Mining for Tree-Query Associations in a Graph

Bart Goethals¹, Eveline Hoekx², Jan Van den Bussche²

¹University of Antwerp, Belgium
²University 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!

<table>
<thead>
<tr>
<th>Transactional category</th>
<th>Single graph category</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>set of many small graphs (transactions)</td>
</tr>
<tr>
<td>frequency</td>
<td>graphs in which the pattern occurs</td>
</tr>
<tr>
<td>examples</td>
<td>Warner, g2ban, NOW, F93, FROM</td>
</tr>
</tbody>
</table>

Our work

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

Parameters and Existential nodes

\[
\begin{align*}
\text{select} & \quad \text{distinct G3 to as x from G G1, G2, G3} \\
\text{where} & \quad \text{G1 from x6 and G1 to G2 from x7 and G1 to G3 from x8 and G2 to G3 from x9 and G3 to x10} \\
\end{align*}
\]

frequency:

\[
\# \{ x | \exists z : (5,2) \in G \land (x,8) \in G \land (z,x) \in G \}
\]

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

Tree-Query Association:

\[ (x_1, x_2, x_3) \quad (5, x, 8) \]

\[ x_1 \quad x_2 \quad x_3 \quad x \quad 5 \]

\[ x_1 \quad x_2 \quad x_3 \quad 3 \quad 3 \]

\[ x_1 \quad x_2 \quad x_3 \quad 3 \quad 5 \]

Features of our algorithm

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

Problem Definition

- Input: a graph G, threshold minsup, a tree query Q thr frequent in G and an threshold minconf
- Output: all tree queries Q such that Q \( \Rightarrow Q \) 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.
       
       \[
       \begin{array}{c}
       x_1 \\
       x_2 \\
       x_3 \\
       \vdots \\
       x_n \\
       \end{array}
       \]
     - Inner loop:
       - For each newly generated tree, generate all queries based on that tree, and test their frequency.
         
         \[
         \begin{array}{c}
         x_1 \\
         x_2 \\
         x_3 \\
         \vdots \\
         x_n \\
         \end{array}
         \]

2. Association Rule mining
   - For each tree query, generate all containment mappings from Q thr to Q, ignoring parameter assignments.

   \[
   (x_1, x_2, x_3) \quad (5, x, 8)
   \]

   - For each containment mapping, generate all parameter assignments such that Q thr \( \Rightarrow Q \) is frequent and confident.

Equivalence checking

1. Equivalent tree queries
   - Queries Q₁ and Q₂ 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

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

Reference data

\[ (x_1, x_2, x_3, x_4, x_5) \]

\[ (x_1, x_2, x_3, x_4, x_5) \]

confidence = 11%

References