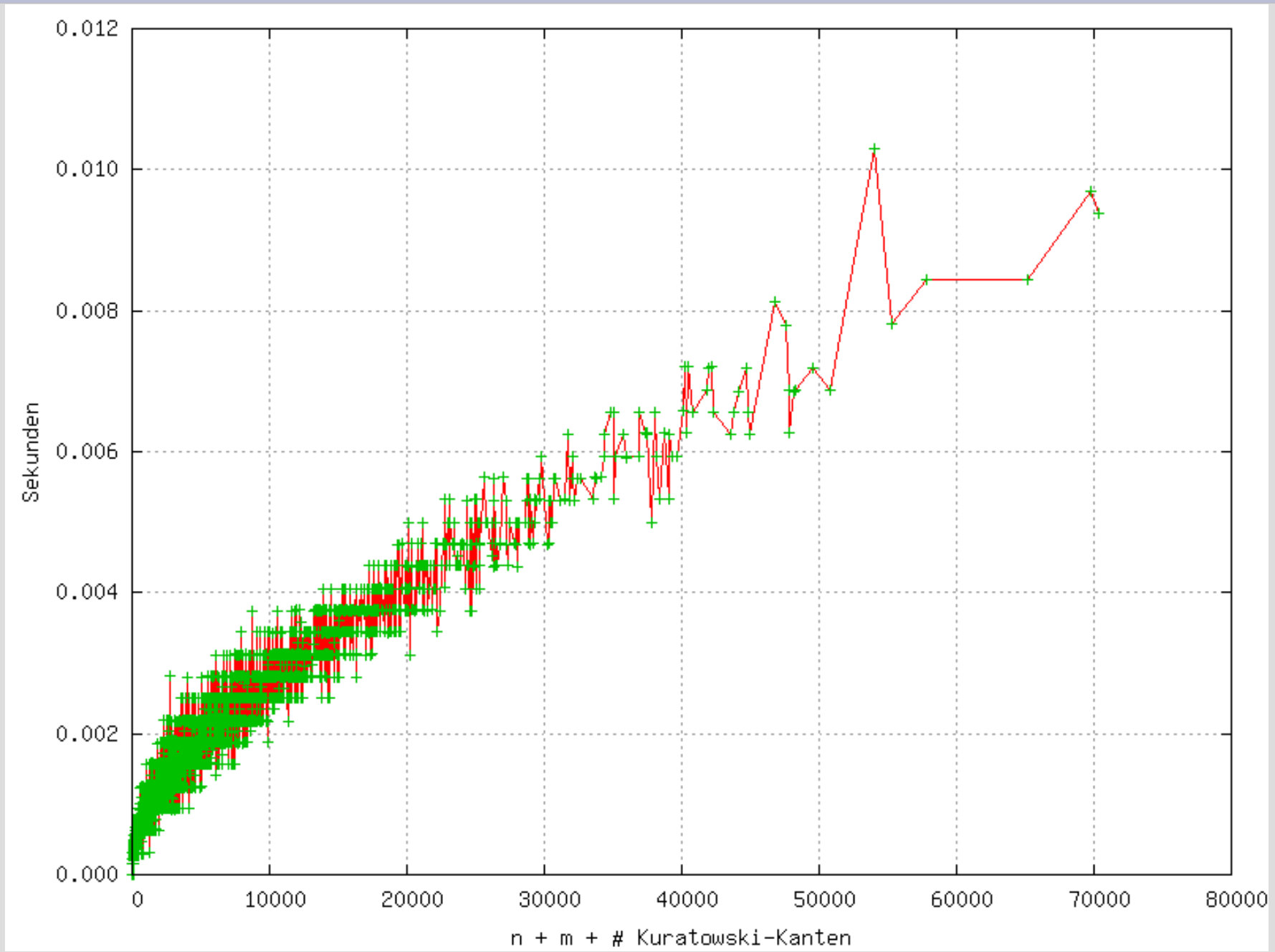
Rome Library - Laufzeit



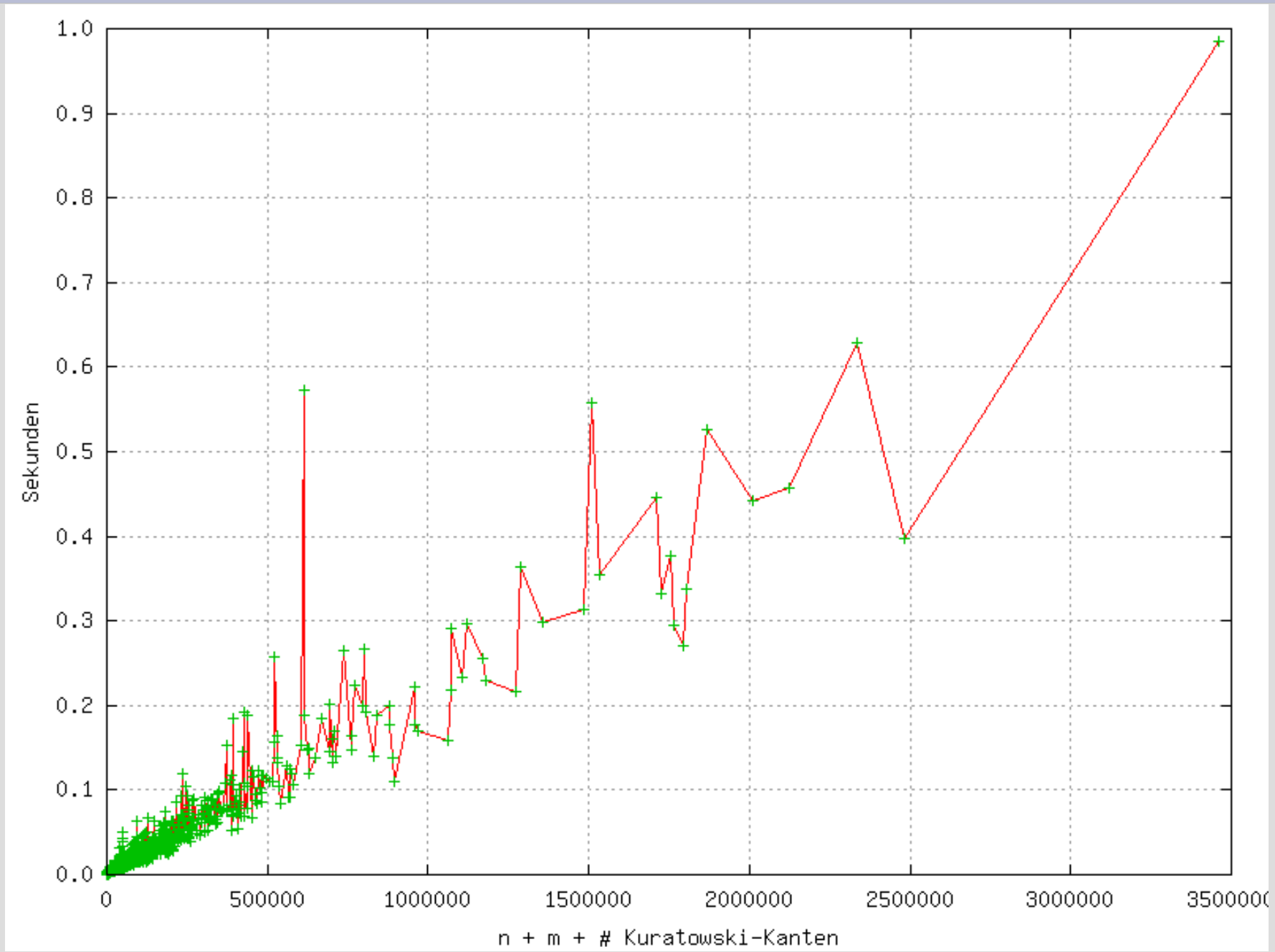
Rome Library – Laufzeit Bundles



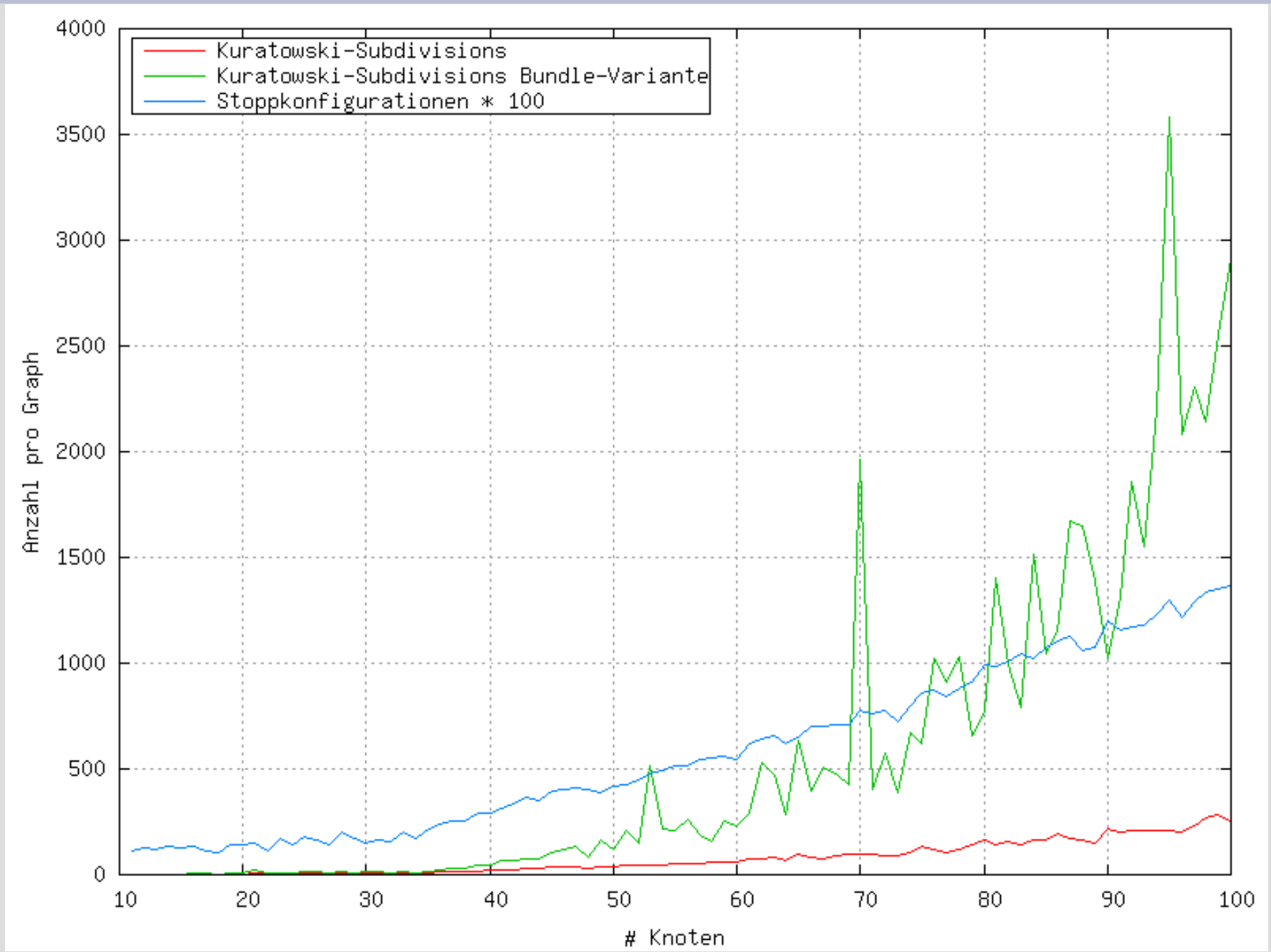
Rome Library – Laufzeit



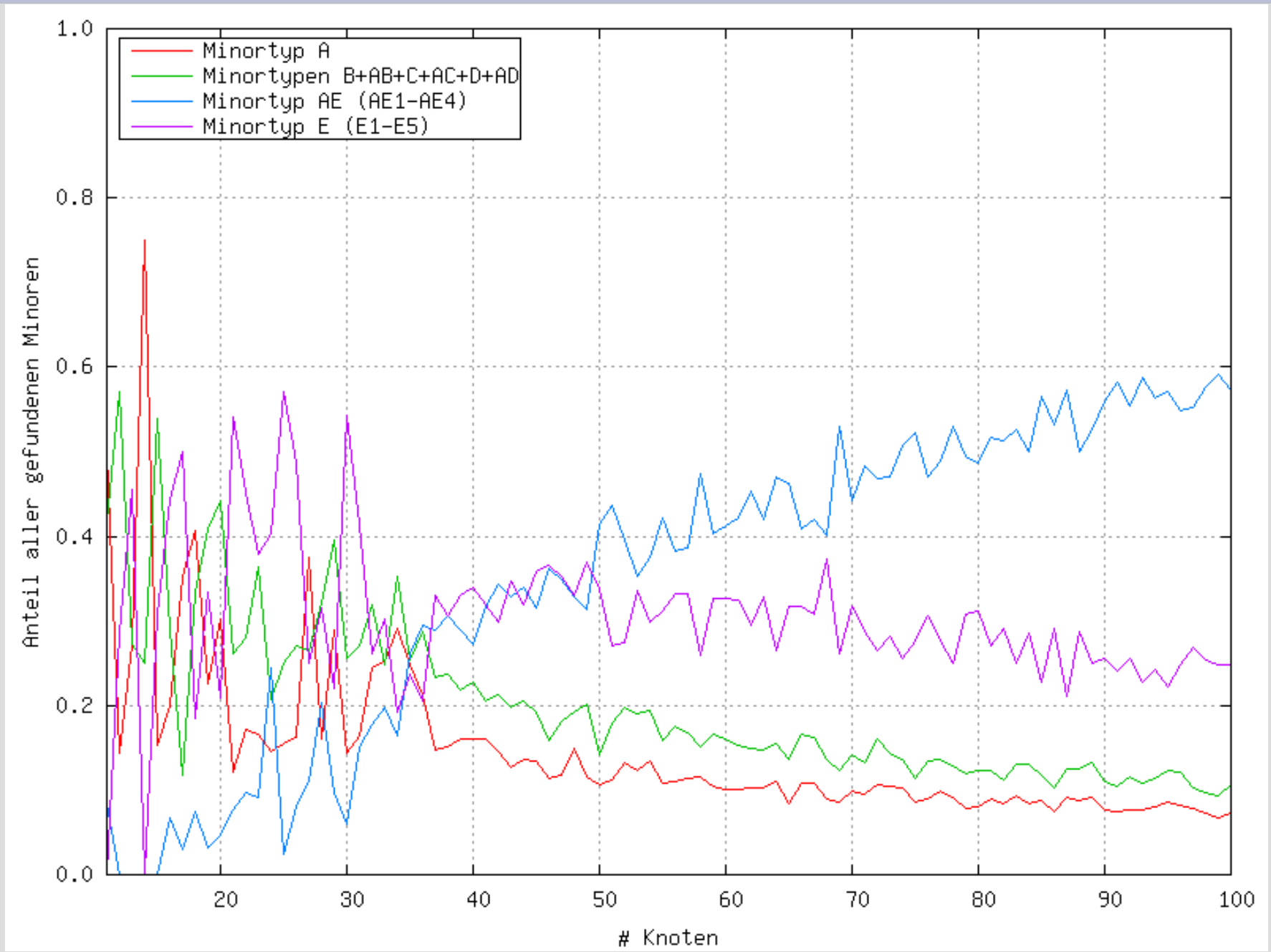
Rome Library – Laufzeit Bundles



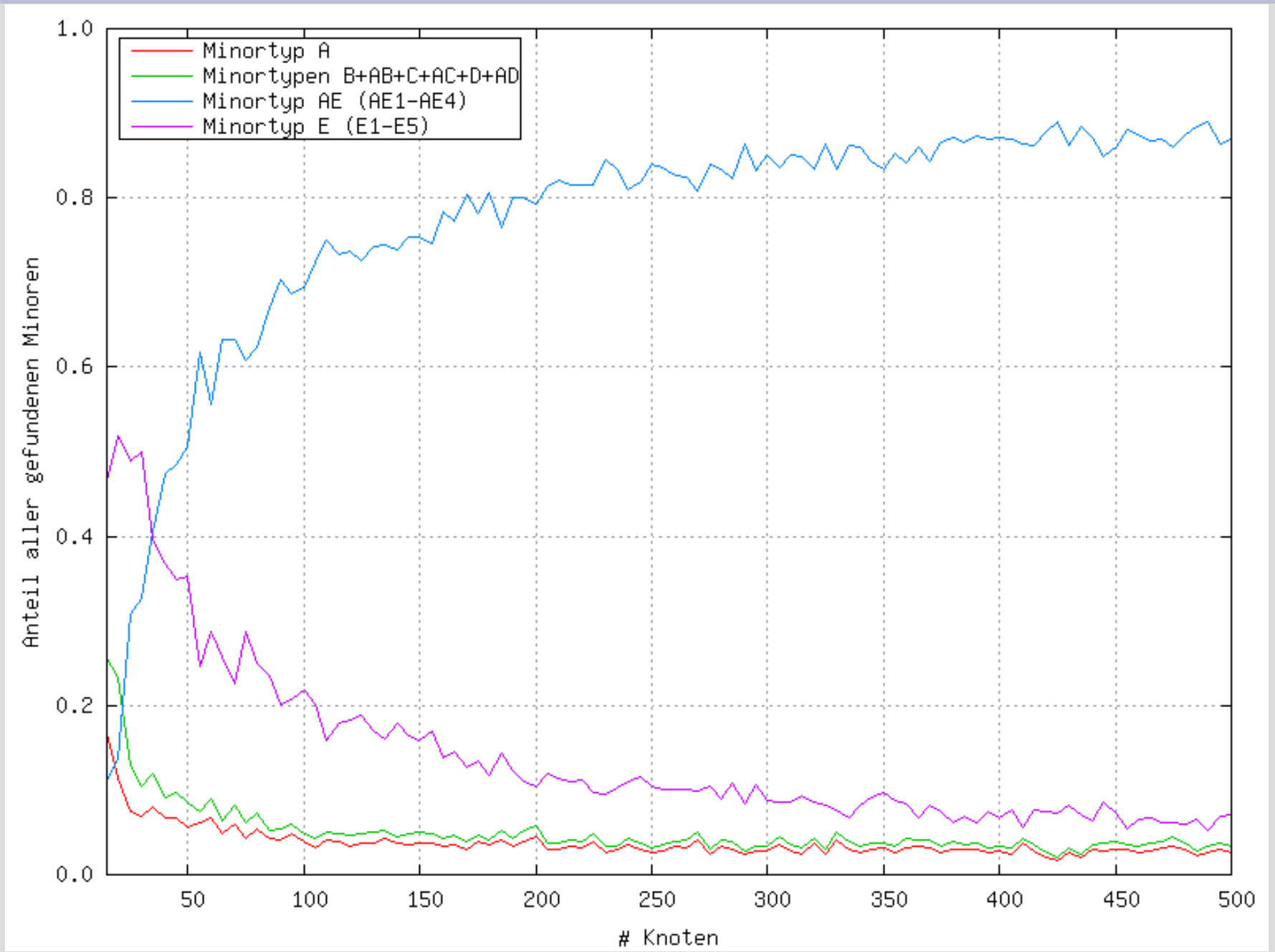
Rome Library – #Subdivisions



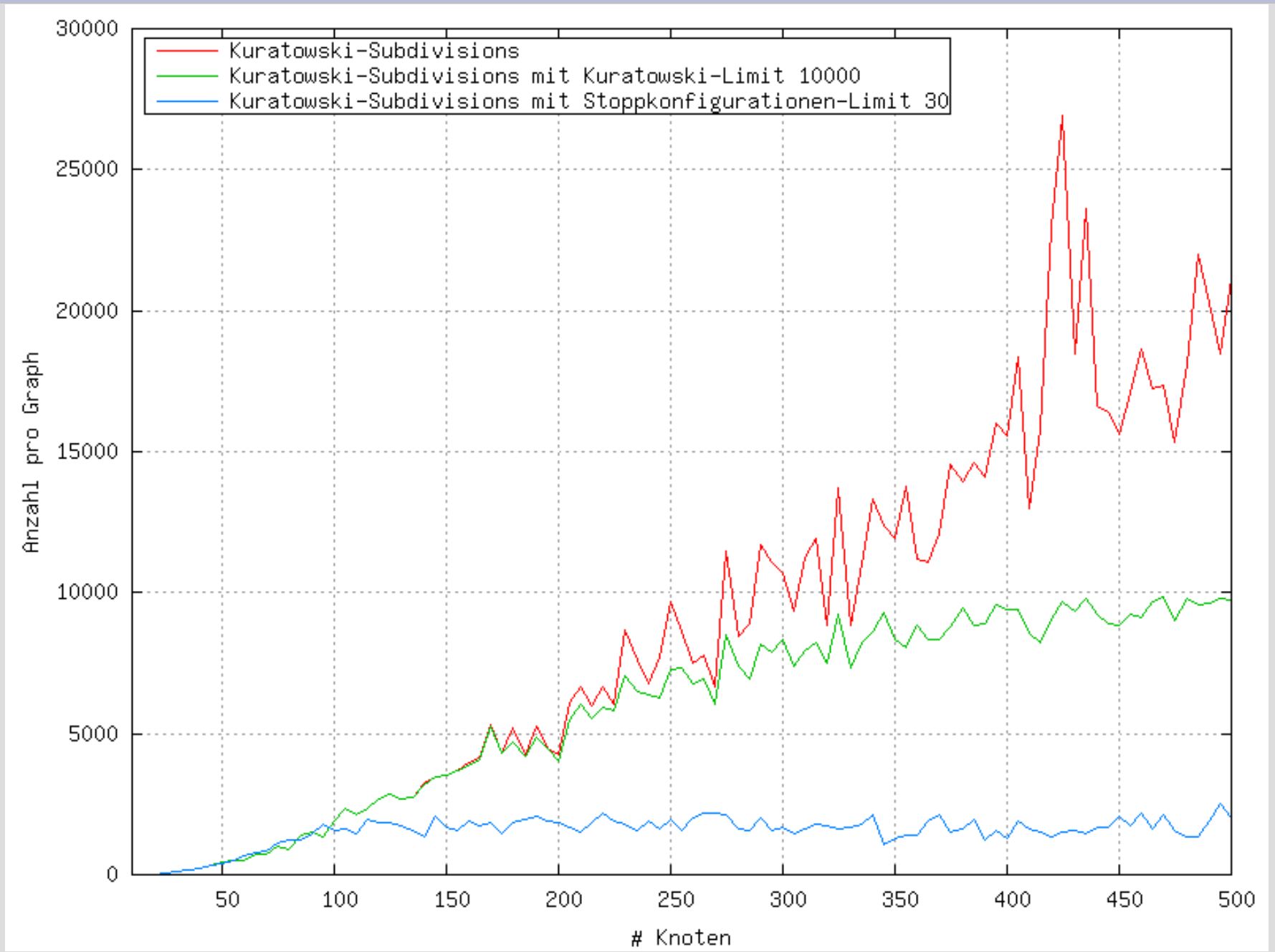
Rome Library – Subdivisions



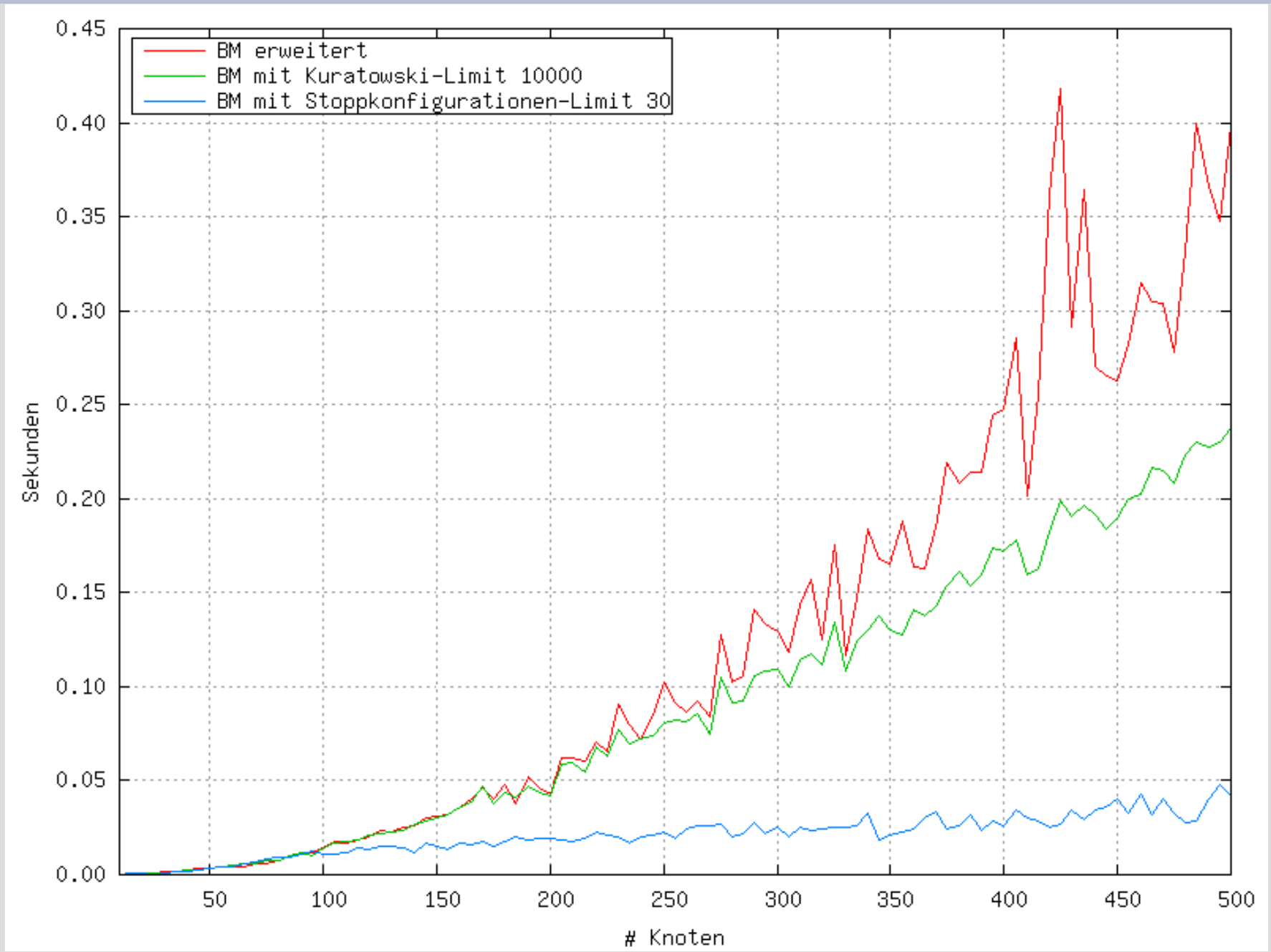
Zufallsgraphen – Subdivisions



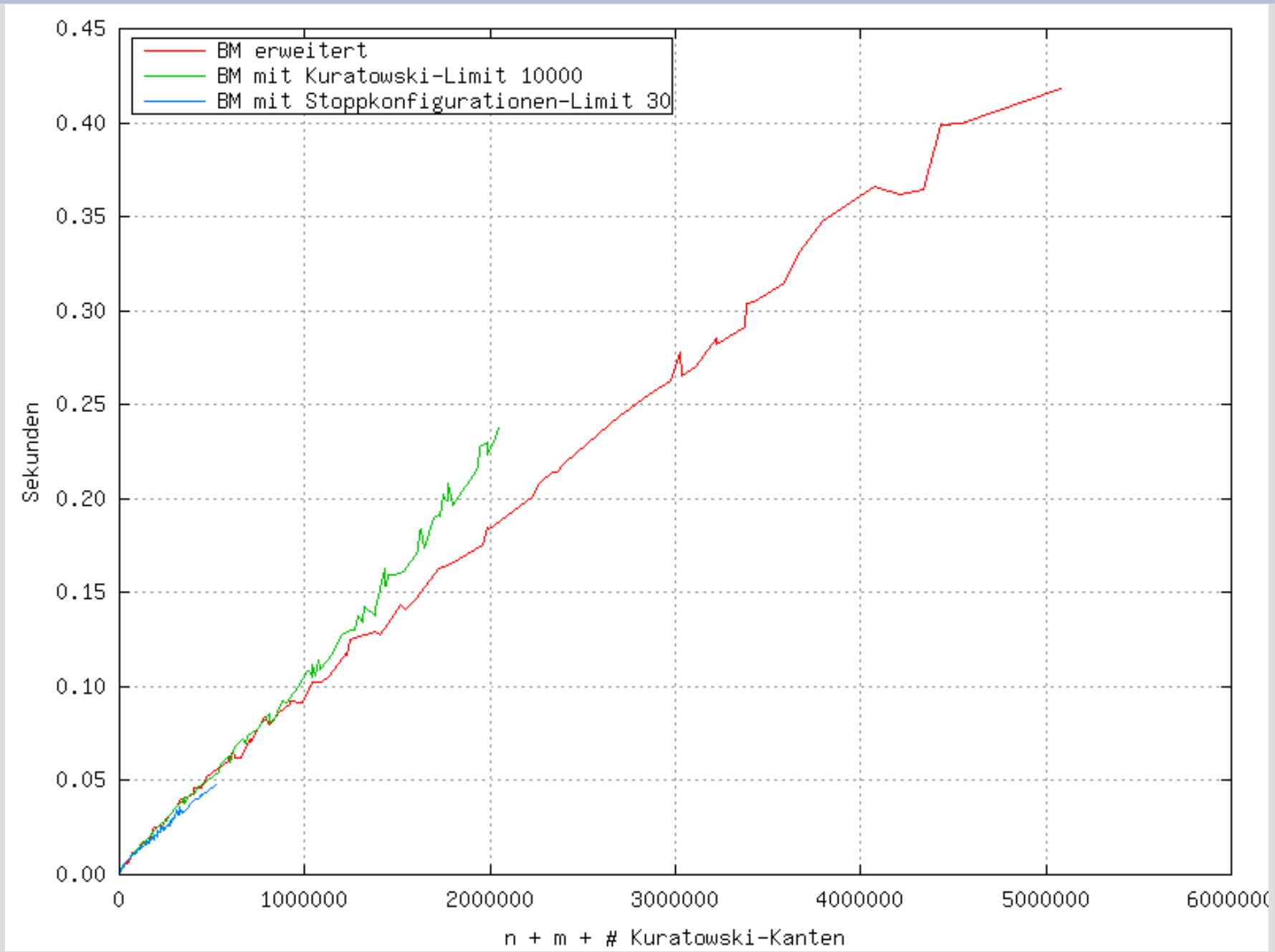
Zufallsgraphen – #Subdivisions



Zufallsgraphen – Laufzeiten



Zufallsgraphen – Laufzeit



Literatur

John M. Boyer, Wendy Myrvold:
On the Cutting Edge: Simplified O(n) planarity by edge addition
JGAA, 8(3) pp. 241-273, 2004

Battista, Boyer, Cortese, Patrignani:
Stop Minding Your P's and Q's: Implementing a Fast and Simple DFS-based Planarity Testing and Embedding Algorithm.
Technical Report RT-DIA-83-2003, 2003

Frank Harary:
Graph Theory
Addison Wesley Publishing, 1996, 6th printing

Reinhard Diestel:
Graph Theory (3rd electronic Edition, auch in Deutsch verfügbar)
http://www.math.uni-hamburg.de/home/diestel/books/graph.theory

Literatur

...the end...