TKE: Mining Top-K Frequent Episodes

Philippe Fournier-Viger[1], Yanjun Yang[1], Peng Yang[1], Jerry Chun-Wei Lin[2], and Unil Yun[3]

1 Harbin Institute of Technology (Shenzhen), China

2 Western Norway University of Applied Sciences (HVL), Norway

3 Sejong University, Republic of Korea

Outline

- Introduction
- Definition
- TKE
- Experiment
- Conclusion

INTRODUCTION

Introduction

- What is *Frequent Episode Mining* (*FEM*)?
  - It is a popular data mining task for analyzing a **sequence of events**. It consists of identifying all **episodes** (subsequences of events) that appear at least *minsup* times.

- However, a major problem of traditional episode mining algorithms is that setting the *minsup* parameter is not intuitive.

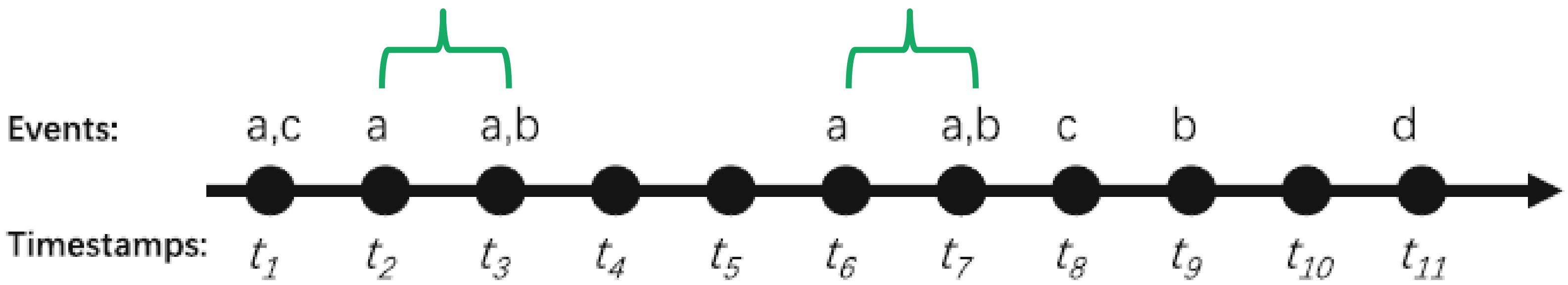
A Major Problem of Frequent Episode Mining

- Selecting an appropriate *minsup* value to find just enough episodes is difficult
  - If *minsup* is set too low
    - Algorithms can have long execution times and find too many episodes.
  - If *minsup* is set too high
    - Algorithms may find few patterns, and hence miss important information.
- The problem is redefined as **top-k frequent episode mining**.

Mining Top-K Frequent Episodes

- An algorithm named **TKE** (**T**op-**K** **E**pisode mining)
  - To find the *k* most frequent episodes
  - Use a internal *minsup* threshold that is initially set to 1
  - Apply a concept of dynamic search, which increases the threshold as quick as possible to reduce the search space

Problem Definition



$winlen = 2$

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

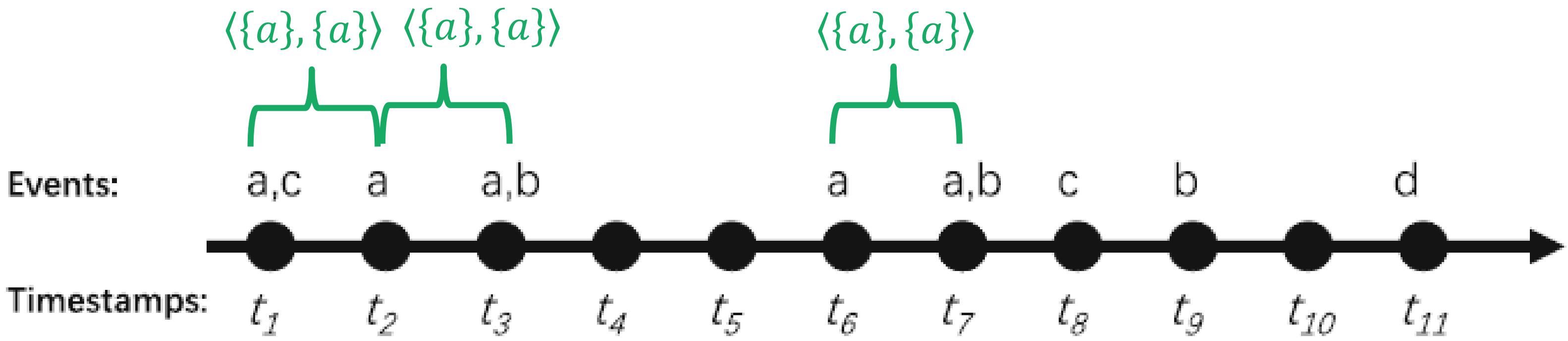
the **head frequency support**, $\sup(\langle\{a\},\{a,b\}\rangle) = 2$

Frequent Episode

- If an episode α having a support that is no less than a user-specified minimum support threshold *minsup*, we call the episode **frequent episode**.

$$sup(\alpha) \geq minsup \rightarrow \alpha \ is \ a \ frequent \ episode$$

Top-K Frequent Episode Mining



$winlen = 2, K = 3$

frequent episodes: $\langle\{a\},\{a\}\rangle, \langle\{a\}\rangle$, and $\langle\{b\}\rangle$

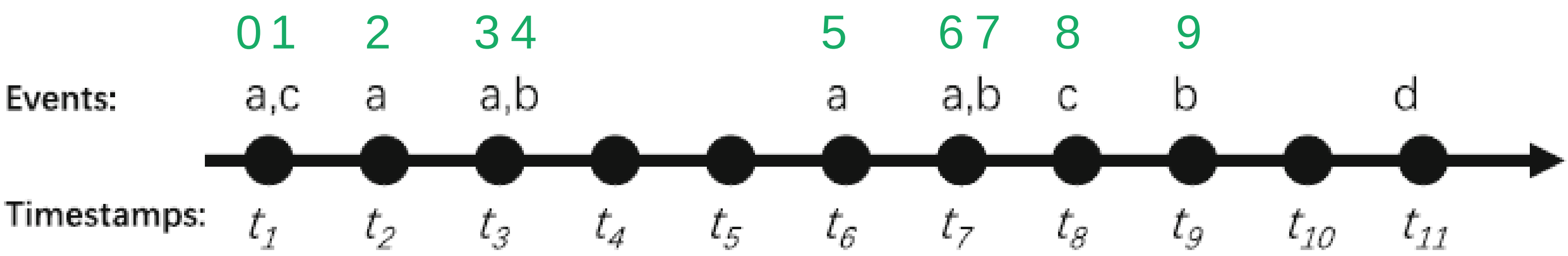
their support value: $3, 5, 3$

THE TKE ALGORITHM

Step 1: Finding the Top-k Events

- First set internal *minsup* = 1
- An optimization: Single Episode Increase (SEI)
  - Set internal *minsup* to the support of the k-th most frequent event.
  - Remove all events having a support less than *minsup*
- Create a *location list* for each frequent event
- Top-k events
  - $\langle\{a\}\rangle, \langle\{b\}\rangle, and\ \langle\{c\}\rangle$, with support values of 5, 3, 2, respectively, and *minsup* = 2

Location List



- $winlen = 2, k = 3, internal\ minsup = 2$
- $locList(a) = \{0, 2, 3, 5, 6\}, locList(b) = \{4, 7, 9\}, locList(c) = \{1, 8\}$
- $sup(e) = |locList(e)|$

Step 2: Finding the Top-k Parallel Episodes

- Combine frequent events found in Step 1
  - Do parallel extension
  - Location lists of new parallel episodes are generated
  - Set internal *minsup* to the support of the k-th most frequent episode.

- Top-k parallel episodes
  - $\langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{c\}\rangle$, and $\langle\{a, b\}\rangle$, with support values of 5, 3, 2, 2, respectively, and *minsup* = 2

Step 3: Re-encoding the Input Sequence Using Parallel Episodes

- A unique identifier is given to each top-k parallel episode

- For instance,
  - The IDs #1, #2, #3, and #4 are assigned to the top-k parallel episodes $\langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{c\}\rangle$, and $\langle\{a, b\}\rangle$, respectively
  - $S' =$

$$\left\langle \begin{array}{c} (\{\#1, \#3\}, t_1), (\{\#1\}, t_2), (\{\#1, \#2, \#4\}, t_3), (\{\#1\}, t_6), \\ (\{\#1, \#2, \#4\}, t_7), (\{\#3\}, t_8), (\{\#2\}, t_9) \end{array} \right\rangle$$

Step 4: Finding the Top-k Composite episodes

- Combine top-k parallel episodes
  - Do serial extension
- Bound list
  - $boundList(\langle\{a\},\{a\}\rangle) = \{[t_1, t_2], [t_2, t_3], [t_6, t_7]\}$
  - $\sup(\langle\{a\},\{a\}\rangle) = |t_1, t_2, t_6| = 3$
- Top-k composite episodes
  - $\langle\{a\}\rangle, \langle\{b\}\rangle, ,$ and $\langle\{a\},\{a\}\rangle$, with support values of 5, 4, and 3, respectively,

Implementation Details

- Adopt priority queues as data structure
- Dynamic search optimization
  - Maintain a priority queue of episodes to generate candidate episodes
  - Always extend the episode that has the highest support
  - The internal *minsup* threshold may be raised more quickly

EXPERIMENT

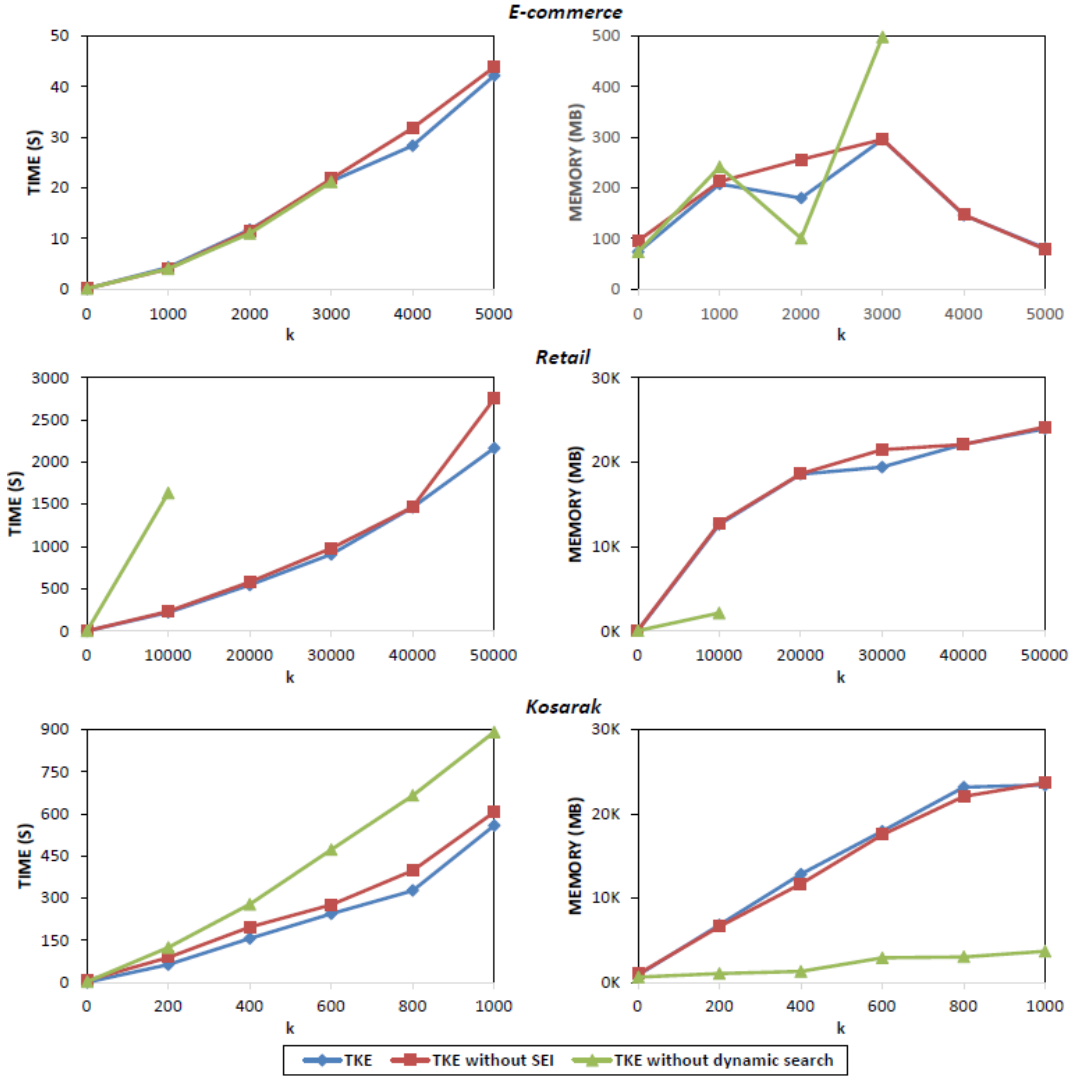
Experiment

- Dataset

Dataset	# Timestamps	# Events	Average event set size
e-commerce	14,975	3,468	11.71
retail	88,162	16,470	10.30
kosarak	990,002	41,270	8.10

- E-commerce and retail are sparse customer transaction datasets
- E-commerce has real timestamps

Influence of k and Optimizations on TKE’s Performance



Performance Comparison with EMMA Set with an Optimal *minsup* Threshold

k	$minsup$	#patterns	TKE runtime (s)	EMMA runtime (s)	TKE memory (MB)	EMMA memory (MB)
1	0.6075	1	3	3	644	239
200	0.3619	159	124	9	1066	788
400	0.3293	210	279	10	1321	2184
600	0.3055	511	473	24	2934	2077
800	0.2862	625	666	29	3038	2496
1000	0.2794	684	891	31	3702	1485

- The runtime and memory of TKE are more than that of EMMA
- Top-k frequent episode mining is more difficult
- Setting *minsup* accurately is a very narrow range of options

CONCLUSION

Conclusion

- Redefined the task of frequent episode mining as top-k frequent episode mining
- A efficient algorithm named TKE for frequent episode mining was proposed
  - Internal *minsup*
  - Dynamic search
- A performance evaluation on real-life data has shown that TKE is efficient

Source code and datasets available in the SPMF open-source data mining library
http://www.philippe-fournier-viger.com/spmf/