Mining Periodic Patterns with a MDL Criterion

E. Galbrun,[1] P. Cellier,[2] N. Tatti,[1,4] A. Termier,[2] and B. Crémilleux[3]

[1] *Aalto University, Finland*
{esther.galbrun,nikolaj.tatti}@aalto.fi
[2] *Univ. Rennes, {INSA, Inria}, CNRS, IRISA, France*
{peggy.cellier,alexandre.termier}@irisa.fr
[3] *Normandie Univ., UNICAEN, ENSICAEN, CNRS – UMR GREYC, France*
bruno.cremilleux@unicaen.fr
[4] *F-Secure, Finland*

ABSTRACT

The quantity of event logs available is increasing rapidly, be they produced by industrial processes, computing systems, or life tracking, for instance. It is thus important to design effective ways to uncover the information they contain. Because event logs often record repetitive phenomena, mining periodic patterns is especially relevant when considering such data. Indeed, capturing such regularities is instrumental in providing condensed representations of the event sequences.

We present an approach for mining periodic patterns from event logs while relying on a Minimum Description Length (MDL) criterion to evaluate candidate patterns. Our goal is to extract a set of patterns that suitably characterises the periodic structure present in the data. We evaluate the interest of our approach on several real-world event log datasets.

Given an event sequence, our goal is to extract a representative collection of cycles,
and more generally, a representative collection of periodic patterns.

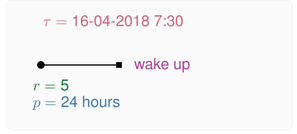
PERIODIC PATTERNS

Example of event sequence recording every day activities

$S = \langle$ (16-04-2018 7:30 , wake up),
(16-04-2018 7:40 , prepare coffee),
...
(16-04-2018 8:10 , take metro),
...
(16-04-2018 11:00 , attend meeting),
...
(16-04-2018 11:00 , eat dinner),
...
(17-04-2018 7:32 , wake up),
(17-04-2018 7:38 , prepare coffee),
...
(20-04-2018 7:28 , wake up),
(20-04-2018 7:41 , prepare coffee),
...
(15-06-2018 7:28 , wake up),
...$\rangle$

Example of simple cycle

On April 16, at 7:30 AM, wake up,
repeat every 24 hours for 5 days



starting point τ: the timestamp of the first occurrence,

event α: the repeating event,

period p: the inter-occurrence distance,

length r: the number of repetitions of the event, and

shift corrections E: a list of time offsets.

Tolerate variation in inter-occurrence distances.
Reconstruct occurrences timestamps of repetitions recursively.

$$t_1 = \tau,$$
$$t_2 = t_1 + p + e_1,$$
$$\ldots,$$
$$t_r = t_{r-1} + p + e_{r-1}.$$

Denote as $cover(C)$ the corresponding set of reconstructed timestamp–event pairs:

$$cover(C) = \{(t_1, \alpha), (t_2, \alpha), \ldots, (t_r, \alpha)\} ,$$

and for a collection $\mathcal{C}$ of cycles

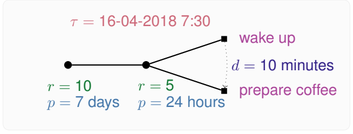
$$cover(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} cover(C) .$$

For a sequence S and cycle collection $\mathcal{C}$ we call *residual* the timestamp–event pairs of S not covered by any cycle in $\mathcal{C}$:

$$residual(\mathcal{C}, S) = S \setminus cover(\mathcal{C}) .$$

Example of complex cycle

On April 16, at 7:30 AM, wake up,
10 minutes later, prepare coffee,
repeat every 24 hours for 5 days,
repeat this every 7 days for 3 months



PROBLEM DEFINITION

Given an event sequence S, find the collection of patterns $\mathcal{P}$ minimising the cost

$$L(\mathcal{P}, S) = \sum_{P \in \mathcal{P}} L(P) + \sum_{o \in residual(\mathcal{P}, S)} L(o) .$$

A MDL CRITERION

The problem definition can be instantiated with different choices of costs.

- We propose costs motivated by the *MDL principle*
- We design a scheme for encoding the input event sequence using cycles and individual occurrences
- Cost of an element = length of the code word assigned under this scheme

The *MDL principle* is a concept from information theory based on the insight that any structure in the data can be exploited to compress the data, and aiming to strike a balance between the complexity of the model and its ability to describe the data.

ALGORITHM

We propose an algorithm with three stages:

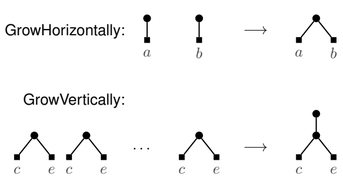
Extracting cycles: extract cycles for each event in turn,
using a dynamic programming routine and a heuristic extracting triples and chaining them

Building tree patterns from cycles: perform combination rounds to generate increasingly complex patterns

Selecting the final pattern collection: solve weighted set cover problem with greedy algorithm

Input: A multi-event sequence S,
a number k of top candidates to keep
Output: A collection of patterns $\mathcal{P}$

$\mathcal{I} \leftarrow$ ExtractCycles(S, k)
$\mathcal{C} \leftarrow \emptyset; \mathcal{V} \leftarrow \mathcal{I}; \mathcal{H} \leftarrow \mathcal{I}$
while $\mathcal{H} \neq \emptyset$ OR $\mathcal{V} \neq \emptyset$ **do**
  $\mathcal{V}' \leftarrow$ CombineVertically$(\mathcal{H}, \mathcal{P}, S, k)$
  $\mathcal{H}' \leftarrow$ CombineHorizontally$(\mathcal{V}, \mathcal{P}, S, k)$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{H} \cup \mathcal{V}; \mathcal{V} \leftarrow \mathcal{V}'; \mathcal{H} \leftarrow \mathcal{H}'$
$\mathcal{P} \leftarrow$ GreedyCover$(\mathcal{C}, S)$
return $\mathcal{P}$

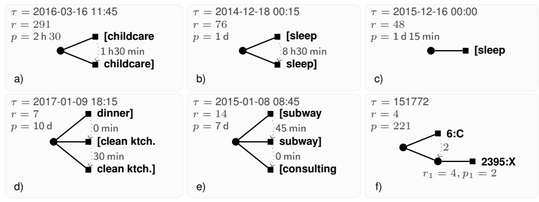


EXPERIMENTS

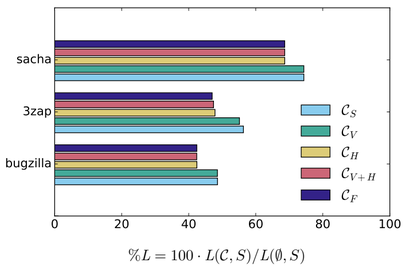
We run experiments on real-world event log datasets.

Life-tracking (a.k.a. quantified self): sacha

Execution traces: 3zap and bugzilla



Example patterns from sacha *(a–e) and* 3zap *(f)*



$$\%L = 100 \cdot L(\mathcal{C}, S) / L(\emptyset, S)$$

Compression ratios for three sequences

FUTURE DIRECTIONS

- Taking into account background knowledge
- Making the algorithm more robust to noise and more scalable
- Designing tailored visualizations

RELATED REFERENCES

- P. Grünwald. *The Minimum Description Length Principle.* MIT Press, 2007.
- J. Vreeken, M. van Leeuwen, and A. Siebes. *Krimp: mining itemsets that compress.* Data Min Knowl Discov, 23(1):169–214, 2011.
- P. Lopez-Cueva et al. *Debugging Embedded Multimedia Application Traces Through periodic pattern mining.* In EMSOFT, 2012.
- E. Galbrun, P. Cellier, N. Tatti, A. Termier, and B. Crémilleux. *Mining periodic patterns with a MDL criterion.* ArXiv e-prints, 2018. arXiv:1807.01706