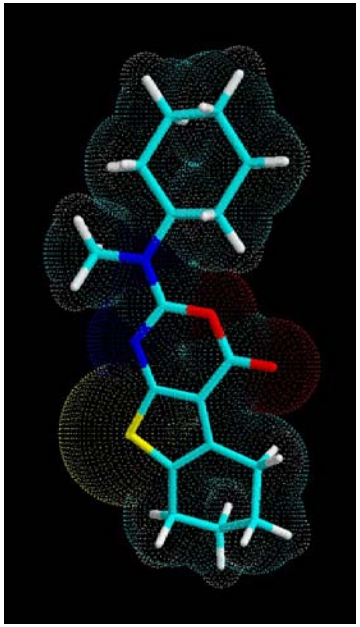
Frequent Connected Subgraph Mining



Tamás Horváth

University of Bonn &
Fraunhofer IAIS, Sankt Augustin, Germany
tamas.horvath@iais.fraunhofer.de

universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer
IAIS

Frequent Connected Subgraph Mining

Outline

- motivation, problem definition, and a negative complexity result
- mining trees with the levelwise search algorithm
- mining bounded tree-width graphs

Notions: Isomorphism and Subgraph Isomorphism

$G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$: labeled graphs

- each vertex and edge is associated with some symbol of an alphabet

G_1 is **isomorphic** to G_2 if there is a bijection $\varphi : V_1 \to V_2$ preserving

- the edges in both directions
  - i.e., $\{u, v\} \in E_1$ if and only if $\{\varphi(u), \varphi(v)\} \in E_2$
- the vertex labels
  - i.e., $\varphi(u)$ has the same label as u for all $u \in V_1$
- the edge labels
  - i.e., $\{\varphi(u), \varphi(v)\}$ has the same label as $\{u, v\}$ for all $\{u, v\} \in E_1$

G_1 is **subgraph isomorphic** to G_2 if G_2 has a subgraph isomorphic to G_1

Mining Frequent Connected Subgraphs

frequent connected subgraph mining problem:

Given a set D of labeled graphs and an integer $t > 0$, list the set of t-frequent connected subgraphs w.r.t. D

- $t > 0$ integer: *frequency threshold*
- t*-frequent subgraph:* subgraph isomorphic to at least t graphs in D

instance of the theory extraction problem:

- D : set of labeled graphs
- $\mathcal{L}$: set of all labeled *connected* graphs
- *interestingness predicate* q_D: for a pattern $H \in \mathcal{L}$, $q_D(H)$ is true iff H is subgraph isomorphic to at least t graphs in D
  - q_D is *anti-monotone*: any connected subgraph of a t-frequent connected graph is also t-frequent

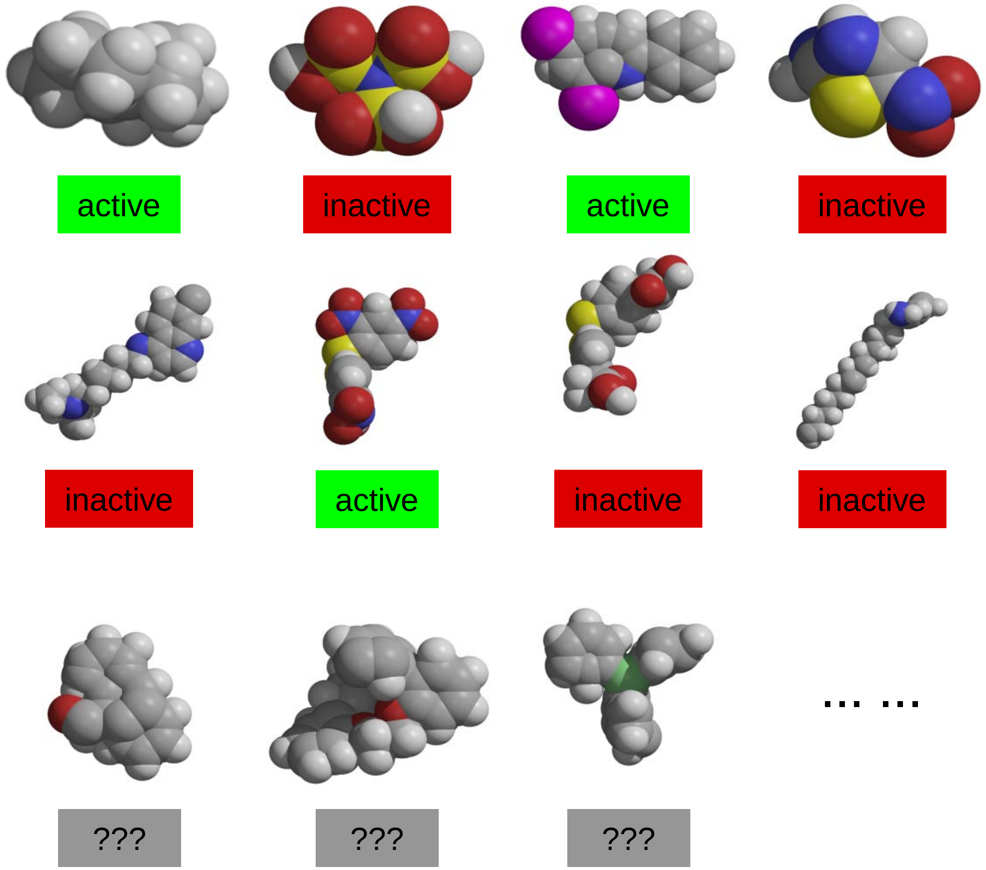
Frequent Subgraph Mining: Motivation

Virtual screening in drug discovery:

select a limited number of candidate compounds from millions of database molecules that are most likely to possess a desired biological activity

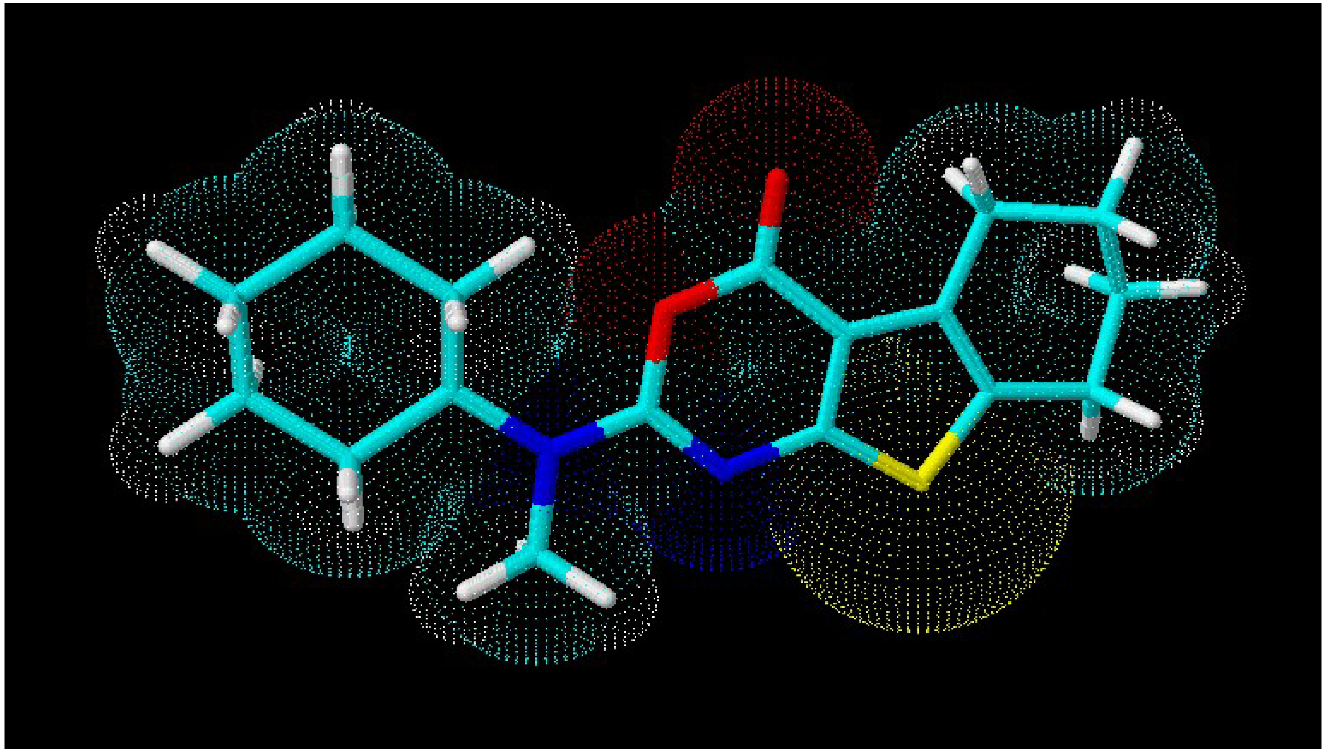
training molecules

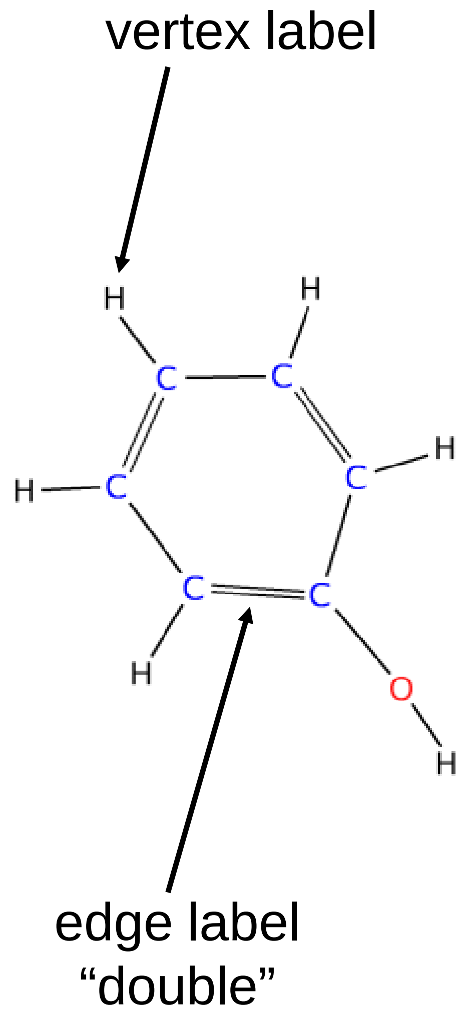
test molecules



Virtual Screening in Drug Discovery

Molecules and their Molecular Graphs





molecules give rise to **labeled undirected graphs**

Virtual Screening in Drug Discovery

approach [Deshpande, Kuramochi, Wale, & Karypis, 2005]:

1. compute the set $\{p_1, \ldots, p_k\}$ of **frequent connected subgraphs** for the molecular graphs of the training molecules

2. assign a **binary colored (feature) vector** v of length k to each training molecule m with

$$v[i] := \begin{cases} 1 & \text{if } p_i \text{ is a subgraph of the molecular graph of } m \\ 0 & \text{o/w} \end{cases}$$

$(i = 1, \ldots, k)$

$\Rightarrow$ each molecule is represented by a colored vertex of the k-dimensional unit hypercube

- green: active
- red: inactive

Virtual Screening in Drug Discovery

approach [Deshpande, Kuramochi, Wale, & Karypis, 2005] (cont'd):

3. compute a hyperplane h in the k-dimensional space that separates the green points from the red ones, as good as possible

   - *Support Vector Machines* [Boser, Guyon, & Vapnik, 1992; Cortes & Vapnik, 1995]:
     - choose the hyperplane with maximum distance from nearest data points

4. for each test molecule, compute the vector as defined in step 2 and predict its activity according to the halfspace defined by h it belongs to

- **empirical experiments:** good predictive performance
- **rest of the lecture:** how to perform step 1?

Enumeration Complexity

the **size** of the output (theory) can be exponential in the size of the input D

⇨ the output cannot be computed in time polynomial in the size of D

enumeration complexities:

a set of S with N elements, say $s_1, \ldots, s_N$, are listed with

- **polynomial delay** if the time before printing s_1, the time between printing s_i and s_{i+1} for every $i=1,\ldots,N-1$, and the termination time after printing s_N is bounded by a polynomial of the size of the input,
- **incremental polynomial time** if s_1 is printed with polynomial delay, the time between printing s_i and s_{i+1} for every $i=1,\ldots,N-1$ (resp. the termination time after printing s_N) is bounded by a polynomial of the combined size of the input and the set $s_1,\ldots, s_i$ (resp. S),
- **output polynomial time** if S is printed in the combined size of the input and the entire set S

A Negative Complexity Result

Thm: The frequent connected subgraph mining problem cannot be solved in output-polynomial time (unless P = NP).

Proof:

reduction: *Hamiltonian path* problem

- **Hamiltonian path problem:**
  Given a graph *G* with *n* vertices, decide whether or not *G* has a Hamiltonian path, i.e., a path containing each vertex of *G* exactly once
  - *NP-complete* problem

Proof of the Negative Complexity Result

- $D = \{G_1, G_2\}$, where
  - G_1 is an arbitrary unlabeled graph with n vertices
  - G_2 is an unlabeled path of length $n-1$ (i.e., has n vertices)
- $\mathcal{L}$: set of all unlabeled connected graphs
- q_D: 2-frequency w.r.t. D

$\Rightarrow$ $Th(\mathcal{L}, D, q_D)$ is the set of 2-frequent paths

- the size of $Th(\mathcal{L}, D, q_D)$ is polynomial in the size of D ($|Th(\mathcal{L}, D, q_D)| \leq n$)
- $|Th(\mathcal{L}, D, q_D)| = n$ if and only if G_1 has a Hamiltonian path
- cannot be computed in output polynomial time, as otherwise the NP-complete Hamiltonian path problem could be decided in polynomial time

Frequent Connected Subgraph Mining

Outline

- motivation, problem definition, and a negative complexity result
- mining trees with the levelwise search algorithm
- mining bounded tree-width graphs

A Generic Levelwise Search Graph Mining Algorithm

Input : database D of transaction graphs and integer $t > 0$
Output: all frequent connected subgraphs

graph class of the transaction graphs

MAIN:

let $S_0 \subseteq \mathcal{G}$ be the set of frequent graphs consisting of a single labeled vertex
for $(l := 0;\ S_l \neq \emptyset;\ l := l + 1)$ **do**
  $C_{l+1} := S_{l+1} := \emptyset$
  forall $P \in S_l$ **do**
    print P
    forall $H \in \rho(P) \cap \mathcal{G}$ satisfying (i) $H \notin C_{l+1}$ and (ii) $\rho^{-1}(H) \subseteq S_l$ **do**
      add H to C_{l+1}
      if SUPPORTCOUNT$(H) \geq t$ **then**
        add H to S_{l+1}

H is a **strong** candidate

add a new labeled edge to P with at least one old vertex in **all possible ways**

SUPPORT COUNT(H, D, t):

counter $:= 0$
forall G in D **do**
  if $H \preccurlyeq G$ **then**
    $\text{counter} := \text{counter} + 1$
return counter

universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer
IAIS

Efficiency Conditions for the Generic Graph Mining Algorithm

Thm: Let $\mathcal{G}$ be a graph class satisfying

(i) $\mathcal{G}$ is closed under taking subgraphs (i.e., $\forall G \in \mathcal{G}$, all subgraphs of G belong to $\mathcal{G}$),

(ii) the membership problem in $\mathcal{G}$ (i.e., does $G \in \mathcal{G}$ hold for any graph G) can be decided in polynomial time, and

(iii) for every $H, G \in \mathcal{G}$ such that H is connected, it can be decided in polynomial time whether H is subgraph isomorphic to G.

If the transaction graphs in D belong to $\mathcal{G}$ then the previous algorithm lists the frequent connected subgraphs with polynomial delay.

Efficiency Conditions for the Generic Graph Mining Algorithm

Proof (sketch):

- the cardinalities (and hence, the sizes) of the sets $\rho(H) \cap \mathcal{G}$ and $\rho^{-1}(H)$ in line 6 are bounded by a polynomial of the size of D,
- both sets can be computed in polynomial time,
- conditions (i) and (iii) together imply that one can define a **canonical string representation** for the graphs in $\mathcal{G}$ that can be computed in time polynomial in the size of D.
  - **canonical string representation**: unique modulo isomorphism (i.e., two graphs have the same canonical strings if and only if they are isomorphic)

$\Rightarrow$ using some advanced (e.g., trie-based) data structure for the storage of S_l and the elements of C_{l+1} generated before H, conditions (i) and (ii) in line 6 can be decided in time polynomial in the size of D

Application: Frequent Subtree Mining in Forests

problem:

Given a set D of labeled *forests* and an integer $t > 0$, list all trees that are subtrees of at least t forests in D.

- **forest**: set of vertex disjoint labeled free trees
- **free tree**: unordered, unrooted tree

Frequent Subtree Mining in Forests

Thm: The frequent connected subgraph mining problem can be solved with *polynomial delay* for **forest** transaction graphs.

proof:

each condition of the previous theorem holds, i.e.,

(i) forests are *closed downward*,

(ii) it can be decided in polynomial time, whether a graph *G* is a forest,

(iii) *subtree isomorphism* can be decided in polynomial time

- in time $O(n^{2.5})$ [Matula,1978]
- can further be improved by a *log* factor [Shamir &Tsur,1999]

Subtree Isomorphism

T tree, F forest; **decide, whether T is subgraph isomorphic to F:**
decide for all trees T' in F, whether T is subgraph isomorphic to T'

T and T' are trees; **decide, whether T is subgraph isomorphic to T':**
decide for some fixed vertex u in T and for all vertices v in T', whether there is a subgraph isomorphism from the tree T **rooted** at u to the tree T' **rooted** at v that maps u to v

- problem is reduced to subtree isomorphism between labeled, **rooted**, unordered trees
- can be solved with a **bottom-up** (recursive) algorithm (next slide)

Bottom-up Subtree Isomorphism algorithm for Rooted Trees

1. T_u: tree T rooted at u; $u_1, \ldots, u_m$: children of u in T_u
   T'_v: tree T' rooted at v; $v_1, \ldots, v_n$: children of v in T_v

2. decide (recursively) whether there is a subgraph isomorphism from the subtree of T_u **rooted** at u_i to the subtree of T'_v **rooted** at v_j that maps u_i to v_j for every $i = 1, \ldots, m$ and $j = 1, \ldots, n$;
   if so, **mark** the pair (i, j) if u and v have the same vertex label, and the edges between u, u_i and between v, v_j have the same edge labels

3. **return** YES (i.e., T_u is subgraph isomorphic to T'_v) if the bipartite graph

   $$(\{u_1, \ldots, u_m\}, \{v_1, \ldots, v_n\}, \{\{u_i, v_j\} : (i, j) \text{ is marked } \})$$

   has a (maximum) matching of size m

   - maximum bipartite matching can be solved in polynomial time

Summary

- mining frequent connected subgraphs in arbitrary transaction graphs is computationally hard
  - cannot be solved in output-polynomial time (unless P = NP)
- for forest transaction graphs, the problem can be solved with polynomial delay
  - obtained by using a generic levelwise search algorithm
  - frequent patterns are not printed immediately after their generation
    - polynomial delay vs. incremental polynomial time

Frequent Connected Subgraph Mining

Outline

- motivation, problem definition, and a negative complexity result
- mining trees with the levelwise search algorithm
- mining bounded tree-width graphs

Positive and Negative Results So Far

frequent connected subgraph mining:

☹ computationally **intractable** for **arbitrary** transaction graphs

- **cannot** be solved in **output-polynomial** time (unless P = NP)

☺ can be solved **efficiently** for **forest** transaction graphs

- with **polynomial delay**

Goal: Generalize the positive result on forests to a broader graph class!

- **What about graphs of bounded tree-width?**
  - parameterized graph class (naturally) generalizing forests

A Generic Levelwise Search Graph Mining Algorithm (recap)

Input : database D of transaction graphs and integer $t > 0$
Output: all frequent connected subgraphs

graph class of the transaction graphs

MAIN:

```
let S_0 ⊆ 𝒢 be the set of frequent graphs consisting of a single labeled vertex
for (l := 0; S_l ≠ ∅; l := l + 1) do
   C_{l+1} := S_{l+1} := ∅
   forall P ∈ S_l do
      print P
      forall H ∈ ρ(P) ∩ 𝒢 satisfying (i) H ∉ C_{l+1} and (ii) ρ^{-1}(H) ⊆ S_l do
         add H to C_{l+1}
         if SupportCount(H) ≥ t then
            add H to S_{l+1}
```

H is a **strong** candidate

add a new labeled edge to P with at least one old vertex in **all possible ways**

SUPPORT COUNT(H, D, t):

```
counter := 0
forall G in D do
   if H ≼ G then
      counter := counter + 1
return counter
```


universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer IAIS

Efficiency Conditions (recap)

Thm: Let $\mathcal{G}$ be a graph class satisfying

(i) $\mathcal{G}$ is closed under taking subgraphs (i.e., $\forall G \in \mathcal{G}$, all subgraphs of G belong to $\mathcal{G}$),

(ii) the membership problem in $\mathcal{G}$ (i.e., does $G \in \mathcal{G}$ hold for any graph G) can be decided in polynomial time, and

(iii) for every $H, G \in \mathcal{G}$ such that H is connected, it can be decided in polynomial time whether H is subgraph isomorphic to G.

If the transaction graphs in D belong to $\mathcal{G}$ then the previous algorithm lists the frequent connected subgraphs with polynomial delay.

Problem:

- ☹ subgraph isomorphism is NP-complete even for graphs of tree-width 2
- ☺ Condition (iii) can be relaxed!

Problem Setting

Given

database D: finite set of labeled graphs of bounded tree-width

- *will be defined*

language $\mathcal{L}$: set of all connected labeled graphs of bounded tree-width,

interestingness predicate q_D: $\varphi \in \mathcal{L}$ is "interesting" w.r.t. D if it is subgraph isomorphic to at least t graphs in D

- $t > 0$ integer: *frequency threshold*

compute the theory of D w.r.t. $\mathcal{L}$ and q_D, i.e.,

$$Th(\mathcal{L}, D, q_D) = \{\varphi \in \mathcal{L} : q_D(\varphi) = \text{true}\}$$

- special case of the theory extraction problem

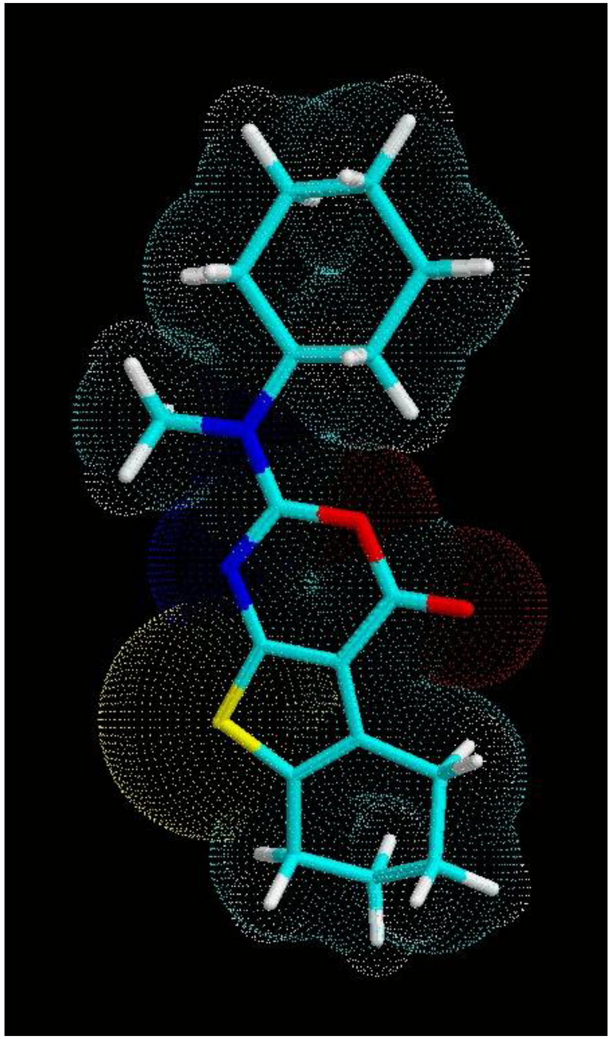
Main Result

Thm [H. & Ramon, 2010]**:**
the frequent connected subgraph mining problem can be solved in *incremental polynomial time* for graphs of *bounded tree-width*

significance of this result:

- efficient pattern mining is possible even for computationally hard pattern matching operators
  - subgraph isomorphism is NP-complete for bounded tree-width graphs
- first positive non-trivial result beyond trees
- positive result for a practically relevant graph class
  - e.g., molecular graphs of most pharmacological compounds have tree-width ≤ 3

Example



NCI Chemical Dataset:

- 250251 compounds

tree-width	*#molecules*	
0	13	*isolated vertices*
1	21950	*trees*
2	221675	*mostly outerplanar*
3	6548	
≥4	65	

Outline for the Rest (Technical Part) of this Topic

- tree-width
- subgraph isomorphism for bounded tree-width graphs
- remarks and open problems

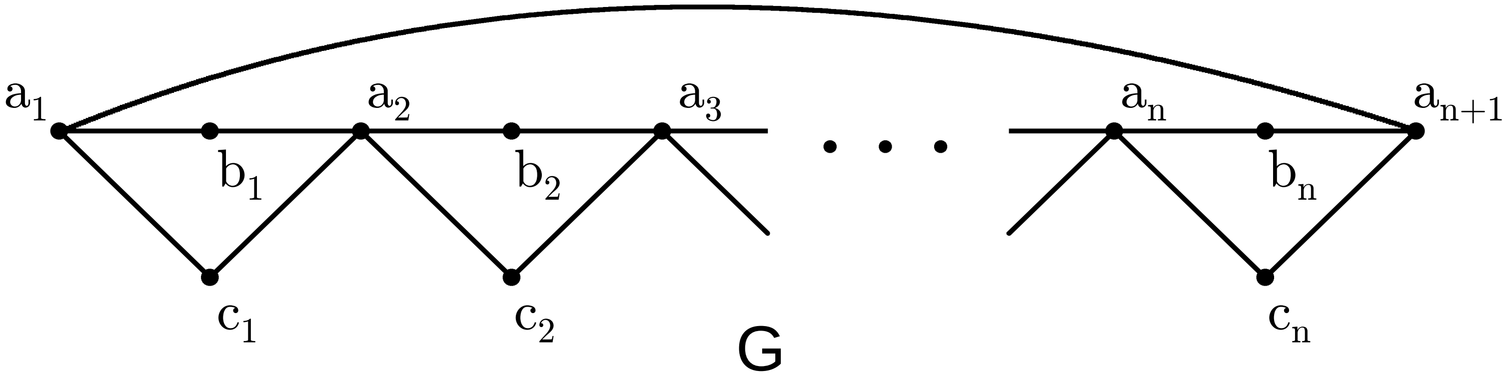
Tree-width (Robertson & Seymour, 1986)

- tree decomposition of a graph $G = (V, E)$: a pair $TD = (T, \mathcal{X})$, where
  - $T = (I, F)$ is an unordered tree,
  - $\mathcal{X} = \{\text{bag}(i) : i \in I\}$ is a family of subsets of V, called **bags**, s.t.
    * $\bigcup_{i \in I} \text{bag}(i) = V$,
    * for every $\{u, v\} \in E$ there is an $i \in I$ with $\{u, v\} \subseteq \text{bag}(i)$,
    * for all $i, j, k \in I$, if j is on the path from i to k in T then $\boxed{\text{bag}(i) \cap \text{bag}(k) \subseteq \text{bag}(j)}$
- tree-width of TD: $\max_i |\text{bag}(i)| - 1$
- tree-width of G: minimum tree-width over any TD
- graphs of bounded tree-width: graphs with tree-width bounded by some constant k

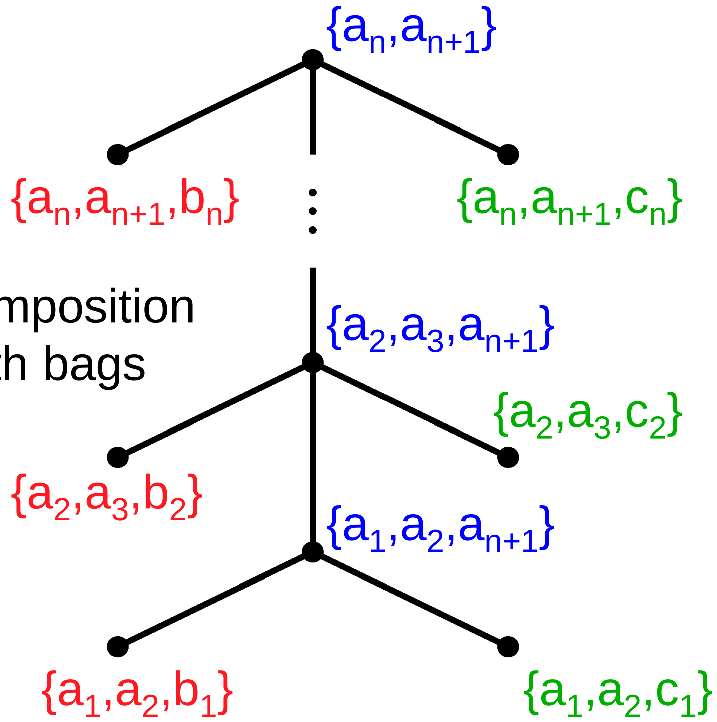
Tree-width (Robertson & Seymour,1986)

- **measure** of the tree-likeness of graphs
  - e.g., the tree-width of trees is 1 and the tree-width of cycles is 2
- **useful** tool in the design of algorithms because
  - many computationally hard problems on graphs become polynomial for graphs of bounded tree-width
  - many practically relevant graph classes have small tree-width
    - e.g., *k*-outerplanar graphs have tree-width at most 3*k*-1

Example



tree decomposition
of G with bags



G has

- $3n+1$ vertices; $4n+1$ edges
- 2^n+n simple cycles
- 2-outerplanar graph
- **tree-width: 2**

Some Properties of Bounded Tree-width Graphs

- ☺ the class of bounded tree-width graphs is *closed downward*
  - any subgraph of a graph of tree-width at most *k* has tree-width at most *k*
- ☺ membership problem can be decided in linear time
  - for **constant** *k*, one can decide in linear time, whether a graph has tree-width at most *k*, and if so, compute a tree-decomposition of tree-width at most *k*
    - [Bodlaender, 1996]
- ☹ subgraph isomorphism remains NP-complete for graphs of bounded tree-width
  - **NP-complete** if the pattern is not k-connected **or** has more than *O*(*k*) vertices of unbounded degree; o/w it can be decided in polynomial time
    - [Gupta & Nishimura, 1996]

Mining Bounded Tree-Width Graphs

⇨ generic mining algoritm: candidate generation/test is **not** directly applicable

- because subgraph isomorphism is NP-complete

⇨ **polynomial delay: open question**

What about incremental polynomial time?

- modify the generic levelwise search graph mining algorithm
  - slide 34
  - **changes:** print frequent patterns directly after their generation (slide 35)

A Generic Levelwise Search Graph Mining Algorithm (recap)

Input : database D of transaction graphs and integer $t > 0$
Output: all frequent connected subgraphs

graph class of the transaction graphs

MAIN:

let $S_0 \subseteq \mathcal{G}$ be the set of frequent graphs consisting of a single labeled vertex
for $(l := 0;\ S_l \neq \emptyset;\ l := l + 1)$ **do**
  $C_{l+1} := S_{l+1} := \emptyset$
  forall $P \in S_l$ **do**
    print P
    forall $H \in \rho(P) \cap \mathcal{G}$ satisfying (i) $H \notin C_{l+1}$ and (ii) $\rho^{-1}(H) \subseteq S_l$ **do**
      add H to C_{l+1}
      if SUPPORTCOUNT$(H) \geq t$ **then**
        add H to S_{l+1}

H is a **strong** candidate

add a new labeled edge to P with at least one old vertex in **all possible ways**

SUPPORT COUNT(H, D, t):

counter $:= 0$
forall G in D **do**
  if $H \preccurlyeq G$ **then**
    $\text{counter} := \text{counter} + 1$
return counter

Modified Levelwise Search Graph Mining Algorithm

Input : database D of transaction graphs and integer $t > 0$
Output: all frequent connected subgraphs

MAIN:

let $S_0 \subseteq \mathcal{G}$ be the set of frequent graphs consisting of a single labeled vertex
print S_0
for $(l := 0;\ S_l \neq \emptyset;\ l := l + 1)$ **do**
  $C_{l+1} := S_{l+1} := \emptyset$
  forall $P \in S_l$ **do**
    forall $H \in \rho(P) \cap \mathcal{G}$ satisfying (i) $H \notin C_{l+1}$ and (ii) $\rho^{-1}(H) \subseteq S_l$ **do**
      add H to C_{l+1}
      if SUPPORTCOUNT$(H) \geq t$ **then**
        print H and add it to S_{l+1}
        add H to S_{l+1}

SUPPORT COUNT(H, D, t):

counter $:= 0$
forall G in D **do**
  if $H \preccurlyeq G$ **then**
    counter $:=$ counter $+ 1$
return counter

universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT
Fraunhofer IAIS

Efficiency Conditions for the Modified Graph Mining Algorithm

Thm: Let $\mathcal{G}$ be a graph class satisfying

(i) $\mathcal{G}$ is closed under taking subgraphs,

(ii) the membership problem in $\mathcal{G}$ can be decided in polynomial time,

(iii*) for any strong candidate pattern H and transaction graph G, $H \preccurlyeq G$ can be decided in time polynomial in the combined size of G and the set of frequent patterns generated before H, and

(iv) isomorphism for $\mathcal{G}$ can be decided in polynomial time.

If the transaction graphs in D belong to $\mathcal{G}$ then the previous algorithm lists the frequent connected subgraphs in incremental polynomial time.

proof: *exercise*

Application to Mining Bounded Tree-Width Graphs

Let $\mathcal{G}$ be the class of graphs of tree-width at most k for some constant k.

(i) $\mathcal{G}$ is closed under taking subgraphs.

(ii) The membership problem in $\mathcal{G}$ can be decided in polynomial time.

- one can decide in linear time, whether a graph has tree-width at most k [Bodlaender, 1996]

(iv) Isomorphism between graphs of tree-width at most k can be decided in polynomial time [Bodlaender, 1990].

$\Rightarrow$ to show that frequent connected subgraph mining in bounded tree-width graphs can be done in incremental polynomial time, we need to prove condition (iii*)

Mining Bounded Tree-width Graphs

H: strong candidate pattern generated by levelwise search

G: transaction graph

- both of tree-width at most k

$\mathcal{F}_H$: H + set of all frequent connected subgraphs generated by the modified levelwise search algorithm before H

Thm: Given H and G above, it can be decided in time poly$(\text{size}(G), \text{size}(\mathcal{F}_H))$, whether H is subgraph isomorphic to G

- poly$(\text{size}(G), \text{size}(\mathcal{F}_H))$: polynom of the combined size of G and the set of frequent patterns computed before H
- incremental polynomial time

rest of this and next talk(s): proof sketch

Main Idea of the Proof

to decide subgraph isomorphism from a strong candidate pattern H into a transaction graph G, we can utilize the information computed earlier for the subgraphs of H

- strong candidate pattern: each of its subgraphs is frequent
- only non-redundant information, certain set of tuples, is actually required
- number of non-redundant tuples is bounded by poly$(\text{size}(G), \text{size}(\mathcal{F}_H))$
- we can identify a superset S of the set of non-redundant tuples such that
  - the size of S is bounded by poly$(\text{size}(G), \text{size}(\mathcal{F}_H))$
  - each tuple in S can be computed in time poly$(\text{size}(G), \text{size}(\mathcal{F}_H))$

Preprocessing Step

compute a nice tree-decomposition $TD(G)$ for every transaction graph G

- nice tree-decomposition: rooted tree with 3 different types of nodes
  - leaf node: it has no children
  - separator node z: has a single child z' such that

$$\text{bag}(z) \subseteq \text{bag}(z')$$

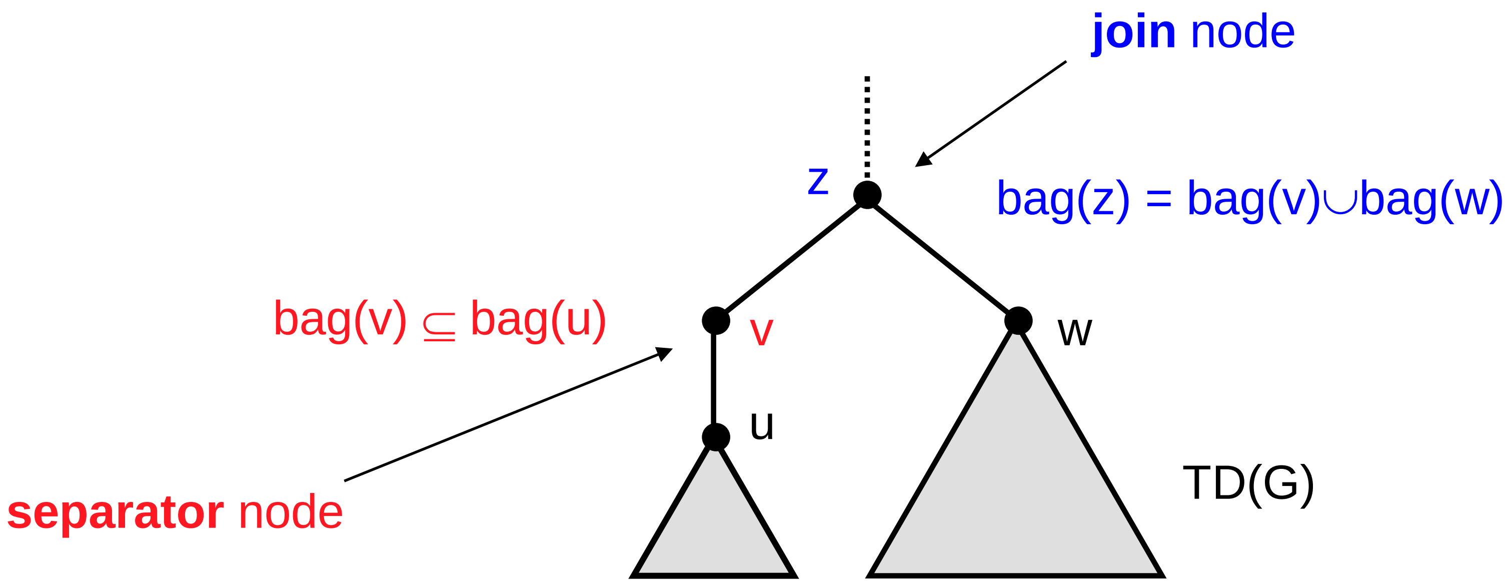
  - join node z: has two children z_1 and z_2 such that

$$\text{bag}(z) = \text{bag}(z_1) \cup \text{bag}(z_2)$$

- size of $TD(G)$ is linear in the size of G
- can be computed in linear time for graphs of bounded tree-width

use $TD(G)$ for the entire mining process

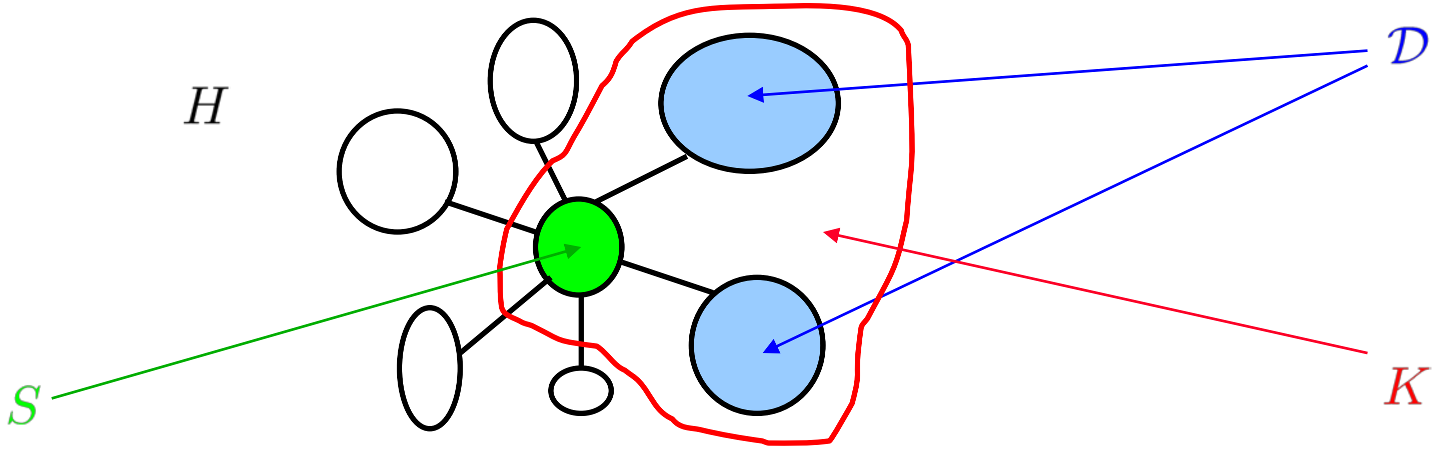
Nice Tree-Decomposition



Iso-Quadruples

given pattern graph H, transaction graph G, and tree-decomposition $TD(G)$

- iso-quadruple of H relative to a node z of $TD(G)$: $(S, \mathcal{D}, K, \psi)$ such that
  - $S \subseteq V(H)$ satisfying $|S| \leq k+1$,
  - $\mathcal{D}$ is a subset of the set of connected components of the induced subgraph $H[V(H) \setminus S]$,
  - K is the induced subgraph $H[S \cup V(\mathcal{D})]$,
  - ψ is an injective function mapping S to bag(z)



Iso-Quadruples

z-characteristic of H for a node z in $TD(G)$:

iso-quadruple $(S, \mathcal{D}, K, \psi)$ of H relative to z such that there is a subgraph isomorphism φ from K into $G_{[z]}$ satisfying

- $\varphi(u) = \psi(u)$ for every $u \in S$,
- $\varphi(v) \notin \text{bag}(z)$ for every $v \in V(\mathcal{D})$

induced subgraph of *G* defined by the union of the bags of *z*'s descendants (*z* is also a descendant of itself)

Lemma: H is subgraph isomorphic to G if and only if there exists an r-characteristic $(S, \mathcal{D}, H, \psi)$ for the root r of $TD(G)$

proof: *exercise*

$\Rightarrow$ check whether an iso-quadruple relative to the root is a characteristic

- the number of iso-quadruples to be checked for the root is bounded by

$$O(|V(H)|^{k+1})$$

Computing Characteristics

[Matoušek & Thomas, 1992; also Hajiaghayi & Nishimura, 2007]:
compute the set of *z*-characteristics for every node *z* in *TD(G)* with **dynamic programming**:

- postorder traversal of *TD(G)*
    - straightforward for *leaf* nodes (next slide),
    - use only the characteristics of the child(ren) for *separator* and *join* nodes (next slides)

notations: for pattern graph *H*, transaction graph *G*, both of bounded tree-width, nice tree-decomposition TD(*G*), and node *z* in TD(*G*):

- $\Gamma(H,z)$ denotes the set of iso-quadruples of *H* relative to *z*
- $\Gamma_{ch}(H,z)$ denotes the set of *z*-characteristics of *H*

Computing Characteristics: Leaf Nodes

Leaf Lemma: Let G and H be graphs of bounded treewidth and z be a leaf node in $TD(G)$. Then, for every $(S, \mathcal{D}, K, \psi) \in \Gamma(H, z)$

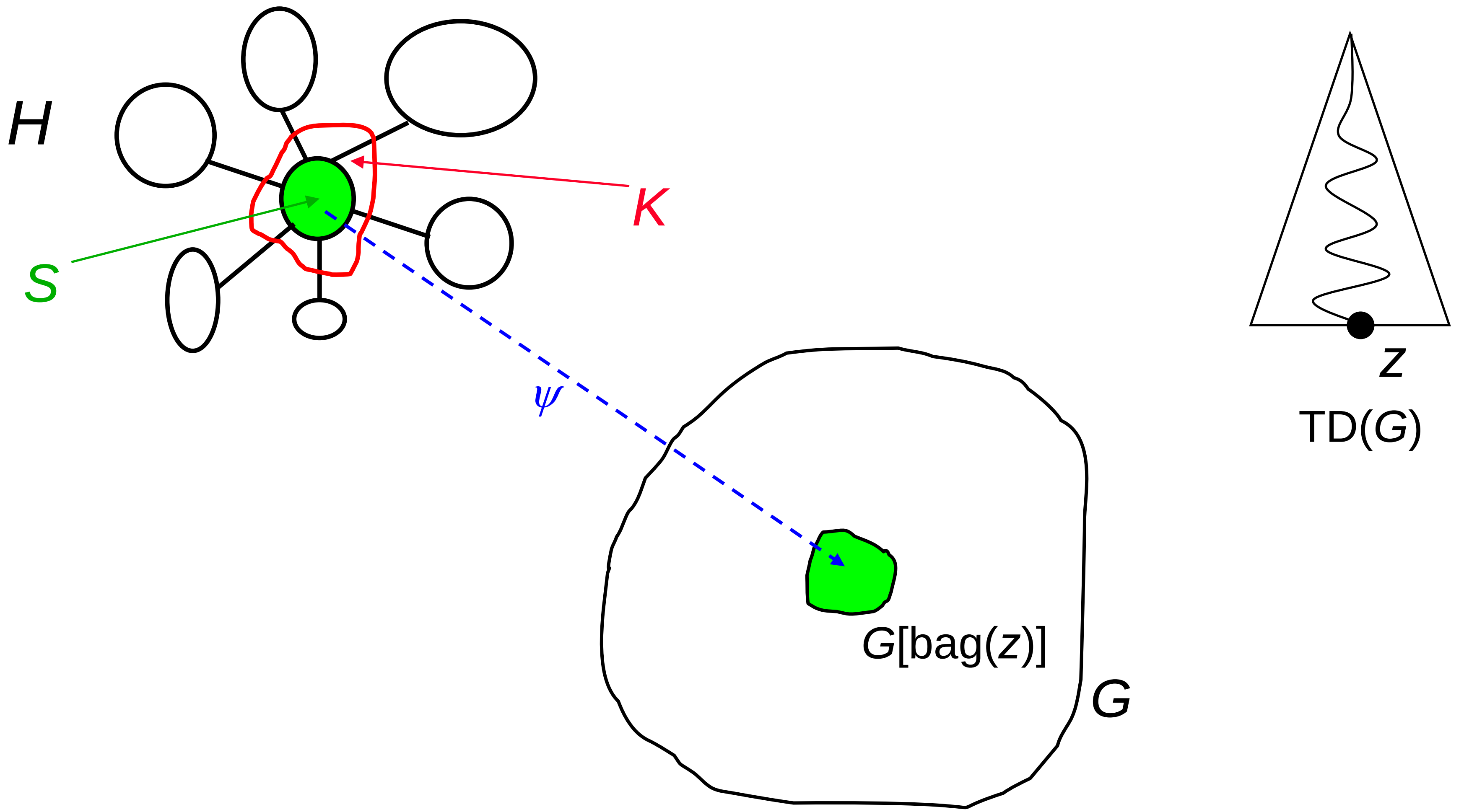
$$(S, \mathcal{D}, K, \psi) \in \Gamma_{\text{ch}}(H, z) \iff$$

$$\mathcal{D} = \emptyset \text{ and } \psi \text{ is a subgraph isomorphism from } H[S] \text{ to } G[\text{bag}(z)]$$

proof: *exercise*

- $\mathcal{D} = \emptyset$ implies $K = H[S]$

Computing Characteristics: Leaf Nodes



Computing Characteristics: Separator Nodes

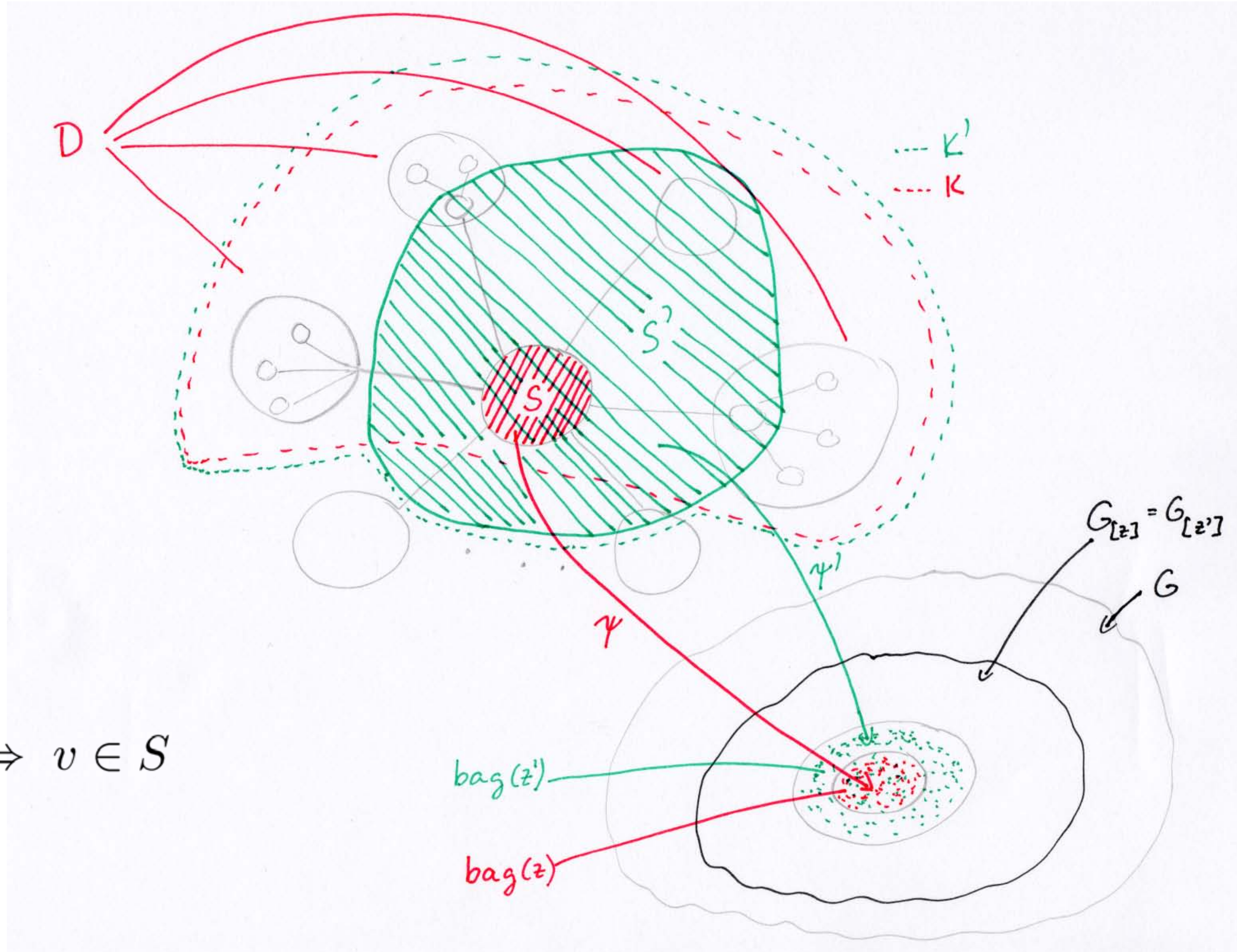
Separator Lemma: Let H, G be as in the previous case and z be a separator node in $TD(G)$ with (a single) child z'. Then for all $(S, \mathcal{D}, K, \psi) \in \Gamma(H, z)$

$$(S, \mathcal{D}, K, \psi) \in \Gamma_{\text{ch}}(H, z) \iff \exists (S', \mathcal{D}', K', \psi') \in \Gamma_{\text{ch}}(H, z') \text{ such that}$$

- $S = \{v \in S' : \psi'(v) \in \text{bag}(z)\}$,
- $\mathcal{D}'$: set of all connected components C of $H[V(H) \setminus S']$ such that C is a subgraph of (a connected component in) $\mathcal{D}$,
- $\psi(v) = \psi'(v)$ for every $v \in S$.

Proof: see [Hajiaghayi & Nishimura, 2007]

Computing Characteristics: Separator Nodes



- $\psi'(v) \in \text{bag}(z) \iff v \in S$
- $\psi = \psi'$ on S

Computing Characteristics: Join Nodes

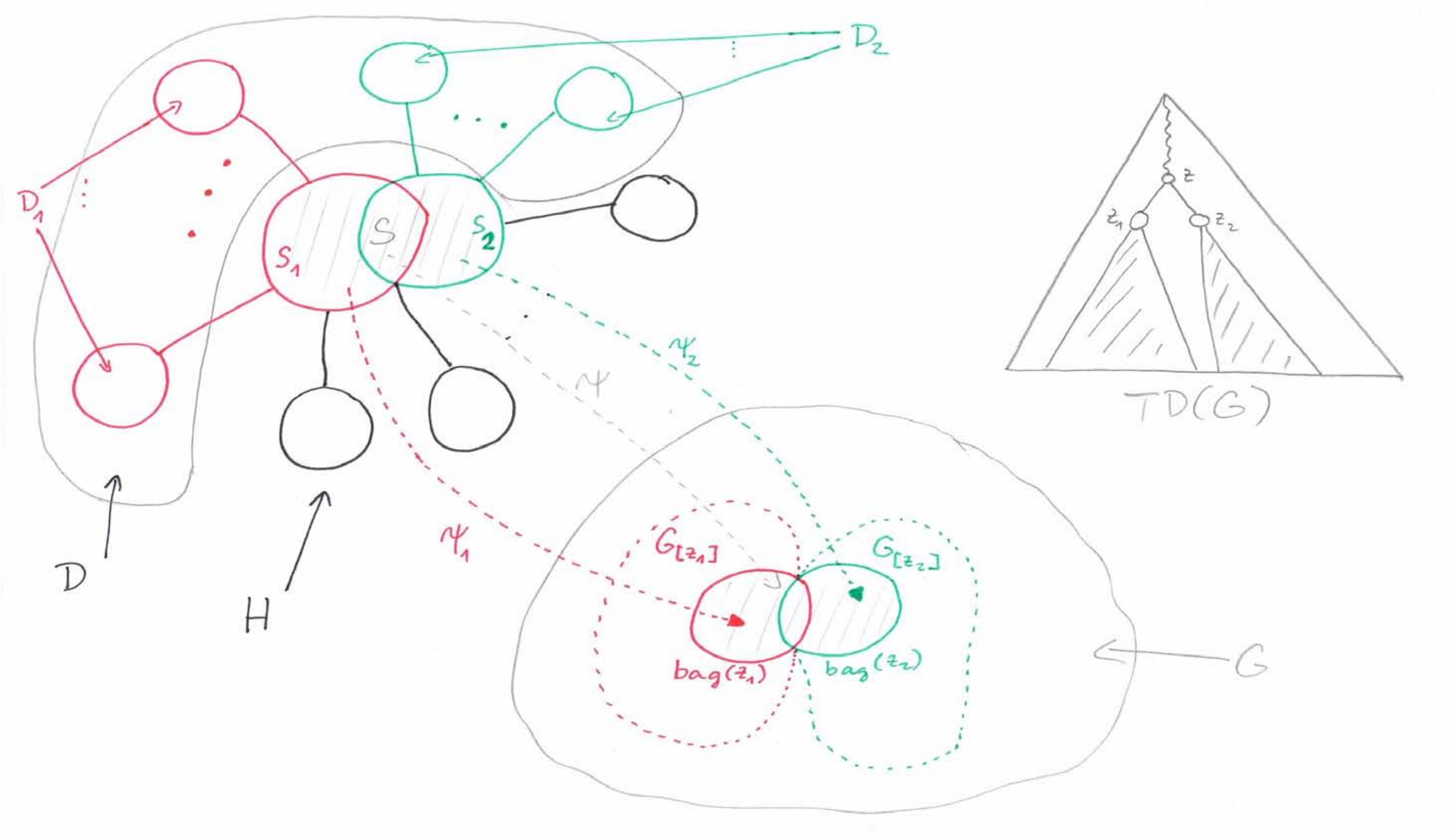
Join Lemma: Let H, G be as in the previous cases and z be a join node in $TD(G)$ with children z_1 and z_2. Then for all $(S, \mathcal{D}, K, \psi) \in \Gamma(H, z)$

$$(S, \mathcal{D}, K, \psi) \in \Gamma_{\text{ch}}(H, z) \iff$$
$$\exists(S_1, \mathcal{D}_1, K_1, \psi_1) \in \Gamma_{\text{ch}}(H, z_1) \wedge \exists(S_2, \mathcal{D}_2, K_2, \psi_2) \in \Gamma_{\text{ch}}(H, z_2) \text{ s.t.}$$

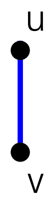
- $S_i = \{v \in S : \psi(v) \in \text{bag}(z_i)\}$ for $i = 1, 2$,
- the connected components of $\mathcal{D}$ is partitioned into $\mathcal{D}_1$ and $\mathcal{D}_2$
  - i.e., $\mathcal{D}_1$ and $\mathcal{D}_2$ are disjoint and $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$
- $\psi_i(v) = \psi(v)$ for every $v \in S_i$ $(i = 1, 2)$,
- ψ preserves the labels and the edges

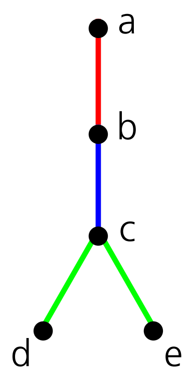
Proof: see [Hajiaghayi & Nishimura, 2007]

Computing Characteristics: Join Nodes



Example

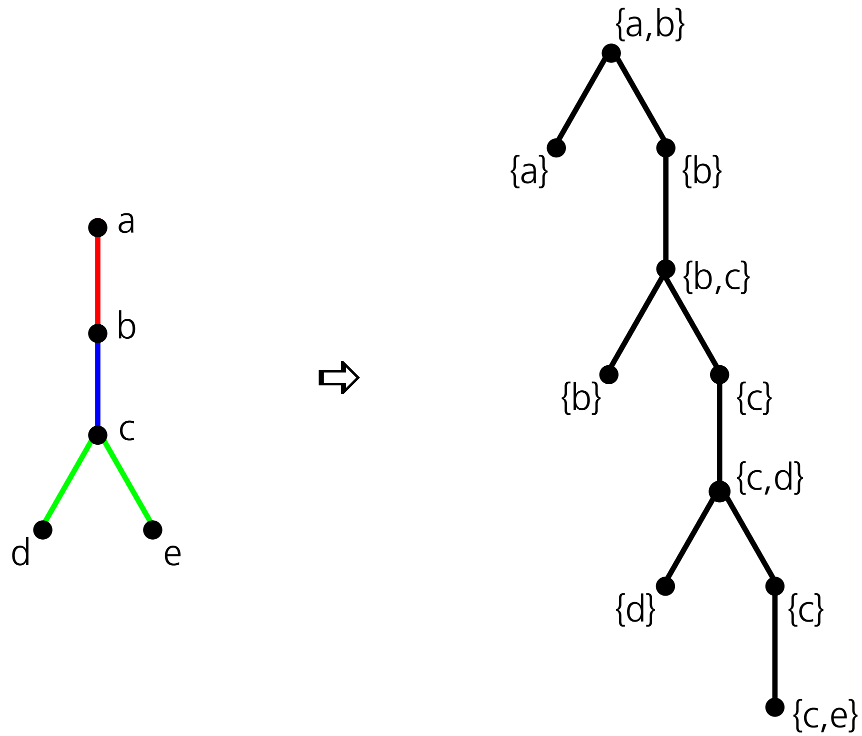
Is pattern *H* 

subgraph isomorphic to transaction graph *G* 

?

- all vertices in *H* and *G* have the same label (*not denoted*)
- edge labels are denoted by colors (i.e., there are 3 edge labels)

Example (cont'd) – Nice Tree-Decomposition



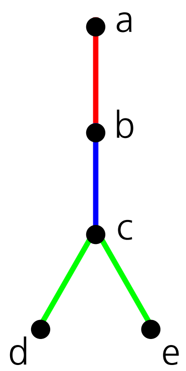
Example – Characteristics

pattern *H*:

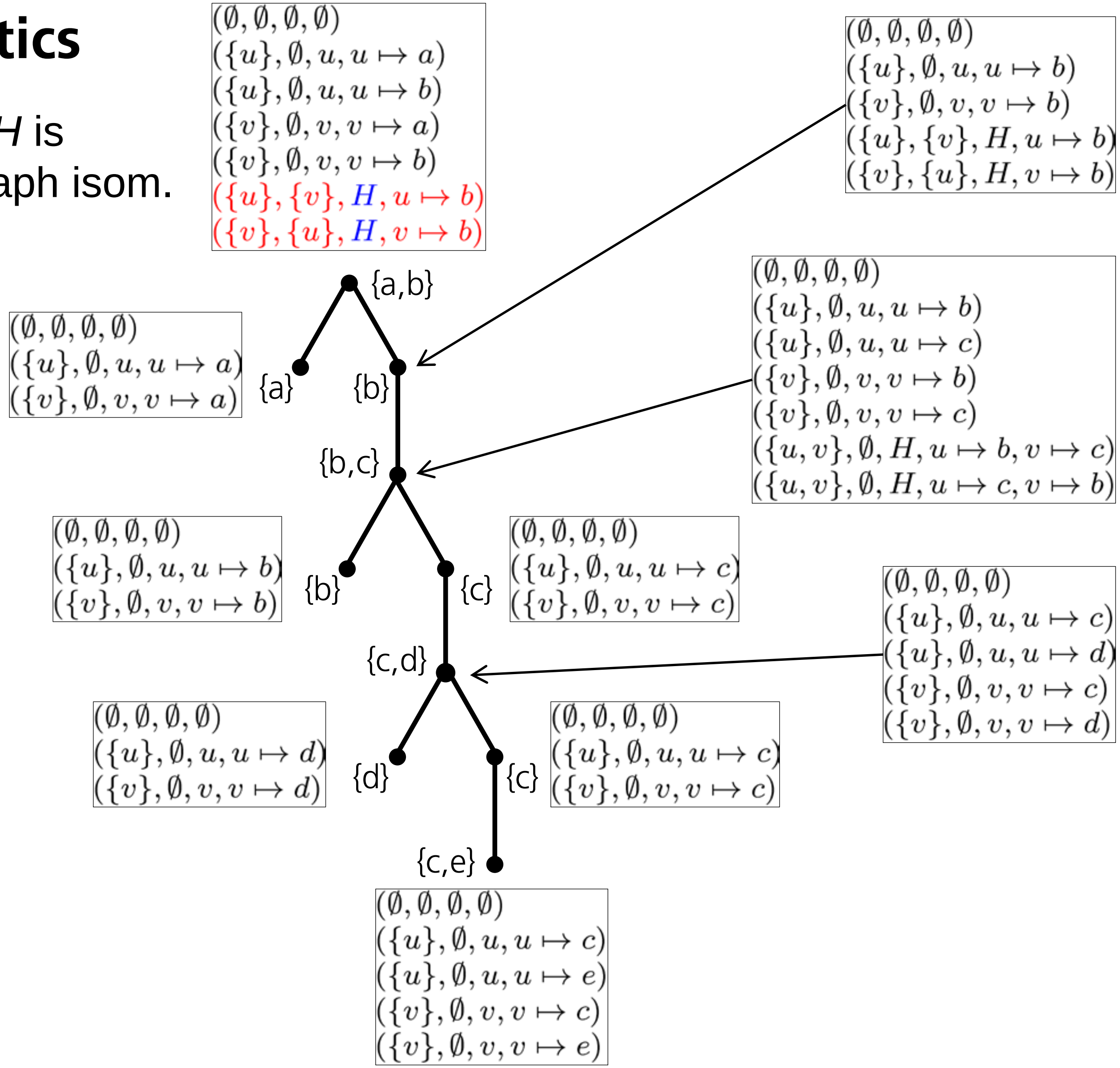
$(S, \mathcal{D}, K)$-triples for possible iso-quadruples:

- $(\emptyset, \emptyset, \emptyset)$
- $(\emptyset, H, H)$
- $(\{u\}, \emptyset, u)$
- $(\{u\}, v, H)$
- $(\{v\}, \emptyset, v)$
- $(\{v\}, u, H)$
- $(\{u, v\}, \emptyset, H)$

transaction graph *G*:



YES, *H* is subgraph isom. to *G*!



Computing Characteristics

problem: the number of iso-quadruples for separator and join nodes can be exponentially large

Thm: For graphs of bounded tree-width and **bounded degree**, the set of z-characteristics can be computed in **polynomial time** for every node z

- [Matoušek & Thomas, 1992]

- we cannot use this result
  - no additional assumption besides bounded tree-width

Equivalent Iso-Quadruples

- H_1, H_2, G: connected graphs of bounded treewidth
- $TD(G)$: nice tree-decomposition of G computed in the preprocessing step
- z: node in $TD(G)$
- $\xi_1 = (S_1, \mathcal{D}_1, K_1, \psi_1) \in \Gamma(H_1, z)$; $\xi_2 = (S_2, D_2, K_2, \psi_2) \in \Gamma(H_2, z)$

Def.: ξ_1 is equivalent to ξ_2 if there is an isomorphism π between K_1 and K_2 such that

- π is a bijection between S_1 and S_2
- $\psi_1(v) = \psi_2(\pi(v))$ for every $v \in S_1$

Notation: $\xi_1 \equiv \xi_2$

Equivalent Iso-Quadruples

Prop. 1: Equivalence of iso-quadruples can be decided in polynomial time.

Proof: *exercise*

Prop. 2: If ξ_1, ξ_2 in the above definition are equivalent then

$$\xi_1 \in \Gamma_{\mathsf{ch}}(H_1, z) \iff \xi_2 \in \Gamma_{\mathsf{ch}}(H_2, z)$$

Proof: *exercise*

$\Rightarrow$ it suffices to store only one representative z-characteristic for each equivalence class of the set of z-characteristics

Non-Redundant Iso-Quadruples

⇒ **given** a strong candidate pattern H generated by levelwise search and a node z in $TD(G)$, it is **sufficient to check** only those iso-quadruples of $\Gamma(H,z)$ for which there exists no equivalent z-characteristic for some frequent pattern listed before H

- non-redundant iso-quadruples of H relative to z
- $\Gamma_{\text{nr}}(H,z)$: set of non-redundant iso-quadruples of H relative to z
- $\Gamma_{\text{nr,ch}}(H,z)$: set of non-redundant z-characteristics of H

Non-Redundant Iso-Quadruples

Prop.: Let H be a strong candidate pattern, G be a transaction graph, both of bounded treewidth, z be a node in $TD(G)$, and $\xi \in \Gamma(H, z)$. Then

$$\xi \in \Gamma_{\text{ch}}(H, z) \iff \exists \xi' \in \bigcup_{P \in \mathcal{F}_H} \Gamma_{\text{nr,ch}}(P, z) \text{ such that } \xi' \equiv \xi$$

– $\mathcal{F}_H$: $\{H\} \cup$ set of frequent patterns computed before H

Proof: *exercise*

Non-Redundant Iso-Quadruples

Theorem

(i) The number of non-redundant iso-quadruples of H relative to z is bounded by

$$O(|V(H)|^{k+1}).$$

- instead of computing $\Gamma_{\mathsf{nr}}(H, z)$, we will efficently compute a superset $\Gamma_{\mathsf{f}}(H, z) \supseteq \Gamma_{\mathsf{nr}}(H, z)$ of cardinality bounded by $O(|V(H)|^{k+1})$

$\Rightarrow$ we must test only for polynomially many iso-quadruples whether they are characteristics

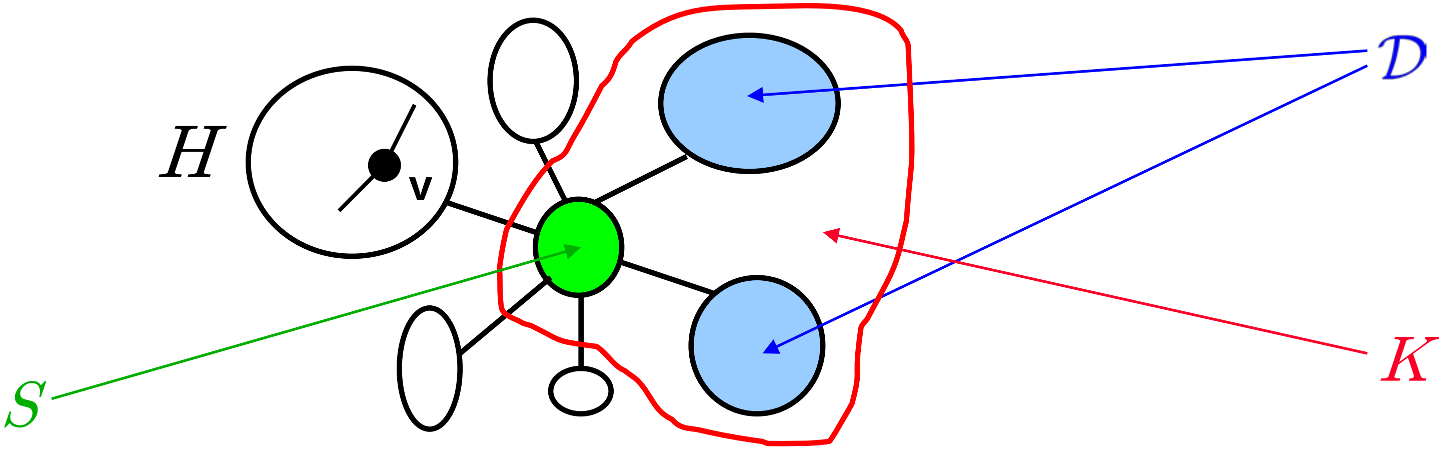
(ii) It can be decided in time polynomial in the combined size of G and the set of frequent patterns listed before H whether an iso-quadruple in $\Gamma_{\mathsf{f}}(H, z)$ is a characteristic.

Claim (i)

Lemma: Let H be a strong candidate pattern generated by levelwise search, z be a node in $TD(G)$, and $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma_{\text{nr}}(H, z)$. Then, for every vertex $v \in V(H) \setminus V(K)$ it holds that

(i) the degree of v in H is at least 2 and

(ii) H has no cycle containing v

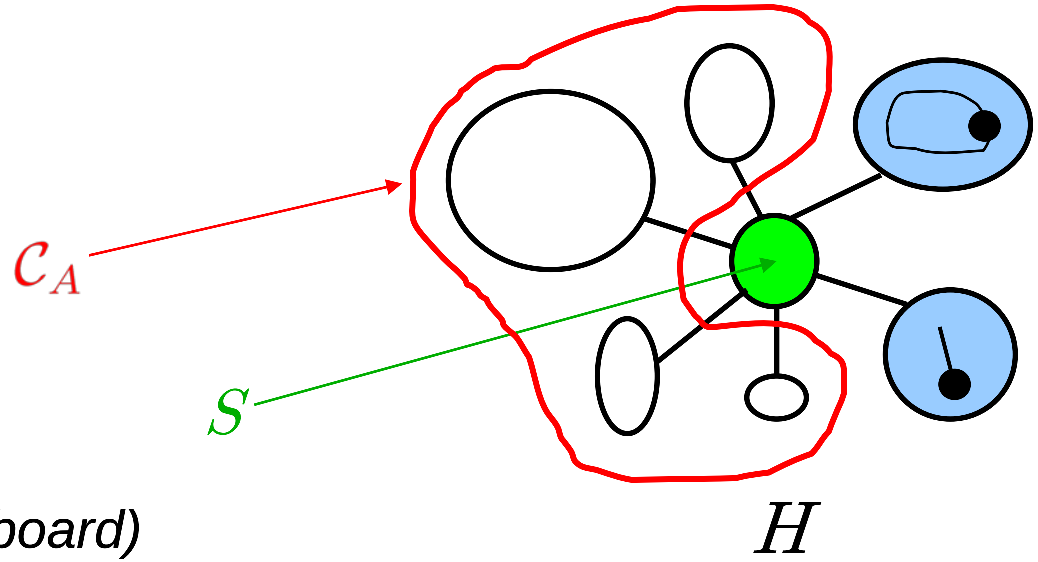


Proof: *(blackboard)*

Claim (i)

Lemma: Let H be a strong candidate pattern generated by levelwise search, S be a subset of $V(H)$ having at most $k+1$ elements, and $\mathcal{C}_A$ be the maximal subset of the set of connected components of $H[V(H) \setminus S])$ such that for every $C \in \mathcal{C}_A$, all vertices of C satisfy (i) and (ii) in the lemma on the previous slide. Then

$$|\mathcal{C}_A| \leq k \ .$$



Proof: *(blackboard)*

Claim (i)

feasible iso-quadruples: for a strong candidate pattern H generated by level-wise search and for a node z in $TD(G)$, an iso-quadruple $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma(H, z)$ is called feasible if it satisfies the previous two conditions, i.e.,

(i) the degree of v in H is at least 2 and

(ii) H has no cycle containing v

for every $v \in V(H) \setminus V(K)$

- set of feasible iso-quadruples in $\Gamma(H, z)$ is denoted by $\Gamma_{\mathsf{f}}(H, z)$

$\Rightarrow \quad \Gamma_{\mathsf{nr}}(H, z) \subseteq \Gamma_{\mathsf{f}}(H, z)$

Claim (i)

Thm.: Let H be a strong candidate pattern generated by levelwise search and let z be a node of $TD(G)$ for some transaction graph G. Then

$$|\Gamma_{\mathsf{f}}(H,z)| \leq O(|V(H)|^{k+1})$$

proof: How many different $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma_{\mathsf{f}}(H,z)$ can we have at most?

- S can be chosen in at most $|V(H)|^{k+1}$ different ways,
- $\mathcal{D}$ can be chosen in at most $2^{|\mathcal{C}_A|} \leq 2^k$ different ways (lemma on slide 61),
- there are at most $(k+1)!$ injective functions from $H[S]$ to $G[\text{bag}(z)]$,

$\Rightarrow$ we have

$$|\Gamma_{\mathsf{f}}(H,z)| \leq 2^k \cdot (k+1)! \cdot |V(H)^{k+1}| \ ,$$

from which the claim directly follows, as k is assumed to be a constant.

Claim (ii): Algorithm Computing Feasible Characteristics

Input:

- connected strong pattern graph H generated by levelwise search and transaction graph G, both of tree-width at most k for some constant k,
- a nice tree-decomposition $TD(G)$ of G with nodes associated with

$$\bigcup_{P \in \mathcal{F}_H \setminus \{H\}} \Gamma_{\text{f,ch}}(P, w)$$

  for every node w in $TD(G)$, and
- a node z in $TD(G)$

Output: set $\Gamma_{\text{f,ch}}(H, z)$ of feasible z-characteristics

Algorithm: *next slide*

Claim (ii): Algorithm FeasibleCharacteristics

next slide

$\Gamma_{\text{f,ch}}(H, z) := \emptyset$
$\Gamma_{\text{f}}(H, z) :=$ FEASIBLEISOQUADRUPLES$(H, \text{bag}(z))$
forall $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma_{\text{f}}(H, z)$ **do** // (ψ is a subgraph isom. from $H[S]$ to $G[\text{bag}(z)]$)
  if z is a leaf node **then**
    if ξ satisfies the condition of the Leaf Lemma **then** add ξ to $\Gamma_{\text{f,ch}}(H, z)$
  else if z is a separator node with child z'
    $\Gamma_{\text{f,ch}}(H, z') :=$ FEASIBLECHARACTERISTICS$(H, G, TD(G), z')$
    if $\exists \xi' \in \mathfrak{S}(\xi)$ and $\exists \xi'' \in \Gamma_{\text{f,ch}}(H, z')$ such that $\xi' \equiv \xi''$ **then**
      add ξ to $\Gamma_{\text{f,ch}}(H, z)$
  else (z is a join node with children z_1 and z_2)
    $\Gamma_{\text{f,ch}}(H, z_1) :=$ FEASIBLECHARACTERISTICS$(H, G, TD(G), z_1)$
    $\Gamma_{\text{f,ch}}(H, z_2) :=$ FEASIBLECHARACTERISTICS$(H, G, TD(G), z_2)$
    if $\left(\exists \xi_1 \in \bigcup_{P \in \mathcal{F}_H} \Gamma_{\text{f,ch}}(P, z_1)\right) \wedge \left(\exists \xi_2 \in \bigcup_{P \in \mathcal{F}_H} \Gamma_{\text{f,ch}}(P, z_2)\right)$ such that $\xi \equiv \oplus_\xi(\xi_1, \xi_2)$
      then add ξ to $\Gamma_{\text{f,ch}}(H, z)$
return $\Gamma_{\text{f,ch}}(H, z)$

to be defined

join operator; will be defined

universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT
Fraunhofer IAIS

I. Computing Feasible Iso-Quadruples (Line 2)

2: $\Gamma_{\mathsf{f}}(H, z) := \textsc{FeasibleIsoQuadruples}(H, \mathsf{bag}(z))$

Lemma: $\Gamma_{\mathsf{f}}(H, z)$ can be computed in time $O(|V(H)|^{k+1}|)$

Proof: By slides 62 and 63, the algorithm below is correct and computes $\Gamma_{\mathsf{f}}(H, z)$ in time $O(|V(H)|^{k+1}|)$.

Input: connected graph H and $\mathsf{bag}(z)$ of a node z in $TD(G)$
Output: $\Gamma_{\mathsf{f}}(H, z)$

```
Y := ∅
forall S ⊆ V(H) satisfying |S| ≤ |bag(z)| do
    let 𝒞 be the set of connected components of H[V(H) \ S]
    compute the subset 𝒞_A ⊆ 𝒞 defined on slide 61
    forall 𝒟' ⊆ 𝒞_A do
        𝒟 := 𝒞 \ 𝒟'
        forall subgraph isomorphisms ψ : S → bag(z) from H[S] to G[bag(z)] do
            add (S, 𝒟, H[S ∪ V(𝒟)], ψ) to Y
return Y
```

II. Leaf Nodes (Line 5)

5: **if** ξ satisfies the condition of the Leaf Lemma **then** add ξ to $\Gamma_{\text{f,ch}}(H, z)$

condition of the Leaf Lemma for $\xi = (S, \mathcal{D}, K, \psi)$:

$\mathcal{D} = \emptyset$ and ψ is a subgraph isomorphism from $H[S]$ to $G[\text{bag}(z)]$

Lemma: The condition of the Leaf Lemma can be decided in constant time.

Proof: ψ is already a required subgraph isomorphism

- see lines 7–8 on the previous slide

III. Separator Nodes (Lines 7-9)

// z is a separator node in $TD(G)$ with child z'

7: $\Gamma_{\text{f,ch}}(H, z') :=$ FEASIBLECHARACTERISTICS$(H, G, TD(G), z')$

8: **if** $\exists \xi' \in \underline{\mathfrak{S}(\xi)}$ and $\exists \xi'' \in \Gamma_{\text{f,ch}}(H, z')$ such that $\xi' \equiv \xi''$ **then**

9: add ξ to $\Gamma_{\text{f,ch}}(H, z)$

let $\xi \in \Gamma_{\text{f}}(H, z)$ be of the form $\xi = (S, \mathcal{D}, K, \psi)$

$\underline{\mathfrak{S}(\xi)}$: set of iso-quadruples $(S', \mathcal{D}', K', \psi') \in \Gamma(H, z')$ satisfying the conditions of the **Separator Lemma**, i.e.,

(S.a) $S = \{v \in S' : \psi'(v) \in \text{bag}(z)\}$,

(S.b) $\mathcal{D}' = \{C :$ C is a connected component of $H[V(H) \setminus S']$ and C is a subgraph of $\mathcal{D}\}$,

(S.c) $\psi(v) = \psi'(v)$ for every $v \in S$.

III. Separator Nodes (Lines 7-9)

Lemma: Suppose that $\Gamma_{\text{f,ch}}(P, z')$ has been computed correctly by the alg. on Slide 65 for every $P \in \mathcal{F}_H$. Then the alg. on Slide 65 computes $\Gamma_{\text{f,ch}}(H, z)$ correctly in time polynomial in $|V(H)|$.

Proof (sketch): (correctness)

Claim: $\mathfrak{S}(\xi) \subseteq \Gamma_{\text{f}}(H, z')$ for all $\xi \in \Gamma_{\text{f}}(H, z)$

Proof (sketch): let $\xi \in \Gamma_{\text{f}}(H, z)$ with $\xi = (S, \mathcal{D}, K, \psi)$

$\Rightarrow$ K is a subgraph of K' for all $\xi' = (S', \mathcal{D}', K', \psi') \in \mathfrak{S}(\xi)$ by (S.b) (see previous slide)

$\Rightarrow$ all $v \in V(H) \setminus V(K')$ satisfy the two conditions on slide 62

$\Rightarrow$ ξ' is feasible

$\Rightarrow$ correctness follows together with Prop. 2 (Slide 56) and the Separator lemma

III. Separator Nodes (Lines 7-9)

Proof (sketch) of the lemma on the previous slide: (efficiency)

1. compute $\mathfrak{S}(\xi)$

- $\mathfrak{S}(\xi)$ has at most $(k+1)! \cdot |V(H)|^{k+1}$ elements and it can be computed in time polynomial in $|V(H)|$

2. for all $\xi' \in \mathfrak{S}(\xi)$, check whether $\xi' \equiv \xi''$ for some $\xi'' \in \Gamma_{\mathsf{f,ch}}(H, z')$

- $|\Gamma_{\mathsf{f,ch}}(H, z')| \leq |\Gamma_{\mathsf{f}}(H, z')| = O(|V(H)|^{k+1})$ by the theorem on Slide 63
- equivalence between iso-quadruples can be decided in polynomial time (Prop. 1 on Slide 56)

$\Rightarrow$ the condition above for ξ' can be decided in polynomial time

IV. Join Nodes (Lines 11-14)

// z is a join node in $TD(G)$ with children z_1 and z_2

11: $\Gamma_{\text{f,ch}}(H, z_1) :=$ FEASIBLECHARACTERISTICS$(H, G, TD(G), z_1)$

12: $\Gamma_{\text{f,ch}}(H, z_2) :=$ FEASIBLECHARACTERISTICS$(H, G, TD(G), z_2)$

13: **if** $\exists \xi_1 \in \bigcup_{P \in \mathcal{F}_H} \Gamma_{\text{f,ch}}(P, z_1) \wedge \exists \xi_2 \in \bigcup_{P \in \mathcal{F}_H} \Gamma_{\text{f,ch}}(P, z_2)$ such that $\xi \equiv \oplus_\xi(\xi_1, \xi_2)$ **then**

14: add ξ to $\Gamma_{\text{f,ch}}(H, z)$

let $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma_{\text{f}}(H, z)$ and $\xi_i = (S_i, \mathcal{D}_i, K_i, \psi_i) \in \Gamma_{\text{f}}(H, z_i)$ $(i = 1, 2)$
// ψ is guaranteed a subgraph isom. from $H[S]$ to $G[\text{bag}(z)]$ by slide 66

ξ_1 and ξ_2 are **join consistent** with ξ if they satisfy the following conditions in the **Join Lemma**:

(J.a) $S_i = \{v \in S : \psi(v) \in \text{bag}(z_i)\}$ for $i = 1, 2$,

(J.b) the connected components of $\mathcal{D}$ are partitioned into $\mathcal{D}_1$ and $\mathcal{D}_2$, and

(J.c) for $i = 1, 2$, $\psi_i(v) = \psi(v)$ for every $v \in S_i$.

IV. Join Nodes (Lines 11-14): The Join Operator

let z be a join node with children z_1 and z_2, and let

- $\xi_0 = (S_0, \mathcal{D}_0, K_0, \psi_0) \in \Gamma(H_0, z_0)$,
- $\xi_1 = (S_1, \mathcal{D}_1, K_1, \psi_1) \in \Gamma(H_1, z_1)$ and $\xi_2 = (S_2, \mathcal{D}_2, K_2, \psi_2) \in \Gamma(H_2, z_2)$

for some graphs H_0, H_1, and H_1

- assumption (w.l.o.g.): K_0, K_1, and K_2 are pairwise vertex disjoint

the **join** of ξ_1 and ξ_2 w.r.t. ξ_0, denoted $\oplus_{\xi_0}(\xi_1, \xi_2)$, is a **quadruple** $\xi = (S, \mathcal{D}, K, \psi)$ with

- $S = \{w_u : u \in \psi_1(S_1) \cup \psi_2(S_2)\}$ is a set of **new** vertices,
- $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$,
- K is the graph ... *(continued on the next slide)*

IV. Join Nodes (Lines 11-14): The Join Operator

- K is the graph obtained from the union of S and $\mathcal{D}$ by labeling all $w_u \in S$ with the label of the corresponding vertex u and connecting

  (i) $w_{\psi_i(u)}$ and $w_{\psi_i(v)}$ by an edge associated with the label of $\{u,v\}$ for all $\{u,v\} \in E(K_i)$ such that $u, v \in S_i$ $(i = 1,2)$,

  (ii) $w_{\psi_i(u)}$ and v by an edge associated with the label of $\{u,v\}$ for all $\{u,v\} \in E(K_i)$ such that $u \in S_i$ and $v \in V(\mathcal{D}_i)$ $(i = 1,2)$, and

  (iii) $w_{\psi_1(u)}$ and $w_{\psi_2(v)}$ by an edge associated with the label of $\{u',v'\}$ for all $\{u',v'\} \in E(K_0)$ such that $\psi_0(u') = \psi_1(u)$, $\psi_0(v') = \psi_2(v)$ for some $u \in S_1$, $v \in S_2$, and $\psi_1(u), \psi_2(v) \notin \psi_1(S_1) \cap \psi_2(S_2)$,

- $\psi : S \to X_z$ is defined by $\psi(w_u) = u$ for every $w_u \in S$.

IV. Join Nodes (Lines 11-14): The Join Operator

Prop.: Let z be a join node with children z_1 and z_2, and let $\xi_i = (S_i, \mathcal{D}_i, K_i, \psi_i) \in \Gamma_{\text{ch}}(H_i, z_i)$ for some graph H_i ($i = 0, 1, 2$) such that $z_0 = z$, and K_0, K_1, and K_2 are pairwise vertex disjoint.
Then $\oplus_{\xi_0}(\xi_1, \xi_2) = (S, \mathcal{D}, K, \psi)$ is uniquely defined and it is an element of $\Gamma(P, z)$ for some pattern P.

Proof (sketch):

- uniqueness is straightforward
- any graph P satisfying $K = P[V(K)]$ is appropriate
- $|S| \leq k + 1$
- since the ψ_i's are all injective, ψ is also injective

universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer IAIS

IV. Join Nodes (Lines 11-14)

Lemma: Let z be a join node in $TD(G)$ with children z_1 and z_2, and suppose that $\Gamma_{\text{f,ch}}(P, z_i)$ has been computed correctly by Alg. FEASIBLECHARACTERISTICS for every $P \in \mathcal{F}_H$ and $i = 1, 2$.
Then Alg. FEASIBLECHARACTERISTICS computes $\Gamma_{\text{f,ch}}(H, z)$ correctly in time polynomial in the combined size of H and the set of frequent patterns computed before H.

Proof (sketch): For the correctness, let $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma_{\text{f}}(H, z)$. Then, as ψ is a subgraph isomorphism from $H[S]$ to $G[\text{bag}(z)]$ by the alg. on slide 66, we have

$$\begin{aligned}
&\xi \in \Gamma_{\text{f,ch}}(H, z) \\
&\quad \Longleftrightarrow \quad \exists \xi_i \in \Gamma_{\text{ch}}(H, z_i) \text{ for } i = 1, 2 \text{ such that } \xi_1, \xi_2, \text{ and } \xi \\
&\qquad\qquad\quad \text{satisfy the conditions of the Join Lemma} \\
&\quad \Longleftrightarrow \quad \exists \xi_i \in \bigcup_{P \in \mathcal{F}_H} \Gamma_{\text{f,ch}}(P, z_i) \text{ for } i = 1, 2 \text{ with } \xi \equiv \oplus_{\xi}(\xi_1, \xi_2)
\end{aligned}$$

IV. Join Nodes (Lines 11-14)

Proof (sketch) of the lemma on the previous slide: (efficiency)

1. the join operator can be computed in polynomial time

2. the number of pairs ξ_1, ξ_2 to be considered is bounded by

$$\text{poly}(\text{size}(H), \text{size}(\mathcal{F}_H))$$

- the last equivalence on the previous slides implies **quadratic** time algorithm in $\text{size}(\mathcal{F}_H)$
- in fact, it can be done in time **linear** in $\text{size}(\mathcal{F}_H)$

3. equivalence of iso-quadruples can be decided in polynomial time
(Prop. 1 on Slide 56)

Putting Together

Thm: The algorithm on the next two slides lists frequent connected subgraphs in incremental polynomial time.

Proof: Using the previous results, it follows by induction on the depth of the tree-decomposition of the transaction graph.

The Mining Algorithm

Input: database DB of graphs of tree-width at most k and integer $t > 0$
Output: all t-frequent connected subgraphs

// (preprocessing of the transaction graphs)
forall G in DB **do**
    compute a nice tree-decomposition $TD(G)$ of G
    forall node z in $TD(G)$ **do** $\Sigma_{G,z} := \{(\emptyset, \emptyset, \emptyset, \emptyset)\}$
// (computing frequent subgraphs consisting of a single vertex)
$S_0 = \emptyset$
forall graphs H consisting of a single labeled vertex **do** $\textsc{Process}(H, S_0)$ *// next slide*
// (computing frequent subgraphs consisting of at least one edge)
for $(l := 0;\ S_l \neq \emptyset;\ l := l + 1)$ **do**
    $C_{l+1} := S_{l+1} := \emptyset$
    forall $P \in S_l$ **do**
        forall $H \in \rho(P)$ satisfying tree-width$(H) \leq k \wedge H \notin C_{l+1} \wedge \rho^{-1}(H) \subseteq S_l$ **do**
            add H to C_{l+1}
            $\textsc{Process}(H, S_{l+1})$ *// next slide*

Function Process (Lines 7 and 14)

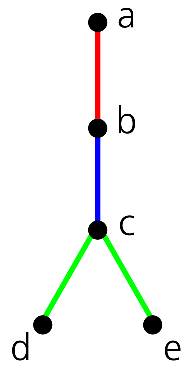
function $\textsc{Process}(H, S)$

counter $:= 0$
forall G in DB **do**
  unmark G
  $r :=$ root of $TD(G)$
  $\Gamma_{\text{f,ch}}(H, r) := \textsc{FeasibleCharacteristics}(H, G, TD(G), r)$ // (Slide 66)
  if $\exists(S, \mathcal{D}, H, \psi) \in \Gamma(H, r)$ equivalent to some $\xi \in \Sigma_{G,r} \cup \Gamma_{\text{f,ch}}(H, r)$ **then**
    counter $:=$ counter $+ 1$ // (H is subgraph isomorphic to G)
    mark G
if counter $\geq t$ **then** // (H is frequent)
  print H and add it to S
  forall G in DB such that G is marked **do**
    forall node z in $TD(G)$ **do** $\Sigma_{G,z} := \Sigma_{G,z} \cup \Gamma_{\text{f,ch}}(H, z)$

Example

mining problem:

list all 1-frequent connected subgraphs of the database consisting of the single transaction graph *G:*

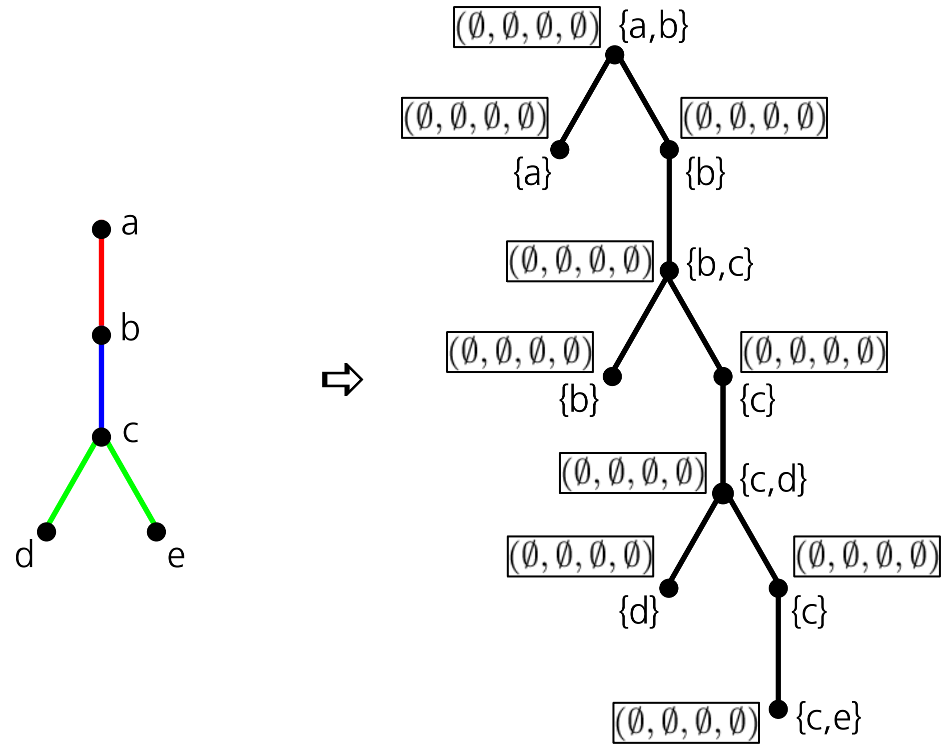


- i.e., all subtrees
- all vertices in *H* and *G* have the same label (*not denoted*)
- edge labels are denoted by colors (i.e., there are 3 edge labels)
  - see also the previous example

Example (cont'd)

Steps 1 – 4 of the alg. on slide 78:

- compute nice tree-decomposition of *G*
- assign the empty iso-quadruple to each node



Example: Feasible Characteristics

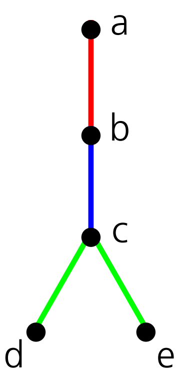
Steps 6-7 of the alg. on Slide 78

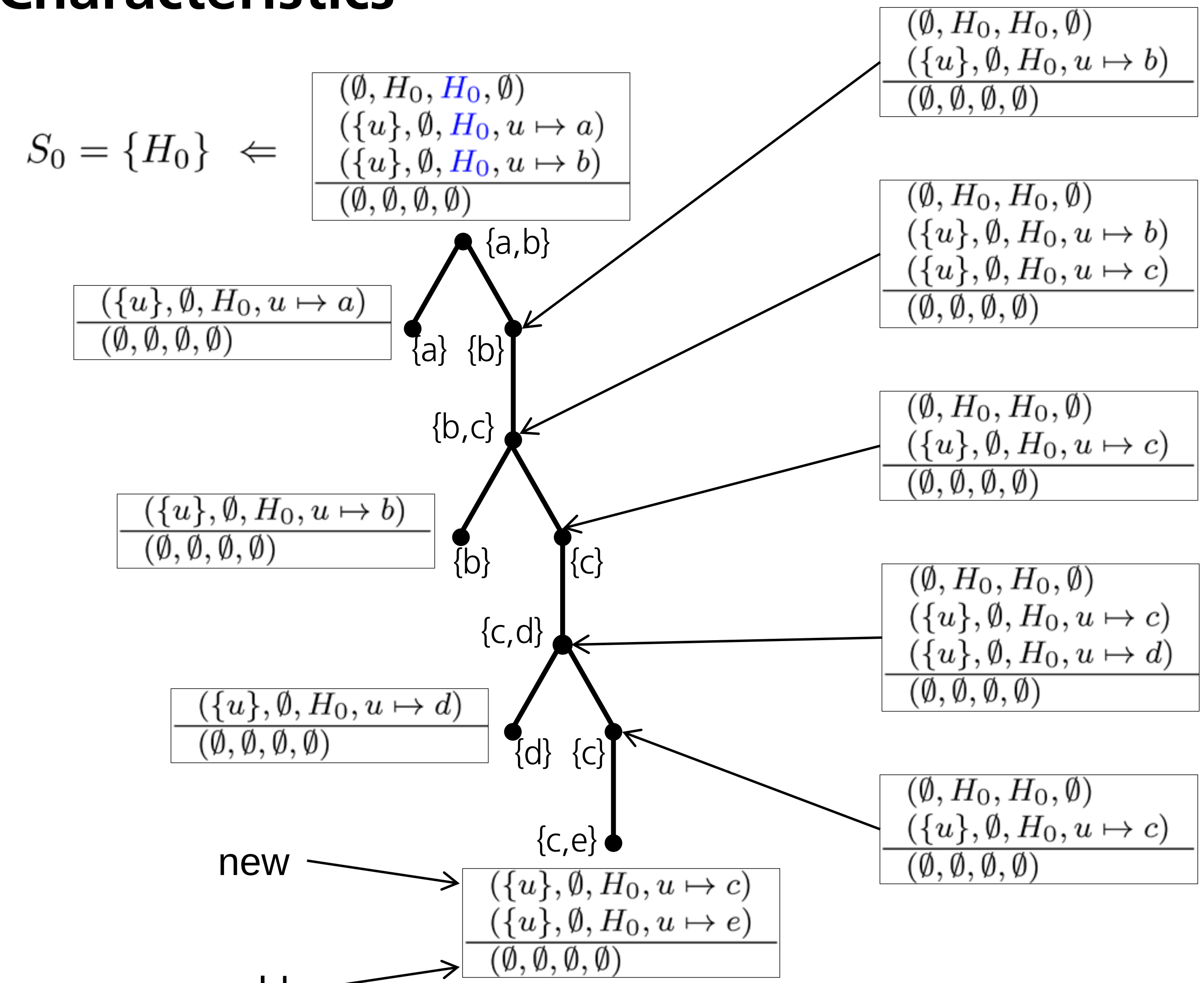
pattern H_0 : $\bullet_u$

$(S, \mathcal{D}, K)$-triples for possible feasible iso-quadruples:

- $\cancel{(\emptyset, \emptyset, \emptyset)}$, $(\emptyset, H_0, H_0)$
- $(\{u\}, \emptyset, H_0)$

transaction graph *G*:





universitätbonn

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer IAIS

Example (cont'd)

Step 12 of the alg. on slide 78

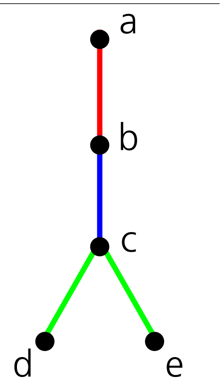
$\rho(\bullet) = \{$•—•, •—•, •—•$\}$

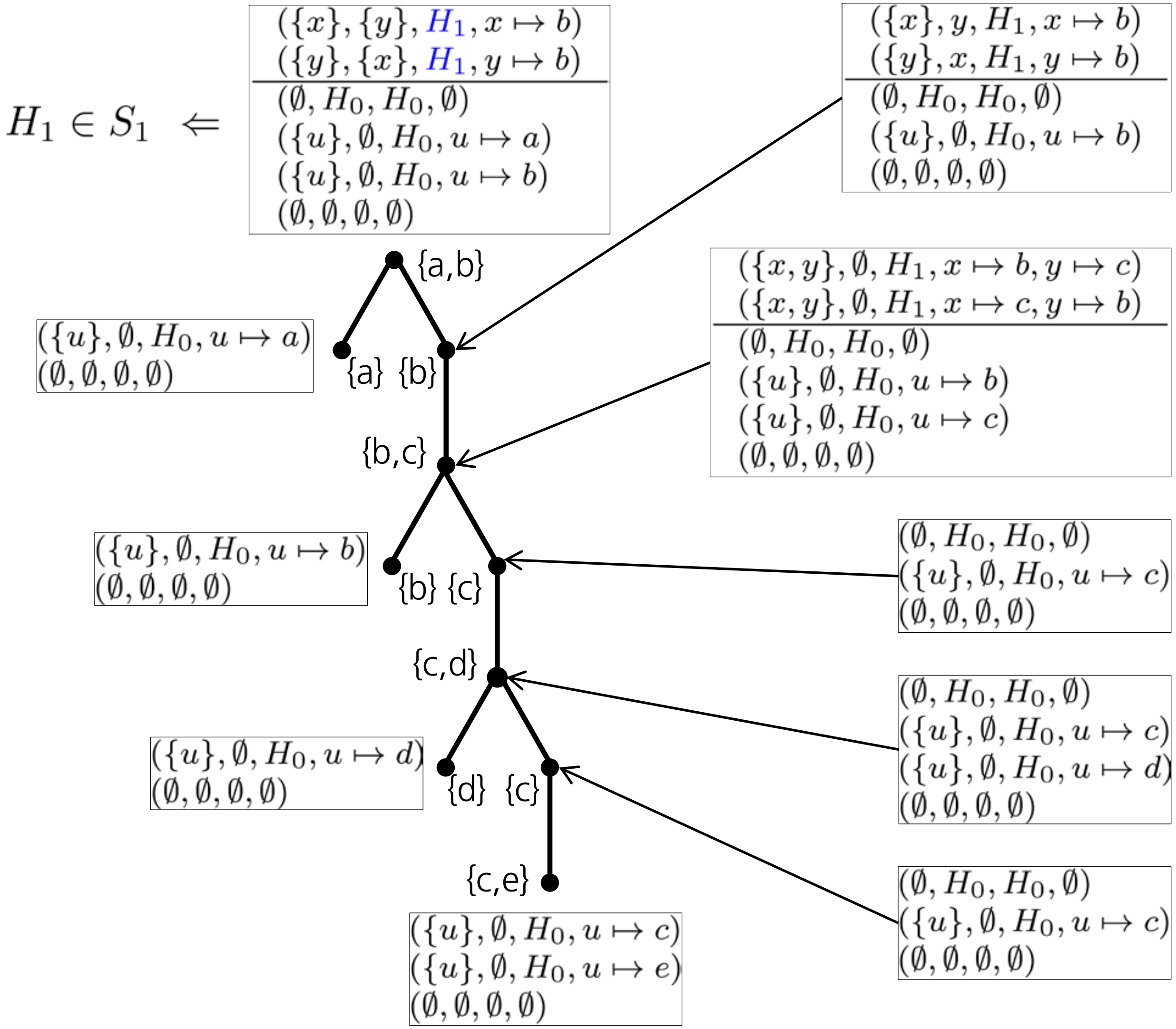
pattern H_1: •—• x y

$(S, \mathcal{D}, K)$-triples for possible feasible iso-quadruples:

- $\cancel{(\emptyset, \emptyset, \emptyset)}$, $(\emptyset, H_1, H_1)$
- $\cancel{(\{x\}, \emptyset, x)}$, $(\{x\}, y, H_1)$
- $\cancel{(\{y\}, \emptyset, y)}$, $(\{y\}, x, H_1)$
- $(\{x, y\}, \emptyset, H_1)$

transaction graph G:

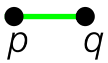




Example (cont'd)

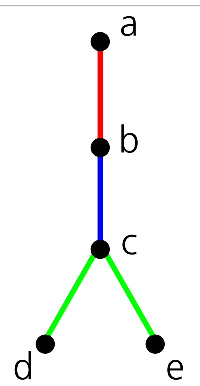
Step 12 of the alg.
on slide 21

$\rho(\bullet)$ = {  }

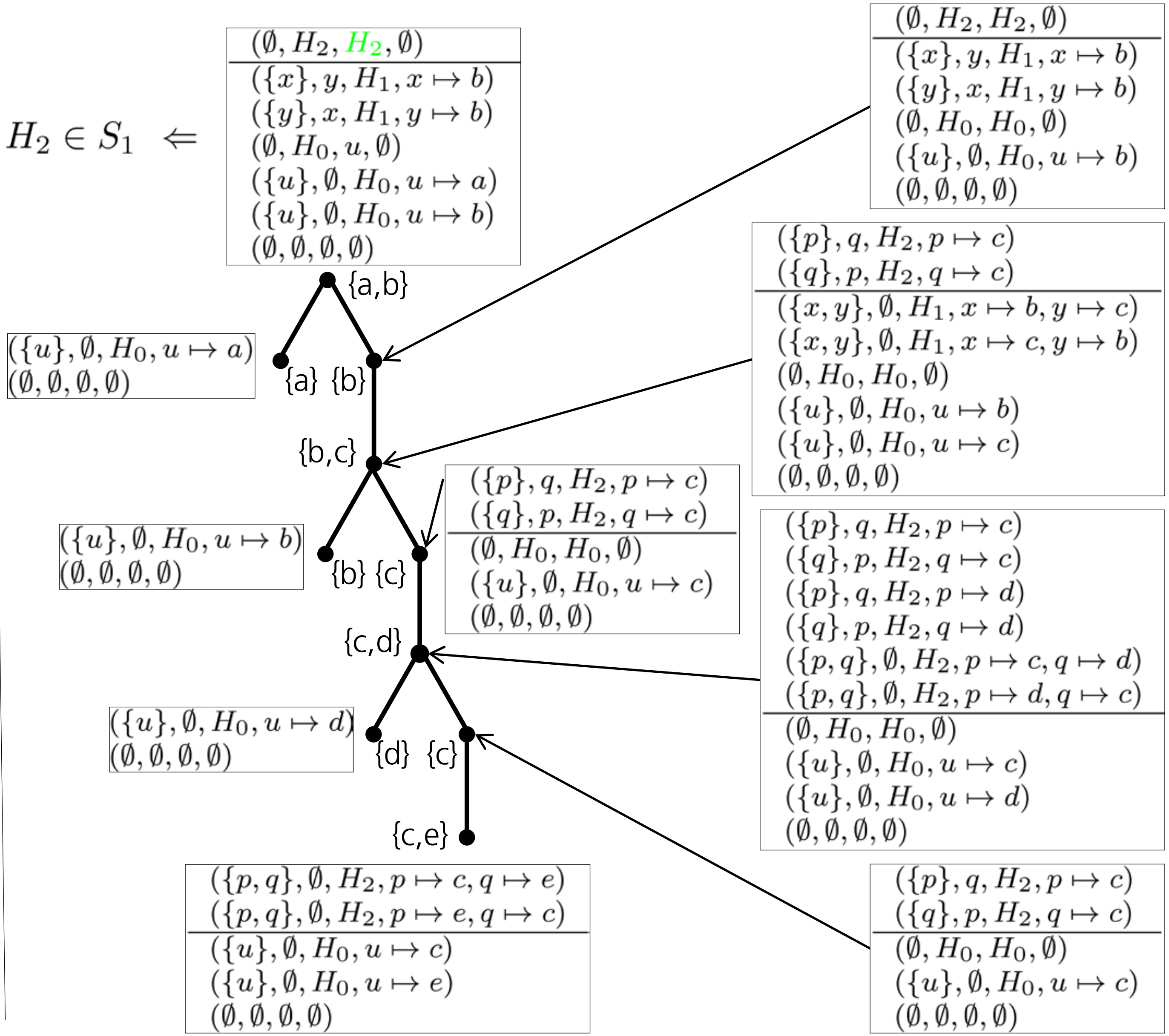
pattern H_2: 

$(S, \mathcal{D}, K)$-triples for possible feasible iso-quadruples:

- $\cancel{(\emptyset, \emptyset, \emptyset)}$, $(\emptyset, H_2, H_2)$
- $\cancel{(\{p\}, \emptyset, p)}$, $(\{p\}, q, H_2)$
- $\cancel{(\{q\}, \emptyset, q)}$, $(\{q\}, p, H_2)$
- $(\{p, q\}, \emptyset, H_2)$

transaction graph G: 

$H_2 \in S_1 \quad \Leftarrow$



Example (cont'd)

and so on…

notice that the algorithm could further be improved because there are redundant characteristics

- because feasible iso-quadruples are processed
- the number of redundant characteristics is polynomial in the combined size of the input and the set of previously generated frequent pattern
- using some advanced data structure, redundant characteristics can be removed in time polynomial in the input

Summary

- efficient pattern mining is possible even for computationally **hard** matching operators
- the technique might be of some independent interest and useful to design efficient algorithms if straightforward dynamic programming requires **exponential space**
- the positive theoretical result of this lecture is **not always practical**
  - e.g., for $k > 4$ (or 5?), no *practical* algorithm is known for deciding whether a graph has tree-width at most k
  - for $k < 4$: fast algorithm [Arnborg, Corneil, Proskurowski, 1987]
    - chemical graphs of pharmacological compounds have mostly tree-width at most 3

open problem: Is it possible to mine frequent connected subgraphs in graphs of bounded tree-width with **polynomial delay**?