Mining for Tree-Query Associations in a Graph

Bart Goethals[1], Eveline Hoekx[2], Jan Van den Bussche[2]
[1]University of Antwerp, Belgium
[2]University of Hasselt, Belgium

Graph Data

A (directed) **graph** over a set of nodes N is a set G of edges: ordered pairs (i,j) with i,j in N.

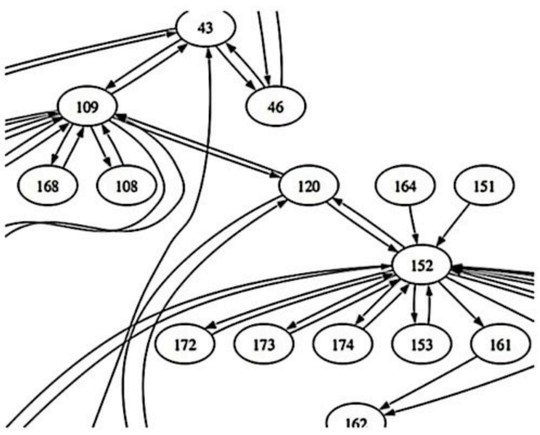


Figure 1: Snapshot of a graph representing the complete metabolic pathway of a human. An node is a compound or enzyme and an edge a reaction.

Graph Mining

⇨ Focus on pattern mining, few work on association rule mining!

	Transactional category	Single graph category
dataset	set of many small graphs (transactions)	single large graph
frequency	#graphs in which the pattern occurs	#copies of the pattern in the large graph
examples	Warmr, gSpan, AGM, FSG, FFSM	Subdue, SEus, SiGraM, Jeh-Widom

Our work

- *Single graph category*
- *Pattern + association rule mining*
- ***Tree Queries**: tree-shaped patterns inspired by conjunctive database queries*

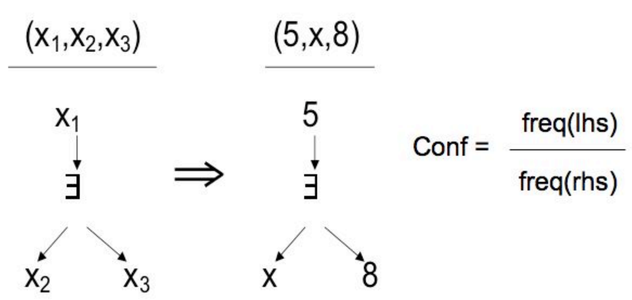
• Parameters and Existential nodes

select distinct G3.to as x
from G G1, G G2, G G3
where G1.from=5 and G1.to=G2.from and G1.to=G3.from and G2.to=8

frequency:
$\#\{x \mid \exists z: (5,z) \in G \wedge (z,8) \in G \wedge (z,x) \in G\}$

• An occurrence of the pattern in G is any homomorphism from the pattern in G.

• **Tree-Query Association:**



Features of our algorithm

1. Pattern mining phase and association rule mining phase
1. Restriction to trees ⇒ efficient algorithms
2. Equivalence checking
3. Apply theory of conjunctive database queries
4. Database oriented implementation

Problem Definition

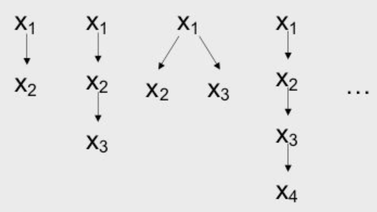
• Input: a graph G, threshold *minsup*, a tree query Q_{left} frequent in G and an threshold *minconf*
• Output: all tree queries Q, such that $Q_{left} \Rightarrow Q$ is frequent and confident in G.

Algorithm

1. Pattern mining

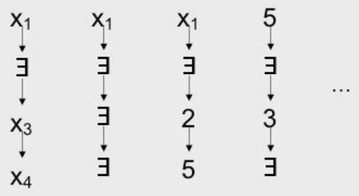
•*Outer loop:*

Generate, incrementally, all possible trees of increasing sizes. Avoid generation of isomorphic trees.



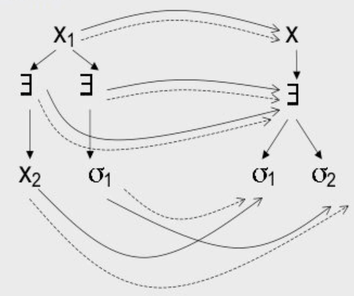
•*Inner loop:*

For each newly generated tree, generate all queries based on that tree, and test their frequency.

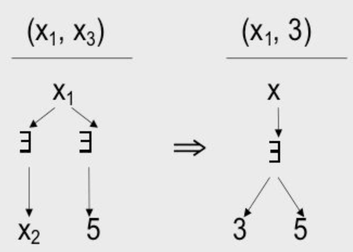


2. Association Rule mining

• For each tree query, generate all containment mappings from Q_{left} to Q, ignoring parameter assignments.



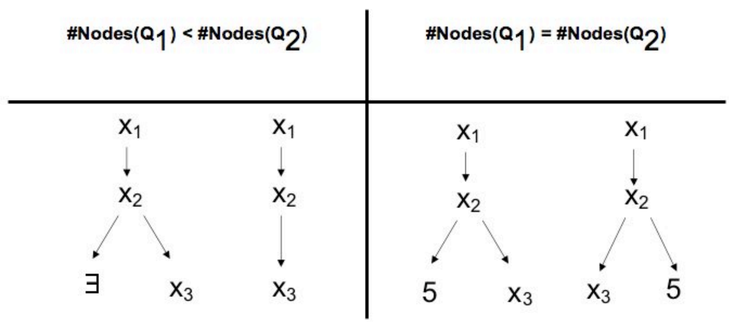
• For each containment mapping, generate all parameter assignments such that $Q_{left} \Rightarrow Q$ is frequent and confident.



Equivalence checking

1. Equivalent tree queries

Queries Q_1 and Q_2 are **equivalent** if same result sets on all graphs G (up to renaming of the distinguished variables).



2. Equivalent Association rules

• Equivalence checking of association rules is as hard as general graph isomorphism testing.
• A special case can be done in polynomial time

Experimental results

• *Performance*

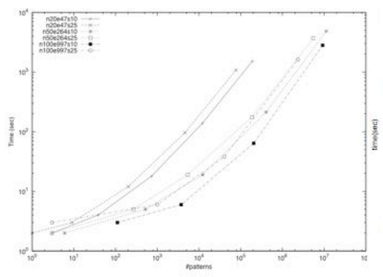


Figure 2: Pattern mining: Performance in terms of number of discovered patterns

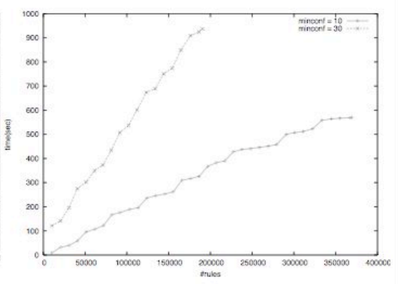
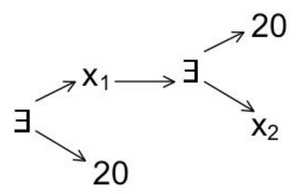
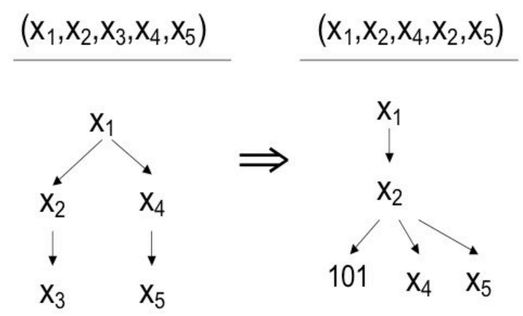


Figure 3: Associations mining: Performance in terms of number of discovered rules

• *Real-life datasets*
(e.g. food web, #nodes=154, #edges=370)





References

1. Bart Goethals, Eveline Hoekx and Jan Van den Bussche, Mining Tree Queries in a graph, in Proceedings of the eleventh ACM SIGKDD International conference on Knowledge Discovery and Data Mining, p61-69, ACM Press 2005.
2. Eveline Hoekx and Jan Van den Bussche, Mining for Tree-Query Associations in a Graph, to appear in Proceedings of the 2006 IEEE International Conference on Data Mining (ICDM 2006).