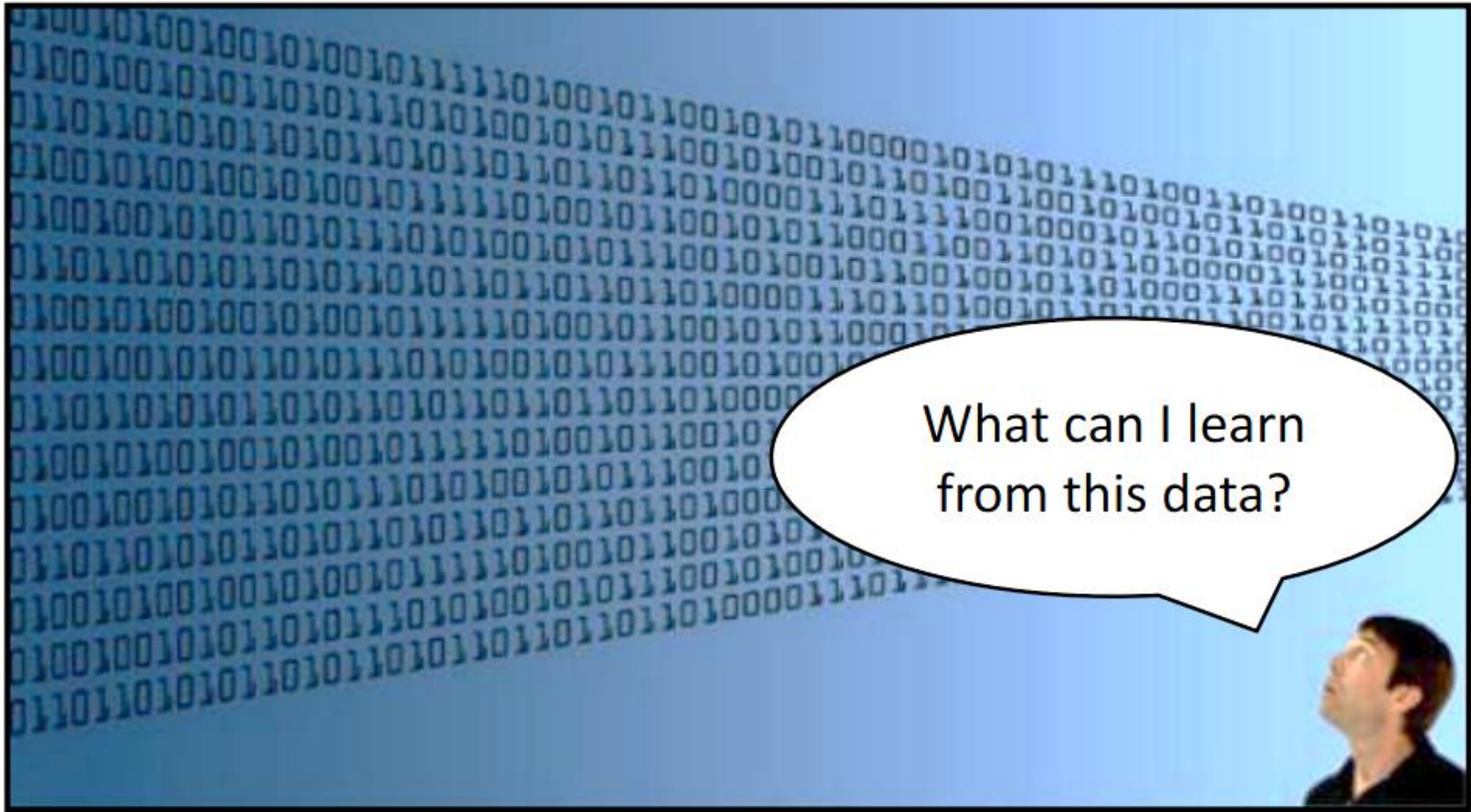


Mining Cost-Effective Patterns

Jiaxuan Li,
Philippe Fournier-Viger
et al.

Fournier-Viger, P., Li, J., Lin, J. C., Chi, T. T., Kiran, R. U. (2020). **Mining Cost-Effective Patterns in Event Logs**. Knowledge-Based Systems (KBS), Elsevier,
DOI: 10.1016/j.knosys.2019.105241

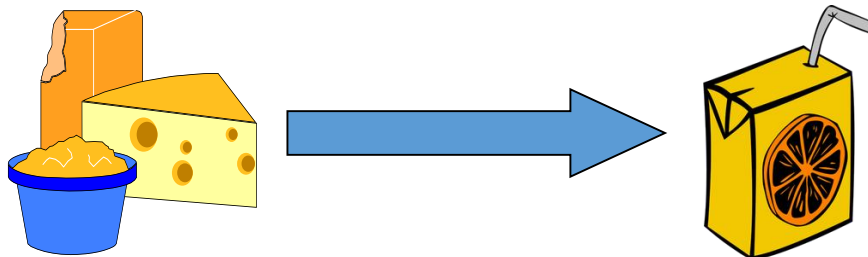
Introduction



- Huge amount of data stored in databases
- A need for algorithms to discover interesting, unexpected and useful **patterns** in data

Discovering patterns

- **Various types of patterns**
 - Itemsets
 - Sequential patterns, sequential rules,
 - Periodic patterns, episodes, subgraphs, etc.
- **Why?**
 - Discover valuable hidden knowledge
 - Interpretable by humans (not a black-box model)



How to find useful patterns?

- There can be **millions of patterns**
- Interestingness measures are used to **select interesting patterns**:
 - support,
 - confidence,
 - utility,
 - periodicity,
 - measures of statistical significance,
 - ...



Frequent Itemset Mining

Finding sets of items that appear frequently in a database

minsup=0,5

Transaction database

Transaction	Items
T1	{ A, B, C, D }
T2	{ A, C, D }
T3	{ D, E }
T4	{ C, D }



Frequent itemsets

Itemset	Support (frequency)
{ A }	50
{ C }	50
{ B }	50
{ C, D }	75
...	...

Useful but does not consider time and other factors such as profit...

High Utility Sequential Pattern Mining

Input

Quantitative sequences with purchase quantities (internal utility)	
sequence 1:	$\langle (a, 3), (b, 3), (c, 1), (b, 4) \rangle$
sequence 2:	$\langle (a, 1), (e, 3) \rangle$
sequence 3:	$\langle (a, 6), (c, 7), (b, 8), (d, 9) \rangle$
sequence 4:	$\langle (b, 3), (c, 1) \rangle$
Unit profits (external utility)	
$a = 5\$$, $b = 1\$$, $c = 2\$$, $d = 1\$$	

a minimum utility threshold (e.g. *minutil* = 30)

Output

All sequences having a *utility* \geq *minutil*)

The sequence $\langle ab \rangle$ is a high utility pattern because:

$$u(\langle ab \rangle) = \underbrace{3 \times 5 + 3 \times 1}_{\text{Sequence 1}} + \underbrace{6 \times 5 + 8 \times 1}_{\text{Sequence 3}} = 56 > \text{minutil}$$

Limitations of high utility pattern mining

- It focused on patterns that have a high utility (importance) but ignores the **cost** to obtain these benefits.
- May find patterns that have **a high utility but a very high cost**
- May miss patterns that have **a low cost but a relatively high utility**

Our proposal

- A novel problem named **Cost-Effective Pattern Mining** that integrates the concept of **utility** with that of **cost**.
- **Goal:** Find cost-efficient patterns that provide utility at a low cost.
- **Cost:** money, time resources consumed, effort

Two problems

Discover **cost-effective patterns (CEPs)** in sequences with cost values and where:

1. the **utility** is **binary** values.



2. the **utility** is **numeric** values.



Problem 1

Mining cost-effective patterns in sequences with cost and binary utility



Sequences with cost and **binary** utility

- A **sequence** is an ordered list of events, each having a cost value.
- The **utility** of a sequence is a **binary value** indicating a good or bad outcome.
- **Example:** **medical pathway data**

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	Positive
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	Negative
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	Positive
S ₄	<(a:2)(b:2)(c:1)(f:2)>	Negative

(e.g. **cured** or **died** after
some medical treatments)

Support of a pattern

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	...
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	...
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	...
S ₄	<(a:2)(b:2)(c:1)(f:2)>	...

- The **number of sequences** in which the pattern appears.
- e.g. the **support** of pattern **ab** is **sup(ab) = 2** because it appears in **two sequences** (**S₁** and **S₄**).

Cost of a pattern

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	...
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	...
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	...
S ₄	<(a:2)(b:2)(c:1)(f:2)>	...

- The **sum of the cost values** of the first occurrence of the pattern in each sequence.
- e.g. the **cost of pattern ab** is $c(\mathbf{ab}) = c(\mathbf{ab}, S_1) + c(\mathbf{ab}, S_4)$
 $= 6 + 4 = 10$

Average cost of a pattern

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	...
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	...
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	...
S ₄	<(a:2)(b:2)(c:1)(f:2)>	...

- The **average of the cost values**
- e.g. the **average cost** of **ab** is $ac(ab) = c(ab) / \text{sup}(ab)$
 $= 10 / 2 = 5$

Problem 1: Finding all cost-effective patterns

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	Positive
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	Negative
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	Positive
S ₄	<(a:2)(b:2)(c:1)(f:2)>	Negative

A pattern p is **cost-effective** if:

$$\text{sup}(p) \geq \text{minsup}$$

$$\text{ac}(p) \leq \text{maxcost}$$

And we measure the **correlation** of a pattern p with the desirable outcome:

$$\text{cor}(p) = \frac{\text{ac}(D_p^+) - \text{ac}(D_p^-)}{\text{Std}} \sqrt{\frac{|D_p^+| |D_p^-|}{|D_p^+ \cup D_p^-|}} \in [-1, 1]$$

a positive correlation is desirable

Pattern	support	average cost	correlation
<ac>	3	5.3	0.80

More details...

The **correlation** of a pattern p :

$$cor(p) = \frac{ac(D_p^+) - ac(D_p^-)}{Std} \sqrt{\frac{|D_p^+||D_p^-|}{|D_p^+ \cup D_p^-|}} \quad \text{where, } ac(D_p^+), ac(D_p^-) \text{ denotes pattern } p\text{'s average cost in positive and negative sequences, respectively.}$$

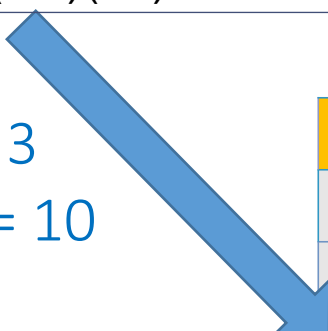
- $ac(D_p^+) - ac(D_p^-)$, indicates the difference in terms of average cost for positive and negative sequences.
- Std , standard deviation of the cost to avoid absolute values.
- $\sqrt{\frac{|D_p^+||D_p^-|}{|D_p^+ \cup D_p^-|}}$, measures distribution difference to indicate patterns' effect on the outcome.
- Correlation values are in the $[-1,1]$ interval.
- The greater positive(negative) the cor measure is, the more a pattern is correlated with a positive (negative) utility.

A full example

Database

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	Positive
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	Negative
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	Positive
S ₄	<(a:2)(b:2)(c:1)(f:2)>	Negative

minsup = 3
maxcost = 10



Cost-effective patterns

Pattern	support	average cost	correlation
a	3	2.7	0.50
b	3	2.3	-0.50
c	4	2.5	0.89
d	3	3.7	0.99
e	3	2.3	0.19
f	3	1.7	0.50
ac	3	5.3	0.80
bc	3	4.7	0.76
cd	3	6.7	0.99

Problem 2

Mining cost-effective patterns in sequences with cost and numeric utility



Sequences with cost and **numeric** utility

- A **sequence** is an ordered list of events, each having a cost value.
- The **utility** of a sequence is a **numeric value** where a higher value indicates a higher benefit.
- **Example:** e-learning data

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	40
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	50
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	60
S ₄	<(a:2)(b:2)(c:1)(f:2)>	70

(e.g. score obtained at an exam)

Utility of a pattern

Sid	<(Event:cost)>	Utility
S1	<(a:4)(b:2)(e:4)(c:4)(d:5)>	40
S2	<(b:3)(c:2)(f:1)(d:1)(e:2)>	50
S3	<(a:2)(f:2)(e:1)(c:3)(d:5)>	60
S4	<(a:2)(b:2)(c:1)(f:2)>	70

The **utility** of a pattern is the sum of the utility of sequences where it appears.

Utility of a pattern p :

$$u(p) = \frac{\sum_{p \subseteq S_s \in S_{ADB}} su(S_s)}{|sup(p)|}$$

$$u(\mathbf{ab}) = \underbrace{(40)}_{\text{Sequence 1}} + \underbrace{(70)}_{\text{Sequence 3}} / 2 = 55$$

Trade-off of a pattern

Sid	<(Event:cost)>	Utility
S1	<(a:4)(b:2)(e:4)(c:4)(d:5)>	40
S2	<(b:3)(c:2)(f:1)(d:1)(e:2)>	50
S3	<(a:2)(f:2)(e:1)(c:3)(d:5)>	60
S4	<(a:2)(b:2)(c:1)(f:2)>	70

It is the ration between the **cost** and **utility** of a pattern p :

Trade-off of a pattern p :

$$tf(p) = \frac{ac(p)}{u(p)} \in (0, +\infty]$$

Lower means more efficient.

$$tf(ab) = 5 / 55 = 0.09$$

$$tf(cd) = 6.7 / 50 = 0.13$$

Thus, pattern (ab) is more efficient than (cd).

Problem 2: Finding all cost-effective patterns

Sid	<(Event:cost)>	Utility
S1	<(a:4)(b:2)(e:4)(c:4)(d:5)>	40
S2	<(b:3)(c:2)(f:1)(d:1)(e:2)>	50
S3	<(a:2)(f:2)(e:1)(c:3)(d:5)>	60
S4	<(a:2)(b:2)(c:1)(f:2)>	70

A pattern p is **cost-effective** if:

$$\text{sup}(p) \geq \text{minsup}$$

$$\text{ac}(p) \leq \text{maxcost}$$

$$u(p) \geq \text{minu}$$



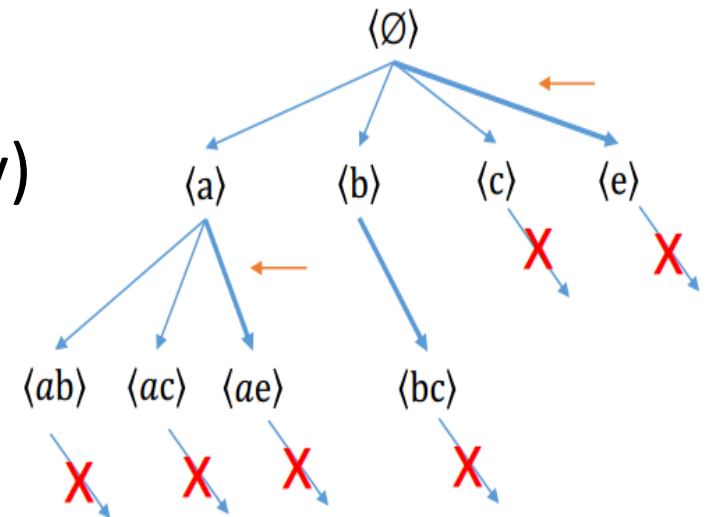
$\text{minsup}=3$ $\text{maxcost}=10$ $\text{minu} = 50$

Utility:50		Utility:53		Utility:55		Utility:56		Utility:60	
pattern	tf	pattern	tf	pattern	tf	pattern	tf	pattern	tf
e	0.05	b	0.04	c	0.05	a	0.05	f	0.03
d	0.07	bc	0.09			ac	0.09		
cd	0.13								

Two algorithms:
CEPB and CEPN

Algorithms

- **CEPB** for Problem 1 (binary utility)
- **CEPN** for Problem 2 (numeric utility)
- Both algorithms explore the search space using a depth-first search.
- Both algorithms adopt a « *pattern-growth* » approach.
- To avoid exploring all possible patterns some **search space reduction techniques** are used →



Reducing the search space using the cost

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	...
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	...
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	...
S ₄	<(a:2)(b:2)(c:1)(f:2)>	...

We propose a **lower-bound** on the **average cost**:

$$AMSC(p) = \frac{1}{minsup} \sum_{i=1,2,..,minsup} c(p, S_i)$$

where $c(p, S_i)$ are sorted in ascending order.

e.g. For $minsup = 2$

$$c(bc, S_4) = 3 \quad c(bc, S_2) = 5 \quad c(bc, S_1) = 6$$

$$AMSC(bc) = (3+5) / 2 = 4$$

Properties of AMSC

$$AMSC(p) = \frac{1}{minsup} \sum_{i=1,2,\dots,minsup} c(p, S_i)$$

Properties of the AMSC:

- I. **Underestimation:** The AMSC of a pattern p is smaller than or equal to its average cost, $AMSC(p) \leq ac(p)$
- II. **Monotonicity:** Let p_x and p_y be two patterns,
If $p_x \subset p_y$ then $AMSC(p_x) \leq AMSC(p_y)$
- III. **Pruning:** For a pattern p , if $AMSC(p) > maxcost$, then pattern p can be eliminated as well as its super-sequences.

Reducing the search space using the utility

We use an upper bound on the utility of a pattern p in a numeric DB:

$$upperu = \frac{1}{minsup} \sum_{i=1,2,\dots,n} u(p, S_i)$$

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	40
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	50
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	60
S ₄	<(a:2)(b:2)(c:1)(f:2)>	70

e.g. $minsup = 2$

$$u(abc, S_1) = 40 \quad u(abc, S_4) = 70$$

$$upperu(p) = \frac{1}{2} (40 + 70) = 55$$

Properties of *upperu*:

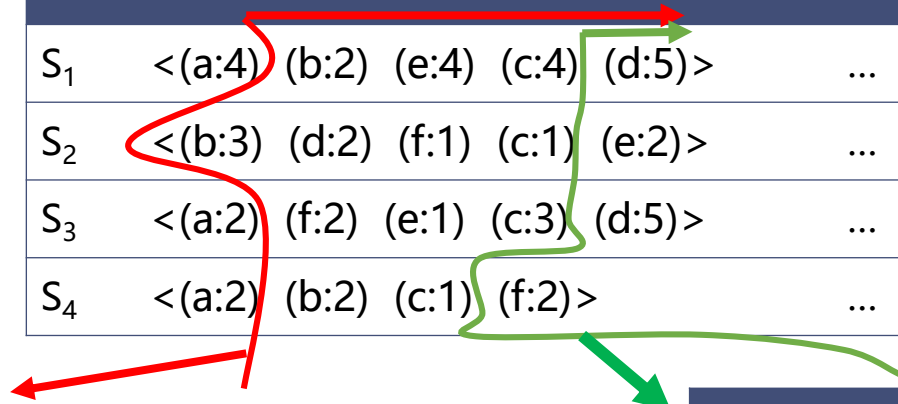
$$upperu = \frac{1}{minsup} \sum_{i=1,2,\dots,n} u(p, S_i)$$

- I. **Overestimation:** The *upperu* of a pattern p is greater than or equal to its cost, $upperu(p) \geq u(p)$
- II. **Anti-monotonicity:** Let p_x and p_y be two patterns,
If $p_x \subset p_y$ then $upperu(p_x) \geq upperu(p_y)$
- III. **Pruning:** For a pattern p , if $upperu(p) < minutility$, then pattern p can be eliminated as well as its supersets.

The CorCEPB and CEPN algorithm

Pattern-growth: when considering a pattern p , the algorithms project the database by the prefix p . Then, use this reduced database to search for larger patterns that extend p .

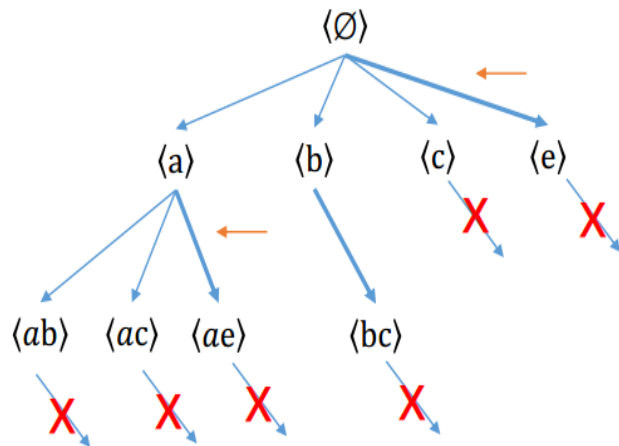
Sid	<(Event:cost)>	Utility
S ₁	<(a:4) (b:2) (e:4) (c:4) (d:5)>	...
S ₂	<(b:3) (d:2) (f:1) (c:1) (e:2)>	...
S ₃	<(a:2) (f:2) (e:1) (c:3) (d:5)>	...
S ₄	<(a:2) (b:2) (c:1) (f:2)>	...



Sid	<(Event:cost)>	Utility
S ₁	<(b:2)(e:4)(c:4)(d:5)>	...
S ₂	<(b:3)(d:2)(f:1)(c:1)(e:2)>	...
S ₃	<(f:2)(e:1)(c:3)(d:5)>	...
S ₄	<(b:2)(c:1)(f:2)>	...

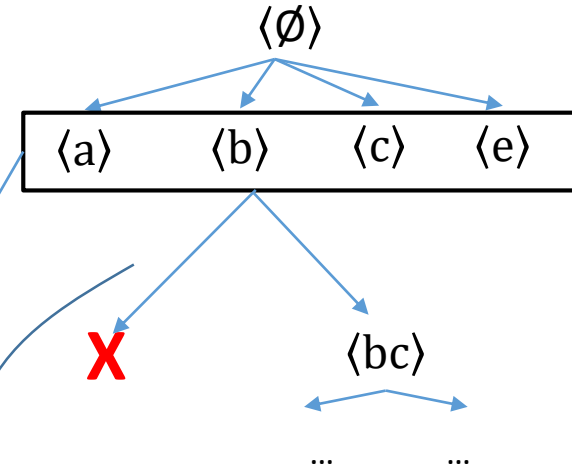
Sid	<(Event:cost)>	Utility
S ₁	<(d:5)>	...
S ₂	<(e:2)>	...
S ₃	<(d:5)>	...
S ₄	<(f:2)>	...

Algorithms



P	sup	ac	$occup$	cor / tf
a	3	2.7	0.22	0.5/
b	3	2.3	0.22	-0.5/
...

P	$sup \wedge AMSC \wedge uo \wedge upperu(case3 \text{ only})$
a	3 2.67 0.11 56.7
b	3 2.33 0.11 53.3
...



$$\begin{aligned}
 &sup(p) \geq \textit{minsup} \quad \wedge \\
 &AMSC(p) \leq \textit{maxcost} \quad \wedge \\
 &upperu(p) \geq \textit{minutility}
 \end{aligned}$$

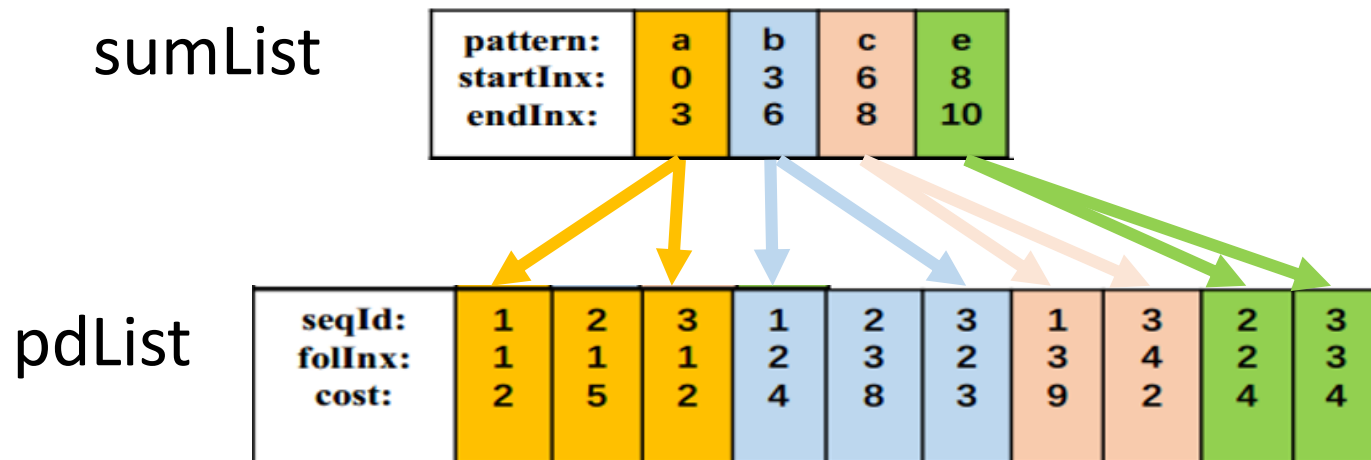
Projected Database Buffer

- **Summary List (sumList)**

sumList is a list of summaries $sumList = \{s_0, s_1, \dots, s_n\}$, each summary is a triple of the form $\{pattern, startInx, endInx\}$ indicating a pattern used to perform a database projection, and two positive integers $startInx$ and $endInx$ indicating that the projected database is stored from the $startInx^{th}$ record to the $(endInx - 1)^{th}$ record of the pdList structure.

- **Projected Database List (pdList)**

pdList is an array of elements $pdList = \{e_0, e_1, \dots, e_n\}$, each element is a triple $\{seqId, folInx, cost\}$ storing the identifier $seqId$ of a projected sequence, an integer $startIndex$ indicating the position in the original sequence where the projected sequence starts, and the pattern's cost in that sequence.



	↓	←	↓							
pattern:	a	b	c	e						
startIdx:	0	3	6	8						
endIdx:	3	6	8	10						
seqId:	1	2	3	1	2	3	1	3	2	3
folIdx:	1	1	1	2	3	2	3	4	2	3
cost:	2	5	2	4	8	3	9	2	4	4

Figure 3: Projected Database Buffer containing the projected databases of $\langle a \rangle$, $\langle b \rangle$, $\langle c \rangle$ and $\langle e \rangle$

	↓	↓								
pattern:	a	b	bc	e						
startIdx:	0	3	6	8						
endIdx:	3	6	8	10						
seqId:	1	2	3	1	2	3	1	3	2	3
folIdx:	1	1	1	2	3	2	3	4	2	3
cost:	2	5	2	4	8	3	13	5	4	4

Figure 5: Accessing the next pattern $\langle b \rangle$ in $sumList[1]$, and then inserting $PD_{\langle bc \rangle}$ in $pdList[6, 8)$

	↓	↓								
pattern:	a	ab	ac	ae						
startIdx:	0	3	6	8						
endIdx:	3	6	8	10						
seqId:	1	2	3	1	2	3	1	3	2	3
folIdx:	1	1	1	2	3	2	3	4	2	2
cost:	2	5	2	6	13	5	11	4	9	6

Figure 7: Accessing the next pattern $\langle a \rangle$ in $sumList[0]$, and then inserting $PD_{\langle ab \rangle}$, $PD_{\langle ac \rangle}$ and $PD_{\langle ae \rangle}$ in $pdList[3, 6)$, $pdList[6, 8)$, $pdList[8, 10)$, respectively

	↓	←	↓							
pattern:	a	b	c	e						
startIdx:	0	3	6	8						
endIdx:	3	6	8	10						
seqId:	1	2	3	1	2	3	1	3	2	3
folIdx:	1	1	1	2	3	2	3	4	2	3
cost:	2	5	2	4	8	3	9	2	4	4

Figure 4: Accessing the next pattern $\langle c \rangle$ in $sumList[2]$

			↓	↓						
pattern:	a	b	bc	e						
startIdx:	0	3	6	8						
endIdx:	3	6	8	10						
seqId:	1	2	3	1	2	3	1	3	2	3
folIdx:	1	1	1	2	3	2	3	4	2	3
cost:	2	5	2	4	8	3	13	5	4	4

Figure 6: Accessing the next pattern $\langle bc \rangle$ in $sumList[2]$

		↓		↓						
pattern:	a	ab	ac	ae						
startIdx:	0	3	5	7						
endIdx:	3	5	7	9						
seqId:	1	2	3	1	2	3	1	3	2	3
folIdx:	1	1	1	2	3	2	3	4	2	3
cost:	2	5	2	6	18	5	11	4	9	6

Figure 8: Accessing patterns $\langle ae \rangle$, $\langle ac \rangle$ and $\langle ab \rangle$ from $sumList[3]$ to $sumList[1]$

Experiments-Runtime of corCEPB

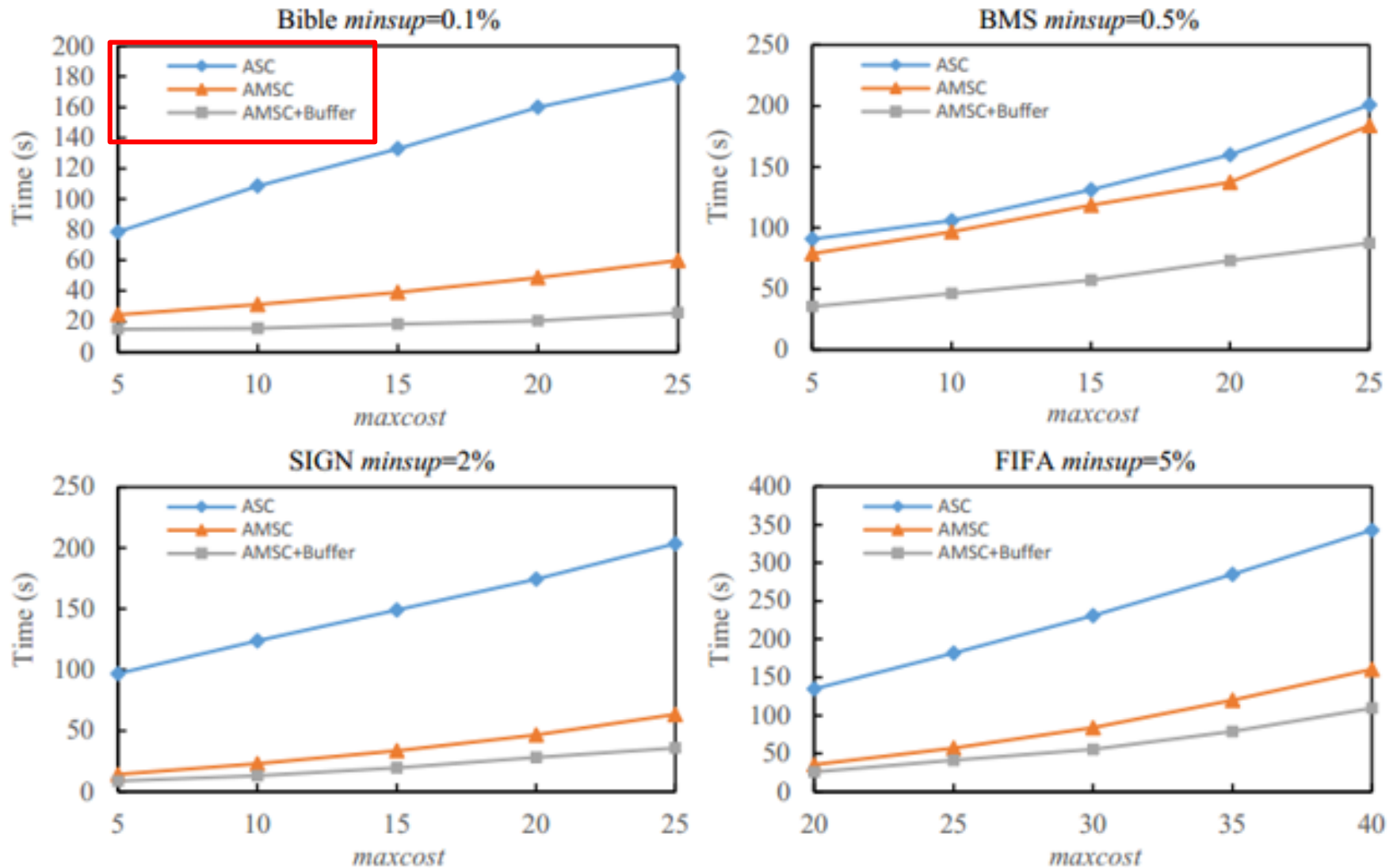


Figure 3-3: Runtime of corCEPB when increasing the *maxcost* threshold.

Experiments - Runtime of CEPN

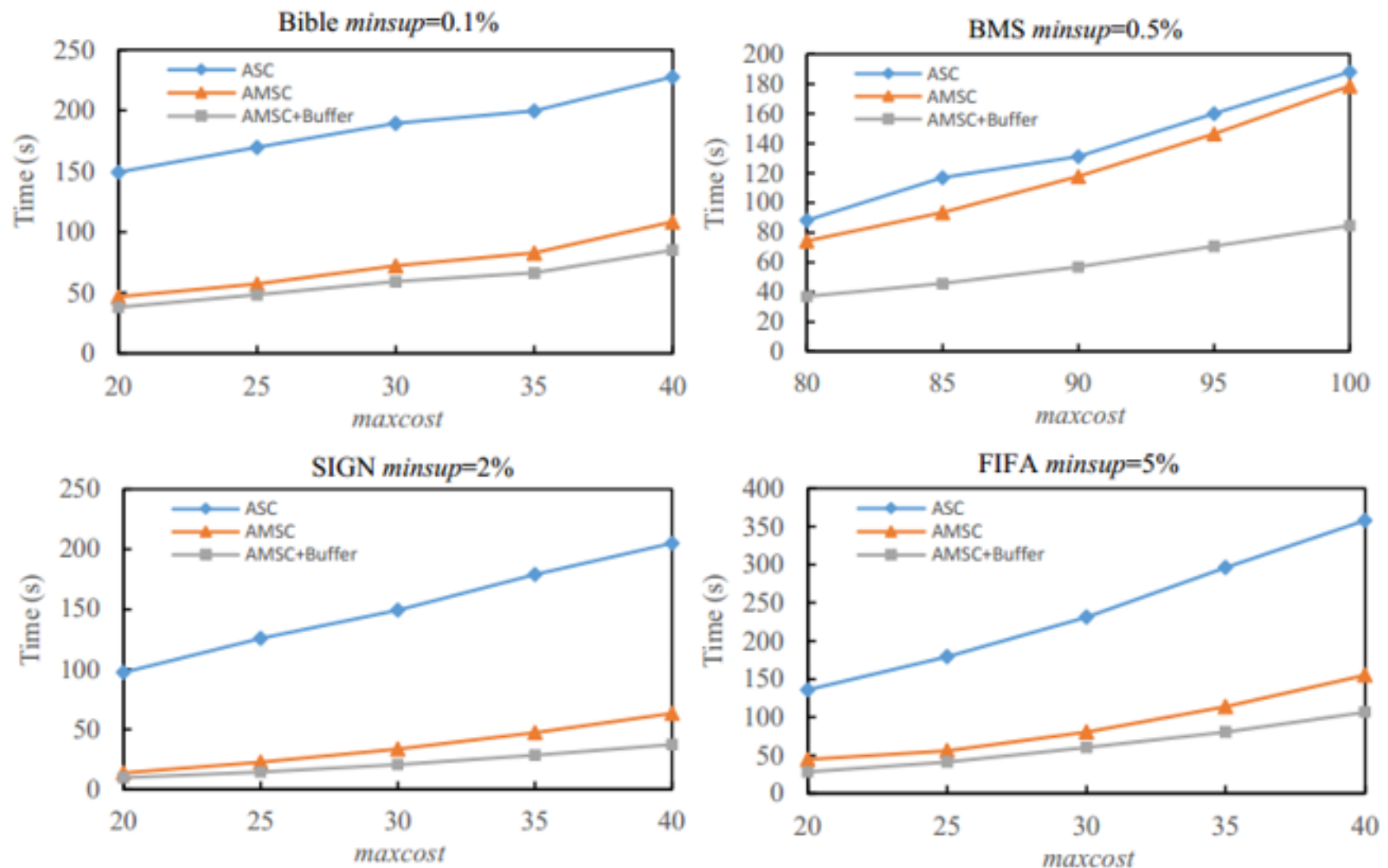


Figure 3-4: Runtime of CEPN when increasing the *maxcost* threshold.

Experiments- Memory usage(1)

Table 11: Memory comparison of CEPB, corCEPB and CEPN on the Bible dataset

<i>maxcost</i>	30		35		40		45		50	
Algorithm	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf
CEPB	2139	2102	4179	3005	4189	3019	4194	3030	4193	3038
corCEPB	596	174	847	307	1123	943	2150	1061	3052	2109
CEPN	1964	592	4152	1122	4151	1188	5377	2156	5384	3889

Table 12: Memory comparison of CEPB, corCEPB and CEPN on the BMS dataset

<i>maxcost</i>	90		95		100		105		110	
Algorithm	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf
CEPB	4228	3289	4223	3289	5616	4112	5604	4112	6028	4112
corCEPB	4170	3291	4203	3292	4171	3292	4211	3289	4216	3289
CEPN	4226	3288	4227	3291	4235	3292	4238	3292	4241	3292

Experiments- Memory usage (2)

Table 13: Memory comparison of CEPB, corCEPB and CEPN on the FIFA dataset

<i>maxcost</i>	5		10		15		20		25	
Algorithm	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf
CEPB	194	190	262	191	262	191	263	191	262	191
corCEPB	194	190	262	193	281	193	263	193	264	193
CEPN	194	190	261	193	281	193	263	193	264	193

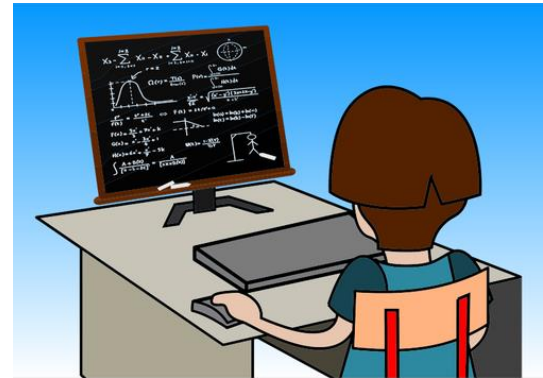
Table 14: Memory comparison of CEPB, corCEPB and CEPN on the SIGN dataset

<i>maxcost</i>	10		15		20		25		30	
Algorithm	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf	ASC	AMSC+Buf
CEPB	2139	857	4185	2034	4188	2170	4203	2161	4205	3387
corCEPB	605	132	1122	293	2142	1116	3071	2124	4227	3357
CEPN	599	217	1074	445	1204	557	2174	855	4244	3243

Case study 1: binary e-learning session sequence

Database

- 115 students
- A **sequence** is a series of learning sessions using the Deeds e-learning system.
- Six sessions: e1, e2... e6
- **Cost**: time to complete a session.
- **Utility**: to *pass* or *fail* the final exam.



Data obtained from

<https://archive.ics.uci.edu/ml/datasets/Data+for+Software+Engineering+Teamwork+Assessment+in+Education+Setting> by Vahdat

Some patterns found

Pattern	Correlation	Average Cost	Support	
$\langle e_1, e_6 \rangle$	0.210	250.2	39	Positive correlation
$\langle e_1, e_2, e_5, e_6 \rangle$	0.209	485.7	34	
$\langle e_2, e_6 \rangle$	0.208	298.4	41	
$\langle e_1, e_2, e_6 \rangle$	0.204	391.9	36	
$\langle e_1, e_5, e_6 \rangle$	0.194	344.3	37	
$\langle e_6 \rangle$	0.193	157.2	50	
$\langle e_1, e_4 \rangle$	-0.004	169.1	41	Correlation ~ 0
$\langle e_1, e_5 \rangle$	0.002	186.0	41	
$\langle e_2, e_3 \rangle$	0.001	284.1	40	
$\langle e_3, e_4, e_5, e_6 \rangle$	0.001	469.5	40	
$\langle e_1, e_4, e_5 \rangle$	0.003	263.2	38	
$\langle e_1, e_2, e_4 \rangle$	-0.003	311.5	36	
$\langle e_2, e_3, e_4 \rangle$	-0.005	358.2	38	
$\langle e_5 \rangle$	-0.147	96.3	53	negative correlation
$\langle e_4, e_5 \rangle$	-0.109	171.0	49	
$\langle e_1, e_3 \rangle$	-0.099	234.6	37	
$\langle e_1, e_3, e_4 \rangle$	-0.081	311.2	35	

Case study 2: numeric e-learning activity sequence

- A **sequence** is the learning activities of a learning session.
- **Cost**: the time to complete an event.
- **Utility**: is the score obtained at the final exam.

minsup = 0.1 and maxcost = 100

- For **Session 6**, the average score is 14.
- The most efficient pattern to obtain this score is (DeedsEs 6 2), which has a trade-off of 0.63.
- For **Session 5**, to obtain the average score of 6 the most efficient pattern is (Study Es 5 2), having a trade-off of 1.35.
- For **Session 4**, the average score of 14 is obtained with the pattern (Study Es 4 2) having a trade-off of 0.71

Some patterns found

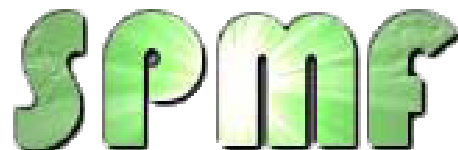
Utility	Pattern	trade-off	Average Cost	Support
1	$\langle Study_Es_6_1, Study_Es_6_1, Study_Es_6_1 \rangle$	48.0	57.6	5
2	$\langle Study_Es_6_1, Study_Es_6_1, Study_Es_6_3 \rangle$	15.0	33.0	5
4	$\langle Study_Es_6_1, Study_Es_6_2, Study_Es_6_2 \rangle$	7.0	32.8	6
5	$\langle Study_Es_6_1, Study_Es_6_1 \rangle$	5.1	27.6	9
6	$\langle Study_Es_6_1, Study_Es_6_1, Deeds_Es_6_1 \rangle$	6.0	40.5	6
7	$\langle Study_Es_6_2, Study_Es_6_2 \rangle$	2.9	20.7	11
8	$\langle Study_Es_6_2, Study_Es_6_2, Deeds_Es_6_2 \rangle$	3.6	31.3	6
9	$\langle Study_Es_6_1 \rangle$	1.2	11.0	20
10	$\langle Study_Es_6_1, Deeds_Es_6_2 \rangle$	2.1	21	13
11	$\langle Study_Es_6_2, Study_Es_6_3 \rangle$	1.56	18.2	16
12	$\langle Study_Es_6_2 \rangle$	0.69	8.9	25
13	$\langle Study_Es_6_3 \rangle$	0.64	8.52	25

Conclusion

We have presented:

- A novel problem of mining **cost-efficient patterns** in sequences with **cost** and **utility** information,
- Two algorithms: **CorCEPB** and **CEPN**
- AMSC-lower bound on cost.
- Upper-bound on utility.
- Optimizations
- A case-study with **e-learning data**.

Source code and **datasets** available in the **SPMF** data mining library.



References

- **Mining Cost-Effective patterns from Sequential Event Log.**

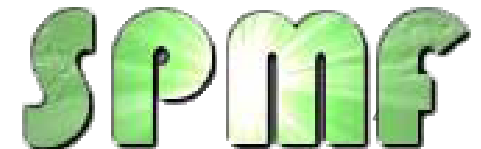
Philippe Fournier-Viger, Jiaxuan Li, Jerry Chun-Wei Lin, Tin Truong Chi, R. Uday Kiran. Knowledge-Based Systems (KBS), Elsevier, 2019. **(SCI,Q1)** Accepted.

- **Discovering and Visualizing Patterns in Utility Sequences.**

Philippe Fournier-Viger, Jiaxuan Li, Jerry Chun-Wei Lin, Tin Truong Chi. Proc. 21st Intern. Conf. on Data Warehousing and Knowledge Discovery (DAWAK, EI), Springer, 2019.

- **Discovering low-cost high utility patterns.**

Jiaxuan Li, Philippe Fournier-Viger, Lin, Jerry Chun-Wei Lin, Tin Truong Chi. 1st International Workshop on Utility-Driven Mining (UDM), in conjunction with the KDD 2018 conference, ACM press, 2018. Oral presentation.





An Open-Source Data Mining Library



[Introduction](#)

[Algorithms](#)

[Download](#)

[Documentation](#)

[Datasets](#)

[FAQ](#)

[License](#)

[Contributors](#)

[Citations](#)

[Performance](#)

[Developers' guide](#)

[Forum](#)

[Mailing-list](#)

[Blog](#)

285994 visitors since
2010-02

Introduction

SPMF is an **open-source data mining mining library** written in **Java**, specialized in **pattern mining**.

It is distributed under the **GPL v3 license**.

It offers implementations of **120 data mining algorithms** for:

- **association rule mining**,
- **itemset mining**,
- **sequential pattern mining**,
- **sequential rule mining**,
- **sequence prediction**,
- **periodic pattern mining**,
- **high-utility pattern mining**,
- **clustering and classification**

The **source code** of each algorithm can be easily integrated in other Java software.

Moreover, SPMF can be used as a **standalone program** with a simple user interface or from the **command line**.

SPMF is fast and lightweight (no dependencies to other libraries).

The current version is **v0.99j** and was released the **16th June 2016**.

<http://www.philippe-fournier-viger.com/spmf/>



Running an algorithm

Choose an algorithm:

CM-SPAM

Choose input file

snake_192_converted.txt

Set output file

test.txt

Choose minsup (%):

0.96

(e.g. 0.5 or 50%)

Min pattern length (optional):

4

(e.g. 1 items)

Max pattern length (optional):

(e.g. 10 items)

Required items (optional):

(e.g. 1,2,3)

Max gap (optional):

(e.g. 1 item)

Show sequence ids? (optional):

(default: false)

Open output file:

☒ using SPMF viewer☐ using text editor

Run algorithm

Algorithm is running...

===== CM-SPAM v0.97 - STATISTICS =====

Total time ~ 135 ms

Frequent sequences count: 447

Max memory (mb) : 39.53382110595703447

minsup 157

Intersection count 2141

Discovered patterns

SPMF - Pattern visualization tool

Patterns:

Pattern	#SUP:
2-1 2-1 2-1 2-1	163
2-1 2-1 2-1 2-1 2-1	160
2-1 2-1 2-1 2-1 2-1 2-1	157
2-1 2-1 2-1 2-1 10-1	162
2-1 2-1 3-1	160
2-1 2-1 2-1 6-1	163
2-1 2-1 2-1 6-1 2-1	163
2-1 2-1 2-1 10-1	163
2-1 2-1 2-1 10-1 2-1	160
2-1 2-1 2-1 10-1 2-1 2-1	158
2-1 2-1 2-1 10-1 3-1	157
2-1 2-1 2-1 10-1 6-1	160
2-1 2-1 2-1 10-1 17-1	161
2-1 2-1 2-1 10-1 17-1 6-1	158
2-1 2-1 2-1 10-1 19-1	158
2-1 2-1 2-1 15-1	161
2-1 2-1 2-1 15-1 2-1	160
2-1 2-1 2-1 17-1	163
2-1 2-1 2-1 17-1 2-1	159
2-1 2-1 2-1 17-1 6-1	161
2-1 2-1 2-1 17-1 6-1 2-1	158
2-1 2-1 2-1 19-1	159
2-1 2-1 6-1 2-1	163
2-1 2-1 6-1 2-1 2-1	158
2-1 2-1 6-1 2-1 6-1	163
2-1 2-1 6-1 2-1 10-1	158
2-1 2-1 6-1 6-1	163
2-1 2-1 6-1 6-1 2-1	160

Number of patterns: 447
File name: test.txt
File size (MB): 0,0152
Last modified: 2016-08-05, 11:08

Search:

Apply filter(s):

Add a filter
Remove selected filter
Remove all filters

Export current view to:
SPMF format

Thank you!

Occupancy of a pattern

Sid	<(Event:cost)>	...
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	...
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	...
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	...
S ₄	<(a:2)(b:2)(c:1)(f:2)>	...

- The **occupancy** of a pattern is the sum of the ratio of events covered by the pattern in each sequence, divided by the support.

- e.g. $occup(ab) = \left(\underbrace{\frac{2}{5}}_{\text{Sequence 1}} + \underbrace{\frac{2}{4}}_{\text{Sequence 4}} \right) / \text{sup}(ab) = 0.45$

Sequence 1 Sequence 4

This measure is used to remove patterns that are short and non-representative of the containing sequences.

Reducing the search space using the occupancy

We use an upper bound on the occupancy of a pattern p :

$$uo(p) = \frac{1}{sup(p)} \cdot \max_{S_1, \dots, S_{sup(p)}} \sum_{i=1}^{sup(p)} \frac{psl[S_i] + ssl[S_i]}{sl[S_i]}$$

where $psl[S_i]$, $ssl[S_i]$ and $sl[S_i]$ is p 's length in S_i , the length of the subsequence after p in S_i , and S_i 's length, respectively.

e.g. $minsup = 2, p = \langle a, b, c \rangle$

$psl[S_1]=psl, [S_4]=3, ssl[S_1]=1, ssl[S_4]=1,$

$sl[S_1]=5, sl[S_4]=4,$

$$uo(p) = \frac{1}{2} \left(\frac{3+1}{5} + \frac{3+1}{4} \right) = 0.9$$

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	...
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	...
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	...
S ₄	<(a:2)(b:2)(c:1)(f:2)>	...

Properties of uo

$$uo(p) = \frac{1}{sup(p)} \cdot \max_{S_1, \dots, S_{sup(p)}} \sum_{i=1}^{sup(p)} \frac{psl[S_i] + ssl[S_i]}{sl[S_i]}$$

- I. **Overestimation:** The uo of a pattern p is greater than or equal to its occupancy, $uo(p) \geq occup(p)$
- II. **Anti-monotonicity:** Let p_x and p_y be two patterns, If $p_x \subset p_y$ then $uo(p_x) \geq uo(p_y)$
- III. **Pruning:** For a pattern p , if $uo(p) < minoccup$, then pattern p can be eliminated as well as its supersets.

Problem 1: Finding all cost-effective patterns

Sid	<(Event:cost)>	Utility
S ₁	<(a:4)(b:2)(e:4)(c:4)(d:5)>	Positive
S ₂	<(b:3)(c:2)(f:1)(d:1)(e:2)>	Negative
S ₃	<(a:2)(f:2)(e:1)(c:3)(d:5)>	Positive
S ₄	<(a:2)(b:2)(c:1)(f:2)>	Negative

A pattern p is **cost-effective** if:

$$\text{sup}(p) \geq \text{minsup}$$

$$\text{ac}(p) \leq \text{maxcost}$$

$$\text{occup}(p) \geq \text{minoccup}$$

And we measure the **correlation** of a pattern p with the desirable outcome:

$$\text{cor}(p) = \frac{\text{ac}(D_p^+) - \text{ac}(D_p^-)}{\text{Std}} \sqrt{\frac{|D_p^+| |D_p^-|}{|D_p^+ \cup D_p^-|}} \in [-1, 1]$$

a positive correlation is desirable

Pattern	support	average cost	correlation
<ac>	3	5.3	0.80

Problem 2: Finding all cost-effective patterns

Sid	<(Event:cost)>	Utility
S1	<(a:4)(b:2)(e:4)(c:4)(d:5)>	40
S2	<(b:3)(c:2)(f:1)(d:1)(e:2)>	50
S3	<(a:2)(f:2)(e:1)(c:3)(d:5)>	60
S4	<(a:2)(b:2)(c:1)(f:2)>	70

A pattern p is **cost-effective** if:

$$\text{sup}(p) \geq \text{minsup}$$

$$\text{ac}(p) \leq \text{maxcost}$$

$$\text{occup}(p) \geq \text{minoccup}$$

$$u(p) \geq \text{minu}$$

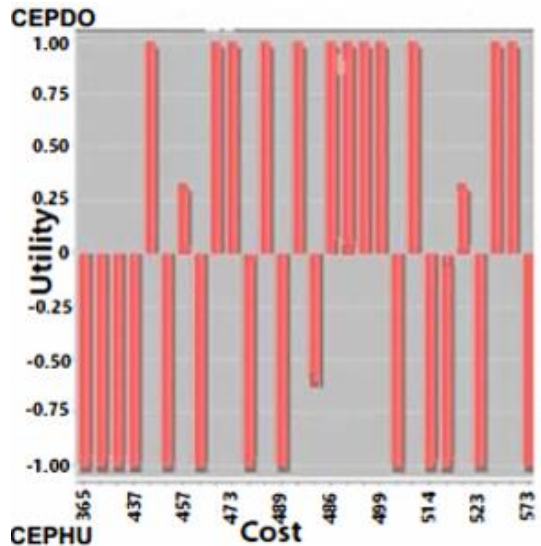


$\text{minsup}=3$ $\text{maxcost}=10$ $\text{minu} = 50$

Utility:50		Utility:53		Utility:55		Utility:56		Utility:60	
pattern	tf	pattern	tf	pattern	tf	pattern	tf	pattern	tf
e	0.05	b	0.04	c	0.05	a	0.05	f	0.03
d	0.07	bc	0.09			ac	0.09		
cd	0.13								

Visualization and Interpretability

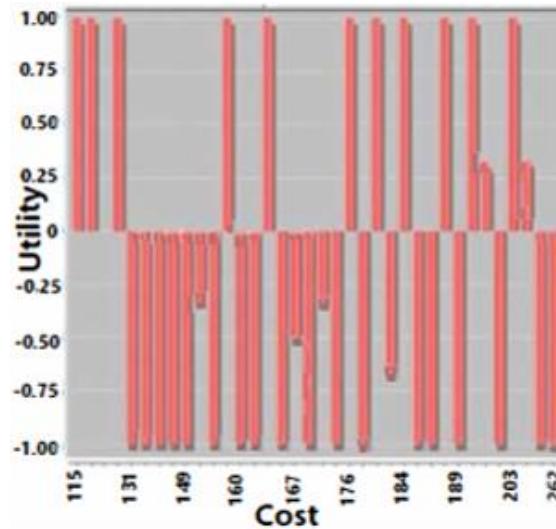
$\langle e_1, e_6 \rangle$



$$\text{cor}(\langle e_1, e_6 \rangle) = 0.210$$

positive correlation

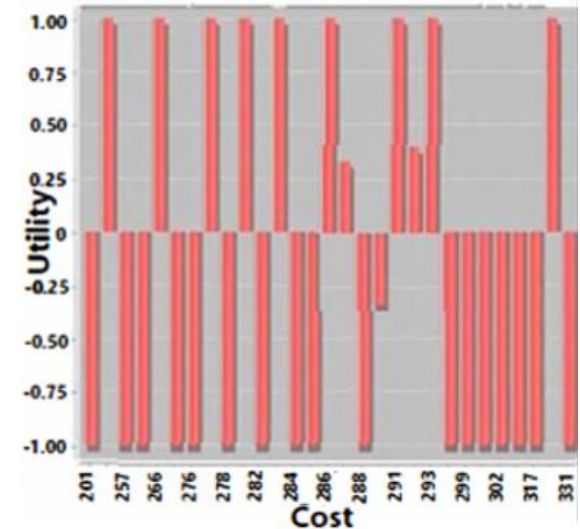
$\langle e_4, e_5 \rangle$



$$\text{cor}(\langle e_4, e_5 \rangle) = -0.109$$

negative correlation

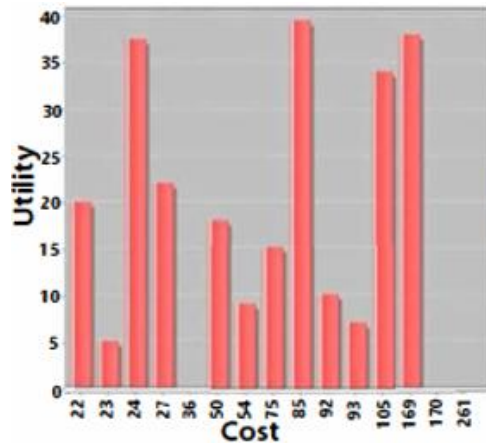
$\langle e_2, e_3 \rangle$



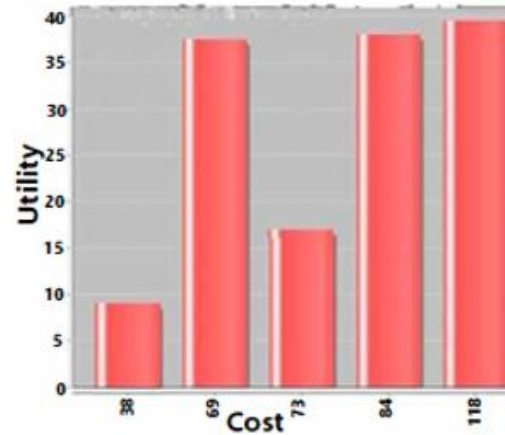
$$\text{cor}(\langle e_2, e_3 \rangle) = 0.001$$

no correlation

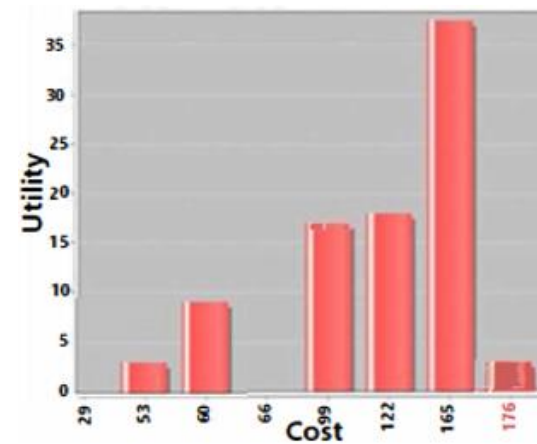
<Study_Es_6_2>
 <Deeds_Es_6_2>
 <Study_Es_6_3>



<Deeds_Es_6_1>
 <Deeds_Es_6_2>
 <Study_Es_6_3><Study_Es_6_3>



<FSM_Es_6_3>
 <Study_Es_6_3><Study_Es_6_3>



$tr(<Study_Es_6_2> <Deeds_Es_6_2> <Study_Es_6_3>) = 1.74,$
cost / utility = 15 / 27

$tr(<Deeds_Es_6_1> <Deeds_Es_6_2> <Study_Es_6_3> <Study_Es_6_3>) = 1.35,$
cost / utility = 21 / 28

$tr(<FSM_Es_6_3> <Study_Es_6_3> <Study_Es_6_3>) = 4.5,$
cost / utility = 21 / 94.8

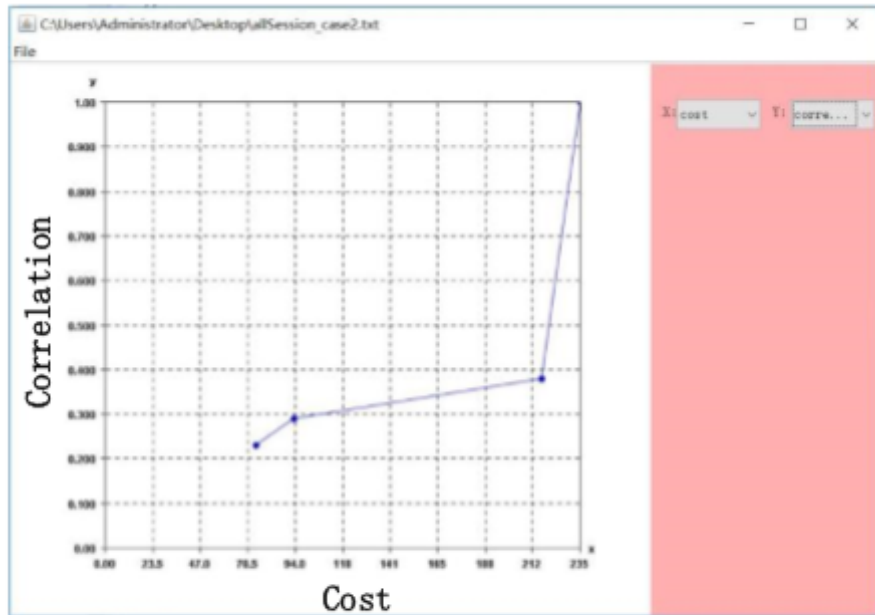


Fig. 3. skyline pattern from CEPDO

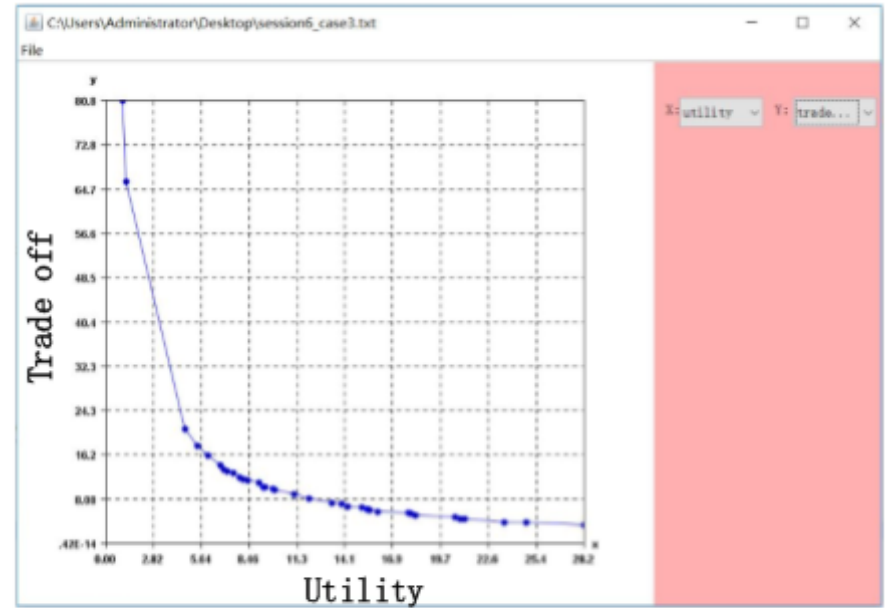


Fig. 4. skyline pattern from CEPHU