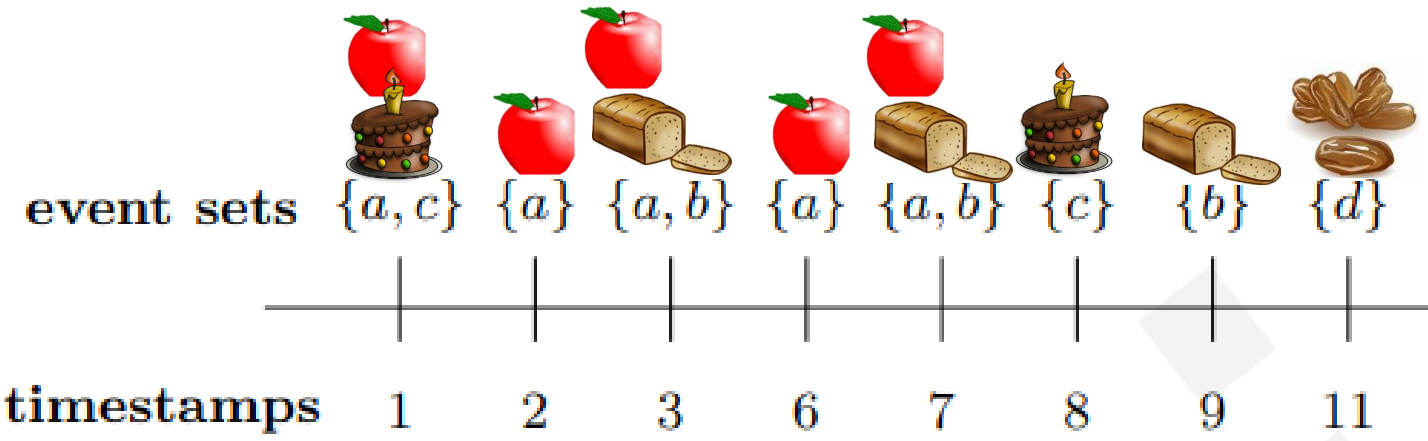
MaxFEM: Mining Maximal Frequent Episodes in Complex Event Sequences

Philippe Fournier-Viger, M. Saqib Nawaz, Yulin He, Farid Nouioua, Unil Yun

Source code and datasets available in the SPMF library

Introduction

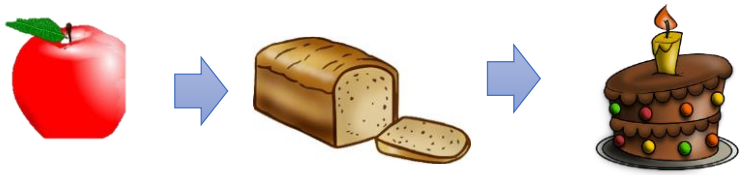
- Analyzing sequence data to discover **frequent patterns** is a popular task in data mining.
- **Discrete sequence**: ordered list of events or symbols.



$$S = \langle (\{a,c\}, 1), (\{a\}, 2), (\{a,b\}, 3), (\{a\}, 6), (\{a,b\}, 7), (\{c\}, 8), (\{b\}, 9), (\{d\}, 11) \rangle$$

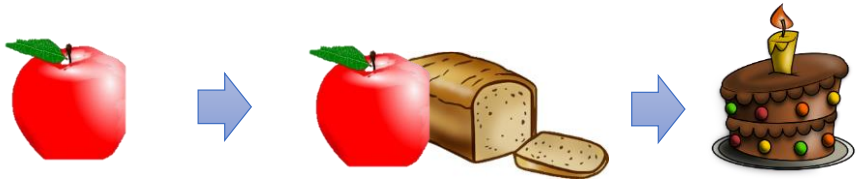
Frequent episode Mining

- **Goal:** finding the **frequent episodes** (subsequences of events) that appear at least *minsup* times in a sequence.
- **Three types of episodes:**
  - **Serial episodes**: all events are sequentially ordered



  - **Simultaneous episodes**: all events appeared at the same time

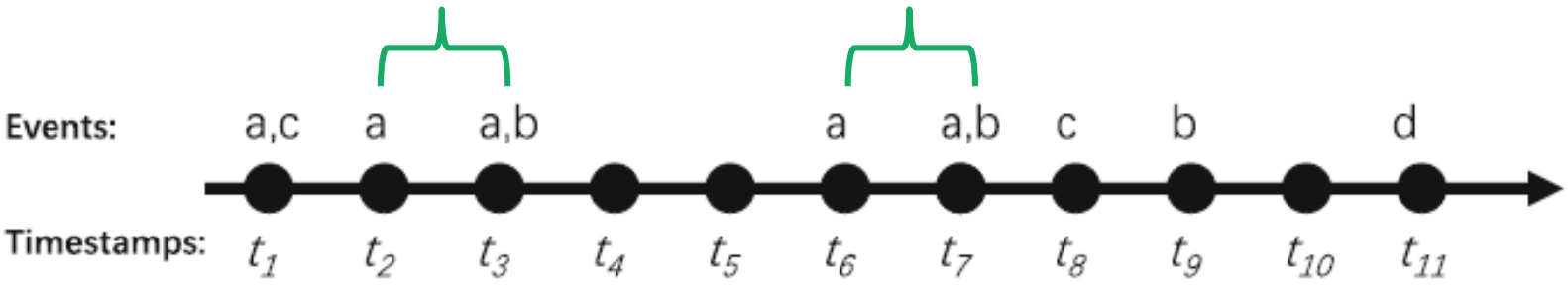
  - **Composite episodes**: events are partially ordered



Frequent episode Mining

- **Several algorithms:**
  - **WINEPI (1995)**: breadth-first search, window-based
  - **MINEPI (1995)**: breadth-first search, minimal occurrences
  - **EMMA** and **MINEPI+ (2008)**: depth-first search, head frequency
  - **TKE (2019)**: find the top-k most frequent episodes
- Use different definitions, data structures and search strategies

How to count the support of episodes?



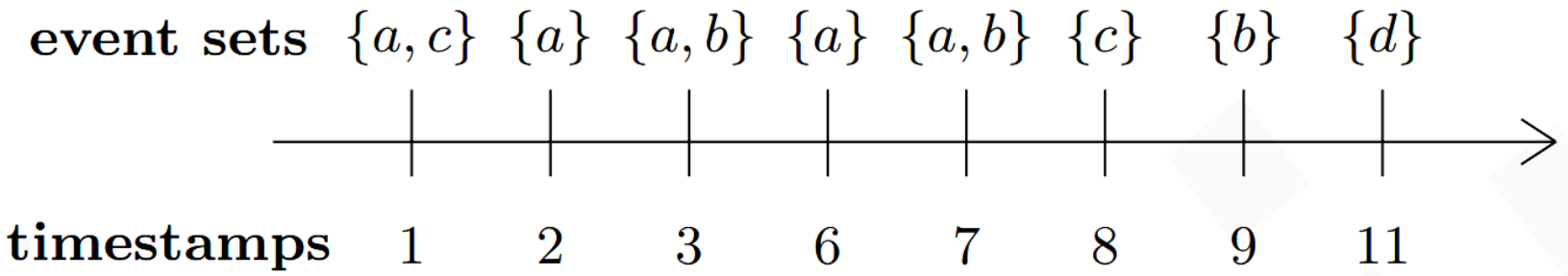
$winlen = 2$

$occSet(\langle\{a\},\{a,b\}\rangle) = \{[t_2, t_3], [t_6, t_7]\}$

the **head frequency support**, $\sup(\langle\{a\},\{a,b\}\rangle) = 2$

Example

Event sequence



Parameters

$winlen = 2$

$minsup = 2$

Frequent episodes

Episode	Support
$\langle\{a, b\}\rangle$	2
$\langle\{a\}, \{b\}\rangle$	2
$\langle\{a\}, \{a, b\}\rangle$	2
$\langle\{a\}, \{a\}\rangle$	3
$\langle\{a\}\rangle$	5
$\langle\{b\}\rangle$	3
$\langle\{c\}\rangle$	2

Limitation of FEM

- FEM algorithms can find **millions of episodes**!
- For each frequent episode, all the sub-episodes are often also frequent.

milk → bread → orange,

milk → bread,

milk → orange

bread → orange

milk

bread

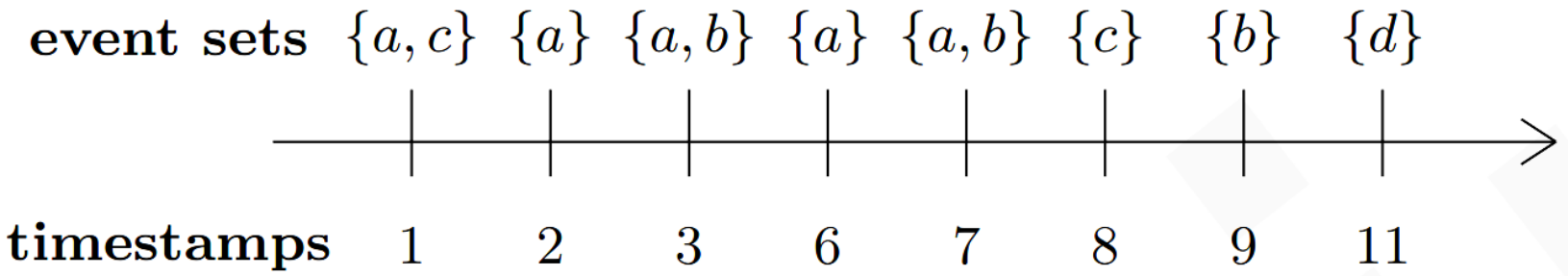
orange

A solution

- Discover only the **maximal episodes**.
- A frequent episode α is **maximal** if it is not a subsequence of another frequent episode β.
- **Benefit**: much less episodes and most of the information is preserved.
- Only one algorithm but for simple sequences (no simultaneous events)
- How to deal with the more general case of finding maximal episodes in a complex sequence?

Example

Event sequence



Parameters

$winlen = 2$

$minsup = 2$

Frequent episodes

Episode	Support	Maximal?
$\langle\{a, b\}\rangle$	2	No
$\langle\{a\}, \{b\}\rangle$	2	No
$\langle\{a\}, \{a, b\}\rangle$	2	**Yes**
$\langle\{a\}, \{a\}\rangle$	3	No
$\langle\{a\}\rangle$	5	No
$\langle\{b\}\rangle$	3	No
$\langle\{c\}\rangle$	2	**Yes**

The MaxFEM algorithm

- A new algorithm: **MaxFEM** (**M**aximal **F**requent **E**pisode **M**ining)
  - To find the *maximal* frequent episodes
  - Extends the EMMA algorithm
  - Applies techniques to keep only maximal episodes and some optimizations

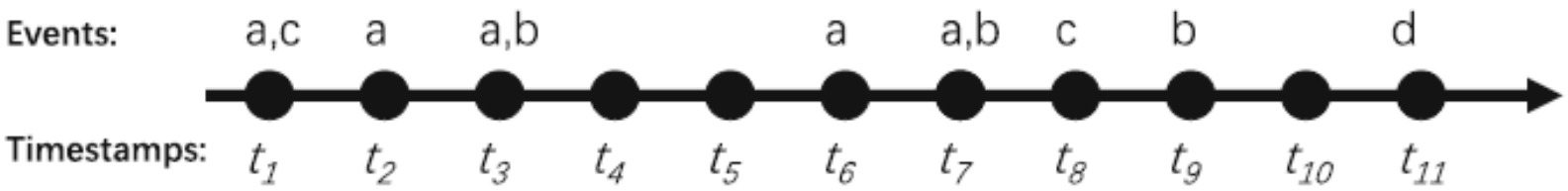
Step 1: Find the Frequent Events

- Scan the sequence to find the frequent events
- *Suppose* *minsup* = 2 and *winlen* = 3

Frequent events

Episode	Support
$\langle\{a\}\rangle$	5
$\langle\{b\}\rangle$	3
$\langle\{c\}\rangle$	2

Step 2: Create Location Lists

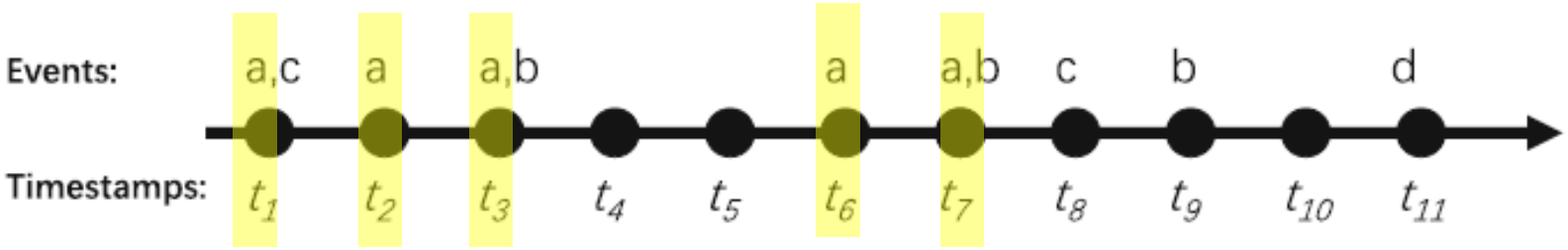


Create a *location list* for each frequent event

Episode	Support	location list
$\langle\{a\}\rangle$	5	$locList(a) = \{1, 2, 3, 6, 7\},$
$\langle\{b\}\rangle$	3	$locList(b) = \{3, 7, 9\}$
$\langle\{c\}\rangle$	2	$locList(c) = \{1, 8\}$

Note: for any episode α, we have $|locList(\alpha)| = \sup(\alpha)$

Step 2: Create Location Lists



Create a *location list* for each frequent event

Episode	Support	location list
$\langle\{a\}\rangle$	5	$locList(a) = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{6}, \mathbf{7}\},$
$\langle\{b\}\rangle$	3	$locList(b) = \{3, 7, 9\}$
$\langle\{c\}\rangle$	2	$locList(c) = \{1, 8\}$

Note: for any episode α, we have $|locList(\alpha)| = \sup(\alpha)$

Step 3: Find the Frequent Parallel Episodes

- Recursively combine frequent events found to create parallel episodes with their locations lists.
- Keep the *frequent* parallel episodes

Frequent episodes

Episode	location list
$\langle\{a\}\rangle$	$\{1, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	$\{1, 8\}$

Frequent parallel episodes

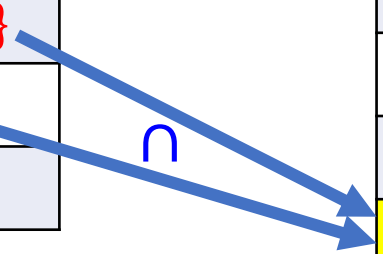
Episode	Support	location list
$\langle\{a\}\rangle$	5	$\{1, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	3	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	2	$\{1, 8\}$

Step 3: Find the Frequent Parallel Episodes

- Recursively combine frequent events found to create parallel episodes with their locations lists.
- Keep the *frequent* parallel episodes

Frequent episodes

Episode	location list
$\langle\{a\}\rangle$	$\{1, 2, \mathbf{3}, 6, \mathbf{7}\}$
$\langle\{b\}\rangle$	$\{\mathbf{3}, \mathbf{7}, 9\}$
$\langle\{c\}\rangle$	$\{1, 8\}$



Frequent parallel episodes

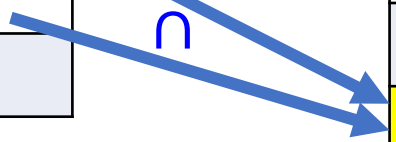
Episode	Support	location list
$\langle\{a\}\rangle$	5	$\{1, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	3	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	2	$\{1, 8\}$
$\langle\{a, b\}\rangle$	??	**{3,7}**

Step 3: Find the Frequent Parallel Episodes

- Recursively combine frequent events found to create parallel episodes with their locations lists.
- Keep the *frequent* parallel episodes

Frequent episodes

Episode	location list
$\langle\{a\}\rangle$	$\{1, 2, \mathbf{3}, 6, \mathbf{7}\}$
$\langle\{b\}\rangle$	$\{\mathbf{3}, \mathbf{7}, 9\}$
$\langle\{c\}\rangle$	$\{1, 8\}$



Frequent parallel episodes

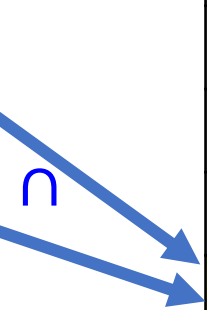
Episode	Support	location list
$\langle\{a\}\rangle$	5	$\{1, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	3	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	2	$\{1, 8\}$
$\langle\{a, b\}\rangle$	**2**	**{3,7}**

Step 3: Find the Frequent Parallel Episodes

- Recursively combine frequent events found to create parallel episodes with their locations lists.
- Keep the *frequent* parallel episodes

Frequent episodes

Episode	location list
$\langle\{a\}\rangle$	$\{\mathbf{1}, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	$\{\mathbf{1}, 8\}$



Frequent parallel episodes

Episode	Support	location list
$\langle\{a\}\rangle$	5	$\{1, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	3	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	2	$\{1, 8\}$
$\langle\{a, b\}\rangle$	2	{3,7}
$\langle\{a, c\}\rangle$	?	{**1**}

Step 3: Find the Frequent Parallel Episodes

- Recursively combine frequent events found to create parallel episodes with their locations lists.
- Keep the *frequent* parallel episodes

Frequent episodes

Episode	location list
$\langle\{a\}\rangle$	$\{\mathbf{1}, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	$\{\mathbf{1}, 8\}$

$\cap$

Frequent parallel episodes

Episode	Support	location list
$\langle\{a\}\rangle$	5	$\{1, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	3	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	2	$\{1, 8\}$
$\langle\{a, b\}\rangle$	2	{3,7}
~~$\langle\{a, c\}\rangle$~~	~~1~~	~~{1}~~

Step 3: Find the Frequent Parallel Episodes

- Recursively combine frequent events found to create parallel episodes with their locations lists.
- Keep the *frequent* parallel episodes

Frequent episodes

Episode	location list
$\langle\{a\}\rangle$	$\{\mathbf{1}, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	$\{\mathbf{1}, 8\}$

Frequent parallel episodes

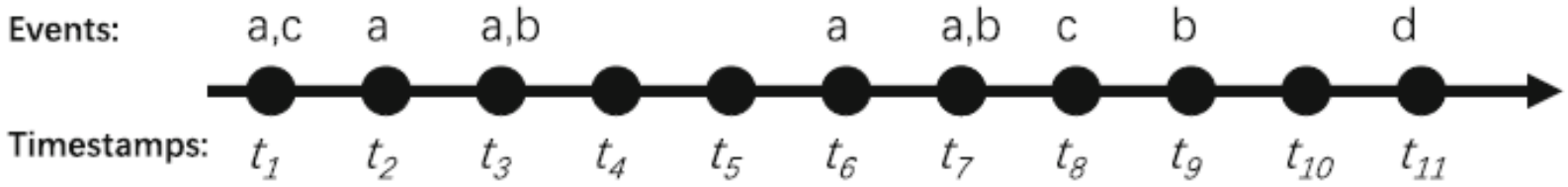
Episode	Support	location list
$\langle\{a\}\rangle$	5	$\{1, 2, 3, 6, 7\}$
$\langle\{b\}\rangle$	3	$\{3, 7, 9\}$
$\langle\{c\}\rangle$	2	$\{1, 8\}$
$\langle\{a, \mathrm{b}\}\rangle$	2	{3,7}

Step 4: Re-encode the Input Sequence Using Parallel Episodes

- A unique identifier is given to each parallel episode

Frequent parallel episodes

Episode	Support	ID
$\langle\{a\}\rangle$	5	#1
$\langle\{b\}\rangle$	3	#2
$\langle\{c\}\rangle$	2	#3
$\langle\{a, b\}\rangle$	2	#4

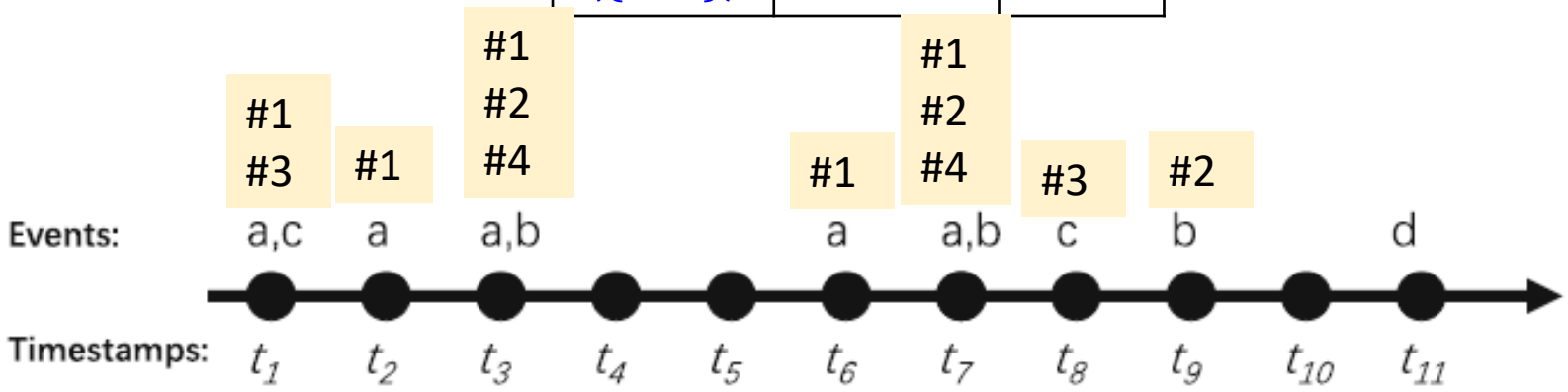


Step 4: Re-encode the Input Sequence Using Parallel Episodes

- A unique identifier is given to each parallel episode

Frequent parallel episodes

Episode	Support	ID
$\langle\{a\}\rangle$	5	#1
$\langle\{b\}\rangle$	3	#2
$\langle\{c\}\rangle$	2	#3
$\langle\{a, b\}\rangle$	2	#4



Step 5: Find Frequent Composite episodes

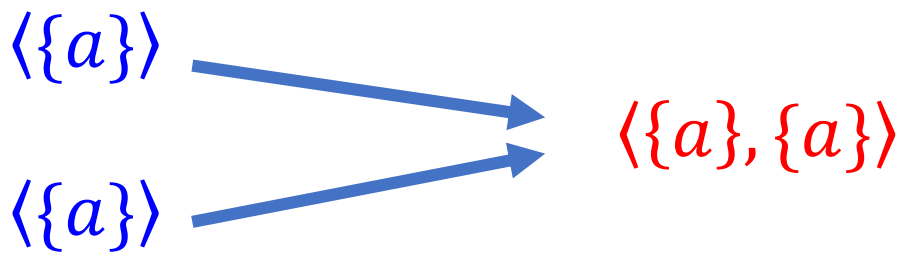
- Parallel episodes are combined by a process called serial extension to create **composite episodes**.

$\langle \{a\} \rangle$

$\langle \{a\} \rangle$

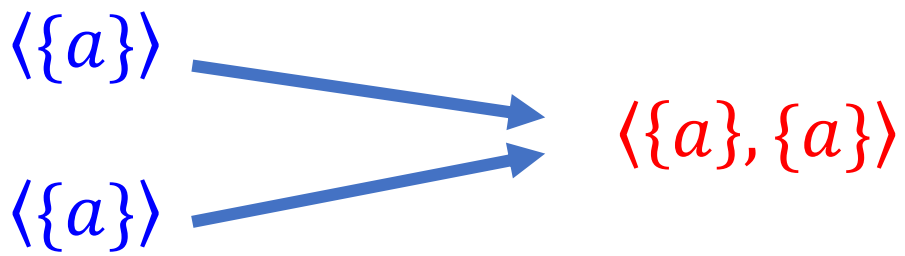
Step 5: Find Frequent Composite episodes

- Parallel episodes are combined by a process called serial extension to create **composite episodes**.



Step 5: Find Frequent Composite episodes

- Parallel episodes are combined by a process called serial extension to create **composite episodes**.
- A **bound list structure** is created for each composite episode, to obtain its support.



$$boundList(\langle\{a\},\{a\}\rangle) = \{[t_1, t_2], [t_2, t_3], [t_6, t_7]\}$$

$$\sup(\langle\{a\},\{a\}\rangle) = |t_1, t_2, t_6| = 3$$

Step 5: Find Frequent Composite episodes

- During the search, to find the maximal episodes:
  - A set W stores the episodes that are currently maximal.
  - When a new episode α is found:
    - **Sub-episode checking:**
      If α is included in an episode β already in W,
      then α is not added to W.
    - **Super-episode checking:**
      If an episode β from W is included in α,
      then β is removed from W

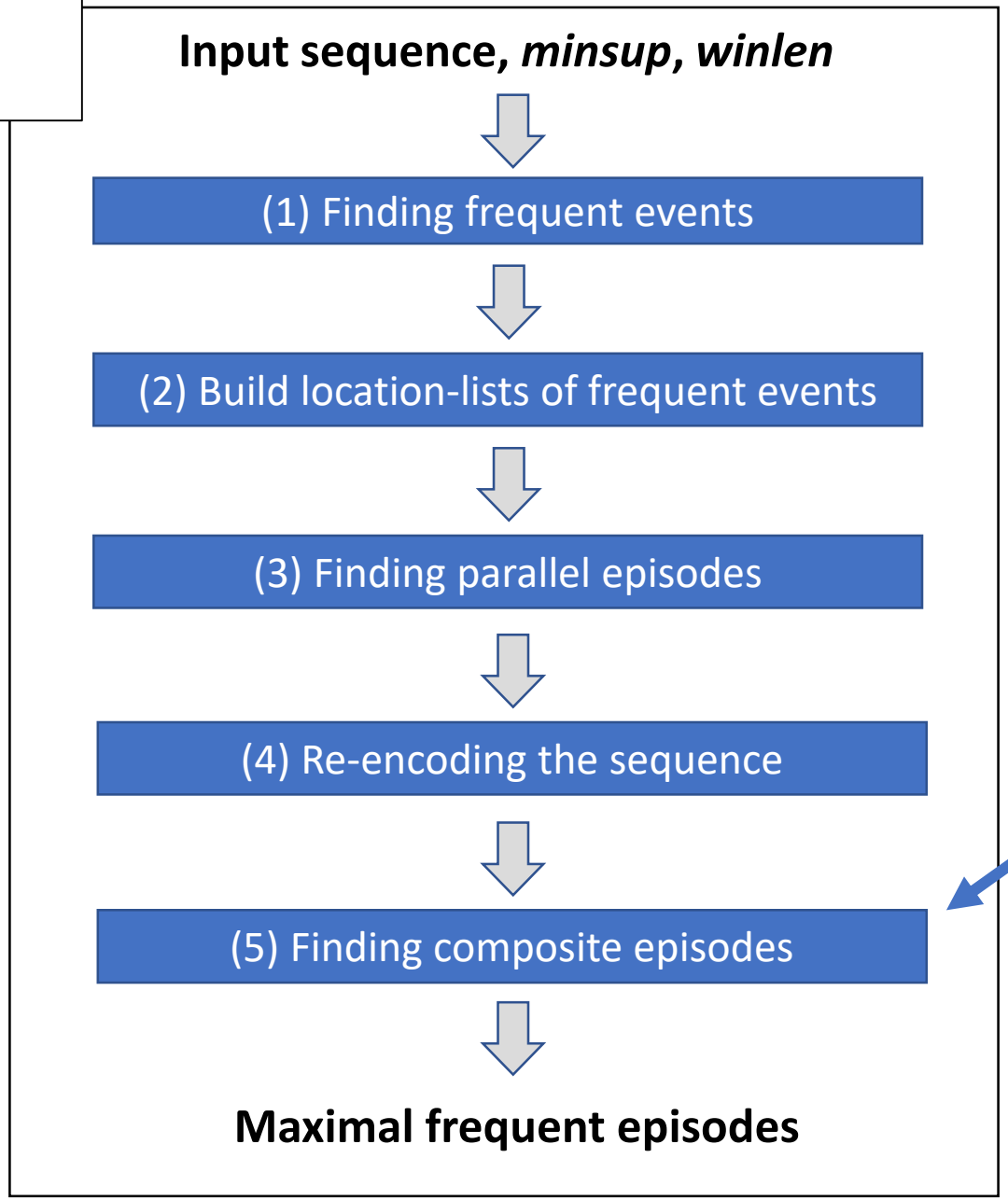
Step 5: Finding Frequent Composite episodes

- Result:

Maximal frequent episodes

Episode	Support
⟨{c}, {a}⟩	2
⟨{a}, {c} ⟩	2
⟨{a}, {a, b} ⟩	3

Flowchart of MaxFEM



Non-maximal episodes are filtered **only** in this step!

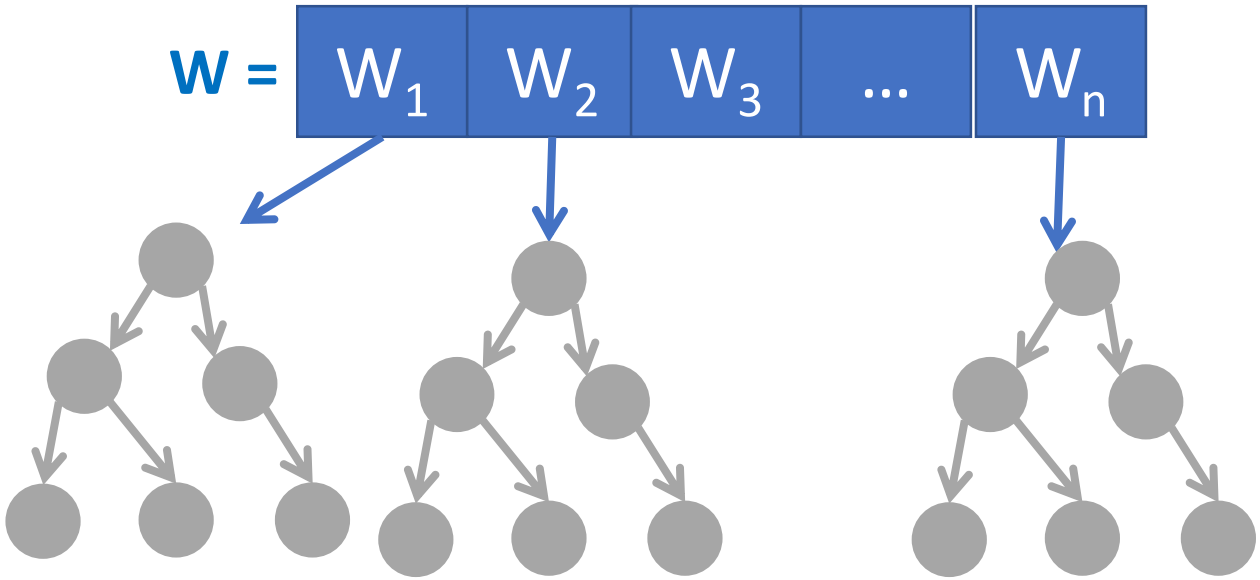
Optimization 1
EFE: Efficient Filtering of Non-maximal episodes

We implement **W** as a List of heaps

The **k-th** list entry contains episodes of size **k**

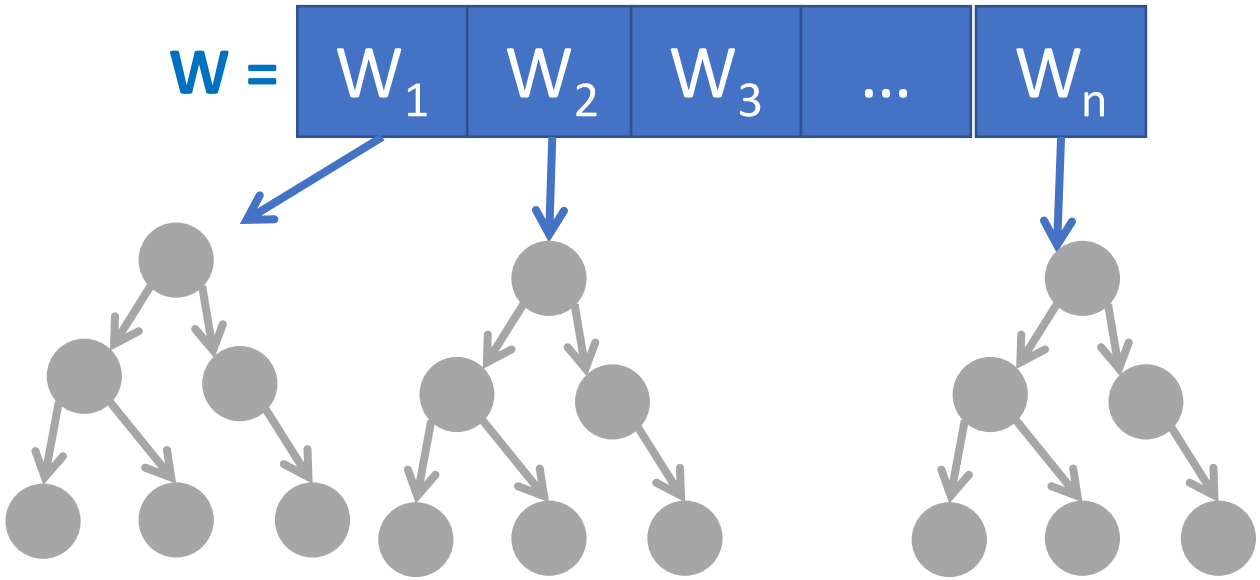
This allows to perform super-episode checking and sub-episode checking only with smaller and larger patterns

Optimization 1
EFE: Efficient Filtering of Non-maximal episodes



- The **sum of events** in each pattern is calculated.
- Each **heap** orders patterns by decreasing sum of events.
- For each pattern S_a found and pattern S_b in Z_k, if $sum(S_a) < sum(S_a)$ we don't need to perform super-episode checking with W_b and any following patterns in W_k.
- Similar for sub-episode-checking

Optimization 1
EFE: Efficient Filtering of Non-maximal episodes



- **Support check optimization**:
  - A pattern cannot be contained in another pattern if its support is smaller.
  - A pattern cannot contain another pattern if its support is larger.

Two more optimizations

- **Strategy 2. Skip Extension checking (SEC)**
  - If a frequent episode ***ep*** is extended by serial extension to form another frequent episode, then it is unnecessary to do super-episode and sub-episode checking for ***ep*** because it is **not maximal.**

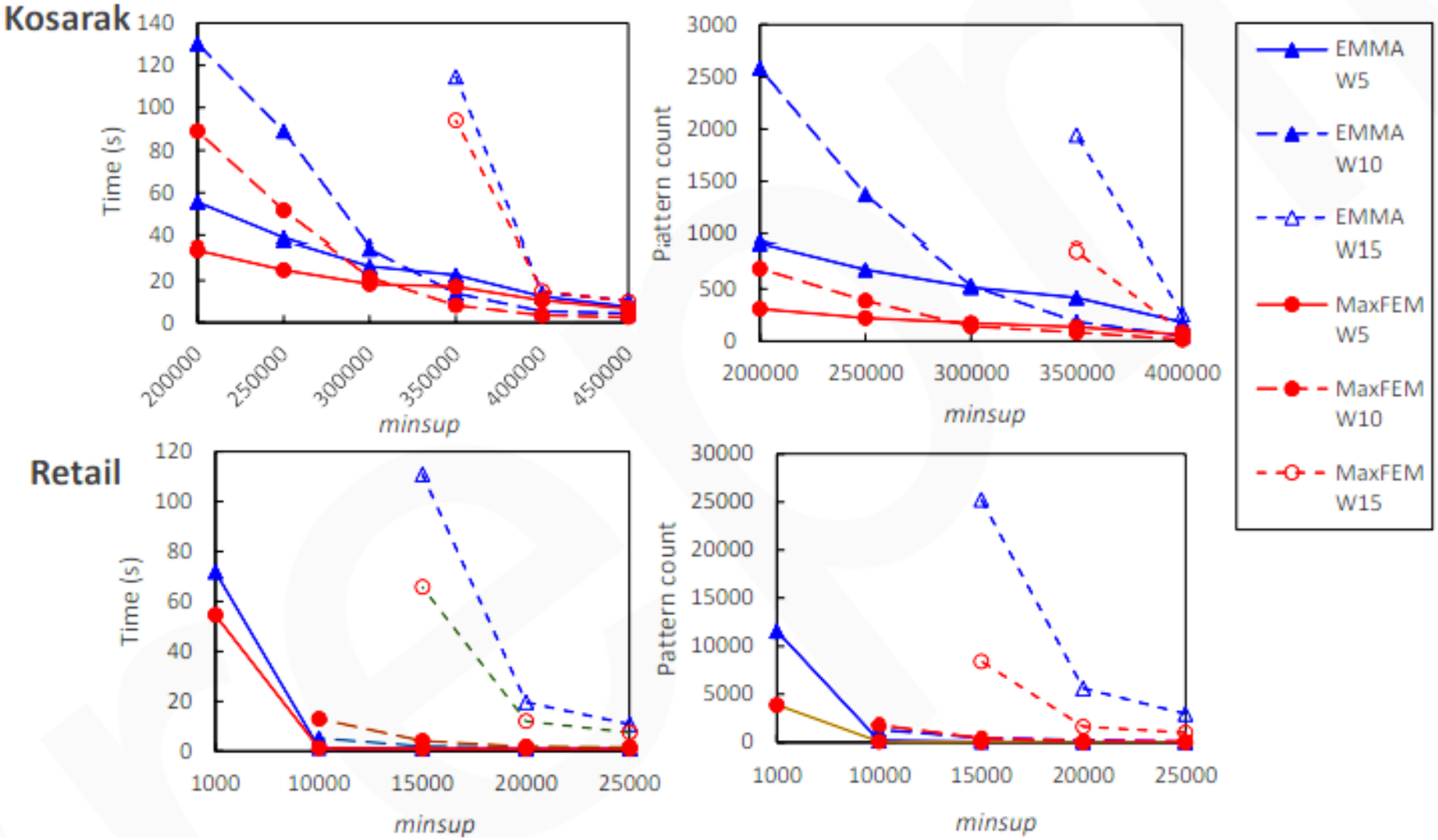
- **Strategy 3. Temporal pruning (TP).**
  - When creating a bound-list, if at any point the number of remaining elements is not enough to satisfy *minsup*, the construction of the bound-list is stopped.

Experiments

- **Two benchmark datasets:**

Dataset	Avg. Sequ. Len.	#Events	#Sequences	Density(%)
Kosarak	8.1	41,270	990,000	0.02
Retail	10.3	16,470	88,162	0.06

- **Compared algorithms:**
  - MaxFEM
  - EMMA
- **Setup:**
  - Java, Windows 11, laptop with Core i7-8565U processor, 16GB RAM
  - **Experiment**: *Winlen* $\in \{5, 10, 15\}$ and *minsup* is varied
  - A 300 second time limit



Two main observations:

- Much less maximal episodes than frequent episodes
  e.g. 694 maximal episodes vs 2,583 episodes on Kosarak for *minsup* = 20,000
- **MaxFEM** is about 10% to 40% faster than **EMMA** (thanks to optimizations)

Conclusion

Contributions

- A new problem of **maximal episode mining** for the general case of a **complex event sequence** and with the **head frequency support** function
- A new efficient algorithm **MaxFEM** with optimizations
- A version to find all frequent episodes called **AFEM**

Research opportunities

- Extend MaxFEM for other frequency functions and sequences types
- Design a parallel and distributed version of MaxFEM

Open source Java data mining software, 240 algorithms
http://www.phillippe-fournier-viger.com/spmf/