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

How to turn data into value (Dream)

"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

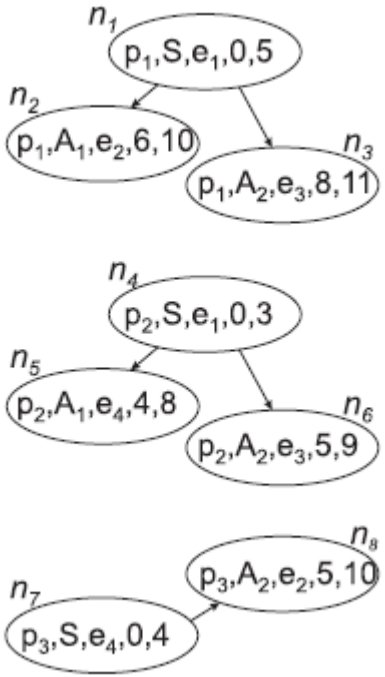
Compression Strategy: STEP 2

- 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

p	A	e	τ_s	τ_d
p_1	S	e_1	0	5
p_1	A_1	e_2	6	4
p_1	A_2	e_3	8	3
p_2	S	e_1	0	3
p_2	A_1	e_4	4	4
p_2	A_2	e_3	5	4
p_3	S	e_4	0	4
p_3	A_2	e_2	5	10

LOG

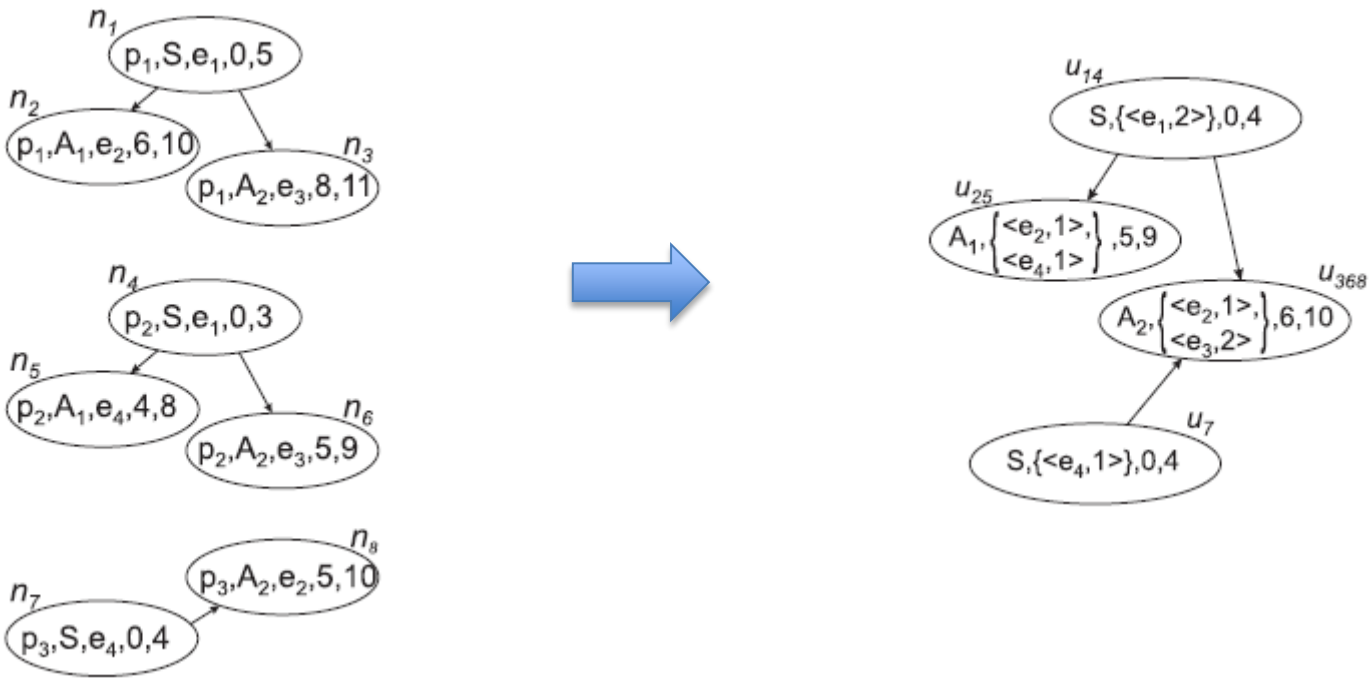


PG

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


n_1
$p_1,S,e_1,0,5$
n_2
$p_1,A_1,e_2,6,10$
n_3
$p_1,A_2,e_3,8,11$
n_4
$p_2,S,e_1,0,3$
n_5
$p_2,A_1,e_4,4,8$
n_6
$p_2,A_2,e_3,5,9$
n_8
$p_3,A_2,e_2,5,10$
n_7
$p_3,S,e_4,0,4$
u_{14}
$S,\{<e_1,2>\},0,4$
u_{25}
$A_1,\{<e_2,1>,<e_4,1>\},5,9$
u_{368}
$A_2,\{<e_2,1>,<e_3,2>\},6,10$
u_7
$S,\{<e_4,1>\},0,4$

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:

$f, \mathcal{G}$: a compression factor and a PG (representing the uncompressed data);

OUTPUT:

$\mathcal{C}$: a compressed precedence graph over $\mathcal{G}$;

begin

$\mathcal{C}$ =*initialize*($\mathcal{G}$);

$\langle \vec{err}_s, \vec{err}_d, \vec{err}_e \rangle$ = *initializeToZero*();

$\langle Err_s, Err_d \rangle$= *normalizationFactors*($\mathcal{C}$);

while size($\mathcal{C}$) $>$ $f \cdot$ size($\mathcal{G}$) **do**

np = *greedySelect*($\mathcal{C}, \vec{err}_s, \vec{err}_d, \vec{err}_e, Err_s, Err_d$);

aggregateAndReplace($\mathcal{C}, \vec{err}_s, \vec{err}_d, \vec{err}_e, np$);

return $\mathcal{C}$;

end.

Querying the compressed data

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

Estimating Queries

- Count Queries

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

- Time Queries

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

Experimental Evaluation

	PC				*SA*			
	Compression ratio f				Compression ratio f			
	5%	3%	1%	0.5%	5%	3%	1%	0.5%
LOG	> 1h				> 1h			
LOG +PREC	10.3min [7.2min .. 16.3min]				41.2min [19.1min .. 62.7min]			
Synopsis	< 1sec	< 1sec	< 1sec	< 1sec	< 1sec	< 1sec	< 1sec	< 1sec
Average Error	0.1%	1.6%	4.8%	8.0%	0.1%	2.5%	6.3%	9.9%
Compr. time	10.5min	10.7min	10.9min	11.1min	23.9min	24.4min	24.9min	25.0min
Amortization	1.02	1.04	1.06	1.08	0.58	0.59	0.60	0.61

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?