Mining Partially-Ordered Episode Rules in an Event Sequence

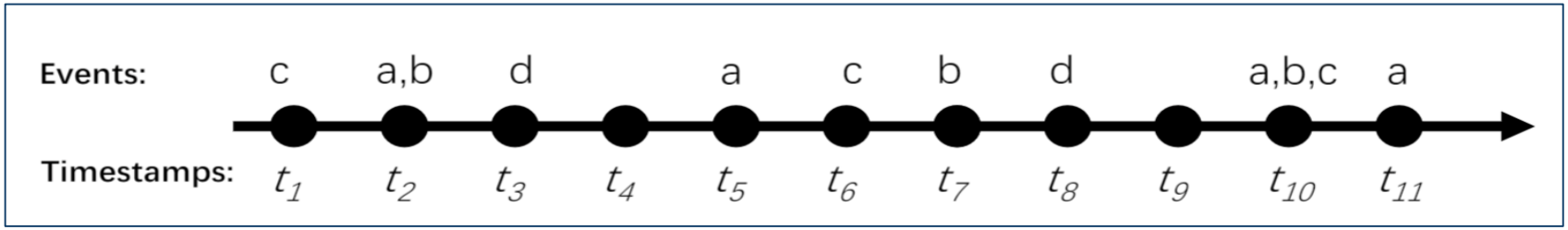
Philippe Fournier-Viger[1], Yangming Chen[1],
Farid Nouioua[2], Jerry Chun-Wei Lin[3]

1 Harbin Institute of Technology (Shenzhen), China
2 University of Bordj Bou Arreridj, Algeria
3 Western Norway University of Applied Sciences (HVL), Norway

1. Introduction

Frequent Episode Mining

Input: a sequence of events with timestamps



Output:

All frequent episode
($occurrence\ count \geq minSup$)

Episode Types:

- parallel episodes,
- serial episodes,
- complex episodes…

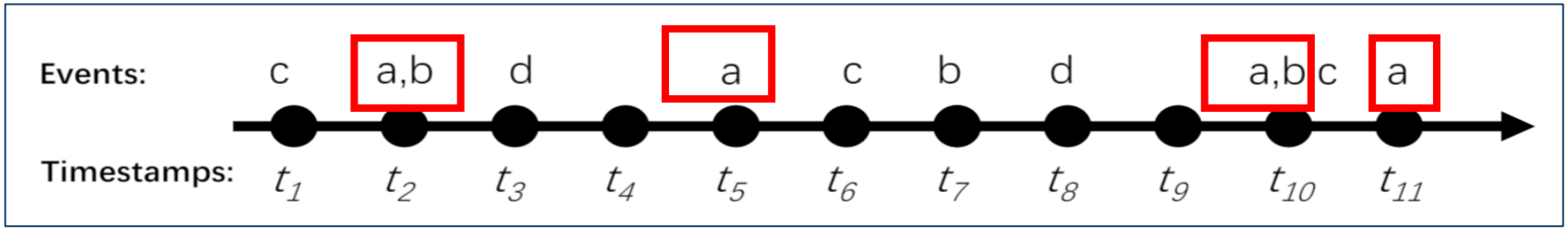
Algorithms:

WINEPI, MINEPI, EMMA, MINEPI+, …

1. Introduction

Frequent Episode Mining

Input: a sequence of events with timestamps



Output:

All frequent episode
($occurrence\ count \geq minSup$)

Example:

episode **<{a,b},{a}>**

Episode Types:

- parallel episodes,
- serial episodes,
- complex episodes…

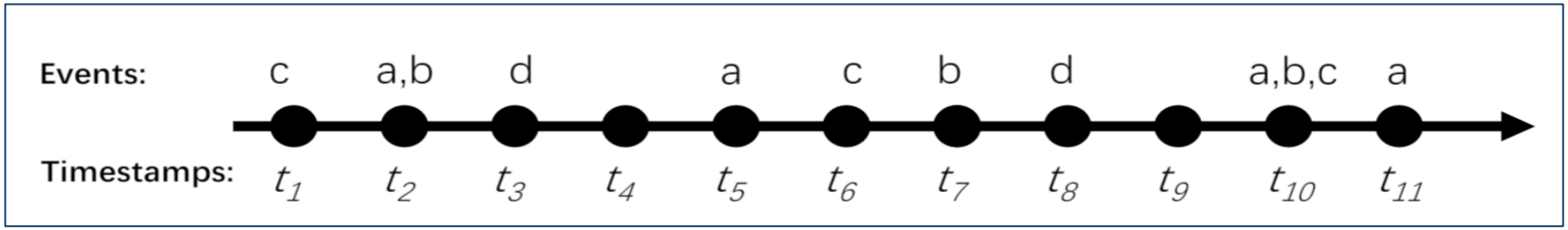
Algorithms:

WINEPI, MINEPI, EMMA, MINEPI+, …

1. Introduction

Episode Rule Mining

Input: a sequence of events with timestamps



Output:

Each episode rule : $X \rightarrow Y$ such that:

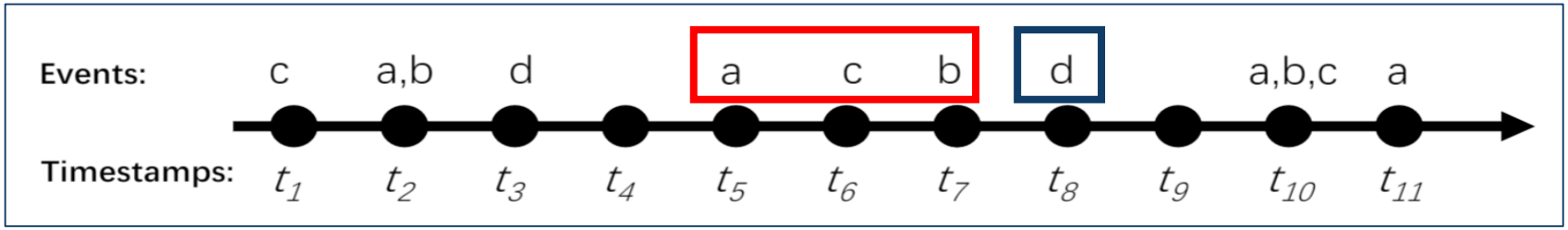
$$(\text{Supp}(X) \geq minSup)$$

$$(\text{conf}(X \rightarrow Y) = \frac{\text{Supp}(X \cap Y)}{Supp(X)} \geq confidence)$$

1. Introduction

Episode Rule Mining

Input: a sequence of events with timestamps



Output:

Each episode rule : $X \rightarrow Y$ such that:

$$(\text{Supp}(X) \geq minSup)$$

$$(\text{conf}(X \rightarrow Y) = \frac{\text{Supp}(X \cap Y)}{Supp(X)} \geq confidence)$$

Example:

$\langle \{a\},\{c\},\{b\} \rangle \rightarrow \langle \{d\} \rangle$

1. Introduction

Related Work

- **Frequent episode rules** (1995, and many papers after)
- **Utility-based episode rules** (ASOC 2017)
  - Discovering episodes rule having a high importance in a sequence of discrete events with weights and quantities
- **Precise-positioning episode rules** (TKDE 2018, ICDE 2017)
  - the elapsed time between the antecedent and the consequent is fixed
- **Distant and essential episode rules** (ESWA 2018)
  - the antecedent is as small as possible in terms of number of events and of occurrence duration
  - consequent is temporally distant from the antecedent

To address these issues

1. Introduction

An Important Limitation

Events within an episode rule must follow a very strict temporal ordering

Example:

$\langle\{a\},\{c\},\{b\}\rangle \rightarrow \langle\{d\}\rangle$

$\langle\{a\},\{b\},\{c\}\rangle \rightarrow \langle\{d\}\rangle$

$\langle\{b\},\{a\},\{c\}\rangle \rightarrow \langle\{d\}\rangle$

$\langle\{b\},\{c\},\{a\}\rangle \rightarrow \langle\{d\}\rangle$

$\langle\{c\},\{b\},\{a\}\rangle \rightarrow \langle\{d\}\rangle$

$\langle\{c\},\{a\},\{b\}\rangle \rightarrow \langle\{d\}\rangle$

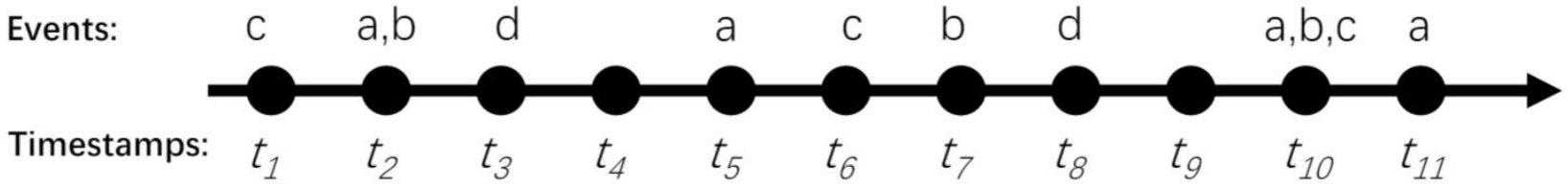
But for applications such as analyzing transactions in a retail store, all these rules may be the same:

$\{a, b, c\} \rightarrow \{d\}$

2. Definition

Event set

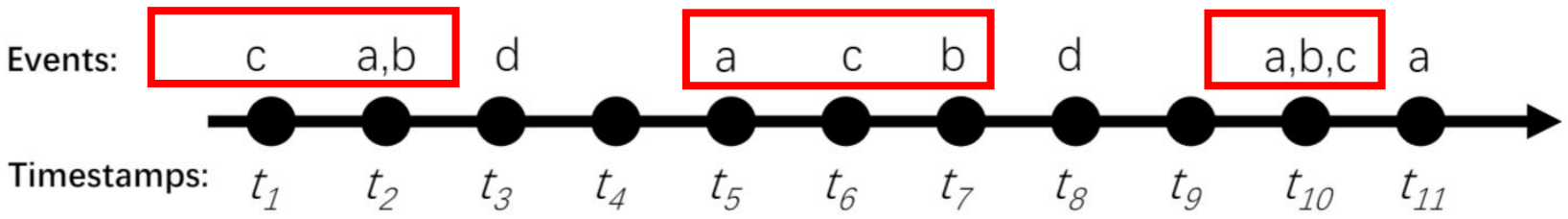
An **event set** is a set of events. e.g. **{a,b,c}**



2. Definition

Event set

An **event set** is a set of events. e.g. **{a,b,c}**



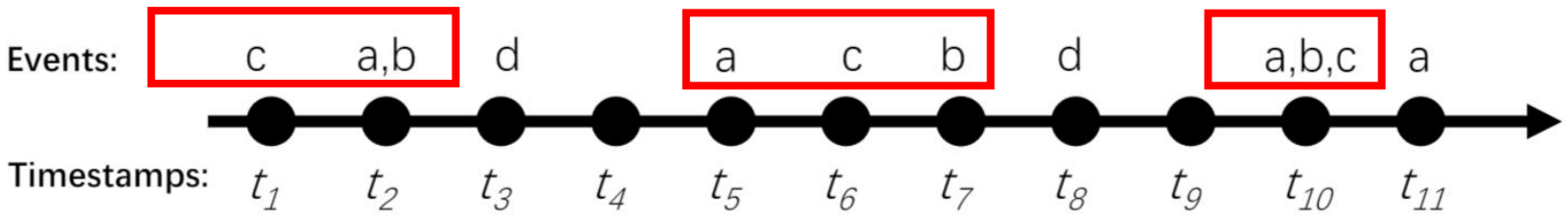
The **non-redundant occurrences** of **{a,b,c}** are:

Occ({a, b, c}) ={[t_1, t_2],[t_5, t_7],[t_{10}, t_{10}]}

2. Definition

Event set

An **event set** is a set of events. e.g. **{a,b,c}**



The **non-redundant occurrences** of **{a,b,c}** are:

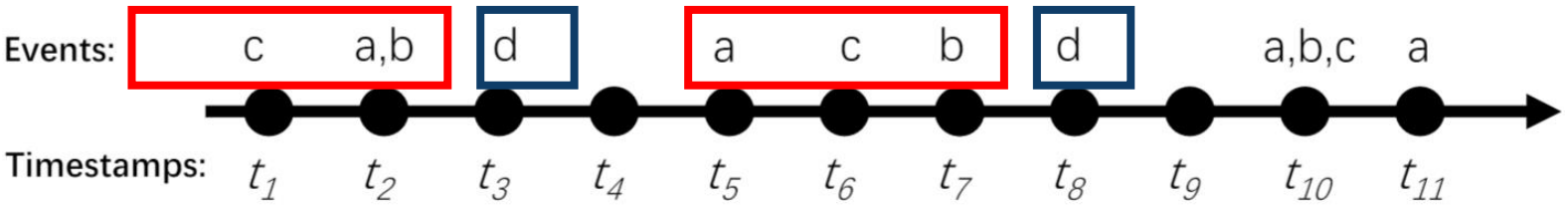
Occ({a, b, c}) ={[t_1, t_2],[t_5, t_7],[t_{10}, t_{10}]}

Note that a time interval such as [t_{10}, t_{11}] is **redundant** to Occ({a, b, c}) because [t_{10}, t_{11}] is overlapping with [t_{10}, t_{10}].

2. Definition

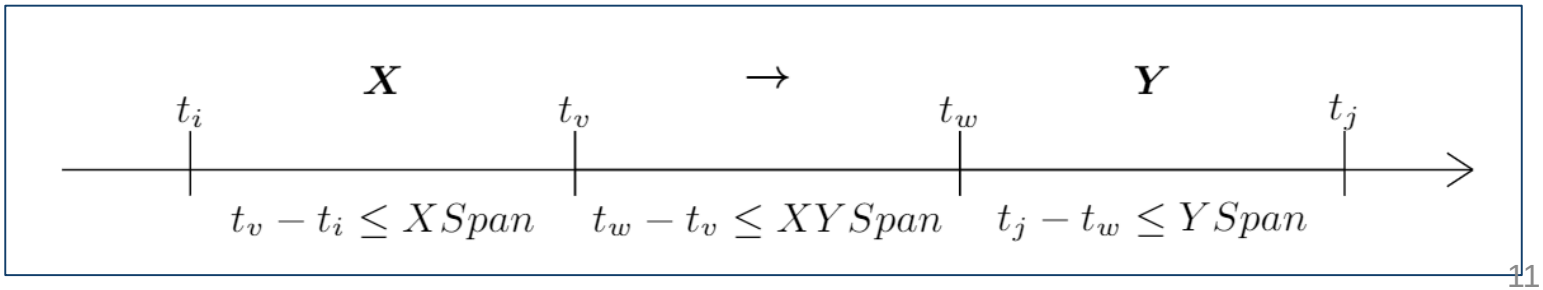
Partially-Ordered Episode Rule（POER）

A new type of episode rules that loosen the ordering constraint.

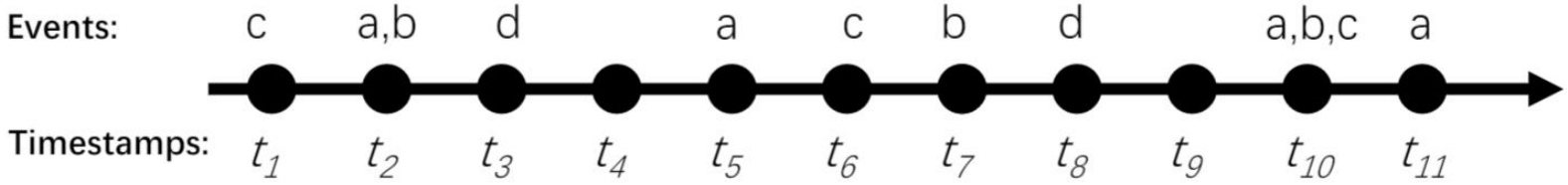


Example: {a, b, c} → {d}
where occ(R) = {[t1, t3],[t5, t8]}

minsup= 3, minconf= 0.6, XSpan= 3, XYSpan= 1, YSpan= 1



3. The POERM algorithm



STEP 1: Find the frequent rule antecedents

minsup= 3

↓

Frequent 1-event sets : {a} , {b}, {c}

↓

extends i-event sets into i+1-event sets

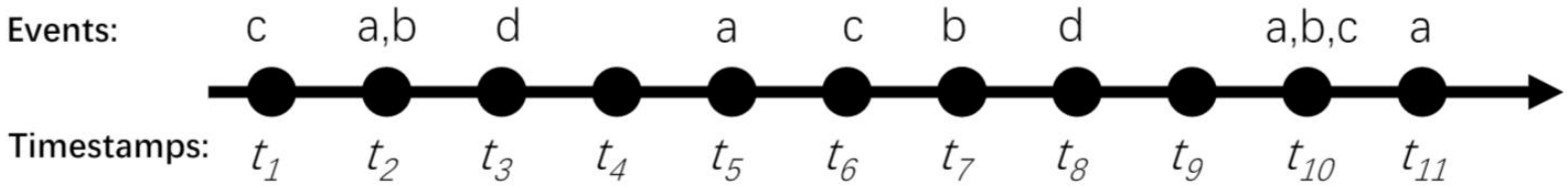
↓

For each occurrences [pos.start, pos.end] of i-event sets X

↓

Search for [pos.end−XSpan+1, pos.start),[pos.end+1, pos.start+XSpan), and [pos.start, pos.end] to extend it

3. The POERM algorithm



STEP 2: Find consequents to make rules

Frequent event sets : <{a} , {b}, {c},{a, b}, {a, c}, {b, c}, {a, b, c}>

↓

Frequent 1-event consequent rule

↓

Extends i-event consequent rule into i+1-event consequent

3 A Baseline Algorithm called POERM-ALL

Find all possible antecedents

Find all possible consequents

Concatenate antecedents with consequents to get POER candidates

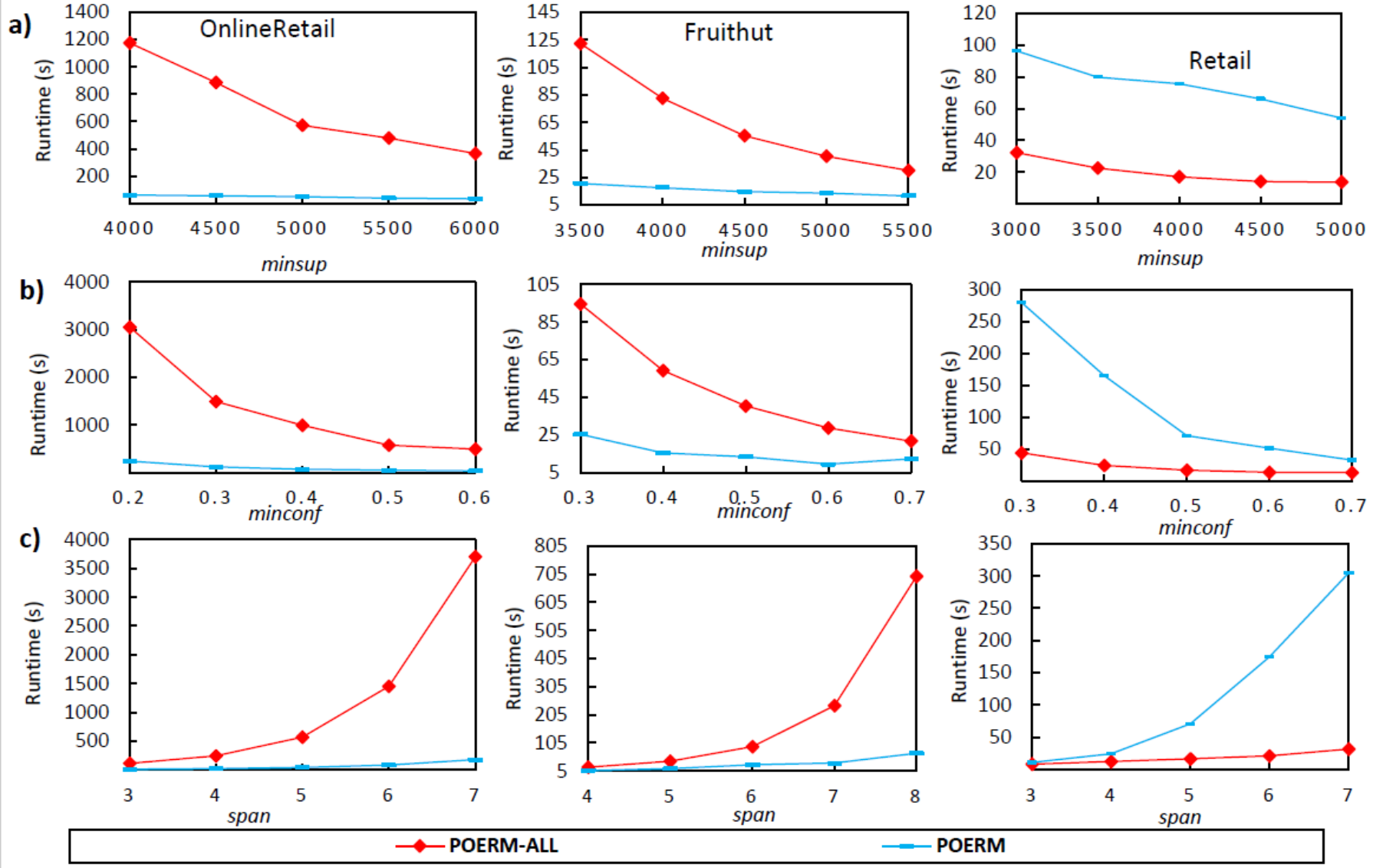
Filter to obtain the POER rules

4 Experimental Evaluation

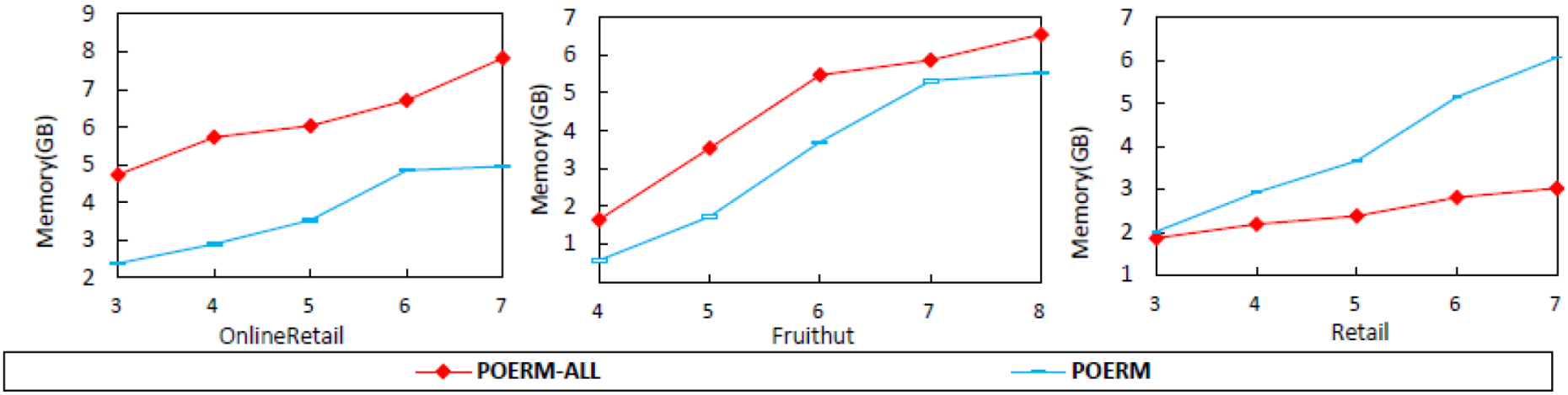
Dataset	# Timestamps	# Events	Average event set size
OnlineRetail	541,909	2,603	4.37
Fruithut	181,970	1,265	3.58
retail	88,162	16,470	10,30

Default Values	minsup	minconf	Span
OnlineRetail	5000	0.5	5
Fruithut	5000	0.5	5
retail	4000	0.5	5

4. Experimental Evaluation (runtime)



4. Experimental Evaluation (memory)



- On OnlineRetail and FruitHut, POERM consumes less runtime and memory then POERM-ALL.
- But on sparse datasets such as Retail, POERM can’t perform better than POERM-ALL.

4. Experimental Evaluation (patterns in real data)

These patterns were discovered in the Fruithut dataset:

Rule	occ(X → Y)	occ(X)
Cucumber Lebanese, Field Tomatoes→Banana Cavendish	4910	6152
Capsicum red, Field Tomatoes→Banana Cavendish	5033	6352
Broccoli, Capsicum red→Field Tomatoes	2343	4043
Nectarine White→Watermelon seedless	2498	5687
Garlic loose, Field Tomatoes→Capsicum red	1752	4409
Cucumber Lebanese, Capsicum red →Eggplant	1236	4098

5. Conclusion

- **Contributions:**
  - A novel type of rules called **partially-ordered episode rules**
  - **An efficient algorithm** named **POERM** (Partially-Ordered Episode Rule Miner) to find these rules
  - **Experimental evaluation** on several benchmark dataset shows that POERM has excellent performance and find interesting rules.
- **Future work:**
  - POERM for streaming data and big data.
  - Considering more complex data types

- **Open-source code, datasets:**
  - SPMF data mining library (over 200 algorithms)

Thanks for listening!

The timestamps that are searched for a candidate ruleX→Y