Mining Periodic Patterns with a MDL Criterion

E. Galbrun P. Cellier N. Tatti
A. Termier B. Crémilleux

ECML PKDD 2018 - Dublin

Aalto University School of Science
UMR IRISA
Inria informatics mathematics
GREYC

Given an 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),
$\ldots\rangle$

Extract activity patterns

$S = \langle$ (**16-04-2018 7:30 , wake up**), $\leftarrow \#1$
(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**), $\leftarrow \#2$
(17-04-2018 7:38 , prepare coffee),
...
(**20-04-2018 7:28 , wake up**), $\leftarrow \#5$
(20-04-2018 7:41 , prepare coffee),
...
(15-06-2018 7:28 , wake up),
$\ldots\rangle$

16-04-2018 7:30, wake up

repeat every 24 hours for 5 days

Event cycles

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

Event cycles

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

Event cycles

starting point τ: the timestamp of the first occurrence,
event α: the repeating event,
period p: the inter-occurrence distance, and
length r: the number of repetitions of the event.

Reconstruct occurrences timestamps of repetitions recursively:

$$
\begin{aligned}
t_1 &= \tau, \\
t_2 &= t_1 + p, \\
&\dots \\
t_r &= t_{r-1} + p.
\end{aligned}
$$

Event cycles

Tolerate variation in inter-occurrence distances,
shift corrections $E = \langle e_1, \dots, e_{r-1} \rangle$.

Reconstruct occurrences timestamps of repetitions recursively:

$$
\begin{aligned}
t_1 &= \tau, \\
t_2 &= t_1 + p + e_1, \\
&\dots \\
t_r &= t_{r-1} + p + e_{r-1}.
\end{aligned}
$$

Event cycles

A cycle is specified by:

event α: the repeating event,

length r: the number of repetitions of the event,

period p: the inter-occurrence distance,

starting point τ: the timestamp of the first occurrence, and

shift corrections E: a list of time offsets.

Hence, a cycle is a 5-tuple $C = (\alpha, r, p, \tau, E)$.

Problem statement (informal)

Given an event sequence, our goal is to extract **a representative collection of periodic patterns called *cycles***.

Cycle cover

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 $\mathcal{C}$:

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

Problem statement

We associate

- a cost $L(o)$ to each individual occurrence
- a cost $L(C)$ to each cycle

Then, we can reformulate our problem as follows:

Problem

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

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

A MDL criterion

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

- We propose costs motivated by the **MDL principle**

A MDL criterion

- We propose costs motivated by the **MDL principle**

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.

A MDL criterion

- 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

More complex periodic patterns

Pattern language so far: cycles over single events

However, several events might recur regularly together and repetitions might be nested with several levels of periodicity.

More expressive pattern language: hierarchy of cyclic blocks, organised as a tree

More complex periodic patterns

Extract activity patterns

$S = \langle$
(**16-04-2018 7:30** , **wake up**),← #1
(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**),← #2
(17-04-2018 7:38 , prepare coffee),
...
(**20-04-2018 7:28** , **wake up**),← #5
(20-04-2018 7:41 , prepare coffee),
...
(15-06-2018 7:28 , wake up),
...$\rangle$

16-04-2018 7:30, wake up

repeat every 24 hours for 5 days

More complex periodic patterns

Extract more complex activity patterns

$S = \langle$ (16-04-2018 7:30 , wake up), $\leftarrow$ #1
(**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), $\leftarrow$ #2
(**17-04-2018 7:38** , **prepare coffee**),
...
(20-04-2018 7:28 , wake up), $\leftarrow$ #5
(**20-04-2018 7:41** , **prepare coffee**),
...
(15-06-2018 7:28 , wake up),
$\dots\rangle$

16-04-2018 7:30, wake up

10 min later, prepare coffee

repeat every 24 hours for 5 days

More complex periodic patterns

Extract more complex activity patterns

$S = \langle$ (16-04-2018 7:30 , wake up), $\leftarrow \#1$ **- 1st week**
(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), $\leftarrow \#2$
(17-04-2018 7:38 , prepare coffee),
...
(20-04-2018 7:28 , wake up), $\leftarrow \#5$
(20-04-2018 7:41 , prepare coffee),
...
(15-06-2018 7:28 , wake up), $\leftarrow \#5$ **- 9th week**
$\ldots \rangle$

16-04-2018 7:30, wake up

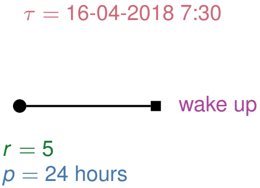
10 min later, prepare coffee

repeat every 24 hours for 5 days

repeat every 7 days for 3 months

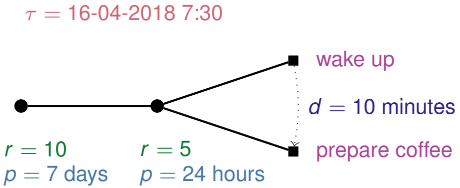
More complex periodic patterns

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



More complex periodic patterns

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



More complex periodic patterns

We extend the encoding for this more expressive pattern language, i.e. define the cost of a pattern P, $L(P)$.

Then, we can extend our problem statement:

Problem

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

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

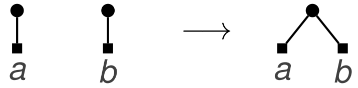
Finding periodic patterns that compress

We have a pattern language and associated encoding.
How do we actually find such patterns?

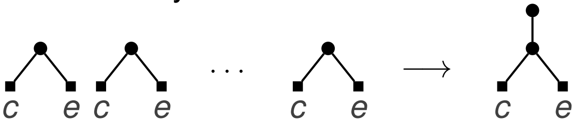
Finding periodic patterns that compress

A natural way to build patterns:
start with cycles and combine them together

GrowHorizontally:



GrowVertically:



Algorithm outline

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

Algorithm outline

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 $\mathcal{P}$

Datasets

We run experiments on real-world event log datasets.

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

Execution traces: 3zap and bugzilla

Example patterns

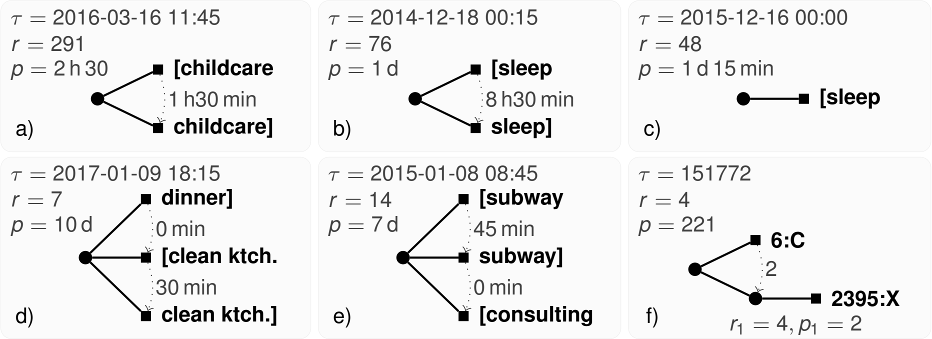


Figure: Example patterns from sacha (a–e) and 3zap (f).

Compression ratios

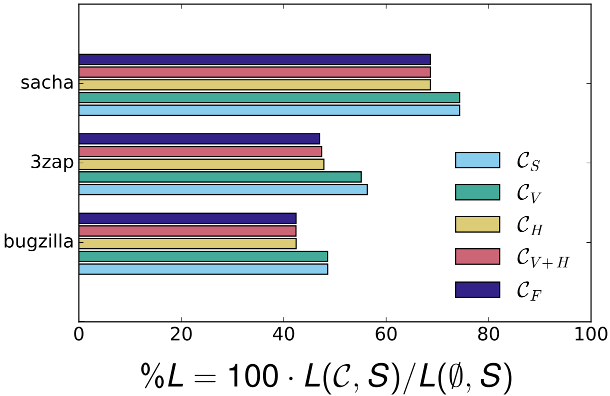


Figure: Compression ratios for three sequences.

Conclusion

We propose

- an approach to mine periodic patterns with a MDL criterion
- an algorithm to put it into practise

In our experiments, we show that we are able to

- extract sets of patterns that compress the input sequences
- identify meaningful patterns

Directions for future work

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

Simple periodic patterns: Cycles

Experiments

Cycle cover

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

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

A cycle C **covers** an occurrence if the corresponding timestamp–event pair belongs to $\mathit{cover}(C)$.

Cycle cover

- Time is represented in an absolute manner
- An event can occur only once at any given timestamp

Hence we do not need to worry about overlapping cycles nor about an order between cycles

Cycle cover

Denote as $\mathit{cover}(\mathcal{C})$ the set of timestamp–event pairs for a collection $\mathcal{C}$ of cycles $\mathcal{C} = \{C_1, \dots, C_m\}$:

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

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

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

Code lengths as costs

$$L(C) = L(\alpha) + L(r) + L(p) + L(\tau) + L(E)$$

Look more closely at

- the range in which each of these pieces of information takes value,
- what values—if any—should be favoured, and
- how the values of the different pieces depend on one another.

Code lengths as costs

$$L(C) = \underline{L(\alpha)} + L(r) + L(p) + L(\tau) + L(E)$$

Cycle event

Events that occur more frequently receive shorter code words:

$$L(\alpha) = -\log(fr(\alpha)) = -\log(\frac{|S^{(\alpha)}|}{|S|})$$

Code lengths as costs

$$L(C) = L(\alpha) + \underline{L(r)} + L(p) + L(\tau) + L(E)$$

Cycle length

The length of a cycle cannot be greater than the number of occurrences of the event:

$$L(r) = \log(\left|S^{(\alpha)}\right|)$$

Code lengths as costs

$$L(C) = L(\alpha) + L(r) + L(p) + L(\tau) + \underline{L(E)}$$

Cycle shift corrections

value digits	Each correction e is represented by $\lvert e \rvert$ ones,
sign digits	prefixed by a single bit indicating the shift direction,
separating digits	separated from the next correction by a zero

Example: $\langle 3, -2, 0, 4 \rangle \rightarrow 0\mathit{111}\mathbf{0}1\mathit{11}\mathbf{0}0\mathbf{0}0\mathit{1111}\mathbf{0}$

$$L(E) = 2\,\lvert E \rvert + \sum_{e \in E} \lvert e \rvert$$

Code lengths as costs

$$L(C) = L(\alpha) + L(r) + \underline{L(p)} + L(\tau) + L(E)$$

Cycle period

The cycle can span at most the time of the whole sequence:

$$L(p) = \log\left(\left\lfloor \frac{\Delta(S) - \sigma(E)}{r - 1} \right\rfloor\right)$$

Code lengths as costs

$$L(C) = L(\alpha) + L(r) + L(p) + \underline{L(\tau)} + L(E)$$

Cycle starting point

The cycle can start anytime between $t_{\text{start}}(S)$ and $t_{\text{end}}(S) - \Delta(C)$:

$$L(\tau) = \log(\Delta(S) - \sigma(E) - (r-1)p + 1)$$

Code lengths as costs

Putting everything together, the cost of a cycle is

$$
\begin{aligned}
L(C) &= L(\alpha) + L(r) + L(p) + L(\tau) + L(E) \\
&= \log(|S|) + \log\left(\left\lfloor \frac{\Delta(S) - \sigma(E)}{r-1} \right\rfloor\right) \\
&+ \log(\Delta(S) - \sigma(E) - (r-1)p + 1) \\
&+ 2\,|E| + \sum_{e \in E} |e|
\end{aligned}
$$

Code lengths as costs

Putting everything together, the cost of a cycle is

$$
\begin{aligned}
L(C) &= L(\alpha) + L(r) + L(p) + L(\tau) + L(E) \\
&= \log(|S|) + \log\left(\left\lfloor \frac{\Delta(S) - \sigma(E)}{r-1} \right\rfloor\right) \\
&+ \log(\Delta(S) - \sigma(E) - (r-1)p + 1) \\
&+ 2\,|E| + \sum_{e \in E} |e|
\end{aligned}
$$

On the other hand, the cost of an individual occurrence is

$$
L(o) = L(t) + L(\alpha) = \log(\Delta(S) + 1) - \log\left(\frac{\left|S^{(\alpha)}\right|}{|S|}\right)
$$

Problem statement

Problem

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

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

Choosing the best period

Given an ordered list of occurrences $\langle t_1, t_2, \ldots t_l \rangle$ of event α

Goal determine the best cycle to cover these occurrences

Choosing the best period

Given an ordered list of occurrences $\langle t_1, t_2, \ldots t_l \rangle$ of event α

Goal determine the best cycle to cover these occurrences

- clearly, α, r, and τ are determined
- need to find p such that $L(C)$ is minimised

Choosing the best period

Given an ordered list of occurrences $\langle t_1, t_2, \ldots t_l \rangle$ of event α

Goal determine the best cycle to cover these occurrences

– clearly, α, r, and τ are determined

– need to find p such that $L(C)$ is minimised

Find p that minimises $L(E)$

$\rightarrow$ let p equal the median of the inter-occurrence distances

Simple periodic patterns: Cycles

Experiments

Compression ratios

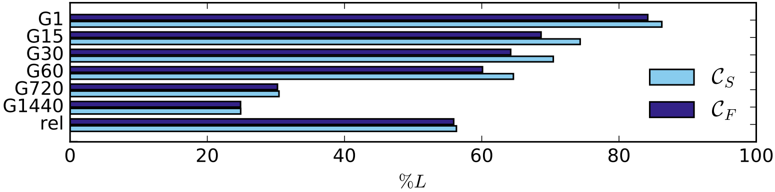


Figure: Compression ratios for sacha sequences with various time granularities.

Running times

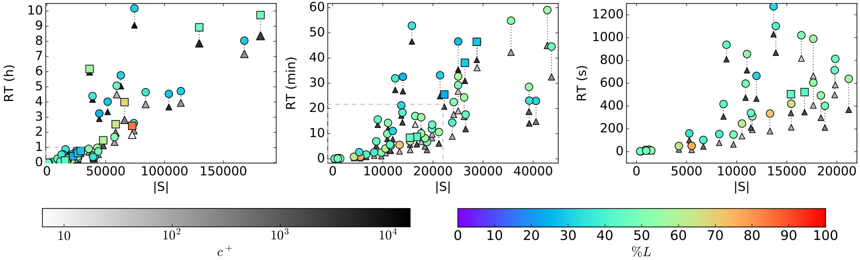


Figure: Running times for sequences from the different datasets, in hours (left) and zoomed-in in minutes (middle) and seconds (right).