Preview

Sequential Activity Database

- A **sequence** is a series of activities, each having a cost.
- The **utility** of a sequence is a binary class.
- Measure the **correlation of** a **pattern** *p* with the desirable outcome.

Sid	<Activity : 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)

What’ s new in this work?

- New problem was defined.
- A tighter bound to reduce the search space.
- A new measure to evaluate patterns.
- Two pruning strategies are designed.

Sequential Activity Database

- A **sequence** is a series of activities, each having a cost.
- The **utility** of a sequence is a positive number.

Sid	<Activity : 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)

The *support* measure

Sid	<Activity : 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 **support** of a pattern p:

$$\mathrm{sup}(p) = |S_s | p \subseteq S_s \in DB|$$

(number of sequences containing p)

e.g. *sup*(<**ab**>)=|{S_1, S_4}| = **2** sequences

This measure is used to remove noise.

The *cost* measure

Sid	<Activity : 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 **cost** of a pattern p:

$$c(p, S_s) = \sum_{v_i \in first(p, S_s)} c(v_i, S_s)$$

$$\mathbf{c(ab,}\ \boldsymbol{S_1}\mathbf{)} = 4 + 2 = \mathbf{6}$$

The **average cost** of a pattern p:

$$ac(p) = \frac{\sum_{p \subseteq S_s \in DB} c(p, S_s)}{|\sup(p)|}$$

$$\mathbf{ac(ab)} = \underbrace{6}_{\textbf{Sequence 1}} + \underbrace{4}_{\textbf{Sequence 4}} / 2 = \mathbf{5}$$

This measure is used to assess the effort or resource spent.

The *occupancy* measure

Sid	<Activity : 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 p:

$$occup(p) = \frac{1}{sup(p)} \sum_{p \subseteq S_s \in SEL} \frac{|p|}{|S_s|}$$

$$\mathbf{occup}(\boldsymbol{ab}) = \frac{1}{2} \cdot \left(\underbrace{\frac{2}{5}}_{\textbf{Sequence 1}} + \underbrace{\frac{2}{4}}_{\textbf{Sequence 4}}\right) = \mathbf{0.45}$$

According to the previous experiments, this measure is used to remove patterns that are **short** and **non-representative** of the containing sequences.

Problem 3:
Finding all cost-effective patterns in a **numeric DB**

Sid	<Activity : 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:

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

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

$$occup(p) \geq minoccup$$

Problem 3:
Finding all cost-effective patterns in a **numeric DB**

Sid	<Activity : 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:

$$\sup(p) \geq minsup$$

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

$$occup(p) \geq minoccup$$

Furthermore, we measure the **trade-off** between the **cost** and **utility** of a **pattern** p :

$$tf(p) = \frac{ac(p)}{u(p)}$$

Average cost

Utility $\quad u(p) = \frac{\sum_{p \subseteq S_s \in DB} su(S_s)}{|\sup(p)|}$

Trade-off values are in the $(0, \infty]$ interval. Lower means more efficient.

More details…

Sid	<Activity : 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

Utility of a pattern *p*:

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

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

Trade-off of a pattern *p*:

$$tf(p) = \frac{ac(p)}{u(p)}$$

$$tf(\mathbf{ab}) = 5 /55 = \mathbf{0.09}$$

$$tf(\mathbf{cd}) = 6.7 /50 = \mathbf{0.13}$$

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

How to reduce the search space? (1)

Sid	<Activity : 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 **tighter 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_1) = 6 c(***bc***,S_2) = 5 c(***bc***,S_4) = 3

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

How to reduce the search space? (2)

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

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

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	<Activity : 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*

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

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.

How to reduce the search space? (3)

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	<Activity : 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. $minsup = 2$ $\quad$ $p = \langle a, b, c \rangle$

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

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

Properties of *upperu*:

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

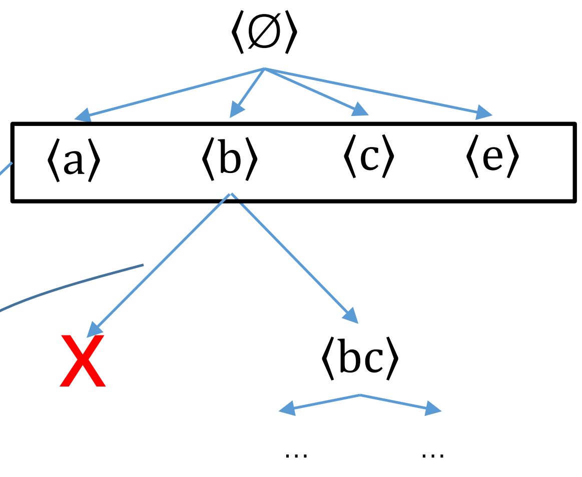
I. **Overestimation**: The $upperu$ of a pattern p is greater than or equal to its cost, $upperu(p) \geq u(p)$

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.

Pattern Growth Algorithms

Sid	<Activity : cost>	Utility
S_1	<~~(a:4)(b:2)(e:4)(c:4)~~(d:5)>	P/40
S_2	<~~(b:3)(c:2)~~(f:1)(d:1)(e:2)>	N/50
S_3	<(a:2)(f:2)(e:1)(c:3)(d:5)>	P/60
S_4	<~~(a:2)(b:2)(c:1)~~(f:2)>	N/70



P	*sup*	*ac*	*occup*	*cor / tf*
a	3	2.7	0.22	0.5/
b	3	2.3	0.22	-0.5/
…	…	…	…	…

P	*sup* ∧ *AMSC* ∧ *uo* ∧ *upperu(case2 only)*			
a	3	2.67	0.11	56.7
b	3	2.33	0.11	53.3
…	…	…	…	…

$$\mathrm{sup}(p) \geq minsup \quad \wedge$$
$$\mathrm{AMSC}(p) \leq \mathrm{maxcost} \quad \wedge$$
$$uo(p) \geq minoccup \quad \wedge$$
$$upperu(p) \geq minutility$$

Case study 2: numeric e-learning DB

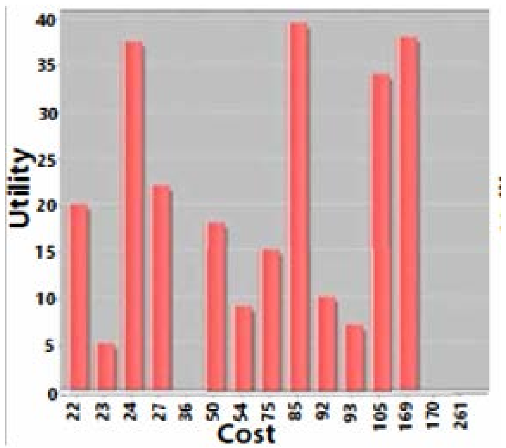
Database

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

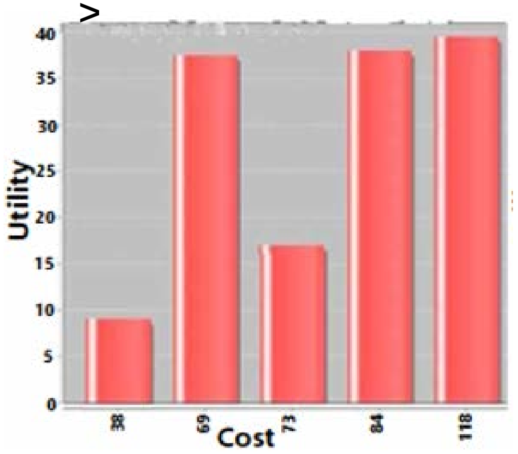
Cost-effective patterns found in learning session 6

Utility	Pattern	trade-off	Average Cost	Support
1	⟨Study_Es_6_1, Study_Es_6_1, Study_Es_6_1⟩	48.0	57.6	5
2	⟨Study_Es_6_1, Study_Es_6_1, Study_Es_6_3⟩	15.0	33.0	5
4	⟨Study_Es_6_1, Study_Es_6_2, Study_Es_6_2⟩	7.0	32.8	6
5	⟨Study_Es_6_1, Study_Es_6_1⟩	5.1	27.6	9
6	⟨Study_Es_6_1, Study_Es_6_1, Deeds_Es_6_1⟩	6.0	40.5	6
7	⟨Study_Es_6_2, Study_Es_6_2⟩	2.9	20.7	11
8	⟨Study_Es_6_2, Study_Es_6_2, Deeds_Es_6_2⟩	3.6	31.3	6
9	⟨Study_Es_6_1⟩	1.2	11.0	20
10	⟨Study_Es_6_1, Deeds_Es_6_2⟩	2.1	21	13
11	⟨Study_Es_6_2, Study_Es_6_3⟩	1.56	18.2	16
12	⟨Study_Es_6_2⟩	0.69	8.9	25
13	⟨Study_Es_6_3⟩	0.64	8.52	25
14	⟨Deeds_Es_6_2⟩	0.62	9.1	28
15	⟨Study_Es_6_2, Deeds_Es_6_2, Study_Es_6_3⟩	1.7	27.0	10
16	⟨FSM_Es_6_1, FSM_Es_6_1, Deeds_Es_6_2, Study_Es_6_3⟩	3.9	64.2	5
17	⟨Deeds_Es_6_2, Study_Es_6_3⟩	0.89	15.6	16
18	⟨Study_Es_6_3, Study_Es_6_3⟩	1.0	18.8	9
20	⟨Deeds_Es_6_1, tudy_Es_6_3, Study_Es_6_3⟩	1.6	32.7	7
21	⟨FSM_Es_6_3, Study_Es_6_3, Study_Es_6_3⟩	4.5	94.8	6
23	⟨Deeds_Es_6_2, Study_Es_6_3, Study_Es_6_3⟩	1.2	27.0	6
24	⟨FSM_Es_6_1, Deeds_Es_6_1, Study_Es_6_3, Study_Es_6_3⟩	3.6	86.3	6
28	⟨Deeds_Es_6_1, Deeds_Es_6_2, Study_Es_6_3, Study_Es_6_3⟩	1.35	38.0	5

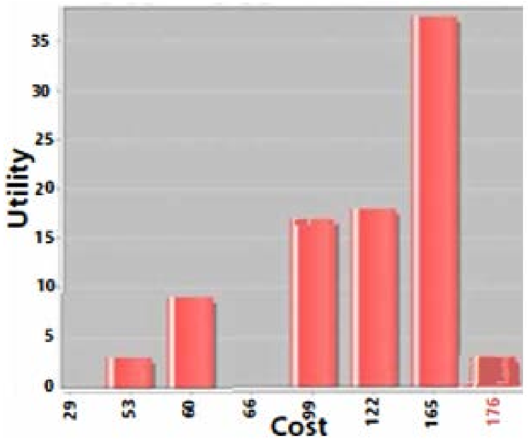
<Study_Es_6_2>
<Deeds _Es_6_2>
<Study_Es_6 _3>



<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>

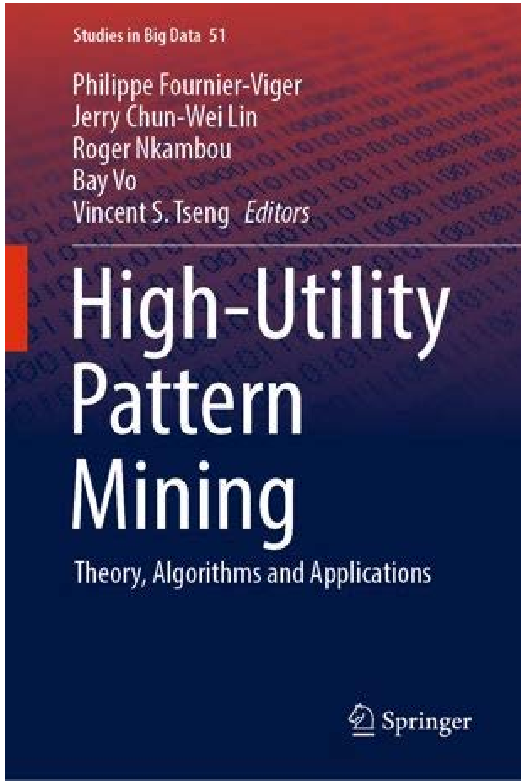


tr(<Study_Es_6_2> <Deeds _Es_6_2><Study_Es_6 _3>)=**1.74**,
cost / utility=27 /15

tr(<Deeds_Es_6 _1><Deeds_Es_6_2><Study_Es_6_3><Study_Es_6_3>)=**1.35**,
cost / utility=28 / 21

tr(<FSM_Es_6_3> <Study _Es_6_3><Study_Es_6 _3>)=**4.5**,
cost / utility=94.8 / 21

The smaller the trade-off, the more effective the pattern.



Open source Java data mining software, 150 algorithms
http://www.phillippe-fournier-viger.com/spmf/