LCIM: Mining Low Cost High Utility Itemsets

M. Saqib Nawaz, Philippe Fournier-Viger, Naji Alhusaini, Yulin He, Youxi Wu, Debdatta Bhattacharya

Source code and datasets available in the SPMF library

Introduction

- Huge amount of data
- A need to **understand** the data.
- **Pattern mining**: using algorithms to identify interesting patterns in data that can be also interpretable.
- **Frequent itemset mining**: finding patterns that appear frequently in the data.
- Recently, much attention on **high utility itemset mining** (HUIM), where the goal is to find values that co-occur in data and have a high utility (importance).

High Utility Itemset Mining

Input: Transaction database + Utility table + Minimum utility threshold (*minutil*)

Transaction database D

TID	Items
T_1	(a,1), (c,1)
T_2	(e,1)
T_3	(a,1), (b,5), (c,1), (d,3), (e,1)
T_4	(b,4), (c,3), (d,3), (e,1)
T_5	(a,1), (c,1), (d,1)
T_6	(a,2), (c,6), (e,2)
T_7	(b,2), (c,2), (e,1)

Utility table

Item	Unit profit
a	5$
b	2$
c	1$
d	2$
e	3$

***minutil* threshold**

***minutil* = 30$**

Output: High Utility Itemsets (HUIs), i.e., itemsets having utility ≥ *minutil*

{b,d}:30$	{a,c}:34$
{b,e}:31$	{b,d,e}:36$
{a,c,e}:31$	{b,c,e}:37$
{b,c,d}:34$	{b,c,d,e}:40$

Limitations

- The focus in HUIM is on the **utility** of patterns (benefits).
- However, the cost or effort required to obtain these benefits is ignored.
- 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

The proposal

- A novel problem named **Low-Cost High Utility Mining** that integrates the concept of **utility** with that of **cost**.
- **Goal**: Find itemsets that provide utility at a low cost.
- **Cost**: money, time resources consumed, effort

Example applications

- **E-learning:**
  - **Transaction:** list of activities taken by a student
  - **Cost:** the study time for each learning activity
  - **Utility**: the grades obtained at the final exam
  - **Low-cost high utility itemsets**: the learning activities that requires a small effort to obtain high grades
- **Medical pathway data**:
  - **Transaction:** list of treatments taken by a patient
  - **Cost:** the time or money spent for each treatment
  - **Utility**: the result of finally being cured or not
  - **Low-cost high utility itemsets**: the treaments that have a small cost and may lead to be cured.

How to combine utility and cost?

- **Simple solution**: subtract the cost from the utility and then apply HUIM algorithms.
  - **Problem**: utility and cost may be expressed in different units (e.g. hours, grades).
- **Solution in this paper:**
  - The cost and utility are modeled separately
  - Allows to use different units
  - Allows to calculate the average utility and average cost, which are more meaningful than total utility and total cost
    (**e.g.** average time spent by student is 30 min)

Need to design a new algorithm!

Related work

- Algorithms for frequent itemset mining: Apriori, FPGrowth, Eclat, etc.
- Algorithms for high utility itemset mining UP-Growth, UI-Miner, FHM, EFIM, REX, etc.
  - No algorithm has integrated the concept of **cost**.
- Algorithms for finding low-cost sequential patterns: CorCEPB, CEPN, CEPB.
  - Analyzing **sequences** is different from analyzing transactions to find itemsets.
  - New algorithms must be designed.

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5),(b,10),(c,1),(d,6),(e,3),(f,5)$	40
T_2	$(b,8),(c,3),(d,6),(e,3)$	20
T_3	$(a,5),(c,1),(d,2)$	18
T_4	$(a,10),(c,6),(e,6),(g,5)$	37
T_5	$(b,4),(c,2),(e,3),(g,2)$	21

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5),(b,10),(c,1),(d,6),(e,3),(f,5)$	40
T_2	$(b,8),(c,3),(d,6),(e,3)$	20
T_3	$(a,5),(c,1),(d,2)$	18
T_4	$(a,10),(c,6),(e,6),(g,5)$	37
T_5	$(b,4),(c,2),(e,3),(g,2)$	21

Let there be a set of **items** $\boldsymbol{I} = \{\boldsymbol{i_1}, \boldsymbol{i_2}, \ldots \boldsymbol{in}\}$

(e.g. learning activities a, b, c, d, e, f, g)

An **itemset** X is a set of items. **e.g.** {a,b}

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5),(b,10),(c,1),(d,6),(e,3),(f,5)$	40
T_2	$(b,8),(c,3),(d,6),(e,3)$	20
T_3	$(a,5),(c,1),(d,2)$	18
T_4	$(a,10),(c,6),(e,6),(g,5)$	37
T_5	$(b,4),(c,2),(e,3),(g,2)$	21

A **database** is a set of transactions $\boldsymbol{D = \{T_1, T_2 \ldots Tm\}}$

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5),(b,10),(c,1),(d,6),(e,3),(f,5)$	40
T_2	$(b,8),(c,3),(d,6),(e,3)$	20
T_3	$(a,5),(c,1),(d,2)$	18
T_4	$(a,10),(c,6),(e,6),(g,5)$	37
T_5	$(b,4),(c,2),(e,3),(g,2)$	21

e.g. Student #5 did learning activity « **b** » , « **c** », « **e** » , « **g** »

Each **transaction** T_i is a set of items ($T_i \subseteq I$)

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5),(b,10),(c,1),(d,6),(e,3),(f,5)$	40
T_2	$(b,8),(c,3),(d,6),(e,3)$	20
T_3	$(a,5),(c,1),(d,2)$	18
T_4	$(a,10),(c,6),(e,6),(g,5)$	37
T_5	$(b,4),(c,2),(e,3),(g,2)$	21

e.g. student **#5** took **4** minutes to do activity **« b ».**

Each **item** i in a **transaction** T has a **cost value** c(i, T) that is a positive number

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5),(b,10),(c,1),(d,6),(e,3),(f,5)$	40
T_2	$(b,8),(c,3),(d,6),(e,3)$	20
T_3	$(a,5),(c,1),(d,2)$	18
T_4	$(a,10),(c,6),(e,6),(g,5)$	37
T_5	$(b,4),(c,2),(e,3),(g,2)$	21

e.g. student **#5** has received 21 points at the final exam

Each **transaction** T has a **utility value** u(T) that is a positive number representing the benefits

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5),(b,10),(c,1),(d,6),(e,3),(f,5)$	40
T_2	$(b,8),(c,3),(d,6),(e,3)$	20
T_3	$(a,5),(c,1),(d,2)$	18
T_4	$(a,10),(c,6),(e,6),(g,5)$	37
T_5	$(b,4),(c,2),(e,3),(g,2)$	21

Three parameters: **minsup >0, minutil >0, maxcost >0**

Output: all the **low-cost high utility itemsets**

Each itemset X such that

its average utility au(X) ≥ minutil,

its average cost ac(X) ≤ maxcost,

its support s(X) ≥ minsup.

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5),(b,10),(c,1),(d,6),(e,3),(f,5)$	40
T_2	$(b,8),(c,3),(d,6),(e,3)$	20
T_3	$(a,5),(c,1),(d,2)$	18
T_4	$(a,10),(c,6),(e,6),(g,5)$	37
T_5	$(b,4),(c,2),(e,3),(g,2)$	21

Example: **X = {b,c}**

Problem definition

Input: A transaction database with utility and cost values

Transaction	Items $(item, cost)$	Utility
T_1	$(a,5), (b,10), (c,1), (d,6), (e,3), (f,5)$	40
T_2	$(b,8), (c,3), (d,6), (e,3)$	20
T_3	$(a,5), (c,1), (d,2)$	18
T_4	$(a,10), (c,6), (e,6), (g,5)$	37
T_5	$(b,4), (c,2), (e,3), (g,2)$	21

Example: **X = {b,c}**

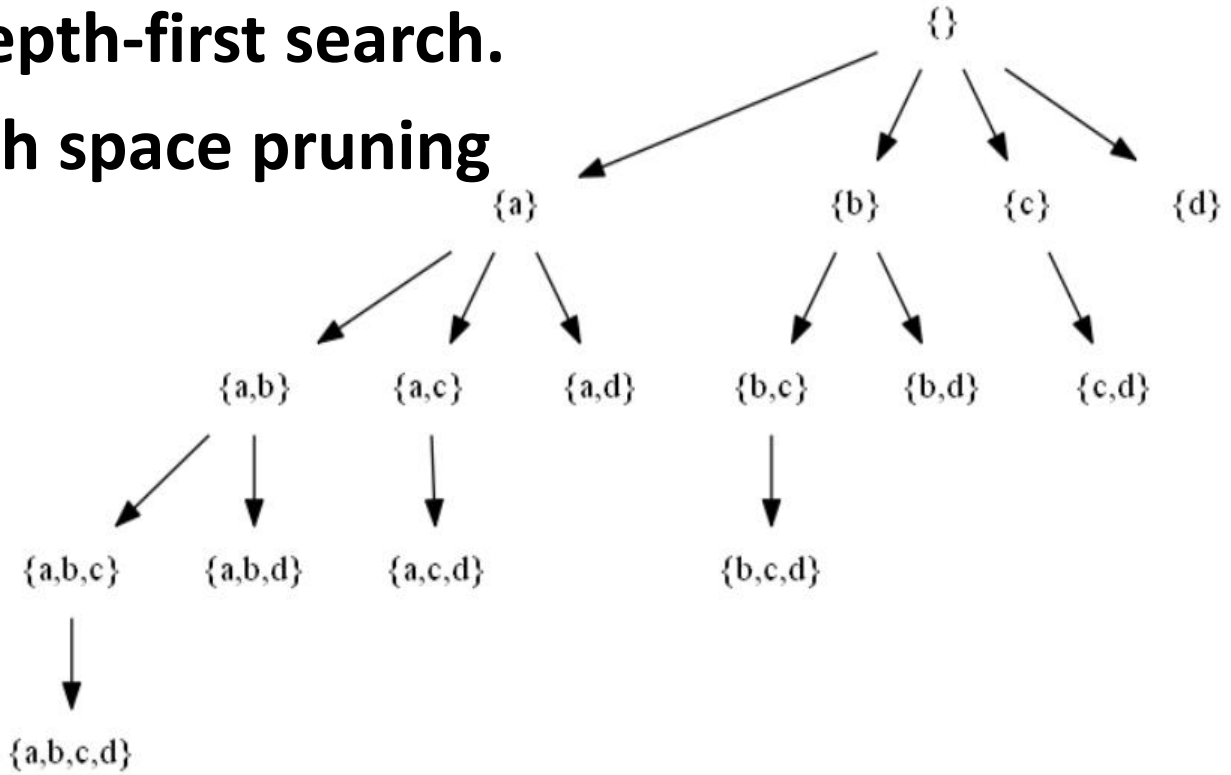
its average utility au(X) = (40 + 20 + 21)/ 3 = **27**

its average cost ac(X) = (10+1+8+3+4+2) / 3 = **9.3**

its support s(X) = 3

The LCIM algorithm

- An **algorithm** that starts from itemsets containing **single items** and recursively combine them to generate larger itemsets.
- Performs a **depth-first search.**
- **Applies search space pruning properties.**



Search space pruning using the support

This is a well-known properties in frequent itemset mining:

Property 1 (Support pruning). For any two itemsets $X \subseteq Y$, $s(X) \geq s(Y)$.

Search space pruning using the cost

Definition 3 (Lower bound on the cost). *Let there be an itemset X, which appears in the transactions $g(X) = \{T'_1, T'_2, \ldots, T'_N\}$. The* ***sequence of cost values of*** *X is the unique sequence $A(X) = (a_i)_{i=1}^{N}$ where $a_i = c(X, T'_i)$. Let $sort(A(X))$ be the unique non decreasing sequence $B = (b_i)_{i=1}^{N}$ satisfying $\forall i \leq K \leq N \ |\{i \in \mathbb{N} | a_i = a_K\}| = |\{i \in \mathbb{N} | b_i = a_K\}|$. The K* ***largest cost values*** *of X is the sequence $A(X)^{(K)} = (c_i)_{i=1}^{K}$ of real numbers satisfying $\forall 1 \leq i \leq K, c_i = b_{N-K+i}$. The* ***average cost bound (ACB)*** *of X is defined as $acb(X) = \frac{\sum_{c_i \in A(X)(minsup)} c_i}{s(X)}$.*

Property 2 (lower bound of the ACB on the average cost). For any itemset X, the average cost bound of X is a lower bound on the average cost of X. In other words, $acb(X) \geq ac(X)$.

Property 3 (anti-monotonicity of the ACB). For any two itemsets $X \subseteq Y$, then $acb(X) \leq acb(Y)$.

Property 4 (Search space pruning using the ACB). For an itemset X, if $acb(X) > maxcost$, X and its supersets are not low cost itemsets.

The cost-list data structure

- To facilitate the calculations, **each itemset** has a special structure called **cost-list.**
- **Example**:

The cost-lists of $\{a\}$, $\{b\}$ and $\{a, b\}$

$L(\{a\}$	
utility	95
cost	20
tids	$\{T_1, T_3, T_4\}$
costs	$\langle 5, 5, 10\rangle$

$L(\{b\}$	
utility	81
cost	22
tids	$\{T_1, T_2, T_5\}$
costs	$\langle 10, 8, 4\rangle$

$L(\{a, b\}$	
utility	40
cost	15
tids	$\{T_1\}$
costs	$\langle 15\rangle$

Algorithm 1: LCIM

input : D: a transaction database,
$minsup, minutil, maxcost$
output : all the low cost itemsets

1 Calculate the support of all items by reading the database;
2 Store the frequent items in a set ω;
3 Etablish the order of support ascending values on ω, and denote it as $\succ$;
4 Create the cost-list of all frequent items in ω;
5 Put each item i from ω in a set γ if $acb(\{i\}) \leq maxcost$ according to $L(\{i\})$;
6 `Search` (γ, $minsup$, $minutil$, $maxcost$);

Algorithm 2: *Search*

input : P: some itemsets with their cost-lists,
$minsup, minutil, maxcost$
output : a set of low cost itemsets

1 **foreach** *itemset* $X \in$ P **do**
2 | **if** $au(X) \geq minutil \wedge ac(X) \leq maxcost$ **then** Output X;
3 | Initialize $XExtend$ as the empty set;
4 | **foreach** *itemset* $Y \in$ P *that can be joined with* X **do**
5 | | $Z \leftarrow X \cup Y$;
6 | | $L(Z) \leftarrow$ `Construct` $(L(X), L(Y))$;
7 | | **if** $s(Z) \geq minsup \wedge acb(Z) \leq maxcost$ **then**
8 | | | $XExtend \leftarrow XExtend \cup \{Z\}$;
9 | | **end**
10 | **end**
11 | `Search` ($XExtend$, $minsup, minutil, maxcost$);
12 **end**

Algorithm 3: Construct procedure

input : $L(X), L(Y)$: the cost utility list of two itemsets X and Y
output : the cost-list $L(Z)$ of $Z = X \cup Y$

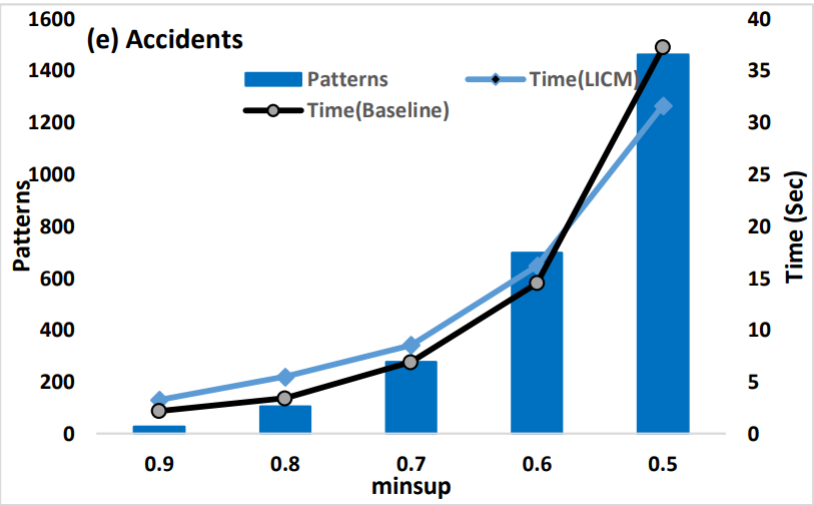
```
1  Initialize a cost-list L(Z) such that L(Z).utility = 0, L(Z).tids = ∅, and L(Z).costs = ∅;
2  foreach transaction T_w ∈ L(X).tids do
3  |   if ∃T_w ∈ L(Y).tids then
4  |   |   L(Z).tids ← L(Z).tids ∪ {T_w};
5  |   |   L(Z).utility ← L(Z).utility + u(T_w);
6  |   |   L(Z).costs ← Merge(L(X).costs, L(Y).costs);
7  |   end
8  end
9  L(Z).cost = ∑ L(Z).costs;
10 return L(Z);
```

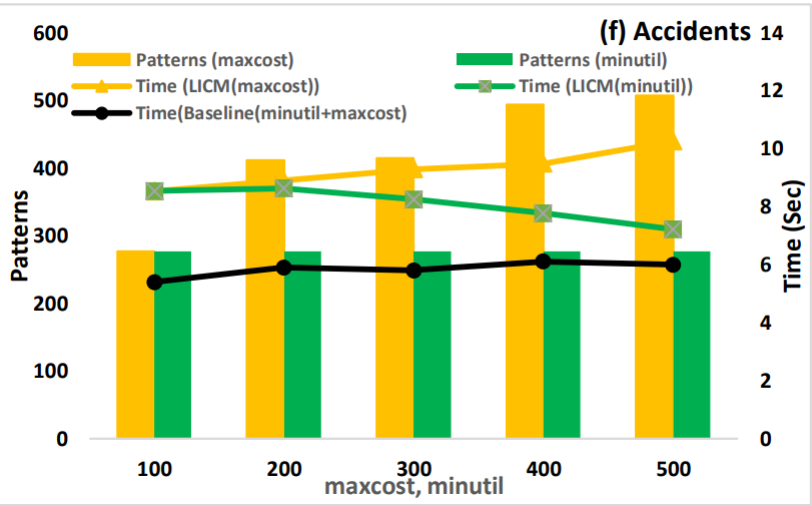
Two optimizations. We also include two performance optimization: (1) *Matrix Support Pruning* (MSP) consists of precalculating the support of all pairs of items in the initial database scan. Then, two itemsets X and Y are not joined as $X \cup Y$ if their last items have a joint support below *minsup*. (2) Efficient List Construction (ELC) the construct procedure is stopped as soon as there are not enough transactions left in $L(X).tids$ to attain $L(Z).tids \geq minsup$.

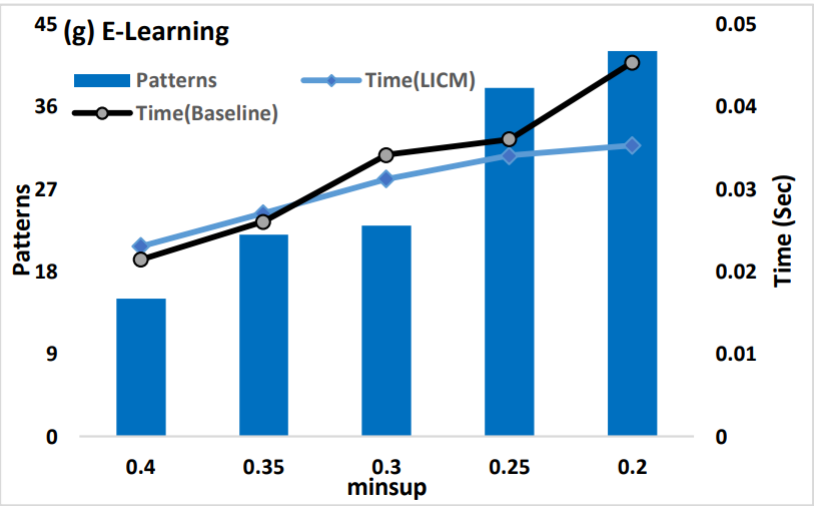
Experiments

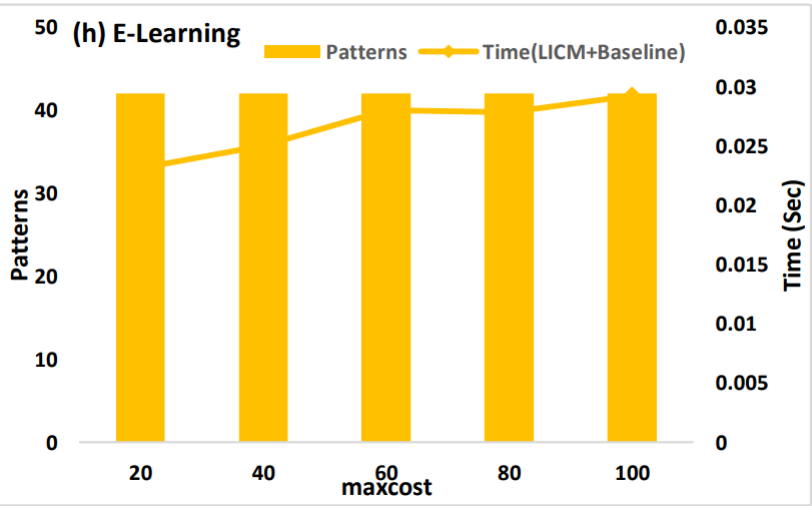
- **Four datasets**
  - **Chess** has 3,196 transactions, 37 distinct items, and an average transaction length of 75 items.
  - **Mushroom** contains 8,416 transactions, 119 distinct items, and the average transaction length is 23 items.
  - **Accidents** contains 340,183 transactions with 468 distinct items, and an average transaction length of 34 items.
  - **E-Learning** contains 54 transactions, 11 distinct items, and the average transaction length is 3.8 items.
- **LCIM vs baseline** (LCIM with no optimizations)

Performance evaluation (1)

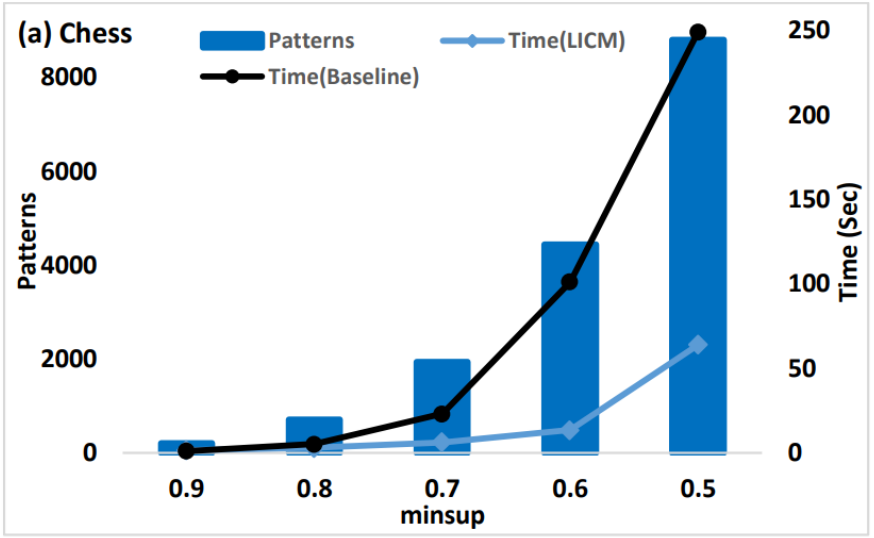


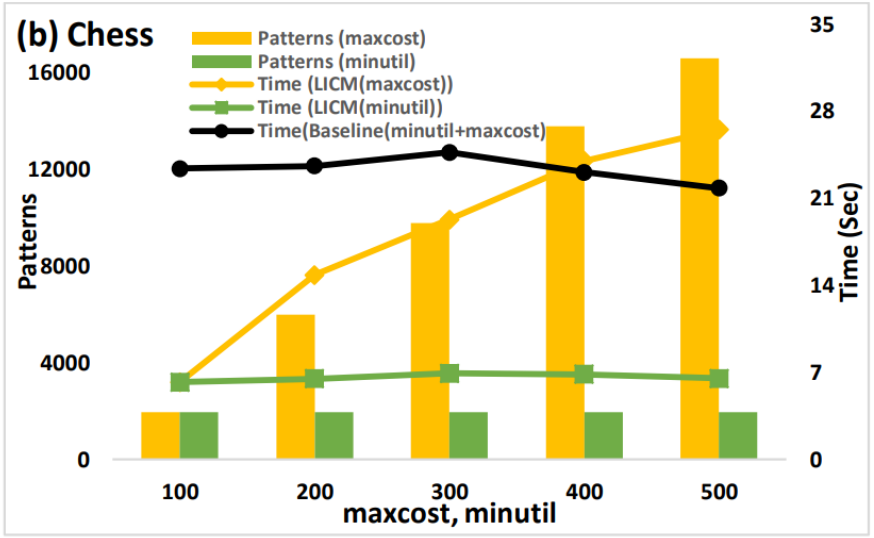


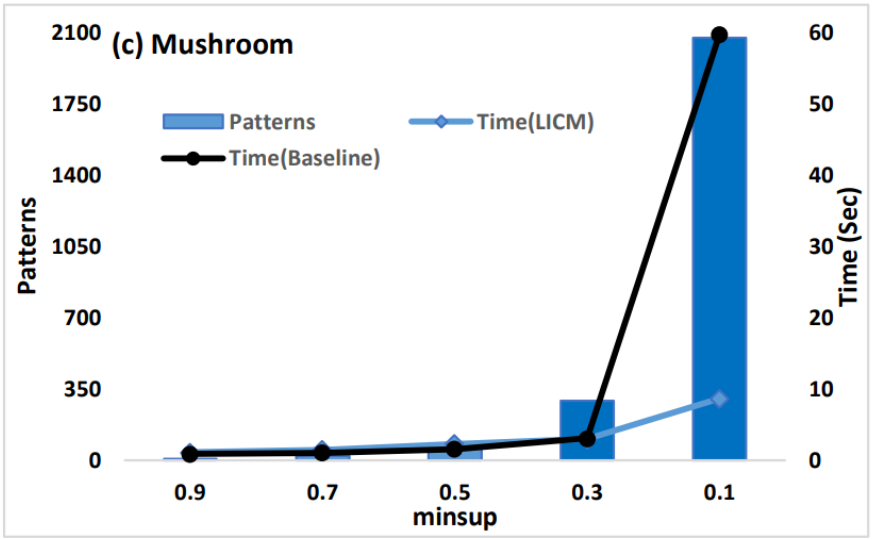


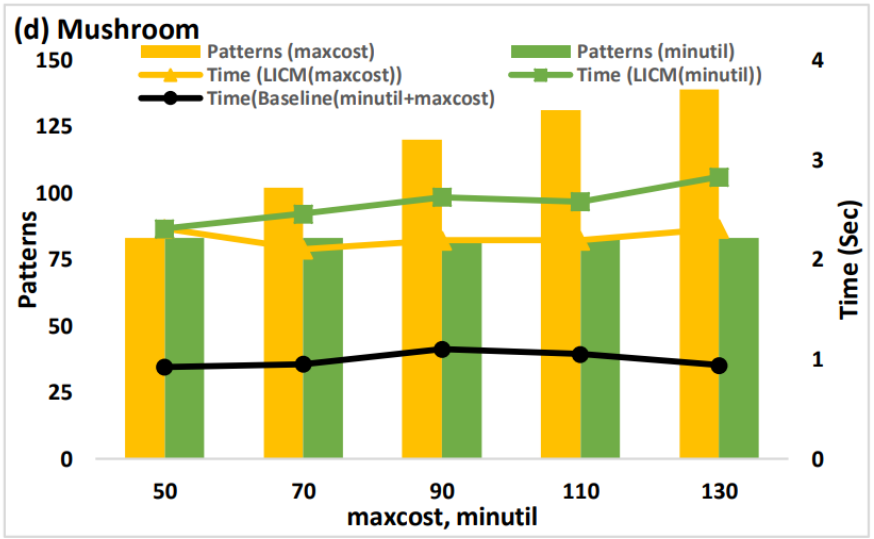


Performance evaluation (2)









Observations

- **LCIM** is not always faster than the **baseline**

- **Increasing maxcost** → execution time increases
  pattern count increases
- **Increasing minsup** → execution time increases
  pattern count increases
- **minutil**→ negligible effect on execution time

Patterns found in e-learning data

- E-learning dataset describe activities done by 54 students during Session 4 of an e-learning environment, and has cost and utility values.
- We found several interesting patterns with **minutil = 10**, **maxcost = 100** and **minsup = 0.2%**

No	Patterns	Avg. Util	Avg. Cost	Sup	Trade-off
1	*Deeds_Es_4_3, Deeds_Es_4_2*	12.53	21.23	13	1.69
2	*Deeds_Es_4_4, Deeds_Es_4_1*	14.07	25.21	14	1.81
3	*Deeds_Es_4_4, Deeds_Es_4_5*	12.42	28.92	14	2.38
4	*Deeds_Es_4_1*	14.21	13.82	23	0.98
5	*Deeds_Es_4_1, Deeds_Es_4_2*	13.64	26.71	14	1.96
6	*Deeds_Es_4_1, Deeds_Es_4_2, Deeds_Es_4_5*	12.76	60.0	13	4.76
7	*Deeds_Es_4_1, Deeds_Es_4_5*	12.5	34.12	16	2.77
8	*Deeds_Es_4_2*	13.26	10.60	23	0.80
9	*Deeds_Es_4_2, Deeds_Es_4_5*	12.47	27.68	19	2.22
10	*Deeds_Es_4_5*	11.62	17.20	24	1.49

A pattern's **trade-off** is the ratio of its average cost to its average utility

Conclusion

Contributions

- A new problem of **low-cost high utility itemset mining**
- A new algorithm **LCIM**
- Search space pruning strategies and optimizations

Research opportunities

- Alternative algorithms and additional optimizations
- Extending the problem (top-k, closed, maximal itemsets…)

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