Process Mining (ProMi)

Winter 2015/16

Matthias Weidlich
II. Process Discovery

Mining Algorithms

Heuristic and Fuzzy Miners
Issues with the $\alpha$-Algorithms

Properties of the $\alpha(+//++)$ – algorithms

- Each (!) direct successorship is processed
- All tasks and successorship relations are considered to be equally important

Consequences:

- $\alpha$ – algorithms are not robust against noise
- $\alpha$ – algorithms provide no means for abstraction
Impact of Noise

Case 1: ABCD
Case 2: ACBD
Case 3: EF
Case L: ABD
Case M: CBD
Case N: CADB

\[ a(W): \]

\[ \alpha \text{-Algorithm} \]

Given \( i_W \), \( (i_W, E) \), and \( p_{(\{E\}, \{F\})} \).
Lack of Abstraction

“Spaghetti-Models”

• Not necessarily a matter of noise!
• The process may be “spaghetti-like”, i.e., the model may be an accurate reflection of reality
• Processes may offer various degrees of freedom in the execution
• There may be a large number of variants, exceptional cases, etc.
Heuristic and Fuzzy Miners

More practical approaches to process discovery

• *Heuristic Miner*: exploits occurrence frequencies to estimate flow probabilities
• *Fuzzy Miner*: introduces measures for significance and correlation to create abstract views of the process model

Common basis:

• Ordering relations of the $\alpha$ – algorithm are used as the foundation
• Relations provide a model to reason about frequencies

Details:

A directed graph where

- Nodes represent activities
- Edges represent the causality relation
- Additional nodes for start (O) and end (X)

Example workflow log: \( W = \{ABCD, ACBD, EF\} \)

Case 1: ABCD
Case 2: ACBD
Case 3: ABCD
Case 4: ACBD
Case 5: EF
Heuristics Miner

Basic idea: Consider frequencies of ordering relations to achieve robustness against noise

Approach:
1) Construction of frequency table
2) Induction of dependency graph
3) Construction of net system from frequency table and dependency graph

Relies on adapted notion of workflow log
Adapted Workflow Log

• We had: $T$ is a set of tasks
• Now: $M(T^*)$ is the set of all multisets (bags) over all finite sequences over $T$
• Then $W \subseteq M(T^*)$ is a workflow log as a multiset

Case 1: ABCD
Case 2: ACBD
Case 3: ABCD
Case 4: ACBD
Case 5: EF

$W = [ABCD, ACBD, ABCD, ACBD, EF] = [ABCD^2, ACBD^2, EF]$
Consider the following frequencies for each task $a \in T$:

- $\#a$ – overall frequency of task $a$
- $\#a > x$ – the frequency of task $a$ being directly followed by another task $x$ in the cases
- $a \rightarrow^L x$ – a local metric indicating the strength of the causality relation between task $a$ and another task $x$

$$a \rightarrow^L x = \frac{\#a > x - \#x > a}{\#a > x + \#x > a + 1}$$
Global Strength of Causality

An additional frequency

\[ a \rightarrow^G x \] – a more global metric indicating the strength of causality

Based on \textit{minimal distance} between occurrences of tasks

\begin{itemize}
  \item Let \( a, x \in T \) be two tasks and \( s \in T^* \) a finite sequence of tasks (the stream of all events in the log)
  \item Derive a sum \( \Delta a \) by doing for each \( a \) in \( s \):
    \begin{itemize}
      \item Determine the distance \( n \) to the next \( x \) in \( s \), i.e., \( x \) follows \( a \) in \( s \) with \( n \) tasks \( t \in T \setminus \{a, x\} \) occurring in between
      \item Add \( \delta^n \) to \( \Delta a \) with \( \delta \in [0,1] \) as a fall factor
    \end{itemize}
  \item Derive a sum \( \Delta x \) by doing for each \( x \) in \( s \):
    \begin{itemize}
      \item Determine the distance \( n \) to the next \( a \) in \( s \), i.e., \( a \) follows \( x \) in \( s \) with \( n \) tasks \( t \in T \setminus \{a, x\} \) occurring in between
      \item Add \( \delta^n \) to \( \Delta x \) with \( \delta \in [0,1] \) as a fall factor
    \end{itemize}
  \item Define \( a \rightarrow^G x = \frac{\Delta a - \Delta x}{\min(\#a, \#x)} \)
\end{itemize}
Example

Consider more traces for the original log and some noise:
\[ W = [ABCD^{10}, ACBD^8, EF^{20}, ABCED, AD] \]

Some frequencies for task C:

- \#C = 19
- \#A > C = 8, \#D > C = 0, \#B > C = 11, \#E > C = 0
- \#C > D = 10, \#C > B = 8, \#C > E = 1
- \( C \rightarrow^L D \approx 0.9, \ C \rightarrow^L B \approx -0.15 \)

Now, considering the stream AABACDAEACBCBDF and \( \delta = 0.8 \)

- \( C \rightarrow^G D \approx 0.64, \ C \rightarrow^G B \approx 0.07 \)
Dependency Score

One approach to filter based on frequencies:
- Consider all task pairs $a, x$
- Check for all frequency counts $\#x > a$, $\#a > x$, $a \rightarrow^L x$, and $a \rightarrow^G x$ whether they exceed predefined thresholds

But: Thresholds are hard to get right

Idea: The resulting model shall be connected
- Define dependency score:
  \[ DS(a, x) = \frac{((a \rightarrow^L x)^2 + (a \rightarrow^G x)^2)}{2} \]
- Filter edges, such that each task is connected to the task for which the DS is maximal
Induction of Dependency Graph

Based on frequency counts and dependency score, derive causality relation:

- $a \rightarrow x$, iff $a \rightarrow^L x > 0$, $a \rightarrow^G x > 0$
  and $x$ is the task for which $DS(a, x)$ is maximal
- $y \rightarrow a$, iff $y \rightarrow^L a > 0$, $y \rightarrow^G a > 0$
  and $y$ is the task for which $DS(y, a)$ is maximal

- $C \rightarrow^L B \approx -0.15$
- $C \rightarrow^G B \approx 0.07$
- $C \rightarrow^L D \approx 0.9$
- $C \rightarrow^G D \approx 0.64$
- $DS(C, B) \approx \frac{0.0225 + 0.0049}{2} \approx 0.014$
- $DS(C, D) \approx \frac{0.81 + 0.4096}{2} \approx 0.61$
Limitations

Filtering of edges based on maximal dependency score prevents discovery of certain control flow structures

- Complex split/join structures
- Short loops
Adaptations

To include complex split/join structures

• Adapt the approach to be more relaxed and introduce tolerance (e.g., 95%) based on maximal dependency score:
  • Given task $a$, let $M$ be the maximal $DS(a, x)$ for all $x$. Then, $a \rightarrow y$, iff $a \rightarrow^L y > 0$, $a \rightarrow^G y > 0$
    and $DS(a, y) > 0.95 \times M$
  • Given task $a$, let $M$ be the maximal $DS(x, a)$ for all $x$. Then, $y \rightarrow a$, iff $y \rightarrow^L a > 0$, $y \rightarrow^G a > 0$
    and $DS(y, a) > 0.95 \times M$

Yet, does not cover loops of lengths 1 or 2
Adaptations Cont.

Heuristic for length 1 loops of task $x$
- Detect high value for $\#x < x$, but $DS(x, x)$ of zero
  - $\#x < x$ is high because of patterns like $...xxxx...$
  - Consider: $x \rightarrow^L x = \frac{\#x>x - \#x>x}{\#x>x+\#x>x+1}$
  - Consider: $x \rightarrow^G x = \frac{\Delta x - \Delta x}{\min(\#x, \#x)}$
- Handle these short loops separately (similar to $\alpha^+ -$ algorithm)

Similar heuristic for length 2 loops of tasks $x$ and $y$
- Detect high value for $\#x > y$ and $\#y > x$, but $DS(x, y)$ and $DS(y, x)$ close to zero
- Additional check (similar to $\alpha^{++} -$ algorithm) to distinguish loop from concurrent execution
Construction of a Net System

General net structure is directly given by the dependency graph.

Transformations needed for splits and joins, i.e., tasks with multiple incoming or outgoing edges:

- Rely on the relations between succeeding (split) or preceding tasks (join) to determine type of split/join.
- Rely on the frequencies for the split (join) and its succeeding (preceding) tasks.

But: Splits and joins are only considered locally (thus most of the discovered model are not sound and require repair actions).
Example

\[ W = [ABCD^2, ACBD^2, EF] \]

Consider A:

- \#B > C = \#B > C = 2
- \#B = \#A = \#C = 4
Summary Heuristic Miner

Copes with noise by means of occurrence frequencies

*Nomen est omen*: Purely heuristic approach

Dependency graph as a means to get a general overview of the processing

Construction of a more elaborated process model (splits/joins, sound model) is still challenging
Fuzzy Miner - Background

Creation of views on process model that are driven by significance and correlation
  • Abstract from undesired details
  • Provide high-level view

Means to cope with processing complexity
  • Aggregation: cluster similar elements
  • Abstraction (aka Projection): remove low-level information
  • Emphasis: highlight significant information
  • Customisation: adapt to context of model use
The Map Analogy

Abstraction
insignificant roads are not shown.

Aggregation
parts of the city are merged.

Customization
Focuses on the intended use and level of detail.

Emphasis
Highways are highlighted by size, contrast and color.
Significance and Correlation

Metrics to control the creation of views

- Significance of individual tasks or binary ordering relations (direct successor, causality)
- Correlation of pairs of tasks

Idea for simplification of process model:

- Highly significant behaviour is *preserved*
- Less significant, but highly correlated behaviour is *aggregated* into clusters
- Less significant and less correlated behaviour is *abstracted*
Significance Metrics

Frequency significance: Frequency of task occurrence
- But: Can be misleading
- “Housekeeping” tasks like archiving, storing documents in regular intervals

Routing significance: Count of distinct predecessors and successors along with their significance
- Splits and joins typically important to understand process logic

Relation significance: Frequency of direct successor relation for pairs of tasks
- In addition, consider difference between relation frequency and frequency significance of source and target nodes
Examples

Task Frequencies

25

12

292

26

14

281

298

298

298

298

266
Examples

Count of predecessors and successors

O → B → C → A → D

O → A → C → B → D

O → D → B → C → A

E → F → G → H → I

I → J → K → X

M

Count of predecessors and successors:

- A: 2 predecessors, 3 successors
- B: 3 predecessors, 1 successor
- C: 2 predecessors, 2 successors
- D: 2 predecessors, 3 successors
- E: 4 predecessors
- F: 3 predecessors, 2 successors
- G: 2 predecessors, 3 successors
- H: 3 successors
- I: 3 predecessors
- J: 3 successors
- K: 3 predecessors
- X: 3 successors
Correlation Metrics

Answer the question: abstract or aggregate?

Proximity correlation: avg distance between execution of tasks in cases

Many more based on attribute values:
  • Originator correlation: tasks conducted by the same roles
  • Data type correlation: tasks have been executed for partially overlapping data (e.g., the same data objects)
Fuzzy Miner Techniques

Start with basic dependency graph induced by causality relation of \( \alpha \) – algorithm

Based on significance and correlation metrics, transform graph by:

- Edge filtering
- Aggregation and abstraction

Transformed graph can be further processed as discussed earlier (e.g., used to construct a WF-system)
Edge Filtering

Baseline solution: Remove insignificant edges
  • But: Tendency to create unconnected clusters of frequent behaviour

Thus: Compute edge utility as weighted sum of its significance and the source-target correlation
  • For each task, maintain incoming and outgoing edge with highest utility value
  • Apart from that, filter based on threshold
Example

Edge Frequencies

Remove Edge
Edge Filtering in Practice
Aggregation and Abstraction

Idea: Tasks with significance below thresholds are *victims* and are aggregated or abstracted.

First phase:
- For each victim, find most highly correlated neighbour.
  - If neighbour is cluster, add victim.
    - Cluster “inherits” incoming and outgoing edges of victim.
  - Otherwise, create singleton cluster with victim.

Second phase:
- For each cluster, check whether all predecessor and successors are also clusters.
  - If so, merge with most highly correlated predecessor or successor cluster, respectively.

Third phase: remove isolated clusters and singleton clusters (edges are preserved transitively).
Example

Task Frequencies

High Correlation
Example

Task Frequencies

High Correlation
Example
Aggregation and Abstraction in Practice
Take Away

Process mining needs to be robust against noise and offer abstraction capabilities.

Heuristics to consider occurrence frequencies to cope with noise.

Graph transformations based on significance and correlation metrics to create abstract views of process model.