Seminar – Maschinelles Lernen

TECHNISCHE UNIVERSITÄT DARMSTADT

Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rootet Trees

Referent: Moritz Huisman

Agenda

- Motivation
- Grundlagen
- Algorithmus
  - Mining frequent ordered Subtrees
  - Mining frequent unordered Subtrees
- Experimentelle Ergebnisse
- Fazit

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

- Motivation
- Graphentheoretische Grundlagen
- Algorithmus
  - Mining frequent ordered Subtrees
  - Mining frequent unordered Subtrees
- Experimentelle Ergebnisse
- Fazit

Die Motivation

- Bäume zur Datenrepräsentation
  - XML-Dokumente
  - Web-Access-Trees
  - Analyse molekularer Evolution

Die Motivation (2)

- Anwendungen für häufige Teilbäume
  - Indizierung- & Zugriffsmethodendesign bei Datenbanken
  - Clustering & Klassifikation
  - Gewinnung von allgemeinen Informationen

Agenda

- Motivation
- Grundlagen
- Algorithmus
  - Mining frequent ordered Subtrees
  - Mining frequent unordered Subtrees
- Experimentelle Ergebnisse
- Fazit

Grundlagen: Graphentheorie (1)

- Graph $G = [V, E, \Sigma, L]$ mit Labelfunktion $L : V \cup E \to \Sigma$

- *ungeordneter* (Wurzel)Baum ist ein azyklischer Graph für den gilt:
  - ∃! spezieller Knoten ohne eingehende Kanten
  - jeder andere Knoten hat genau eine eingehende Kante
  - eindeutiger Pfad von der Wurzel zu jedem anderen Knoten

- *geordneter* Baum: Definierte Ordnung über die Kinder

Grundlagen: Graphentheorie (2)

- Baum *t* wird als Teilbaum von Baum *s* bezeichnet, gdw.
  - $V_t \subseteq V_s$
  - $E_t \subseteq E_s$
  - Labels von *E* und *V* aus s werden in t erhalten

- Wenn Anzahl der Knoten in *t* < der Knoten in *s*, bezeichnet man t als *echten* Teilbaum

Grundlagen: Mining häufiger Teilbäume

TECHNISCHE UNIVERSITÄT DARMSTADT

- D bezeichnet eine Datenbank
- Jede Transaktion s $\in$ D ist ein gelabelter Baum
  - s kann *geordnet* oder *ungeordnet* sein

- gelabelter Teilbäume (Muster) t, die abhängig von D *geordnet* oder *ungeordnet* ist
  - $\sigma_t(s) = 1$ wenn t ein Teilbaum von s, sonst 0.
  - $support(t) = \sum_{s \in D} \sigma_t(s)$
  - *t* wird als *häufig* bezeichnet, wenn *support(t)* ≥ *minsup*

Grundlagen: Abgeschlossen & Maximal häufige Teilbäume

- häufiger Teilbaum *t* ist *abgeschlossen*, wenn keiner seiner echten Superbäume den gleichen *support* wie *t* hat
- häufiger Teilbaum *t* ist *maximal*, wenn keiner seiner echten Superbäume häufig ist

  $\rightarrow \quad M \subseteq C \subseteq F$

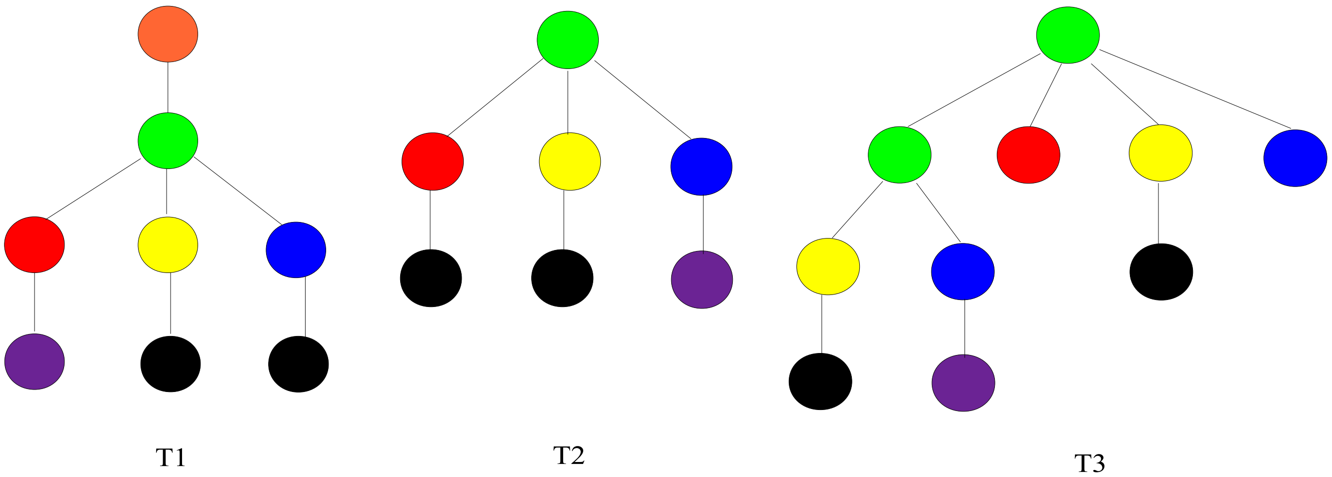
- Meist gibt es weniger abgeschlossen & maximal häufige Teilbäume als häufige Bäume
- Durch das Mining nach abgeschlossen & maximal häufigen Teilbäumen geht „kaum“ Information verloren
  - die Gesamtanzahl von t ist <u>nicht</u> verfügbar

Agenda

- Motivation
- Graphentheoretische Grundlagen
- Algorithmus
  - Mining ordered Subtrees
  - Mining unordered Subtrees
- Experimentelle Ergebnisse
- Fazit

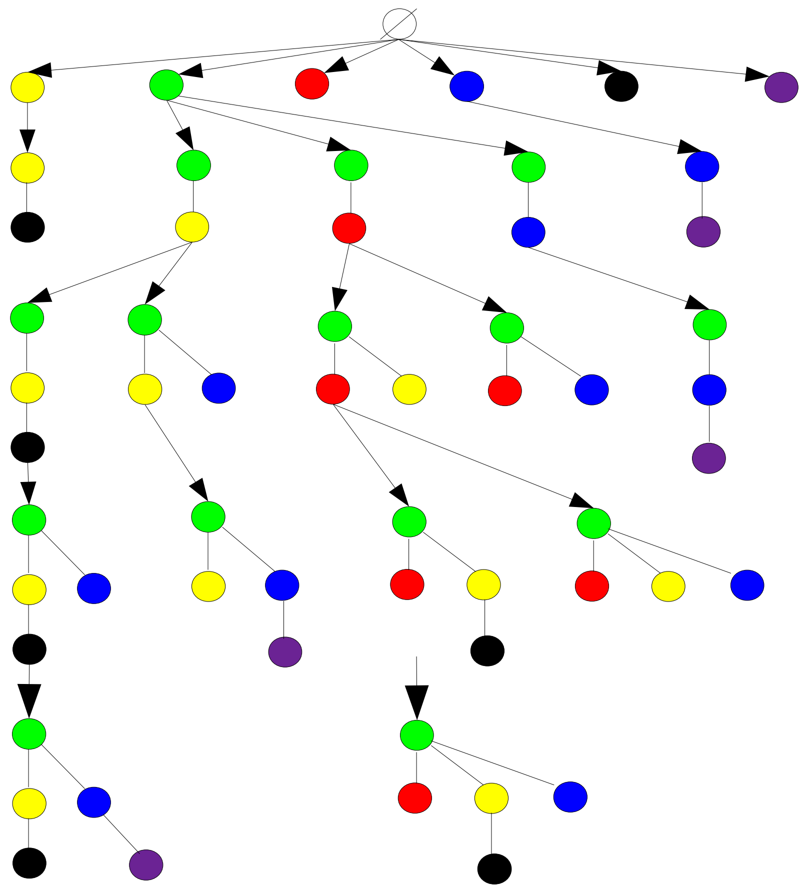
Input

- Datenbank mit 3 Transaktionen:



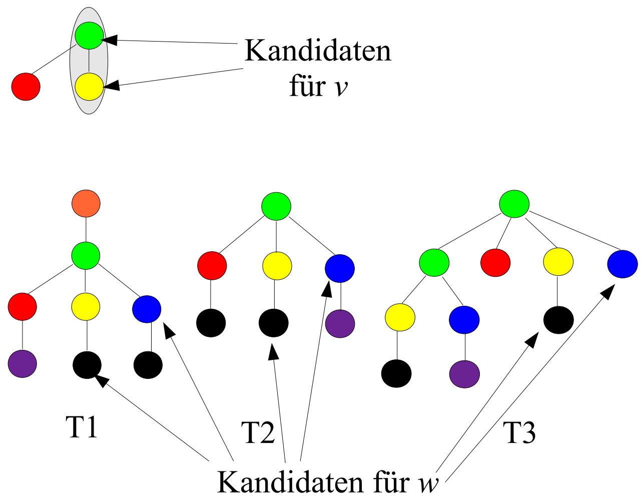
- *minsup* = 2

Enumeration Tree

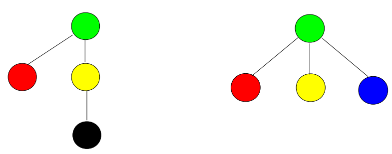


- dient der systematischen Aufzählung aller häufigen Teilbäume
- Teilbaumkandidaten werden aus ihrem eindeutigen Elternknoten generiert
- dazu wird eine *Rightmost Extension* durchgeführt

Rightmost Extension



Es resultieren also folgende Nachfolger:



- um Knoten w zu einem häufigen Teilbaum t hinzufügen zu können, muss der Elternknoten v von w auf dem *rightmost path* von t liegen
- w muss im neuen Teilbaum t' der letzte Knoten nach DFS-Traversierung sein
- t' muss häufig sein

The Blanket

TECHNISCHE UNIVERSITÄT DARMSTADT

- für einen häufigen Teilbaum t bezeichnen wir die Menge aller unmittelbaren, häufigen Nachfolger als B_t
  - $t' \in B_t$, mit genau einem Knoten mehr als t

- $t' \setminus t$ bezeichne den zusätzlichen Knoten w, Label & Position

Maximalität & Abgeschlossenheit

- Definition von Maximalität und die Abgeschlossenheit häufiger Teilbäumen mit Hilfe der Blankets:
  - ein häufiger Teilbaum ist maximal, gdw. $B_t = \emptyset$
  - Ein häufiger Teilbaum ist abgeschlossen, gdw. für jedes $t' \in B_t$ gilt: *support(t') < support(t)*
- Abgeschlossenheit und Maximalität lassen sich also durch die Blankets bestimmen, anstatt eine Nachbearbeitung vorzunehmen

Das Pruning

TECHNISCHE UNIVERSITÄT DARMSTADT

- nicht alle Zweige des Enumeration Trees enthalten abgeschlossen und/oder maximal häufige Teilbäume

- Occurrence-Matching:
  - für ein $t' \in B_t$ werden t und t' als occurrence-matched bezeichnet, wenn für jedes Vorhandensein von t in einer Transaktion der Datenbank wenigstens ein zugehöriges Vorhandensein von t' besteht

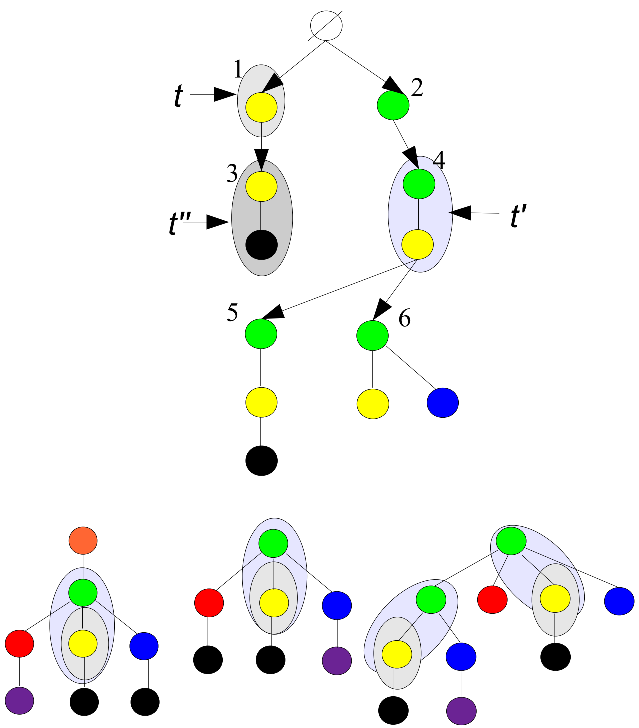
- Transaction-Matching:
  - für ein $t' \in B_t$ werden t und t' als transaktion-matched bezeichnet, wenn für jede Transaktion s $\in D$ mit $\sigma_t(s) = 1$, dann $\sigma_{t'}(s) = 1$

Left-Blanket Pruning

- Blanket lässt sich in Left-Blanket $B_{\text{t\_left}}$ und Right-Blanket $B_{\text{t\_right}}$ unterteilen.
  - $B_{\text{t\_right}} = \{t' \in B_t \mid$ t' ist ein Kind von t im Enumeration Tree$\}$
  - $B_{\text{t\_left}} = B_t \setminus B_{\text{t\_right}}$

- Theorem 1: Wenn für einen häufigen Teilbaum t im Enumeration Tree ein $t' \in B_{\text{t\_left}}$, so dass t und t' occurrence-matched sind, dann sind weder t noch dessen Nachfahren im Enumeration Tree abgeschlossen und deswegen kann t vom Enumeration Tree abgeschnitten werden.

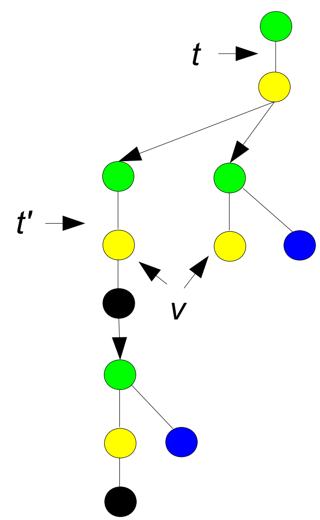
Left-Blanket Pruning (2)

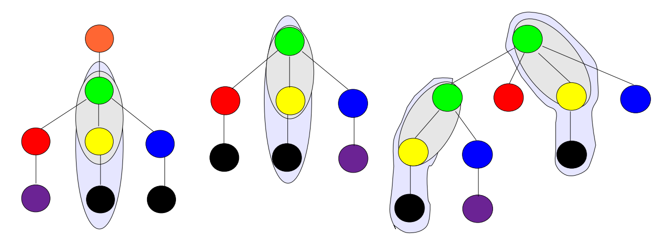


- Theorem 1 folgt aus Lemma 6: If ∃ ein häufigen Teilbaum t im Enumeration-Tree und ∃ $t' \in B_{t\_left}$ existiert, so dass t und t' occurrence-matchen, dann ist 1) t nicht abgeschlossen und 2) für jedes $t'' \in B_{t\_right}$ existiert wenigstens ein Superbaum $t''' \in B_{t''\_left}$, so dass t'' und t''' occurrence-matchen

Right-Blanket Pruning (1)

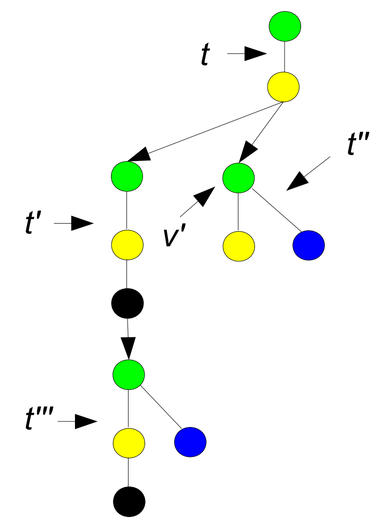
- wenn ein Occurrence-matching mit t' in $B_{\text{t\_right}}$ auftritt, kann t nicht gepruned werden
- je nach Position von $t' \setminus t$ aber Kinder von t
- Theorem 2: Wenn für einen häufigen Teilbaum t im Enumeration-Tree ein $t' \in B_{\text{t\_right}}$ existiert, so dass t und t' occurrence-matched sind und der Vaterknoten von $t' \setminus t$ v ist, dann braucht t nicht um weitere rightmost Vertexes an echten Vorfahren von v erweitert werden

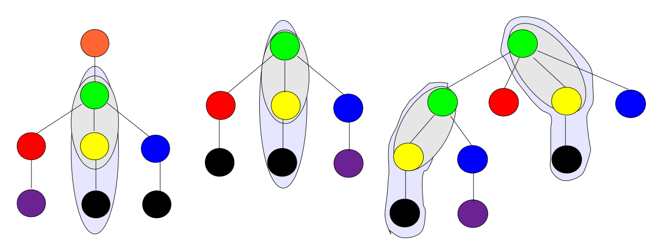




Right-Blanket Pruning (2)

- Wenn nun $t'' \in B_{\text{t\_right}}$ mit Vaterknoten v' von $t'' \setminus t$, der Vorfahr von v ist, existiert, dann ist v auf dem Leftmost-Path von t''. Da $t' \setminus t$ Kindknoten von v ist, ist er ebenso auf dem Leftmost-Path. Also muss ein $t''' \in B_{\text{t''\_left}}$ existieren, so dass t'' und t''' occurrence-matchen





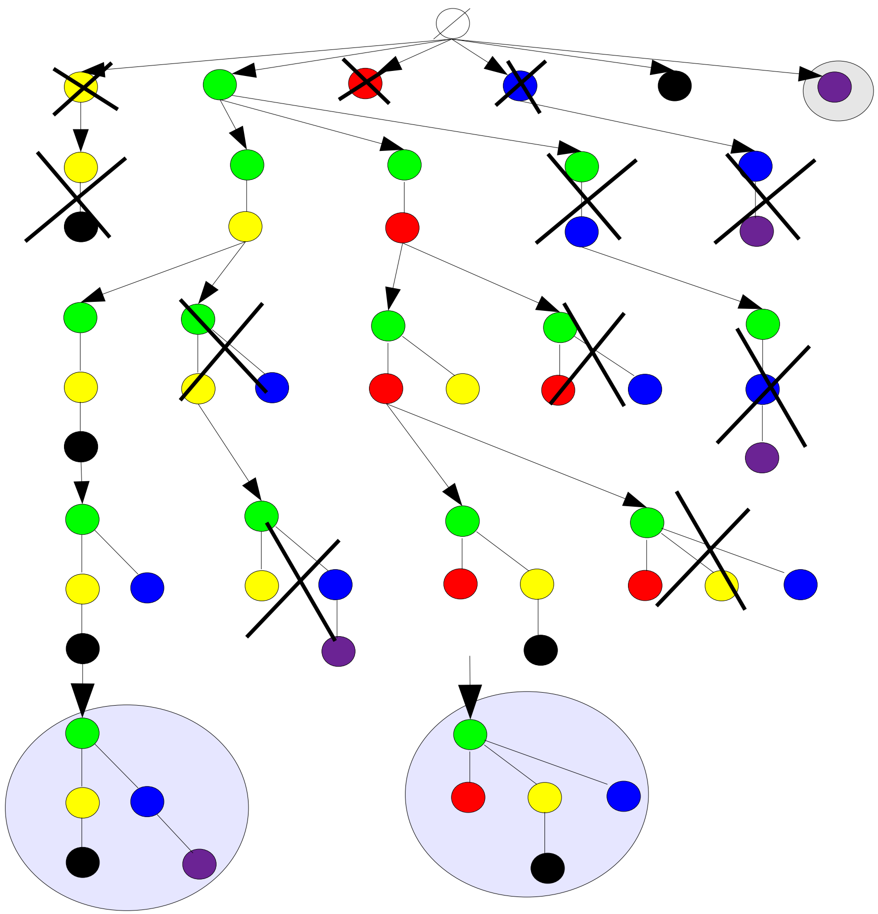
Teilmengen der Blankets

TECHNISCHE UNIVERSITÄT DARMSTADT

- In machen Fällen wird nur eine Teilmenge von B_t benötigt
- Unterscheidung der Teilmengen
  - B_{t}^{OM} = {t' ϵ B_t | t' und t sind occurrence-matched}
  - B_{t}^{TM} = {t' ϵ $B_t \setminus B_{\mathrm{t}}^{OM}$ | t' und t sind transaktion-matched}
  - $B_{\mathrm{t}}^{F} = B_t \setminus (B_{\mathrm{t}}^{OM} \cup B_{\mathrm{t}}^{TM})$
- Diese Mengen werden weiterhin in left und right eingeteilt, wie vorhin gesehen
- ∃ t' ϵ $B_{\mathrm{t\_left}}^{OM}$ → t kann abgeschnitten werden
- ∃ t' ϵ $B_{\mathrm{t\_right}}^{\bar{O}M}$ → evtl. können einige t' abgeschnitten werden. Außerdem ist t nicht abgeschlossen

- Wenn B_{t}^{TM} = Ø, dann ist t abgeschlossen
- Wenn B_{t}^{F} = Ø, dann ist t maximal

Ergebnis des Algorithmus



Berechnung der Teilmengen

- B_t^{OM} = Schnitt über alle Superbaumkandidaten aus jedem Auftreten von t
- B_t^{TM} = Schnitt aus der Vereinigung der Superbaumkandidaten aus den Transaktionen
- B_t^{F} = Vereinigung aller Superbaumkandidaten
  - außerdem muss der Support aller Kandidaten berechnet und gespeichert werden

Heuristische Berechnungsreihenfolge

1. Berechne B_t^{OM}. If $\exists t' \in B_{\mathrm{t\_left}}^{OM}$ then prune t
   else if $\exists t' \in B_{\mathrm{t\_right}}^{OM}$ then wende Theorem 2 an.
2. If $B_t^{OM} = \emptyset$ berechne B_{t}^{TM} . $B_t^{TM} \neq \emptyset$ ist t nicht abgeschlossen (also muss B_t^F nicht berechnet werden) else ist t abgeschlossen
3. Untersuche t im Enumeration-Tree. If irgendein Kind t' von t häufig ist, muss B_t^F nicht berechnet werden, da t nicht maximal ist.
4. If $B_{\mathrm{t\_right}}^{OM} = \emptyset$ und $B_{\mathrm{t}}^{TM} = \emptyset$ und keine Kinder von t im Enumeration-Tree häufig sind, berechne $B_{\mathrm{t\_left}}^F$. If $B_{\mathrm{t\_left}}^F = \emptyset$ then t ist maximal, else t ist closed

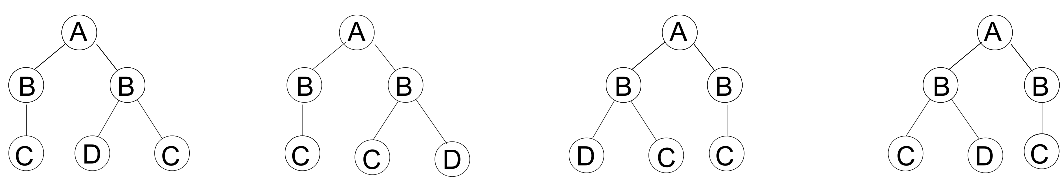
Agenda

- Motivation
- Graphentheoretische Grundlagen
- Algorithmus
  - Mining ordered Subtrees
  - Mining unordered Subtrees
- Experimentelle Ergebnisse
- Fazit

Depth-First Canonical Form

- Definition zweier Symbole für Depth-First String Encoding
  - $ $\stackrel{\text{def}}{=}$ Backtrack
  - # $\stackrel{\text{def}}{=}$ Ende der Codierung

- Rekursive Definition der DFCF:
  1. Einzelner Knoten ist trivial
  2. Sei r die Wurzel des ungeordneten Baums t mit N Kindern $r_1, \ldots, r_N$. Zuerst DFCF's für die Teilbäume $t_{r_1}, \ldots, t_{r_N}$ bestimmen. Diese werden von links nach rechts in lexikographischer Ordnung ihrer Depth-First String Encoding's“ sortiert



ABC$$BD$C# *ABC$$BC$D#* *ABD$C$$BC#* *ABC$D$$BC#*

Anpassung des Blanket-Konzepts

- Zuteilung zu $B_{\mathrm{t\_left}}$ und $B_{\mathrm{t\_right}}$ muss verfeinert werden
  - neben der Position von $t' \setminus t$ ist dessen Label relevant

  - dazu erfolgt Berechnung des Bereichs, den die Label annehmen können
- Diese Berechnung wird auch bei der Erweiterung von t im Enumeration-Tree benötigt

Erweitertes Right-Blanket Pruning

- Theorem 3: Wenn für einen häufigen Teilbaum t im Enumeration-Tree häufiger ungeordneter Teilbäume ein $t' \in B_{t\_right}$ existiert, so dass t und t' occurrence-matched sind und v der Vaterknoten von $t' \setminus t$ ist, dann
  1. muss t nicht durch weitere Knoten an Vorfahren von v erweitert werden
  2. muss t nicht durch weitere Knoten an v erweitert werden, deren Label lexikographisch größer als $t' \setminus t$ ist

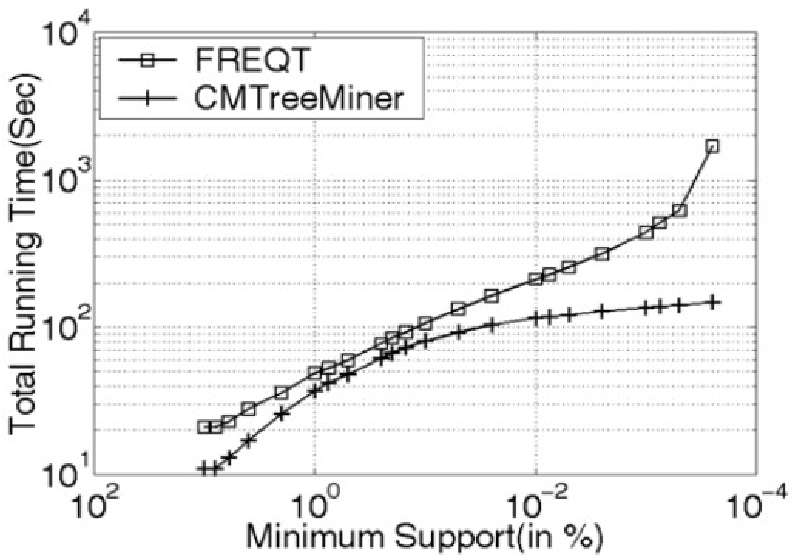
Agenda

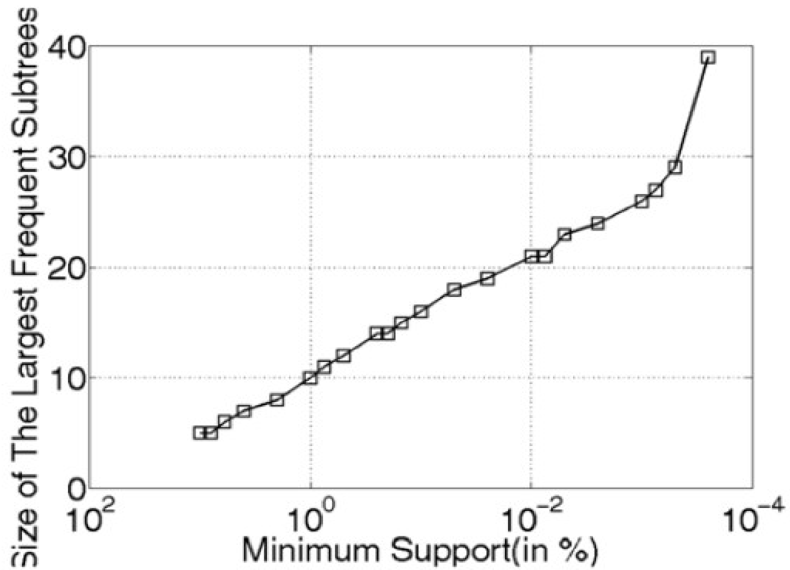
- Motivation
- Graphentheoretische Grundlagen
- Algorithmus
- **Experimentelle Ergebnisse**
- Fazit

Experimente mit geordneten Bäumen (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

- synthetische Daten
- Kosten für Nachbearbeitung bei FREQT werden vernachlässigt
- 1.000.000 Transaktionen; durchschnittlich 6,94 Knoten pro Baum; 100 verschiedene Knotenlabels

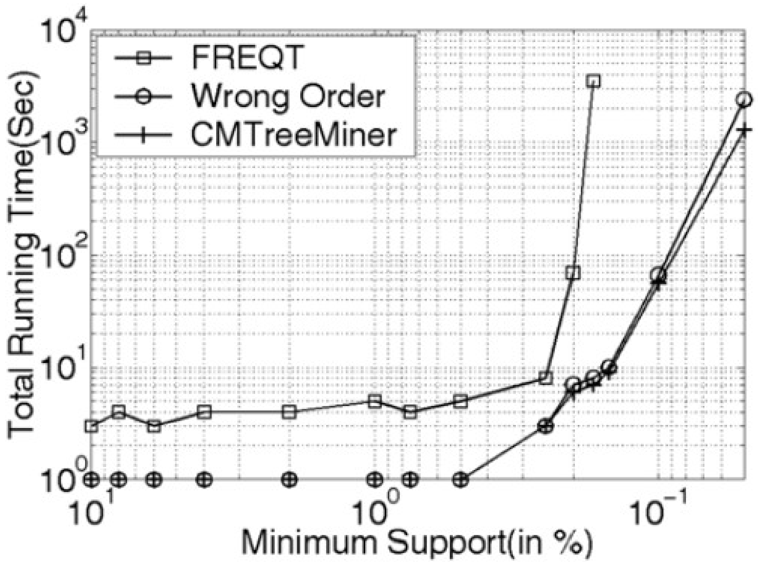


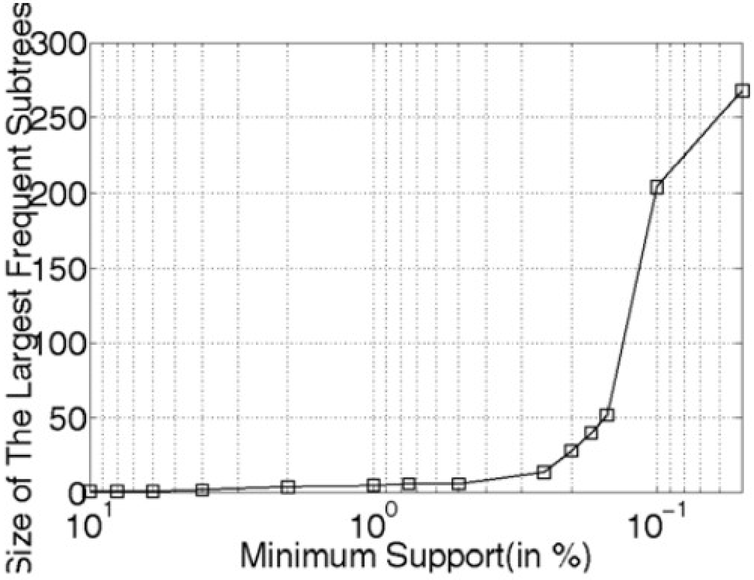


Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rootet Trees

Experimente mit geordneten Bäumen (2)

- Zugriffsbäume der Nutzer der Website des Fachbereichs Informatik am RPI (Rensselaer Polytechnic Institute (RPI))
- 59.691 User Access Trees, die 13.361 eindeutige Labels/Web Pages
- Durchschnittlich 12 Knoten pro Baum

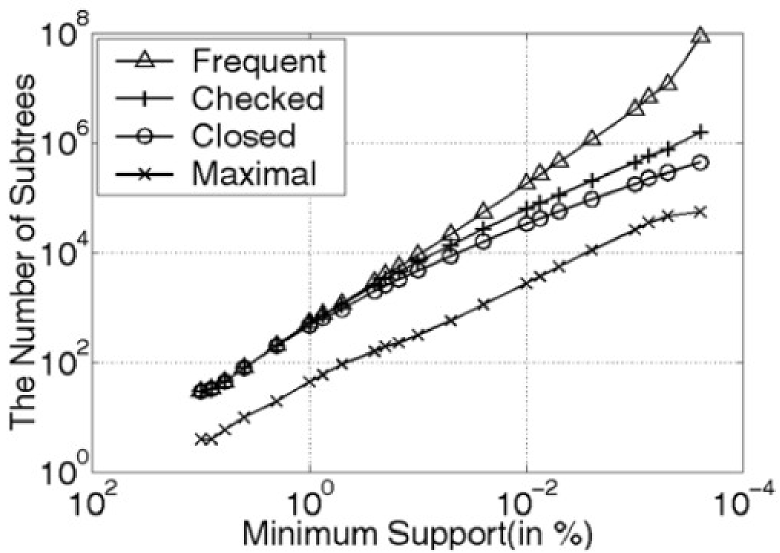


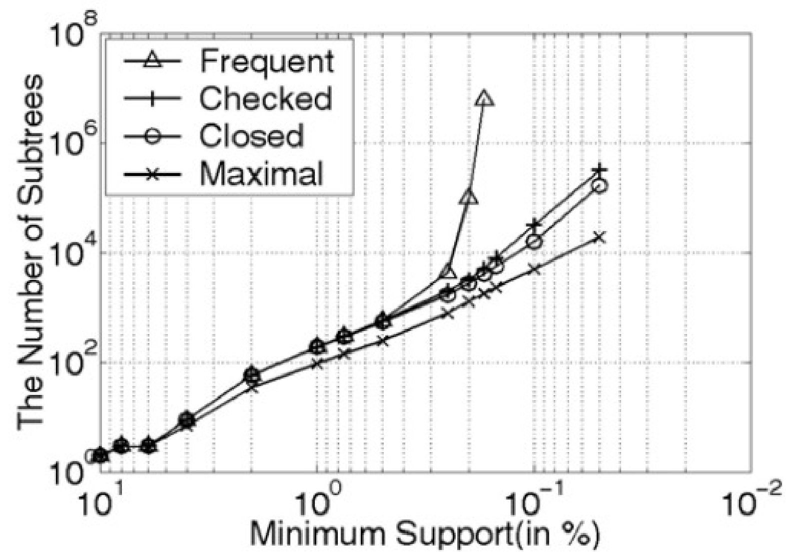


Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rootet Trees

Steigende Anzahl häufiger Teilbäume bei sinkendem Minsup

- Zeiteinsparungen von CMTreeMiner lassen sich durch geringere Anzahl abgeschlossen häufiger Teilbäume gegenüber der Anzahl häufiger Teilbäume erklären
- Geringere Anzahl geschlossen häufiger Teilbäume können durch Korrelation zwischen den häufigen Teilbäumen erklärt werden. Diese Korrelationen bestehen in realen Anwendungen

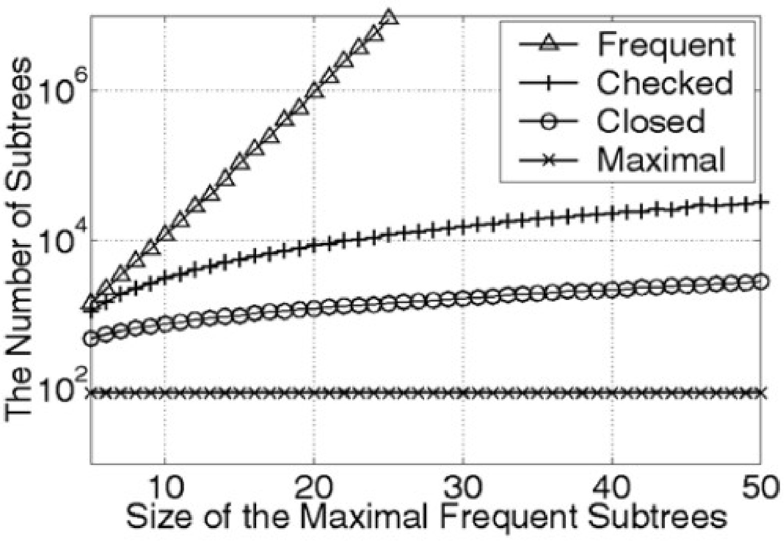


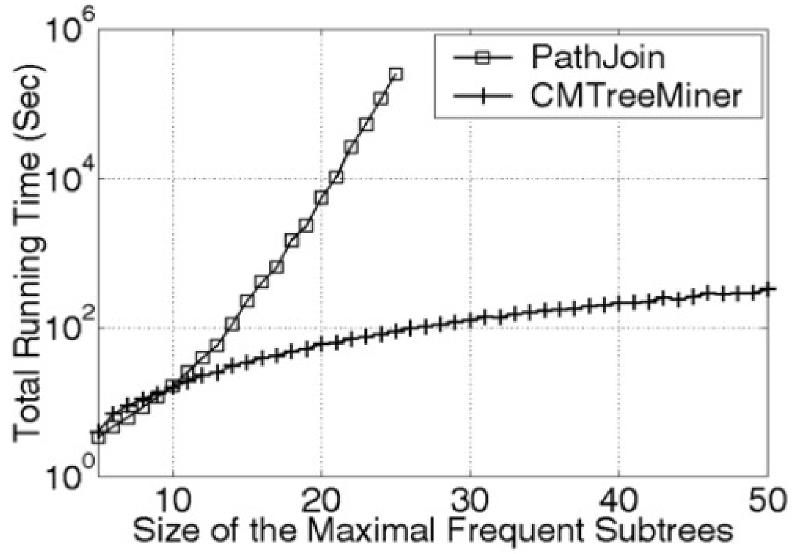


Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rootet Trees

Experimente mit ungeordneten Bäumen (1)

- exponentieller Anstieg der Anzahl aller häufigen Teilbäume(untere Schranke für die Anzahl an Bäumen, die PathJoin überprüft)
- Anzahl Teilbäume, die von CMTreeMiner überprüft werden wächst polynomial
- synthetische Datenmenge: 100.000 Transaktionen á 50 Knoten
- Minimum Support = 1%

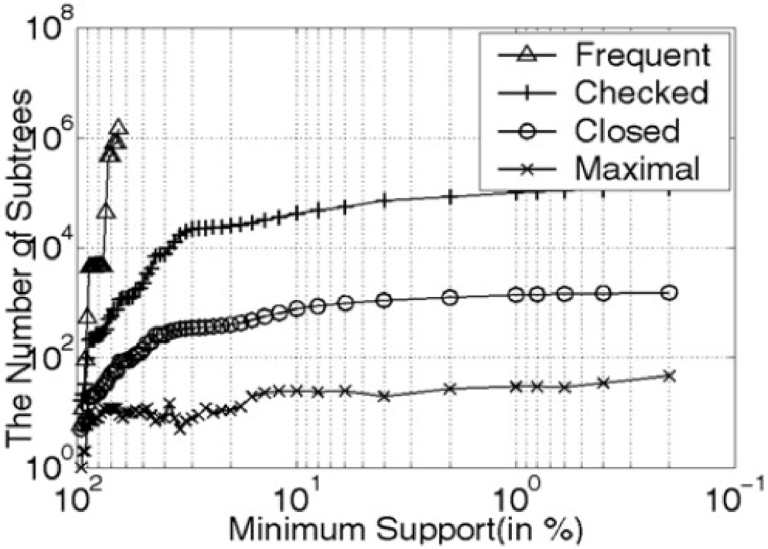


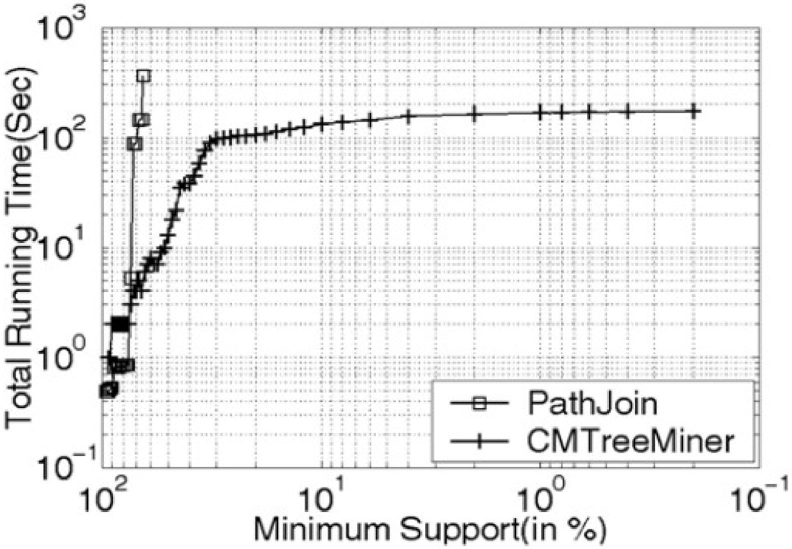


Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rootet Trees

Experimente mit ungeordneten Bäumen (2)

- Daten bestehen aus IP Multicast-Trees, die während eines NASA Shuttlestarts gemessen wurden
- 1000 Transaktionen; 333 Knoten\Labels





Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rootet Trees

Agenda

- Motivation
- Graphentheoretische Grundlagen
- Algorithmus
- Experimentelle Ergebnisse
- Fazit

Fazit

- Erster Algorithmus, der statt einer Nachbearbeitung direkt das Mining abgeschlossener und maximal häufiger Teilbäume vornimmt
- Verwendung für geordnete und ungeordnete Bäume
- Pruning und heuristische Berechnungsreihenfolge verringern die Laufzeit des CMTreeMiner's insbesondere bei niedrigem minsup
- Durch den geringeren Rechenaufwand ist CMTreeMiner Algorithmen überlegen, die alle häufigen Teilbäume minen

Vielen Dank für eure Aufmerksamkeit

Anhang

Der Pseudo-Code (1)

Algorithm ***CMTreeMiner***(*D, minsup*)
1: $CL \leftarrow \emptyset$, $MX \leftarrow \emptyset$; 2: $C \leftarrow$ frequent 1-trees; 3: ***CM-Grow***(C, CL, MX, D, *minsup*); 4: return CL, MX;

Der Pseudo-Code (2)

Subroutine ***CM-Grow***(*C, CL, MX, D, minsup*)

for each $t \in C$ ***do***
  $E \leftarrow \emptyset$;
  compute B_t^{OM} ;
  if $B_t^{OM} = \emptyset$ ***then*** compute B_t^{TM} ;
  if $\exists t' \in B_{\text{t\_left}}^{OM}$ ***then continue***;
  else
    for each vertex v on the rightmost path of t ***do***
    (in a bottom-up fashion)
      for each valid new rightmost vertex w of t ***do***
        $t' \leftarrow t$ plus vertex w, with v as w´s parent;
        if support$(t') \geq minsup$ ***then*** $E \leftarrow E \cup t'$;
      if $\exists t' \in B_{\text{t\_right}}^{OM}$ s.t. v is the parent of $t' \backslash t$ ***then***
        break;
  if $E \neq \emptyset$ ***then CM-Grow***$(E, CL, MX, D, minsup)$;
  if $B_t^{OM} = \emptyset$ and $B_t^{TM} = \emptyset$ ***then***
    $CL \leftarrow CL \cup t$;
  if $E = \emptyset$ ***then***
    compute $B_{\text{t\_left}}^{F}$;
    if $B_{\text{t\_left}}^{F} = \emptyset$ ***then*** $MX \leftarrow MX \cup t$;
  break;

Enumeration DAG (Directed Acyclic Graph)

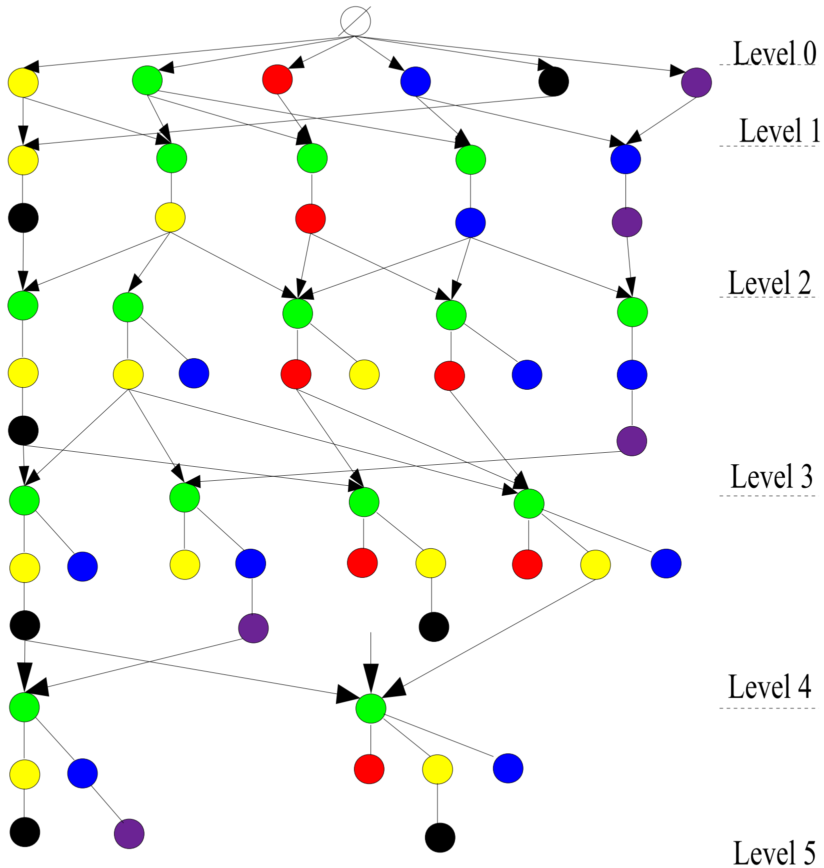


Abb. 2, in Anlehnung an Fig 2 (a) aus [1] Seite 4

- jedes Level enthält die häufigen Teilbäume der jeweiligen Größe
- gerichtete Kanten repräsentieren Teilbaum-Superbaum Beziehung, sie zeigen vom Teilbaum zum Superbaum
- Zusammenhang: Enumerationtree ist ein Spannbaum vom Enumeration DAG
- Kinder eines Teilbaums(Knotens) sind dessen Blanket

Speicherbedarf bei synthetischen Daten (ungeordnete Bäume)

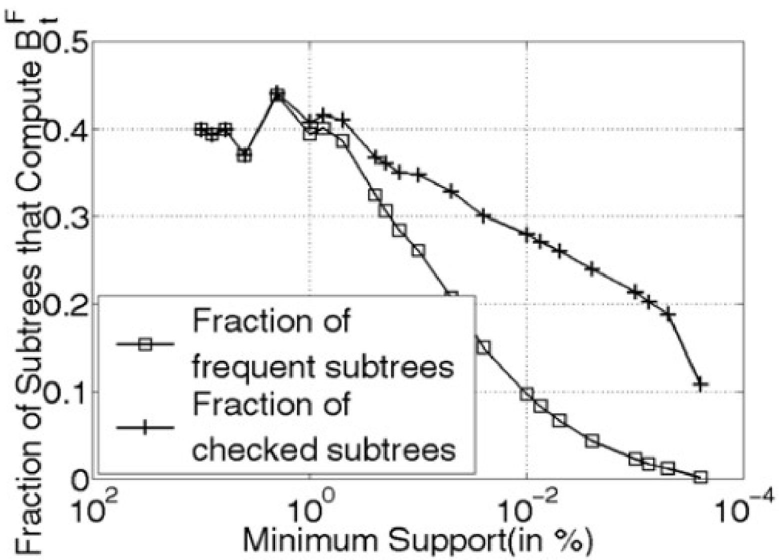
- PathJoin
  - |I| > 25 : 1GB HS ausgeschöpft
  - ~ 3 Tage Rechenzeit bei |I| = 25

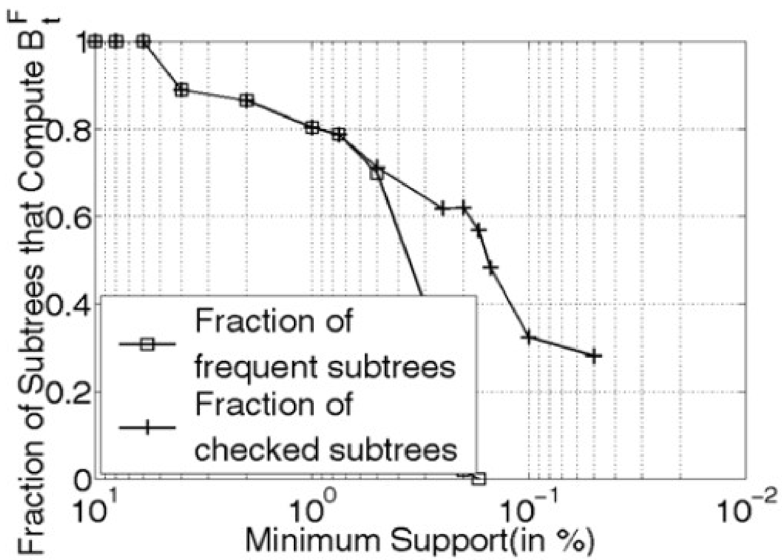
- CMTreeMiner
  - |I| = 26 : 302MB
  - |I| = 50 : 567MB
  - 90 Sekunden Rechenzeit bei |I| = 25

- **Lemma 1:** *Jeder Teilbaum eines häufigen Baumes ist häufig und jeder Superbaum eines nicht häufigen Baumes ist nicht häufig*
- **Lemma 2:** $M \subseteq C \subseteq F$
- **Lemma 3:** *Aus M können alle häufigen Bäume extrahiert werden. Gleichermaßen können aus alle häufigen Bäume mit deren Support aus C extrahiert werden*

Berechnung von B_t^F

- sinkende Support-Thresholds führen zu einem geringeren Anteil an Teilbäumen, für die B_t^F berechnet wird
- Autoren führen dies auf auf ein Pruning nahe der Wurzel zurück





Mining Closed and Maximal Frequent Subtrees from Databases of Labeled Rootet Trees

Speicherbedarf

Reale Datenmenge und Support-Threshold = 0,00025%

- FREQT
  - ~254MB
- CMTreeMiner
  - ~256MB

synthetische Daten und Support-Threshold = 0,17%

- FREQT
  - ~176MB
- CMTreeMiner
  - ~159MB

- FREQT speichert nur das Auftreten des „rightmost-Vertex“ pro häufigem Teilbaum
- CMTreeMiner speichert das Auftreten aller Knoten in allen häufigen Teilbäumen
- Performance von CMTreeMiner sollte bei sinkender Anzahl unterschiedlicher Labels sinken, da unterschiedliches Auftreten der gleichen Pattern Ansteigt und somit der Speicherbedarf
  → Thema für zukünftige Arbeit