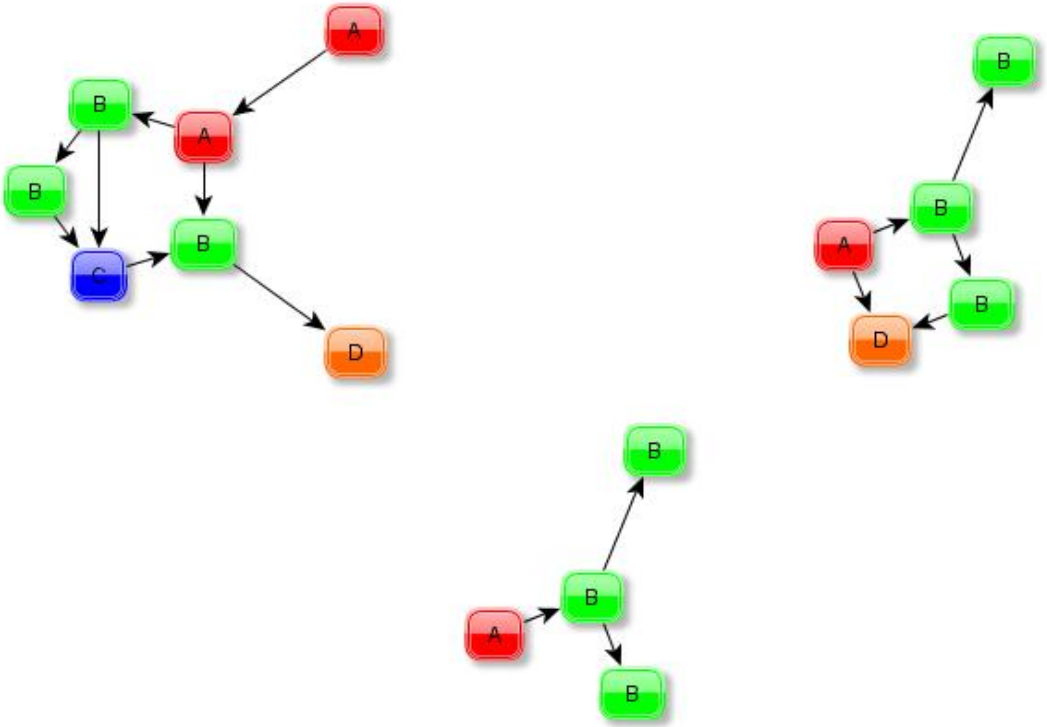
# Graph Mining

## Discovering Frequent Subgraphs
## Discovering Frequent Geometric Subgraphs

Angela Eigenstetter und Christian Wirth

TECHNISCHE UNIVERSITÄT DARMSTADT



# FSG und GFSG Allgemein

TECHNISCHE UNIVERSITÄT DARMSTADT

- Apriori Ansatz
  - Candidate Generation, Frequency Counting
  - Über eine Menge von Graphen
  - „Support" Grenze: min. Anzahl von Graphen die das Pattern beinhalten

- Isomorphie Test
  - über kanonische Labels (FSG)
  - Geometrische Transformationen (GFSG)

# Anwendungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Allgemeiner Ansatz
- Chemische Verbindungen
  - Knoten = Atom
  - Kante = Verbindung
  - Kantenlabel = Verbindungstyp
  - Häufige Subgraphen dienen der Klassifizierung
- Andere Anwendungen möglich
- Athlon MP 1800+ (1,53Ghz), 2Gb Ram

# Frequent Subgraph Discovery Algorithm

# Übersicht

TECHNISCHE UNIVERSITÄT DARMSTADT

- Allgemeine Informationen
- Candidate Generation
- Frequency Counting
- Ergebnisse

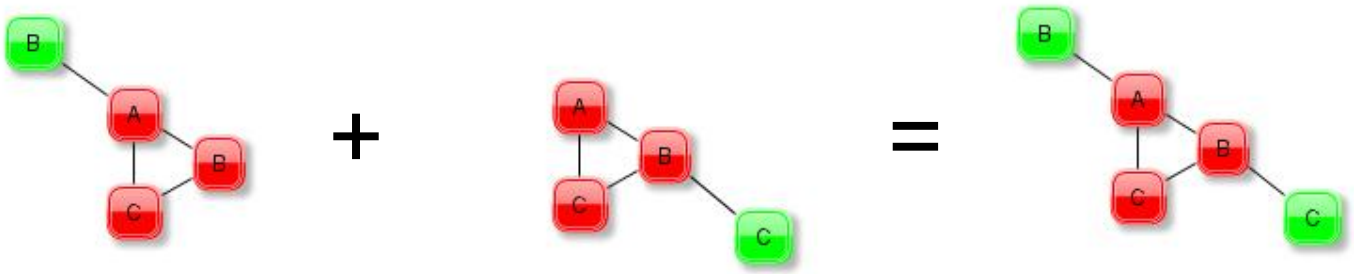
# Allgemein

TECHNISCHE UNIVERSITÄT DARMSTADT

- Generalisiert
- (3D-) Strukturinformationen als Kantenlabel
- Datenstrukturbasierte Verbesserungen

# Candidate Generation I

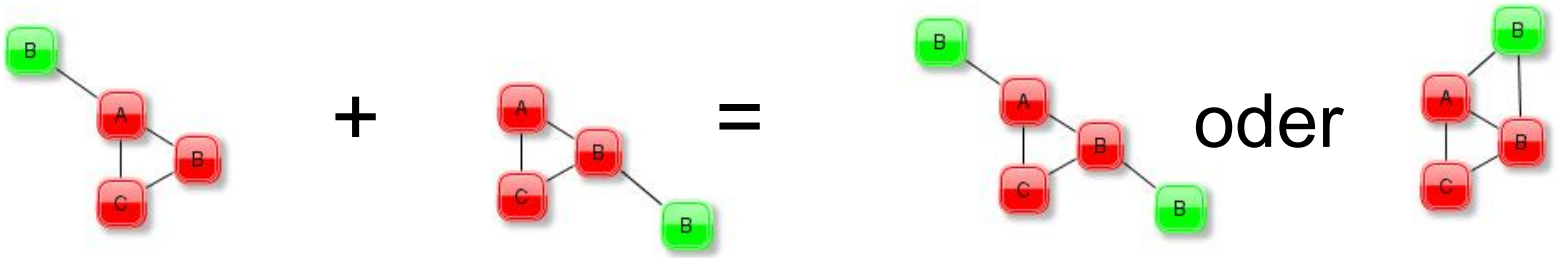
- Zusammenfassen zweier k-Subgraphen
  - Jede mögliche Kombination
  - Auch zwei gleiche Subgraphen
- Gemeinsamer k-1 Kern
  - Isomorph
  - Primärer Subgraph
- Erzeugt k+1 Subgraph



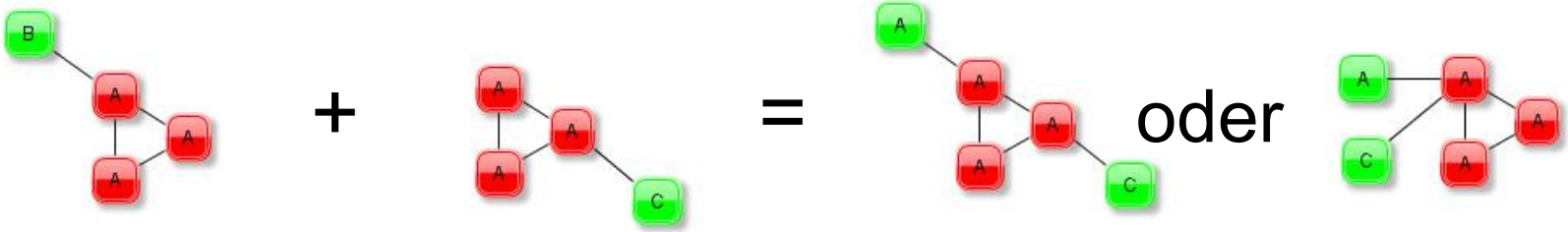
# Candidate Generation II

TECHNISCHE UNIVERSITÄT DARMSTADT

- Muss nicht eindeutig sein
  - Gleiches Label




  - Automorphismus



# Frequency Counting

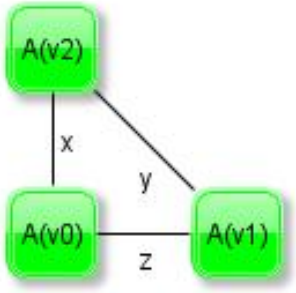
TECHNISCHE UNIVERSITÄT DARMSTADT

- Transaction Identifier List
  - Liste je Pattern
  - Beinhaltet alle Graphen die dieses Pattern enthalten

- Erstellen der k+1 Liste
  - Berechnen der Überschneidungen
    - Liste aller Graphen die in beiden Listen vorkommen
    - Länge < Grenzwert => Prune
  - Berechnen der Isomorphie
    - Canonical Labeling für alle Elemente
    - Neue TID List
    - Länge < Grenzwert => Prune

# Canonical Labeling I

- Allgemein
  - Nutzung der Knoten Invarianten
    - Grad
    - Label
    - Nachbarschaft




| | | v0 | v1 | v2 |
|---|---|---|---|---|
| | | A | A | A |
| v0 | A | | z | x |
| v1 | A | z | | y |
| v2 | A | x | y | |

Label: AAA zxy

- Erstellung
  - Kanonisches Label
    - Oberes Dreieck der Adjazenzmatrix
    - Labelreihe + Matrixspalten
    - Maximierung des Labels durch Permutation

| | | v1 | v0 | v2 |
|---|---|---|---|---|
| | | A | A | A |
| v1 | A | | z | y |
| v0 | A | z | | x |
| v2 | A | y | x | |

Maximiertes Label: AAA zyx

# Canonical Labeling II

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem: Berechnung aller Permutationen
  - O(|V|!)
  - Bereits bei kleinen Graphen problematisch
- Lösung: Partitionierung
  - Äquivalenzklassen
  - Knoten Invarianten
- Erstellung
  - Iterativ
  - Erst Grad & Label
  - Dann Nachbarschaft

# Canonical Labeling - Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

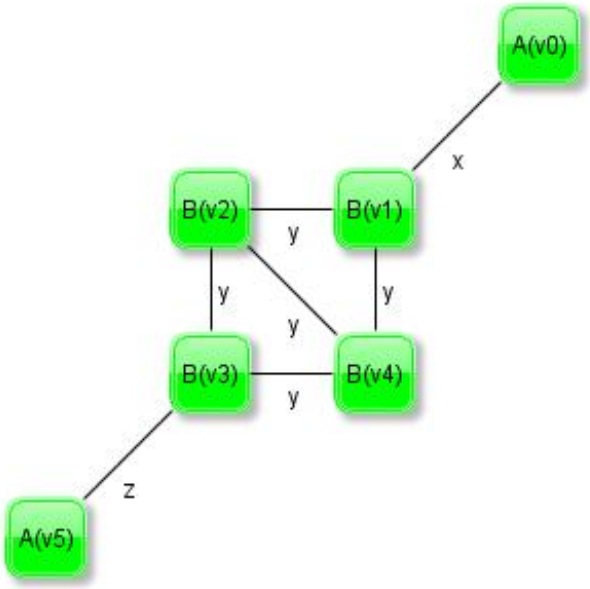
- 1. Partitionen nach Grad und Label
- 2. Repartitionierung nach Nachbarschaftsliste
- 3. Sortieren der Partitionen
- 4. Maximierung der Labels je Partition
- 5. Erstellen des Labels

# Canonical Labeling – Beispiel I

TECHNISCHE UNIVERSITÄT DARMSTADT

- Erstellen der Partitionen
  - Eine Partition je Grad und Label

1. ***Partitionierung***
2. Repartitionierung
3. Sortieren
4. Maximierung des Labels
5. Erstellen des Labels



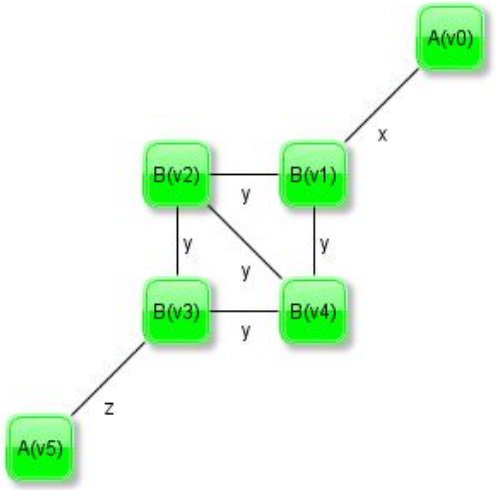

Partitionen:
- p0 : Label = B und Grad = 3
  - v1, v2, v3, v4
- p1 : Label = A und Grad = 1
  - v0, v6

# Canonical Labeling – Beispiel II

TECHNISCHE UNIVERSITÄT DARMSTADT

- Erstellen der Nachbarschaftslisten
  - Partitionsnummer
  - Edgelabel

1. Partitionierung
2. ***Repartitionierung***
3. Sortierung
4. Maximierung des Labels
5. Erstellen des Labels




Partitionen:
- p0 : Label = B und Grad = 3
  - v1, v2, v3, v4
- p1 : Label = A und Grad = 1
  - v0, v6

Nachbarschaften:
- p0
  - v1 : (p0,y),(p0,y),(p1,x)
  - v2 : (p0,y),(p0,y),(p0,y)
  - v3 : (p0,y),(p0,y),(p1,z)
  - v4 : (p0,y),(p0,y),(p0,y)
- p1
  - v0 : (p0,x)
  - v5 : (p0,z)

# Canonical Labeling – Beispiel III

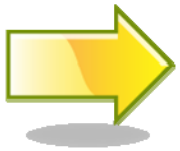
TECHNISCHE UNIVERSITÄT DARMSTADT

- Repartitionierung
  - Erstellen neuer Partitionen nach Nachbarschaft
  - Kann erneutes Repartitionieren verursachen

1. Partitionierung
2. ***Repartitionierung***
3. Sortieren
4. Maximierung des Labels
5. Erstellen des Labels

Partitionen:
- p0 : Label = B und Grad = 3
  - v1, v2, v3, v4
- p1 : Label = A und Grad = 1
  - v0, v6

Nachbarschaften:
- p0
  - v1 : (p0,y),(p0,y),(p1,x)
  - v2 : (p0,y),(p0,y),(p0,y)
  - v3 : (p0,y),(p0,y),(p1,z)
  - v4 : (p0,y),(p0,y),(p0,y)
- p1
  - v0 : (p0,x)
  - v5 : (p0,z)



Partitionen:
- p0 : Label = B, Grad = 3 und Nachbarschaft = (p0,y),(p0,y),(p0,y)
  - v2, v4
- p1 : Label = B, Grad = 3 und Nachbarschaft = (p0,y),(p0,y),(p1,x)
  - v1
- p2 : Label = B, Grad = 3 und Nachbarschaft = (p0,y),(p0,y),(p1,z)
  - v3
- p3 : Label = A und Grad = 1 und Nachbarschaft = (p0,x)
  - v0
- p4 : Label = A und Grad = 1 und Nachbarschaft = (p0,z)

# Canonical Labeling – Beispiel IV

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sortierung
  - Isomorphe Graphen haben gleiche Partitionen
  - Reinfolge egal solange Eindeutig
  - Sollte schnelles Pruning ermöglichen
    - Große Variationen am Anfang
    - Sortieren nach Grad

1. Partitionierung
2. Repartitionierung
3. ***Sortieren***
4. Maximierung des Labels
5. Erstellen des Labels

Partitionen:

- p0 : Label = B, Grad = 3 und
  Nachbarschaft = (p0,y),(p0,y),(p0,y)
- p1 : Label = B, Grad = 3 und
  Nachbarschaft = (p0,y),(p0,y),(p1,x)
- p2 : Label = B, Grad = 3 und
  Nachbarschaft = (p0,y),(p0,y),(p1,z)
- p3 : Label = A und Grad = 1 und
  Nachbarschaft = (p0,x)
- p4 : Label = A und Grad = 1 und
  Nachbarschaft = (p0,z)

p2 > p0 > p1 > p4 > p3

# Canonical Labeling – Beispiel V

TECHNISCHE UNIVERSITÄT DARMSTADT

- Maximierung des Labels
  - Partitionsreinfolge definiert
  - Reinfolge innerhalb der Partition
    - Maximierung
    - Vertauschung

- 1. Partitionierung
- 2. Repartitionierung
- 3. Sortieren
- ***4. Maximierung des Labels***
- 5. Erstellen des Labels

| | | v3 | v2 | v4 | v1 | v5 | v0 |
|---|---|---|---|---|---|---|---|
| | | B | B | B | B | A | A |
| v3 | B | | y | y | | z | |
| v2 | B | y | | y | y | | |
| v4 | B | y | y | | y | | |
| v1 | B | | y | y | | | x |
| v5 | A | z | | | | | |
| v0 | A | | | | x | | |
| | | p2 | p0 | | p1 | p4 | p3 |

P0 : mögliche Label
- yyy
- keine Permutationen möglich

# Canonical Labeling – Beispiel VI

TECHNISCHE UNIVERSITÄT DARMSTADT

- Erstellen des Labels
  - Reihe der Labels
  - Spaltenkonkatenation

1. Partitionierung
2. Repartitionierung
3. Sortieren
4. Maximierung des Labels
***5. Erstellen des Labels***

| | | v3 | v2 | v4 | v1 | v5 | v0 |
|---|---|---|---|---|---|---|---|
| | | B | B | B | B | A | A |
| v3 | B | | y | y | | z | |
| v2 | B | y | | y | y | | |
| v4 | B | y | y | | y | | |
| v1 | B | | y | y | | | x |
| v5 | A | z | | | | | |
| v0 | A | | | | x | | |
| | | p2 | p0 | | p1 | p4 | p3 |

Label:
BBBBAA yyy0yyz000000x0

# Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Apriori
  - Generieren der Kandidaten
    - Primäre Subgraphen
    - Mehrere Kandidaten
  - Häufigkeit testen
    - Transaction Identifier Lists
    - Isomorphie über Canonical Labels
  - Canonical Labels
    - Partitionen nach Vertex Invarianten
    - Maximieren des Labels

# Verbesserungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vertex Stabilisation
  - Große Partitionen haben O(k!) Permutationen
  - Kann verkleinert werden
  - Zufälliges Vertex mit neuer Partition
  - Erzeugt ~ (k/2) neue Partitionen
  - Muss für alle k Möglichkeiten berechnet werden
  - O(k(k/2)!) < O(k!)

- Datenbank Unterteilung
  - Unterteilen der Graphmenge
  - Berechnen der Häufigkeit je Teil
  - ∑ Häufigkeiten ergibt Obergrenze => Pruning
  - Speicherersparnis

# Ergebnisse

TECHNISCHE UNIVERSITÄT DARMSTADT

- Keine Vergleiche mit anderen Algorithmen

- Evaluation der Eigenschaften
  - Auf natürlichem Datenset
    - Verhältnis Graphen/Laufzeit
    - Verhältnis Support/Laufzeit
    - Speicherbedarf/Datenbank Unterteilungen
  - Auf synthetischem Datenset
    - Verhältnis Anzahl Vertexlabel/Laufzeit
    - Verhältnis Patterngröße/Laufzeit
    - Verhältnis Graphgröße/Laufzeit

- System: Athlon MP 1800+ (1,53Ghz), 2Gb Ram

# Ergebnisse I

TECHNISCHE UNIVERSITÄT DARMSTADT

- Natürliches Datenset
  - Predictive Toxicology Evaluation (PTE) Challenge
  - Developmental Therapeutics Program (DTP)
  - Beides Molekularstrukturen

| Support threshold | Runtime $t$[sec], Size of Largest Frequent Pattern $k^*$, and Number of Frequent Patterns $\|\mathcal{F}\|$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTE $\|\mathcal{D}\| = 340$ | | | DTP $\|\mathcal{D}\| = 50,000$ | | | DTP $\|\mathcal{D}\| = 100,000$ | | | DTP $\|\mathcal{D}\| = 200,000$ | | |
| [%] | $t$[sec] | $k^*$ | $\|\mathcal{F}\|$ | $t$[sec] | $k^*$ | $\|\mathcal{F}\|$ | $t$[sec] | $k^*$ | $\|\mathcal{F}\|$ | $t$[sec] | $k^*$ | $\|\mathcal{F}\|$ |
| 10.0 | 3 | 11 | 844 | 74 | 9 | 351 | 156 | 9 | 360 | 337 | 9 | 373 |
| 9.0 | 3 | 11 | 977 | 80 | 9 | 400 | 169 | 10 | 420 | 366 | 10 | 442 |
| 8.0 | 4 | 11 | 1323 | 87 | 11 | 473 | 184 | 11 | 490 | 401 | 11 | 512 |
| 7.0 | 4 | 12 | 1770 | 94 | 11 | 562 | 200 | 11 | 591 | 437 | 11 | 635 |
| 6.0 | 6 | 13 | 2326 | 109 | 12 | 782 | 230 | 12 | 813 | 503 | 12 | 860 |
| 5.0 | 9 | 14 | 3608 | 122 | 12 | 1017 | 259 | 12 | 1068 | 570 | 12 | 1140 |
| 4.0 | 16 | 15 | 5935 | 146 | 13 | 1523 | 316 | 13 | 1676 | 705 | 13 | 1855 |
| 3.0 | 60 | 22 | 22758 | 186 | 14 | 2705 | 398 | 14 | 2810 | 894 | 14 | 3004 |
| 2.0 | 459 | 25 | 136927 | 263 | 14 | 5295 | 571 | 14 | 5633 | 1343 | 15 | 6240 |
| 1.0 | — | — | — | 658 | 16 | 19373 | 1458 | 16 | 20939 | 3776 | 17 | 24683 |

- Ergebnis
  - Verhältnis Graphen/Laufzeit: Sehr gut, fast linear
  - Verhältnis Support/Laufzeit: Starke Abhängigkeit, Datenabhängig

# Ergebnisse II

TECHNISCHE UNIVERSITÄT DARMSTADT

- Datenbank Unterteilungen zur Speicherreduktion

| $\|\mathcal{D}\|$ | Runtime [sec] | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Number of Partitions | | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 10 | 20 | 30 | 40 | 50 |
| 100,000 | 1432 | 1878 | 2032 | 2189 | 2356 | 2924 | 3899 | 4842 | 6122 | 7459 |
| 200,000 | 3698 | 4494 | 5095 | 5064 | 5538 | 6418 | 7856 | 9516 | 11165 | 12670 |

| $\|\mathcal{D}\|$ | Maximum amount of memory for storing TID lists [Mbytes] | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Number of Partitions | | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 10 | 20 | 30 | 40 | 50 |
| 100,000 | 53.8 | 27.0 | 18.1 | 13.6 | 11.0 | 5.6 | 2.9 | 2.0 | 1.5 | 1.2 |
| 200,000 | 118 | 59.1 | 39.5 | 29.6 | 23.9 | 12.1 | 6.2 | 4.2 | 3.2 | 2.6 |

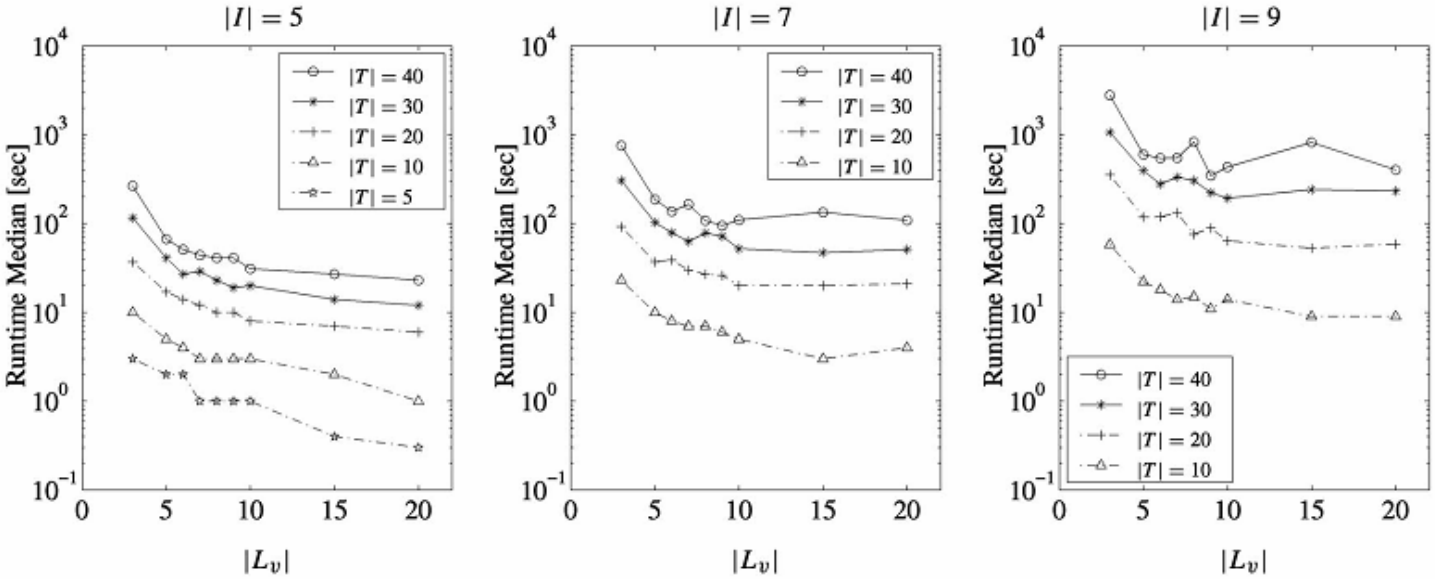
- Ergebnis
  - Starke Reduktion bis ~ 20 Unterteilungen
  - Geringer Anstieg der Laufzeit

# Ergebnisse III

TECHNISCHE UNIVERSITÄT DARMSTADT

- Synthetisches Datenset
  - Diverse Kontrollparameter
    - Anzahl Graphen |D|
    - Durchschnittliche Anzahl Kanten je Graph |T|
    - Durchschnittliche Anzahl Kanten je FSG |I|
    - Durchschnittliche Anzahl FSG je Graph |S|
    - Anzahl der unterschiedlichen Kantenlabels $|L_E|$
    - Anzahl der unterschiedlichen Knotenlabels $|L_V|$

# Ergebnisse IV




- Ergebnis
  - Die Laufzeit sinkt mit $|L_V|$, weniger Automorphismen, weniger Isomorphismen
  - Die Laufzeit steigt mit |I|, komplexere Patterns
  - Die Laufzeit steigt mit |T|, mehr Pattern je Graph möglich

# Frequent Geometric Subgraph Discovery Algorithm

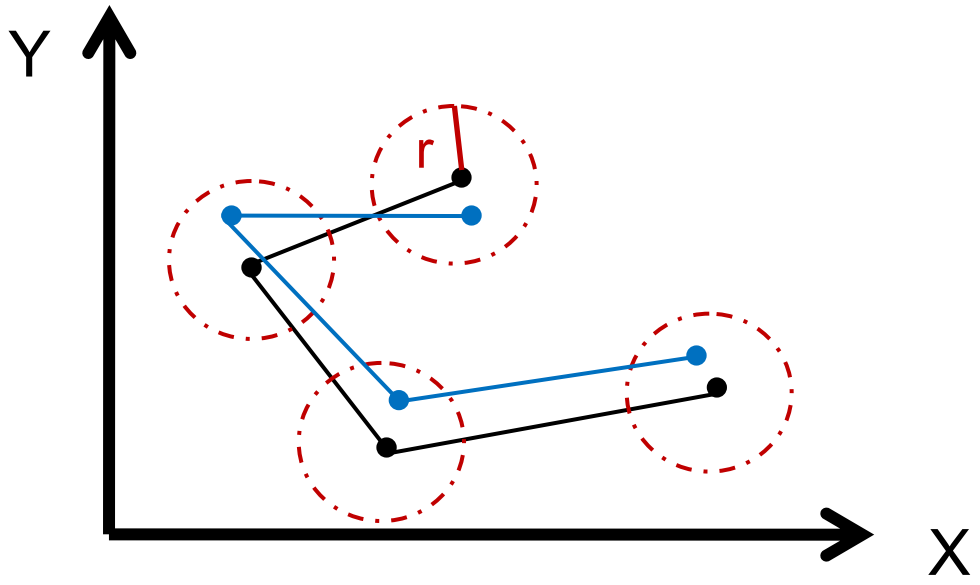
# Übersicht

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wichtige Begriffe
- Isomorphismus in geometrischen Graphen
- Parameter und High Level Struktur des gFSG Algorithmus
  - Candidate Generation
  - Frequency Counting
  - Iterative Shape Adjustment
- Experimentelle Evaluation

# Begriffe I

- Geometric Graph
  - Ungerichteter Graph
  - Mit Labeln, nicht eindeutig
  - Jeder Knoten hat eine zwei bzw. dreidimensionale Koordinaten
- Coordinate Matching Tolerance r



# Begriffe II

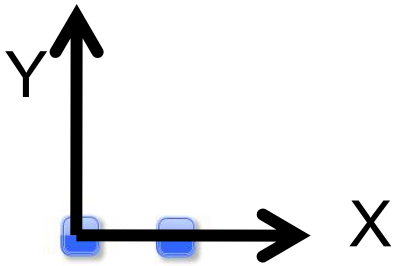
TECHNISCHE UNIVERSITÄT DARMSTADT

- r tolerant frequent Subgraph
  - D Menge von Graphen
  - $0 < \sigma \leq 1.0$
  - Ein Subgraph ist frequent, wenn er mindestens $\sigma|D|$ % aller Graphen gefunden wird (Minimum Support)

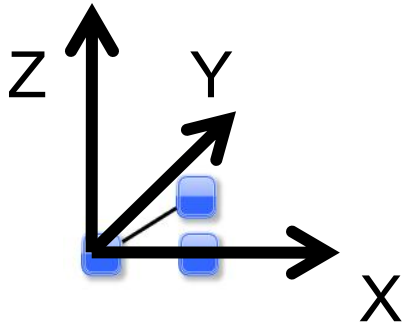
# Begriffe III

TECHNISCHE UNIVERSITÄT DARMSTADT

- Geometric Configuration
  - Koordinatensystems ergibt sich aus Kanten des Graphen
  - 2D : Kante des Graphen entspricht X-Achse




  - 3D : zwei Kanten (nicht parallel) mit einem gemeinsamen Knoten als XY-Ebene




=> Ermöglicht translations- und rotations-invariante Isomorphismen

# Geometric Configuration Example

TECHNISCHE UNIVERSITÄT DARMSTADT

- Skalierung des Graphen
  eine Kante besitzt Länge eins

$\Rightarrow$ Erlaubt uns skalen-invariante Isomorphismen zu finden

# Geometric Graph Isomorphism

TECHNISCHE UNIVERSITÄT DARMSTADT

- Finde geometrische Transformation zwischen $g_1$ und $g_2$
  - Rotation, Skalierung und Translation
  - Nur zwischen den Knoten
- Überprüfe die Topologie (und Label) jeder Transformation
  - Topologie: Struktur des Graphen

# Geometric Graph Isomorphism

TECHNISCHE UNIVERSITÄT DARMSTADT

1. $g_1$ und $g_2$ haben unterschiedliche Anzahl Knoten oder Kanten
   -> false
2. Zufällige geometrische Konfiguration von $g_2$
   1. Für jede geometrische Konfiguration von $g_1$
   2. Für jeden Knoten v in $g_1$
      1. Finde den nächsten Knoten u in $g_2$ mit gleichem Label
      2. Distanz zwischen den Knoten größer r -> break
      3. u wird dem Knoten v zugeordnet

   3. Bijektion gefunden
   4. Ist Bijektion eine zulässiger topologischer Isomorphismus zwischen $g_1$ und $g_2$
      -> return true
3. Return false

# Parameter von gFSG

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eingabe
  - Graphenmenge D
  - Support threshold σ
  - Matching Tolerance r
  - Shape Adjustment Type type
  - Anzahl der Shape Adjustment Iterationen N
- Ausgabe
  - Alle Subgraphen die den Minimum Support erfüllen
    - Frequent Subraphs (FSG) genannt

# GFSG High Level Structure

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Spezifizieren aller FGS der Größe 1,2 und 3 in D
2. Solange es 2 FGS der Größe k-1 gibt
   1. Generiere alle Kandidaten $g^k$ der Größe k (Candidate Generation)
   2. Für jeden Kandidaten der Größe k
      1. Berechne die Menge S der Graphen t ϵ D in denen der Kandidat vorkommt (Frequency Counting)
      2. Kommt der Kandidat in keinem der Graphen vor fahre mit nächstem Kandidaten fort
      3. Gegebenenfalls wird die geometrische Form des Kandidaten angepasst (Iterative Shape Adjustment)
   3. Verwirft Subgraphen $g^k$ die den Minimum Support nicht erfüllen

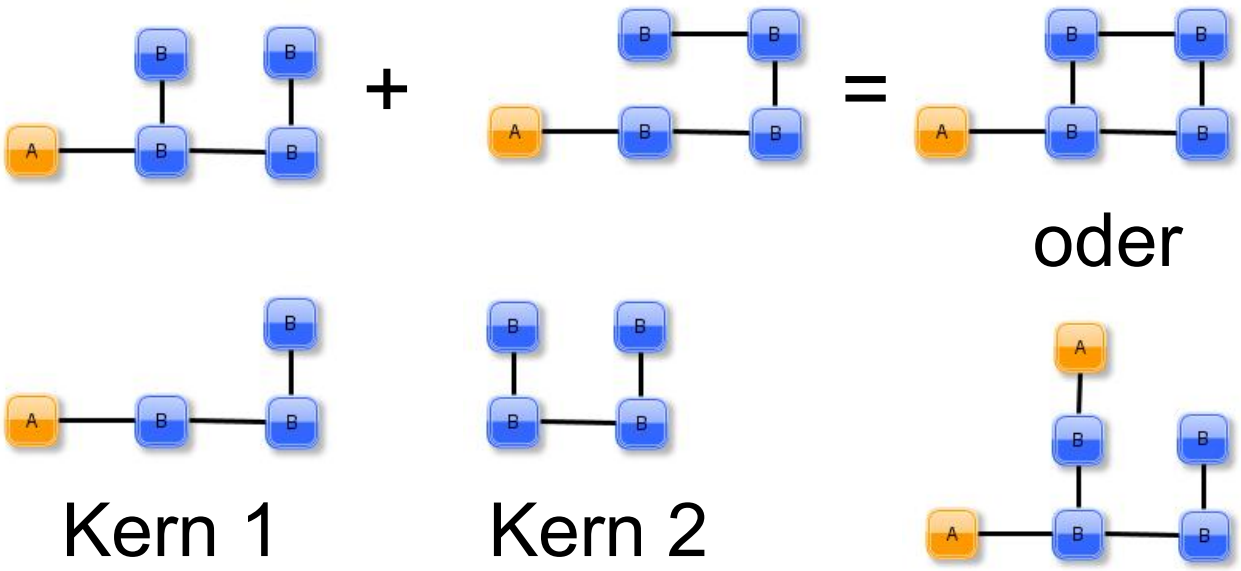
# GFSG High Level Structure

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Spezifizieren aller FGS der Größe 1,2 und 3 in D
2. Solange es 2 FGS der Größe k-1 gibt
   1. **Generiere alle Kandidaten $g^k$ der Größe k (Candidate Generation)**
   2. Für jeden Kandidaten der Größe k
      1. Berechne die Menge S der Graphen t ϵ D in denen der Kandidat vorkommt (Frequency Counting)
      2. Kommt der Kandidat in keinem der Graphen vor fahre mit nächstem Kandidaten fort
      3. Gegebenenfalls wird die geometrische Form des Kandidaten angepasst (Iterative Shape Adjustment)
   3. Verwirft Subgraphen $g^k$ die den Minimum Support nicht erfüllen

# Candidate Generation

- Zusammenfügen FGS der Größe k zu einem der Größe k+1
  - müssen gleichen k-1 Subgraph (Kern) enthalten
  - nicht unbedingt eindeutig
    - Kern hat mehrere Authomorphismen
    - FSG hat mehrere geometrische Kerne



# Candidate Generation Algo

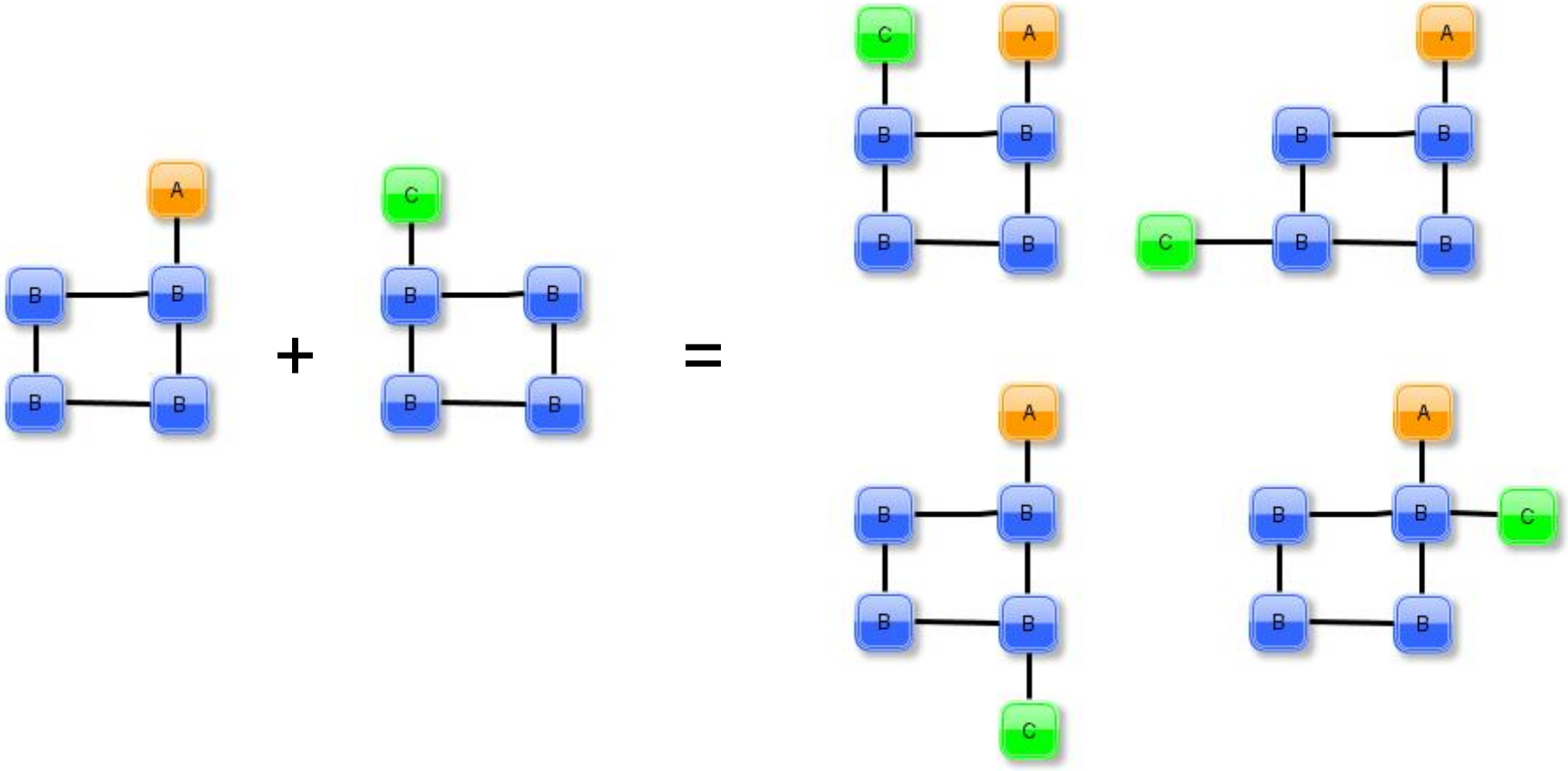
TECHNISCHE UNIVERSITÄT DARMSTADT

1. Für jedes Paar $g_i$, $g_j$ von FSG der Größe k
   1. Für jede Kante e des 1. FSG

      1. Entferne Kante e aus FSG -> k-1 Subgraph
      2. Enthält der 2. FSG $g_j$ den k-1 Subgraph
         -> gleicher Kern

         1. Zusammenfügen der beiden FSG $g_i$ und $g_j$
            zu einem neuen Subgraphen der Größe k+1
         2. k+1 Subgraph wurde bereits generiert -> continue
         3. Subgraph ist neuer Kandidat Größe k+1

# Join Beisipiel



# GFSG High Level Structure

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Spezifizieren aller FGS der Größe 1,2 und 3 in D
2. Solange es 2 FGS der Größe k-1 gibt
   1. Generiere alle Kandidaten $g^k$ der Größe k
   2. Für jeden Kandidaten der Größe k
      1. Berechne die Menge S der Graphen t ϵ D in denen der Kandidat vorkommt
      2. Kommt der Kandidat in keinem der Graphen vor fahre mit nächstem Kandidaten fort
      3. Gegebenenfalls wird die geometrische Form des Kandidaten angepasst
   3. Verwirft Subgraphen $g^k$ die den Minimum Support nicht erfüllen

# GFSG High Level Structure

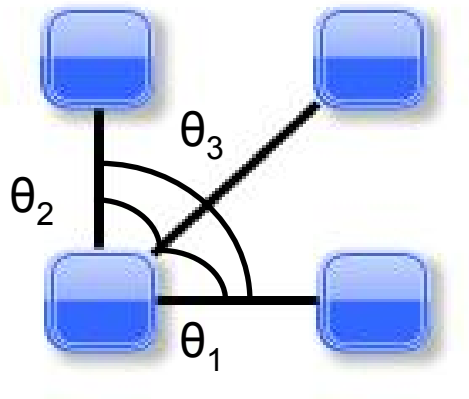
TECHNISCHE UNIVERSITÄT DARMSTADT

1. Spezifizieren aller FGS der Größe 1,2 und 3 in D
2. Solange es 2 FGS der Größe k-1 gibt
   1. Generiere alle Kandidaten $g^k$ der Größe k
   2. Für jeden Kandidaten der Größe k
      1. **Berechne die Menge S der Graphen t ϵ D in denen der Kandidat vorkommt**
      2. Kommt der Kandidat in keinem der Graphen vor fahre mit nächstem Kandidaten fort
      3. Gegebenenfalls wird die geometrische Form des Kandidaten angepasst
   3. Verwirft Subgraphen $g^k$ die den Minimum Support nicht erfüllen

# Count Frequency (EAL)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Edge Angle List (eal)
  - Multimenge von Winkeln zwischen jedem Kantenpaar mit gemeinsamem Knoten




$eal(g) = \{\theta_1, \theta_2, \theta_3\}$

  - Winkel sind invariant gegenüber geometrischen Transformationen
  - wenn g isomorph zu einem Subgraphen aus t ϵ D dann eal(g) Teilmenge eal(t)

# Count Frequency (TID List)

TECHNISCHE UNIVERSITÄT DARMSTADT

- TID List (Transaction ID List)
  - Liste von Graphen (Transactions) in denen ein FSG enthalten
  - Berechnung der Häufigkeit eines Kandidaten der Größe k+1 :
    1. Berechne Schnittmenge der TIDs der beiden zugrunde liegenden FSG der Größe k
    2. Größe der Schnittmenge kleiner als Minimum Support =>Subgraph der Größe k+1 verworfen
    3. Sonst
       Berechne Häufigkeit mit Subgraph Isomorphismus auf den Graphen die sich in der Schnittmenge befinden

# Count Frequency (Hybrid)

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Finden der frequent edge angles
2. TID list für jeden frequent edge angle
   der Graphen die eine Instanz des Winkels enthalten

$$tid(\theta)=\{t_1,t_2,\ldots,t_n\}$$

3. Kandidat g hat eine $eal(g)=\{\theta_1,\theta_2,\ldots,\theta_n\}$
4. Berechnung der Schnittmenge der TID Listen der verschiedenen edge angles in g

$$I=Schnitt_i(tid(\theta_i))$$
$$\theta_i \in eal(g)$$

=>Wenn g den Minimum Support erfüllt ist $|I|\geq\sigma|D|$

# Frequency Counting

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bewertung der drei Ansätze
  - Edge Angle List
    - Sehr langsam
  - TID List
    - Schnell verbraucht aber sehr viel Speicher
    - Speichere Listen von allen FSG der Größe k
  - Hybrid
    - 5 mal schneller als Edge Angle List
    - 2 mal langsamer als TID List
    - Verbraucht allerdings viel weniger Speicher

=> Hybrid wurde in allen Experimenten verwendet

# GFSG High Level Structure

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Spezifizieren aller FGS der Größe 1,2 und 3 in D
2. Solange es 2 FGS der Größe k-1 gibt
   1. Generiere alle Kandidaten $g^k$ der Größe k
   2. Für jeden Kandidaten der Größe k
      1. Berechne die Menge S der Graphen t ϵ D in denen der Kandidat vorkommt
      2. Kommt der Kandidat in keinem der Graphen vor fahre mit nächstem Kandidaten fort
      3. Gegebenenfalls wird die geometrische Form des Kandidaten angepasst
   3. Verwirft Subgraphen $g^k$ die den Minimum Support nicht erfüllen

# GFSG High Level Structure

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Spezifizieren aller FGS der Größe 1,2 und 3 in D
2. Solange es 2 FGS der Größe k-1 gibt
   1. Generiere alle Kandidaten $g^k$ der Größe k
   2. Für jeden Kandidaten der Größe k
      1. Berechne die Menge S der Graphen t ϵ D in denen der Kandidat vorkommt
      2. Kommt der Kandidat in keinem der Graphen vor fahre mit nächstem Kandidaten fort
      3. **Gegebenenfalls wird die geometrische Form des Kandidaten angepasst**
   3. Verwirft Subgraphen $g^k$ die den Minimum Support nicht erfüllen

# Anpassung der geometrischen Form

TECHNISCHE UNIVERSITÄT DARMSTADT

- Geometrische Konfiguration eines Kandidaten nicht unbedingt optimal
  - Problem: Einige vorkommen des Kandidaten können nicht gefunden werden
  - Lösung: Anpassung der From des Kandidaten

- Führe Frequency Counting für jeden Kandidaten mehrmals durch
- In jeder Iteration :
  - Berechne Menge der vorkommenden isomorphen Subgraphen
  - Berechne Durchschnitt der Knotenkoordinaten
- Diese Koordinaten sind die neuen Koordinaten des Kandidaten (Angepasste Form)

# Anpassung der geometrischen From

TECHNISCHE UNIVERSITÄT DARMSTADT

- Terminierung
  - Simple Adjustment (SA)
    - Terminiert nach N Iterationen (Benutzerspezifiziert)
  - Supporting Transaction Monitoring (STM)
    - Wie Simple Adjustment, aber ...
    - ...frühere Terminierung möglich, wenn die Menge der gefundenen Subgraphen sich in zwei aufeinanderfolgenden Iterationen nicht verändert
  - Downward Closure Check (DWC)
    - Wie Supporting Transaction Monitoring, aber...
    - ...wenn Veränderungen des Kandidaten zu groß werden kann es passieren, dass die k-1 Subgraphen des Kandidaten nicht mehr den Minimum Support erfüllen => Terminiert

# Experimentelle Evaluierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Skalierbarkeit bezüglich der Datenbankgröße D
- Skalierbarkeit bezüglich der Graphengröße T
- Effektivität der Formanpassung

  - σ Minimum Support Threshold
  - t Laufzeit in Sekunden
  - l Größe des größten FSG
  - # f  Gesamtzahl der gefunden FSG

# Skalierbarkeit bezüglich der Datenbankgröße

TECHNISCHE UNIVERSITÄT DARMSTADT

| $\sigma$ | Total Number of Transactions $D$ | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $D = 1000$ | | | $D = 2000$ | | | $D = 5000$ | | | $D = 10000$ | | | $D = 20000$ | | |
| % | $t$[sec] | $l$ | $\#f$ | $t$[sec] | $l$ | $\#f$ | $t$[sec] | $l$ | $\#f$ | $t$[sec] | $l$ | $\#f$ | $t$[sec] | $l$ | $\#f$ |
| 5.0 | 8 | 6 | 119 | 14 | 6 | 113 | 34 | 6 | 114 | 75 | 5 | 117 | 179 | 6 | 111 |
| 4.5 | 9 | 6 | 137 | 20 | 6 | 138 | 45 | 6 | 139 | 83 | 5 | 132 | 209 | 6 | 126 |
| 4.0 | 10 | 6 | 168 | 22 | 6 | 157 | 52 | 6 | 160 | 96 | 6 | 151 | 244 | 6 | 154 |
| 3.5 | 12 | 6 | 206 | 30 | 6 | 209 | 57 | 6 | 184 | 110 | 6 | 185 | 281 | 6 | 182 |
| 3.0 | 14 | 7 | 236 | 35 | 6 | 246 | 73 | 7 | 236 | 126 | 6 | 217 | 321 | 6 | 224 |
| 2.5 | 20 | 7 | 314 | 55 | 7 | 329 | 85 | 7 | 287 | 150 | 6 | 259 | 357 | 7 | 268 |
| 2.0 | 26 | 7 | 415 | 72 | 7 | 430 | 124 | 7 | 404 | 205 | 7 | 352 | 522 | 7 | 359 |
| 1.5 | 48 | 7 | 687 | 107 | 7 | 613 | 218 | 8 | 630 | 410 | 7 | 552 | 842 | 7 | 526 |
| 1.0 | 123 | 8 | 1393 | 315 | 8 | 1395 | 460 | 9 | 1189 | 1107 | 8 | 1295 | 1974 | 8 | 1019 |
| 0.5 | 694 | 10 | 4960 | 1478 | 10 | 4623 | 2108 | 10 | 3593 | 4621 | 9 | 3869 | 9952 | 9 | 3354 |
| 0.25 | 2043 | 13 | 14235 | 5674 | 12 | 15232 | 8972 | 12 | 11103 | 17421 | 9 | 10929 | 41895 | 11 | 11177 |

# Skalierbarkeit bezüglich der Graphengröße

TECHNISCHE UNIVERSITÄT DARMSTADT

| $\sigma$ | Average Transaction Size $T$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $T = 14$ | | | $T = 19$ | | | $T = 23$ | | | $T = 28$ | | |
| % | $t$[sec] | $l$ | $\#f$ | $t$[sec] | $l$ | $\#f$ | $t$[sec] | $l$ | $\#f$ | $t$[sec] | $l$ | $\#f$ |
| 5.0 | 15 | 6 | 74 | 21 | 6 | 93 | 37 | 6 | 116 | 92 | 6 | 201 |
| 4.5 | 16 | 6 | 86 | 26 | 6 | 112 | 46 | 6 | 142 | 102 | 6 | 236 |
| 4.0 | 17 | 6 | 110 | 29 | 6 | 130 | 54 | 6 | 166 | 115 | 7 | 277 |
| 3.5 | 19 | 7 | 127 | 34 | 6 | 162 | 64 | 6 | 205 | 128 | 7 | 309 |
| 3.0 | 22 | 7 | 154 | 41 | 6 | 196 | 73 | 6 | 249 | 175 | 7 | 408 |
| 2.5 | 27 | 7 | 195 | 59 | 6 | 247 | 96 | 7 | 302 | 331 | 7 | 658 |
| 2.0 | 36 | 7 | 264 | 81 | 6 | 325 | 142 | 7 | 420 | 543 | 8 | 993 |
| 1.5 | 53 | 7 | 386 | 138 | 6 | 502 | 291 | 7 | 729 | 1002 | 8 | 1599 |
| 1.0 | 92 | 9 | 680 | 284 | 8 | 927 | 612 | 9 | 1385 | 2530 | 10 | 3936 |
| 0.5 | 406 | 9 | 2072 | 1438 | 9 | 2859 | 3050 | 9 | 4620 | 9923 | 12 | 13178 |
| 0.25 | 1226 | 10 | 5358 | 4997 | 10 | 8949 | 10824 | 12 | 15232 | 29686 | 14 | 38788 |

# Effektivität der Formanpassung

TECHNISCHE UNIVERSITÄT DARMSTADT

| Dataset | $\sigma$ [%] | Adjustment | $N$ | $t$[sec] | #$f$ | |
|---|---|---|---|---|---|---|
| DTP 5,000 compounds | 2.0 | None | — | 124 | 404 | |
| | | SA | 2 | 155 | 541 | + 34 % |
| | | | 5 | 209 | 562 | |
| | | | 10 | 248 | 558 | |
| | | STM | 2 | 138 | 543 | |
| | | | 5 | 187 | 542 | |
| | | | 10 | 233 | 536 | |
| | | DWC | 2 | 138 | 543 | |
| | | | 5 | 187 | 542 | |
| | | | 10 | 248 | 557 | |
| DTP 10,000 compounds | 1.0 | None | — | 1107 | 1295 | |
| | | SA | 2 | 1180 | 1637 | + 25 % |
| | | | 5 | 1509 | 1638 | |
| | | | 10 | 1720 | 1678 | |
| | | STM | 2 | 1026 | 1618 | |
| | | | 5 | 1383 | 1666 | |
| | | | 10 | 1676 | 1699 | |
| | | DWC | 2 | 1123 | 1631 | |
| | | | 5 | 1461 | 1628 | |
| | | | 10 | 1737 | 1683 | |

# Danke für die Aufmerksamkeit

Fragen???