A compression-based framework for the efficient analysis of business process logs

Bettina Fazzinga, Sergio Flesca, Filippo Furfaro, Elio Masciari, Luigi Pontieri
Context

BUSINESS PROCESS LOGS ANALYSIS
Data, Data, Data

- Large volumes, Large diversification of traces:
  - Users
  - Connections
  - Actions
  - Contents
"Let's say you want to save millions of dollars — you just push this button here..."
How to turn data into value (Reality)

• Process Mining
• Conformance Checking
• Querying
• Exploratory Analysis

However sometimes...
Bite off more than you can chew?

• **Problem**: Typical Log Data Set size scale to Terabytes
  – Queries can be slow
  – Impractical for exploratory analysis

• **Solution**: Log data compression
  – Queries become scalable
  – Quick identification of interesting data portion
  – E.g. *how long the execution of a group of activities lasted on average?*
Our technique in a short

• Tuple merging by storing aggregate information about:
  – Executors
  – Starting Time of activities (AVG)
  – Duration (AVG)

• Guided by heuristic
  – Limiting approximation errors
  – Process structure preserving
Our technique in a short

• Allowed queries against the synopsis
  – Meaningful Execution Patterns
  – Selection Conditions On:
    • Precedence Relationships between activities
    • Time Constraints
    • Executor Constraints
    • Duration Constraints
  – Aggregate Statistics
    • MIN/MAX/AVG/COUNT
Compression Strategy: STEP 1

• Process Instances stored in LOG relation
  – Process Identifier
  – Activity being executed
  – Executors
  – Starting Time
  – Duration

• Tuples describe a process instance step
Compression Strategy: STEP 2

• Graph Based Representation of LOG (Precedence Graph - PG)
  – Nodes are LOG tuples
  – Directed Edges model precedence relationships between activities within the same process instance
  – Precedence can be strict or loose
LOG vs PG

LOG

<table>
<thead>
<tr>
<th>p</th>
<th>A</th>
<th>e</th>
<th>τ_s</th>
<th>τ_d</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>S</td>
<td>e1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>p1</td>
<td>A1</td>
<td>e2</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>p1</td>
<td>A2</td>
<td>e3</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>p2</td>
<td>S</td>
<td>e1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>p2</td>
<td>A1</td>
<td>e4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>p2</td>
<td>A2</td>
<td>e3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>p3</td>
<td>S</td>
<td>e4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>p3</td>
<td>A2</td>
<td>e2</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

PG
Compression Strategy: STEP 3

• Compressed Precedence Graph - CPG
  – Nodes represent a set of executions of the same activity within different process instances
  – Result of merging the nodes of a precedence graph over the same activity
    • Number of executions of the activity, grouped by executor
    • average starting times and durations
PG vs CPG
What we lose?

- Errors due to time, duration and executor approximations

\[ err(u) = \sqrt{\left( \frac{err_s(u)}{Err_s} \right)^2 + \left( \frac{err_d(u)}{Err_d} \right)^2 + (err_e(u))^2} \]
STEP 4: The compression Algorithm

Algorithm 1

INPUT:
\( f, G \): a compression factor and a PG (representing the uncompressed data);

OUTPUT:
\( C \): a compressed precedence graph over \( G \);

begin
\( \text{initialize}(G) \);
\( \langle \text{err}_s, \text{err}_d, \text{err}_e \rangle = \text{initializeToZero}() \);
\( \langle \text{Err}_s, \text{Err}_d \rangle = \text{normalizationFactors}(C) \);
while size(C) > \( f \cdot \text{size}(G) \) do
\( np = \text{greedySelect}(C, \text{err}_s, \text{err}_d, \text{err}_e, \text{Err}_s, \text{Err}_d) \);
\( \text{aggregateAndReplace}(C, \text{err}_s, \text{err}_d, \text{err}_e, np) \);
return C;
end.
Querying the compressed data

- COUNT
- MIN
- MAX
- AVG
- Conditions about time, duration and executors
- homomorphism between PG and CPG
Estimating Queries

• Count Queries

\[ \tilde{Q} = \sum_{u \in \text{Start}(C)} \left( |T(u)| \times \left( 1 - \prod_{h \in \mathcal{H}(\pi,C) \land h(r_{\pi}) \in \Lambda(u)} (1 - \tilde{P}(h)) \right) \right) \]

• Time Queries

\[ \tilde{Q} = \frac{\sum_{h \in \mathcal{H}(\pi,C)} \left( |T(h(r_{\pi}))| \times \tilde{P}(h) \times \psi_{h(\pi)} \right)}{\sum_{h \in \mathcal{H}(\pi,C)} \left( |T(h(r_{\pi}))| \times \tilde{P}(h) \right)} \]
## Experimental Evaluation

<table>
<thead>
<tr>
<th></th>
<th>$PC$</th>
<th></th>
<th></th>
<th></th>
<th>$SA$</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compression ratio $f$</td>
<td>5%</td>
<td>3%</td>
<td>1%</td>
<td>0.5%</td>
<td>5%</td>
<td>3%</td>
<td>1%</td>
</tr>
<tr>
<td>LOG</td>
<td>$&gt; 1 \text{h}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$&gt; 1 \text{h}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$LOG + PREC$</td>
<td>10.3\text{min} [7.2\text{min .. 16.3\text{min}}]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>41.2\text{min} [19.1\text{min .. 62.7\text{min}}]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synopsis</td>
<td>$&lt; 1 \text{sec}$</td>
<td>$&lt; 1 \text{sec}$</td>
<td>$&lt; 1 \text{sec}$</td>
<td>$&lt; 1 \text{sec}$</td>
<td>$&lt; 1 \text{sec}$</td>
<td>$&lt; 1 \text{sec}$</td>
<td>$&lt; 1 \text{sec}$</td>
<td>$&lt; 1 \text{sec}$</td>
</tr>
<tr>
<td>Average Error</td>
<td>0.1%</td>
<td>1.6%</td>
<td>4.8%</td>
<td>8.0%</td>
<td>0.1%</td>
<td>2.5%</td>
<td>6.3%</td>
<td>9.9%</td>
</tr>
<tr>
<td>Compr. time</td>
<td>10.5\text{min}</td>
<td>10.7\text{min}</td>
<td>10.9\text{min}</td>
<td>11.1\text{min}</td>
<td>23.9\text{min}</td>
<td>24.4\text{min}</td>
<td>24.9\text{min}</td>
<td>25.0\text{min}</td>
</tr>
<tr>
<td>Amortization</td>
<td>1.02</td>
<td>1.04</td>
<td>1.06</td>
<td>1.08</td>
<td>0.58</td>
<td>0.59</td>
<td>0.60</td>
<td>0.61</td>
</tr>
</tbody>
</table>
Conclusion & Future Works

- Speed-up for exploratory analysis
- Good compression factors
- Tolerable Errors
- Error Guarantee
- Hierarchies over data
- Data Updates
- Distributed approaches
- Scientific Data
THANK YOU

QUESTIONS?