Control-flow Discovery from Event Streams

Andrea Burattin$^1$, Alessandro Sperduti$^1$, Wil M. P. van der Aalst$^2$

$^1$ University of Padua, Italy
$^2$ Eindhoven University of Technology, The Netherlands

July 11, 2014
Typical Process Mining Scenario

## Typical Event Log

<table>
<thead>
<tr>
<th>Event #</th>
<th>Activity</th>
<th>Orig.</th>
<th>Time</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>U₁</td>
<td>2014-01-01</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>U₁</td>
<td>2014-01-02</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>U₂</td>
<td>2014-01-03</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>U₂</td>
<td>2014-01-04</td>
<td></td>
</tr>
</tbody>
</table>

**Case Id: C₁**

<table>
<thead>
<tr>
<th>Event #</th>
<th>Activity</th>
<th>Orig.</th>
<th>Time</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>U₁</td>
<td>2014-01-01</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>U₁</td>
<td>2014-01-02</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>U₂</td>
<td>2014-01-03</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>U₂</td>
<td>2014-01-04</td>
<td></td>
</tr>
</tbody>
</table>

**Case Id: C₂**

<table>
<thead>
<tr>
<th>Event #</th>
<th>Activity</th>
<th>Orig.</th>
<th>Time</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>U₁</td>
<td>2014-01-02</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>U₁</td>
<td>2014-01-03</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>U₃</td>
<td>2014-01-04</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>U₃</td>
<td>2014-01-05</td>
<td></td>
</tr>
</tbody>
</table>
Typical Event Log

<table>
<thead>
<tr>
<th>Event #</th>
<th>Activity</th>
<th>Orig.</th>
<th>Time</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>U₁</td>
<td>2014-01-01</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>U₁</td>
<td>2014-01-02</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>U₂</td>
<td>2014-01-03</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>U₂</td>
<td>2014-01-04</td>
<td>...</td>
</tr>
</tbody>
</table>

**Case Id: C₁**

<table>
<thead>
<tr>
<th>Event #</th>
<th>Case Id</th>
<th>Activity</th>
<th>Orig.</th>
<th>Time</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C₁</td>
<td>A</td>
<td>U₁</td>
<td>2014-01-01</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>C₂</td>
<td>A</td>
<td>U₁</td>
<td>2014-01-02</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>C₁</td>
<td>B</td>
<td>U₁</td>
<td>2014-01-02</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>C₂</td>
<td>B</td>
<td>U₁</td>
<td>2014-01-03</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>C₁</td>
<td>C</td>
<td>U₂</td>
<td>2014-01-03</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>C₁</td>
<td>E</td>
<td>U₂</td>
<td>2014-01-04</td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>C₂</td>
<td>D</td>
<td>U₃</td>
<td>2014-01-04</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>C₂</td>
<td>E</td>
<td>U₃</td>
<td>2014-01-05</td>
<td>...</td>
</tr>
</tbody>
</table>

**Case Id: C₂**

<table>
<thead>
<tr>
<th>Event #</th>
<th>Case Id</th>
<th>Activity</th>
<th>Orig.</th>
<th>Time</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C₁</td>
<td>A</td>
<td>U₁</td>
<td>2014-01-01</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>C₂</td>
<td>A</td>
<td>U₁</td>
<td>2014-01-02</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>C₁</td>
<td>B</td>
<td>U₁</td>
<td>2014-01-02</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>C₂</td>
<td>B</td>
<td>U₁</td>
<td>2014-01-03</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>C₁</td>
<td>C</td>
<td>U₂</td>
<td>2014-01-03</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>C₁</td>
<td>E</td>
<td>U₂</td>
<td>2014-01-04</td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>C₂</td>
<td>D</td>
<td>U₃</td>
<td>2014-01-04</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>C₂</td>
<td>E</td>
<td>U₃</td>
<td>2014-01-05</td>
<td>...</td>
</tr>
</tbody>
</table>
Representation of an event stream $\sigma$:

- **Boxes represent events**:  
  - Background colors represent the case id  
  - Letters inside are the activity names

<table>
<thead>
<tr>
<th>Time</th>
<th>E</th>
<th>F</th>
<th>E</th>
<th>G</th>
<th>F</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>H</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case: $C_X$</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Case: $C_Y$</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Case: $C_Z$</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The stream miner continuously receives events and, using the latest observations, updates the process model (e.g., a Petri Net)
Peculiarities of the stream mining problem
Peculiarities of the stream mining problem

1. Cannot store the entire stream (approximation)
Peculiarities of the stream mining problem

1. **Cannot store the entire stream (approximation)**

2. **Backtracking not feasible over streams** (algorithms required to make one pass over data → scale linearly w.r.t. the number of processed items)
Peculiarities of the stream mining problem

1. Cannot store the entire stream (approximation)
2. Backtracking not feasible over streams (algorithms required to make one pass over data \(\rightarrow\) scale linearly w.r.t. the number of processed items)
3. The approach must deal with variable system conditions, such as fluctuating stream rates
Peculiarities of the stream mining problem

1. Cannot store the entire stream (approximation)
2. Backtracking not feasible over streams (algorithms required to make one pass over data → scale linearly w.r.t. the number of processed items)
3. The approach must deal with variable system conditions, such as fluctuating stream rates
4. It is important to quickly adapt the model to cope with unusual data values (concept drifts)
Historical Background

Our approaches are based on Heuristics Miner, quite old (∼ 2003) but still one of the most used algorithm.
Our approaches are based on Heuristics Miner, quite old (∼ 2003) but still one of the most used algorithm.

Fundamental metric is “dependency measure” between two activities (e.g. $a$, $b$):

$$a \Rightarrow b = \frac{|a > b| - |b > a|}{|a > b| + |b > a| + 1} \in [-1, 1]$$

Where:

- $|a > b|$ is the number of times that $a > b$ holds in the log
- $a > b$ holds if $a$ executed at time $t$ and $b$ at $t + 1$
Given the dependency measure for all activity pairs and a threshold $\tau_{dep}$, the algorithm builds a directed dependency graph.
Given the dependency measure for all activity pairs and a threshold \( \tau_{\text{dep}} \), the algorithm builds a directed dependency graph.

If both \( a \Rightarrow b > \tau_{\text{dep}} \) and \( a \Rightarrow c > \tau_{\text{dep}} \) then:

- **Relation ambiguity between** \( b \) **and** \( c \):  
  - **XOR**: either \( b \) or \( c \) is executed  
  - **AND**: both \( b \) and \( c \) are executed
Given the dependency measure for all activity pairs and a threshold $\tau_{\text{dep}}$, the algorithm builds a directed dependency graph.

If both $a \Rightarrow b > \tau_{\text{dep}}$ and $a \Rightarrow c > \tau_{\text{dep}}$ then:

- **Relation ambiguity between $b$ and $c$:**
  - **XOR**: either $b$ or $c$ is executed
  - **AND**: both $b$ and $c$ are executed

Heuristics Miner proposes the “AND-measure”

$$a \Rightarrow (b \land c) = \frac{|b > c| + |c > b|}{|a > b| + |a > c| + 1} \in [0, 1]$$

If $a \Rightarrow (b \land c) > \tau_{\text{and}}$ then AND relation, XOR otherwise.
Basic data structure for HM is **Direct Following Matrix**

Given activities $A$, $B$, $C$, $D$ and a log $L$, $|a > b|$ is the number of times that $a$ is directly followed by $b$ (within the same process instance) in the log $L$.

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>0</td>
<td>52</td>
<td>64</td>
<td>91</td>
</tr>
<tr>
<td>$B$</td>
<td>52</td>
<td>0</td>
<td>24</td>
<td>87</td>
</tr>
<tr>
<td>$C$</td>
<td>64</td>
<td>24</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>$D$</td>
<td>91</td>
<td>87</td>
<td>13</td>
<td>0</td>
</tr>
</tbody>
</table>
Proposed Approaches

Our Proposal

We present three approaches, based on Heuristics Miner, for process discovery from event streams:

(SW) Heuristics Miner with Sliding Window *(as baseline)*
(LC) Heuristics Miner with Lossy Counting
(LCB) Heuristics Miner with Lossy Counting with Budget
## Proposed Approaches

### Our Proposal

We present three approaches, based on Heuristics Miner, for process discovery from event streams:

- **(SW)** Heuristics Miner with Sliding Window *(as baseline)*
- **(LC)** Heuristics Miner with Lossy Counting
- **(LCB)** Heuristics Miner with Lossy Counting with Budget

### Fundamental Principle

Recent observations are more important than older ones
Basic idea is to iterate these steps

1. Collect events for a given time span
2. Generate a finite event log
3. Apply the “offline version” of the algorithm
Frequency Counting with Lossy Counting

Given

- Max approximation error $\epsilon$
- Variables: $A$, $B$, $C$

Lossy Counting uses a data structure $\mathcal{D} = \{(\text{var, freq, max error})\}$

- Bucket size is $w = \left\lceil \frac{1}{\epsilon} \right\rceil$
- $b_{\text{current}} = \left\lfloor \frac{\text{no. of observed items}}{w} \right\rfloor$
Frequency Counting with Lossy Counting

Given

- Max approximation error $\epsilon$
- Variables: $A$, $B$, $C$
- Bucket size is $w = \left\lceil \frac{1}{\epsilon} \right\rceil$
- $b_{current} = \left\lceil \frac{\text{no. of observed items}}{w} \right\rceil$

Lossy Counting uses a data structure $D = \{(\text{var}, \text{freq}, \text{max error})\}$

Data sequence: $A\ A\ B\ A\ B\ C\ C\ B\ C\ A\ A\ B\ C$

Buckets: $b_{current}$, $b_{current} + 1$

If $B$ not present then
(B, $f = 1$, $\Delta = b_{current} - 1$)
else
Update frequency $f$ of $B$

Remove all elements s.t. $f + \Delta \leq b_{current}$
Lossy Counting Demo

\[ f \xrightarrow{\Delta} 0 0 0 0 0 \]
\[ f \xleftarrow{\Delta} 0 0 \]

Time

Remove if \( f + \Delta \leq 1 \)

End of bucket 1

Beginning of bucket 2
Lossy Counting Demo

Time

$b_{current} = 2$

Remove if $f + \Delta \leq 2$

Beginning of bucket 2

End of bucket 2

Beginning of bucket 3
Lossy Counting Demo

- Time
- $b_{current} = 3$
- Remove if $f + \Delta \leq 3$

- Beginning of bucket 3
- End of bucket 3
- Beginning of bucket 4
Comparison between Lossy Counting frequencies and true frequencies

These inequalities hold: \( f \leq F \leq f + \Delta \leq f + \epsilon N \)
Comparison between Lossy Counting frequencies and true frequencies

These inequalities hold: \( f \leq F \leq f + \Delta \leq f + \epsilon N \)

Lossy Counting with Budget Idea

New bucket when there is no more space, then \( \epsilon = \frac{1}{\text{bucket size}} \)
To count direct following relations we need

\[ D_{\text{rel}} \] Actual relations frequencies, tuples: \((a_s, a_t, f, \Delta)\)

\[ D_{\text{act}} \] Latest activity names, tuples: \((a, f, \Delta)\)

\[ D_{\text{cases}} \] Latest activity of a case, tuples: \((c, a, f, \Delta)\)
To count direct following relations we need

- $D_{\text{rel}}$ Actual relations frequencies, tuples: $(a_s, a_t, f, \Delta)$
- $D_{\text{act}}$ Latest activity names, tuples: $(a, f, \Delta)$
- $D_{\text{cases}}$ Latest activity of a case, tuples: $(c, a, f, \Delta)$

With a certain periodicity, model update

- Activities from $D_{\text{act}}$
- Dependencies and AND/XOR rules from $D_{\text{rel}}$
Adaptation of LC/LCB to HM

To count direct following relations we need

- $\mathcal{D}_{rel}$ Actual relations frequencies, tuples: $(a_s, a_t, f, \Delta)$
- $\mathcal{D}_{act}$ Latest activity names, tuples: $(a, f, \Delta)$
- $\mathcal{D}_{cases}$ Latest activity of a case, tuples: $(c, a, f, \Delta)$

With a certain periodicity, model update

- Activities from $\mathcal{D}_{act}$
- Dependencies and AND/XOR rules from $\mathcal{D}_{rel}$

Actually, we show that updates on $\mathcal{D}$ data structures affect only local parts of the model (incremental update of the process model)
Evaluation Datasets

Artificial dataset characteristics:

- Three randomly generated processes (to simulate concept drifts)
- Most complex model has 3 splits (1 AND and 2 XOR)
- Longest process has 16 activities
- Stream with 17,265 events

Real-world dataset (BPI Challenge 2012 log) characteristics

- Dutch Financial Institute
- 36 activities
- 262,198 events, among 13,087 process instances
Evaluation on Artificial Dataset

Model-to-model Metric

Assess the correspondence between original and discovered model

![Graph showing model-to-model similarity and observed events for different metrics and parameters like LCB (B = 300), LCB (B = 100), LC (ε = 0.000001), LC (ε = 0.001), SW (W = 300), SW (W = 100).]
Evaluation on Artificial Dataset

Space Requirements

Space expressed as number of stored items

No. stored items

0 250 500 750 1000 1250 1500 1750 2000

Observed events

LCB (B = 300)  
LCB (B = 100)  
LC (ε = 0.000001)  
LC (ε = 0.001)  
SW (W = 300)  
SW (W = 100)
Evaluation on Artificial Dataset

Time Requirements

Time required to process each event

![Graph showing time required to process each event](image)

- LCB (B = 300)
- LCB (B = 100)
- LC (ε = 0.000001)
- LC (ε = 0.001)
- SW (W = 300)
- SW (W = 100)
Evaluation on BPI Challenge 2012

Precision Metric

Precision of the discovered models

![Graph showing precision over observed events for SW, LC, and LCB.]

Time required to process an event:
- **SW**: 24.59ms
- **LC**: 5.68ms
- **LCB**: 2.56ms
Conclusions

- We addressed the problem of discovering process models from event streams
- Three approaches proposed, based on Heuristics Miner (with Sliding Window, with Lossy Counting, and with Lossy Counting with Budget)
- Experimental results on both artificial and real dataset, with improvements in terms of quality of the mined models, execution time, and space requirements as well

Future Work

- Improve the process analyst to mine different perspectives
- Animations to point out process drifts locations
End.
Heuristics Miner with SW

**Input:** $S$: event stream; $M$: memory; $\text{max}_M$: maximum memory size; 
*perform mining*: mining update periodicity

1. **forever do**
2. \( e \leftarrow \text{observe}(S) \) /* Observe a new event, where \( e = (c_i, a_i, t_i) \) */
3. /* Memory update */
4. \( \text{if} \ \text{size}(M) = \text{max}_M \ \text{then} \)
5. \hspace{1em} \( \text{shift}(M) \)
6. \( \) insert \( (M, e) \)
7. /* Mining update */
8. \( \text{if} \ \text{perform mining} \ \text{then} \)
9. \hspace{1em} \( L \leftarrow \text{convert}(M) \) /* Conversion of the memory into an event log that can be used with Heuristics Miner */
10. \hspace{2em} \( \text{HeuristicsMiner}(L) \)
11. **end**

**Algorithm 1:** Heuristics Miner with SW
Heuristics Miner with Lossy Counting

Input: $S$ event stream; $\epsilon$: approximation error

1. Initialize the data structure $D_A$, $D_C$, $D_R$

2. $N \leftarrow 1$

3. $w \leftarrow \left\lceil \frac{\epsilon}{N} \right\rceil$
   /* Bucket size */

4. forever do
   /* Event $e = (c, a, t)$ */
   e ← observe($S$)
   $b_{curr} = \left\lceil \frac{N}{w} \right\rceil$
   /* current bucket id */

   /* Update the $D_A$ data structure */
   if $\exists (a, f, \Delta) \in D_A$ such that $a = a_i$
   then
     Remove the entry $(a, f, \Delta)$ from $D_A$
     $D_A \leftarrow D_A \cup\{(a, f + 1, \Delta)\}$
   else
     $D_A \leftarrow D_A \cup\{(a_i, 1, b_{curr} - 1)\}$
   end

   /* Update the $D_C$ data structure */
   if $\exists (c, a_{last}, f, \Delta) \in D_C$ such that $c = c_i$
   then
     Remove the entry $(c, a_{last}, f, \Delta)$ from $D_C$
     $D_C \leftarrow D_C \cup\{(c, a_i, f + 1, \Delta)\}$
   else
     $D_C \leftarrow D_C \cup\{(c_i, a_i, 1, \Delta)\}$
   end

   /* Update the $D_R$ data structure */
   Build relation $r_i$ as $a_{last} \rightarrow a_i$
   if $\exists (r, f, \Delta) \in D_R$ such that $r = r$,
   then
     Remove the entry $(r, f, \Delta)$ from $D_R$
     $D_R \leftarrow D_R \cup\{(r, f + 1, \Delta)\}$
   else
     $D_R \leftarrow D_R \cup\{(r_i, 1, b_{curr} - 1)\}$
   end

   else
     $D_C \leftarrow D_C \cup\{(c_i, a_i, 1, \Delta)\}$
   end

   /* Periodic cleanup */
   if $N = 0 \mod w$
   then
     foreach $(a, f, \Delta) \in D_A$ s.t. $f + \Delta \leq b_{curr}$ do
     Remove $(a, f, \Delta)$ from $D_A$
   end

     foreach $(c, a, f, \Delta) \in D_C$ s.t. $f + \Delta \leq b_{curr}$ do
     Remove $(c, a, f, \Delta)$ from $D_C$
   end

     foreach $(r, f, \Delta) \in D_R$ s.t. $f + \Delta \leq b_{curr}$ do
     Remove $(r, f, \Delta)$ from $D_R$
   end

   end

   $N \leftarrow N + 1$

   Update the model as described in Section ??.

   For the directly follows relations, use the frequencies in $D_R$. 

end
Evaluation on BPI Challenge 2012

Precision vs Fitness Metric

Precision (SW)

Fitness (SW)

Precision (LCB)

Fitness (LCB)

Precision (LC)

Fitness (LC)
Space required by LCB (with $B = 300$) to store activities ($D_A$), relations ($D_R$) and cases ($D_C$)