哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

FHUQI-Miner: Fast High Utility Quantitative Itemset Mining

Presented by:

Mourad NOUIOUA

Supervised by:
Prof. Philippe Fournier-Viger

Outline

- Introduction
  - Frequent itemset mining
  - High utility itemset mining
- Limitation of HUIM
- Solution: FHUQI-Miner
- Conclusion

Frequent Itemset Mining

Input: Transaction Database + Minimum Support Threshold (minsup)

minsup = 3

A transaction database

TID	Items
T_1	a, c
T_2	e
T_3	a, b, c, d, e
T_4	b, c, d, e
T_5	a, c, d
T_6	a, c, e
T_7	b, c, e

Output: Frequent itemsets, i.e, itemsets having support ≥ minsup.

Frequent itemsets (with support ≥ *minsup=3*)

{a}:4	{b,e}:3
{b}:3	{c,d}:3
{c}:6	{c,e}:4
{d}:3	{b,c,e}:3
{e}:5	…
{a,c}:4	

Algorithms

- Apriori（VLDB 1994）
- Apriori-TID（VLDB 1994）
- Eclat（TKDE 2000）
- FP-Growth（ACM SIGMOD 2000）

Outline

- Introduction
  - Frequent itemset mining
  - High utility itemset mining
- Limitation of HUIM
- Solution: FHUQI-Miner
- Conclusion

High Utility Itemset Mining

Input: Transaction Database + Profit table + Minimum Utility Threshold (minutil)

A transaction database

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)

External utility values

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

a *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$

High Utility Itemset Mining

Several algorithms

- Two-Phase (PAKDD 2005)
- IHUP (TKDE, 2010)
- UP-Growth (KDD 2011)
- HUI-Miner (CIKM 2012)
- FHM (ISMIS 2014)
- EFIM (KAIS 2017)
- mHUIMiner (PAKDD 2017)

Key idea

Calculate an upper-bound on the utility of itemsets (**e.g.** the TWU) that is anti-monotonic to be able to prune the search space.

If $twu(X) < minutil$ then X is not HUI with all its extensions.

High Utility Itemset Mining

TWU pruning strategy

A 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)

External utility values

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

a *minutil* threshold

***minutil* = 45$**

TWU({b,c,d})=u(T3)+u(T4)=25+17=42

{b,c,d} is not HUI with all its supersets.

$$\forall X \in D, TWU(X) \geq u(X)$$

$u(\{b,c,d\}) = (5 \times 2) + (1 \times 1) + (3 \times 2) + (4 \times 2) + (3 \times 1) + (3 \times 2) = 34$

Outline

- Introduction
  - Frequent itemset mining
  - High utility itemset mining
- Limitation of HUIM
- Solution: FHUQI-Miner
- Conclusion

Limitation

High utility itemset mining

- The discovered patterns do not provide information about quantities.

High utility Quantitative itemset mining

- Discover all sets of items that have a high utility while also providing information about item quantities that led to this utility.

HUIM
Coffee, Cookies, Eggs

$\Longrightarrow$

HUQIM
Coffee:3, Cookies:2, Eggs:6

Limitation

Consider quantity information

- Mining high utility quantitative association rules HUQA: (DAWAK 2007).
- Vertical mining of high utility quantitative itemsets VHUQI: (IEEE GrC 2014).
- Efficient mining of high utility quantitative itemsets HUQI-Miner: (ICDMW:2019).

HUQI-Miner

- Proposed algorithms still have very long runtimes due to the very large search space.

Could we make a more efficient algorithm <u>with faster execution time</u>?

Outline

- ❑ Introduction
  - Frequent itemset mining
  - High utility itemset mining
- ❑ Limitation of HUIM
- ❑ Solution: FHUQI-Miner
- ❑ Conclusion

Solution: FHUQI-Miner

- HUQIM associates a quantity or range of quantities to each item.

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

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

Larger search space

Items

a, b, c, d

HUQIM ⇒

Q-items

(a,2), (b,5), (c,2), (d,1), (c,3), (b,4), (b,6), (d,1)

Solution: FHUQI-Miner

- In HUQIM, there are two kind of itemsets to be discovered: **Exact Q-itemsets** and **Range Q-itemsets.**
- **Range Q-itemsets** do not exist explicitly in the database. They are obtained by **combining exact Q-items.**
- **Exact Q-itemsets** **e.g. [(A,2),(C,6)].**
- **Range Q-itemsets** **e.g. [(A,5),(B,<u>2</u>,<u>4</u>).**

4-2+1 = 3

Lower bound quantity

Upper bound quantity

Q-interval

Solution: FHUQI-Miner

FHUQI-Miner

- A novel algorithm named **FHUQI-Miner** (**Fast High Utility Quantitative Itemset** miner) is proposed to:

  Find **High Utility Quantitative Itemsets.**

  Input: the minimum utility threshold (minutil).

  Output: Exact and range Q-itemsets having high utilities.

How to calculate the utility?

A transaction database

TID	Items
T_1	(a,2), **(b,5), (c,2)**, (d,1)
T_2	**(b,4), (c,3)**
T_3	(a,2), (c,2)
T_4	(a,2), (b,6), (d,1)

External utility values

Item	Unit profit
a	3$
b	**1$**
c	**2$**
d	2$

- The utility of the exact Q-itemset [(a,2)(c,2)] is calculated as follows:

u([(a,2),(c,2)]) = u(**[(a,2),(c,2)]**,T_1) + u(**[(a,2),(c,2)]**,T_3) = $(2 \times 3 + 2 \times 2) + (2 \times 3 + 2 \times 2)$ =20.

- The utility of the range Q-itemset [(b,4,5)(c,2,3)] is calculated as follows:

u([(b,4,5),(c,2,3)]) = u([(b,5),(c,2)],T_1) + u([(b,4),(c,3)],T_2) = $(5 \times 1 + 2 \times 2) + (4 \times 1 + 3 \times 2)$ =19.

Q-itemsets Combination

- The **combination process** is performed to produce high utility range Q-itemsets by merging Q-itemsets with consecutive quantities.

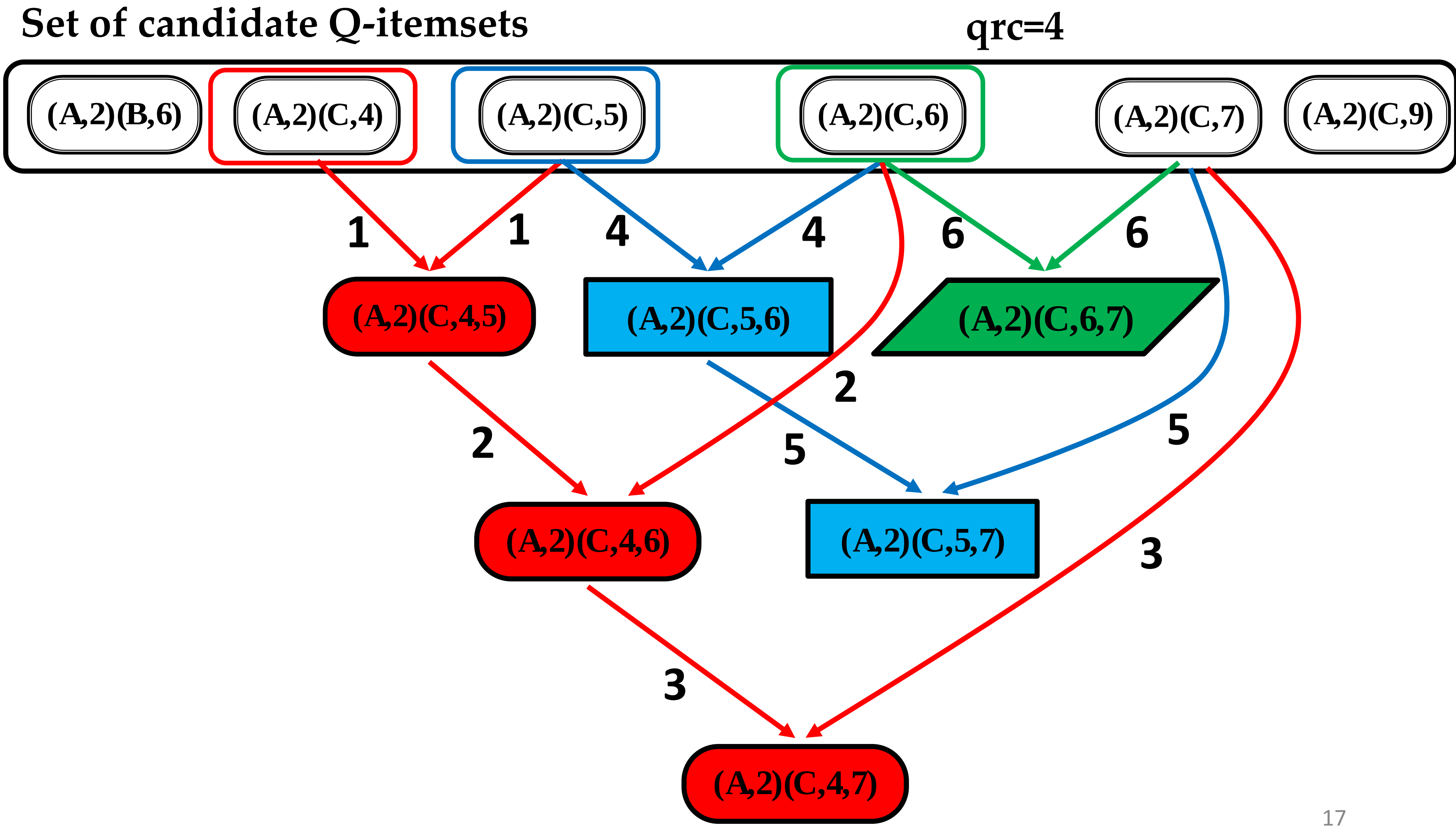
- It is required to define:

  1. The **combine-method** (***Combine-All, Combine-Min, Combine-Max***).
  2. The set of **candidate Q-itemsets**.
  3. The **quantitative related coefficient (*qrc*).**

Q-itemsets Combination

Combine-All

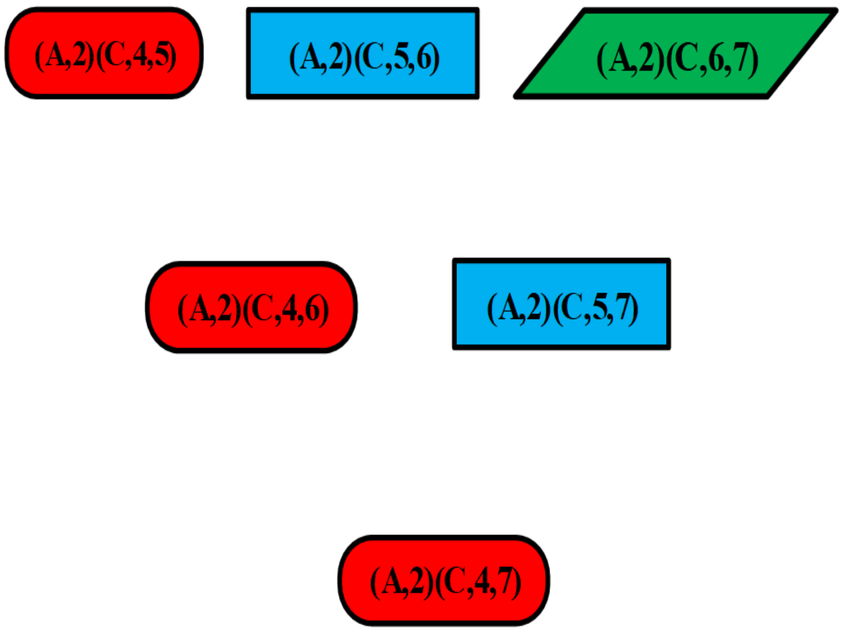
Set of candidate Q-itemsets



Q-itemsets Combination

Combine-All

- **Combine-All method keeps high utility range Q-itemsets from colored Q-itemsets.**



- **If all generated Q-itemset are HUIs, then Combine-All outputs:**

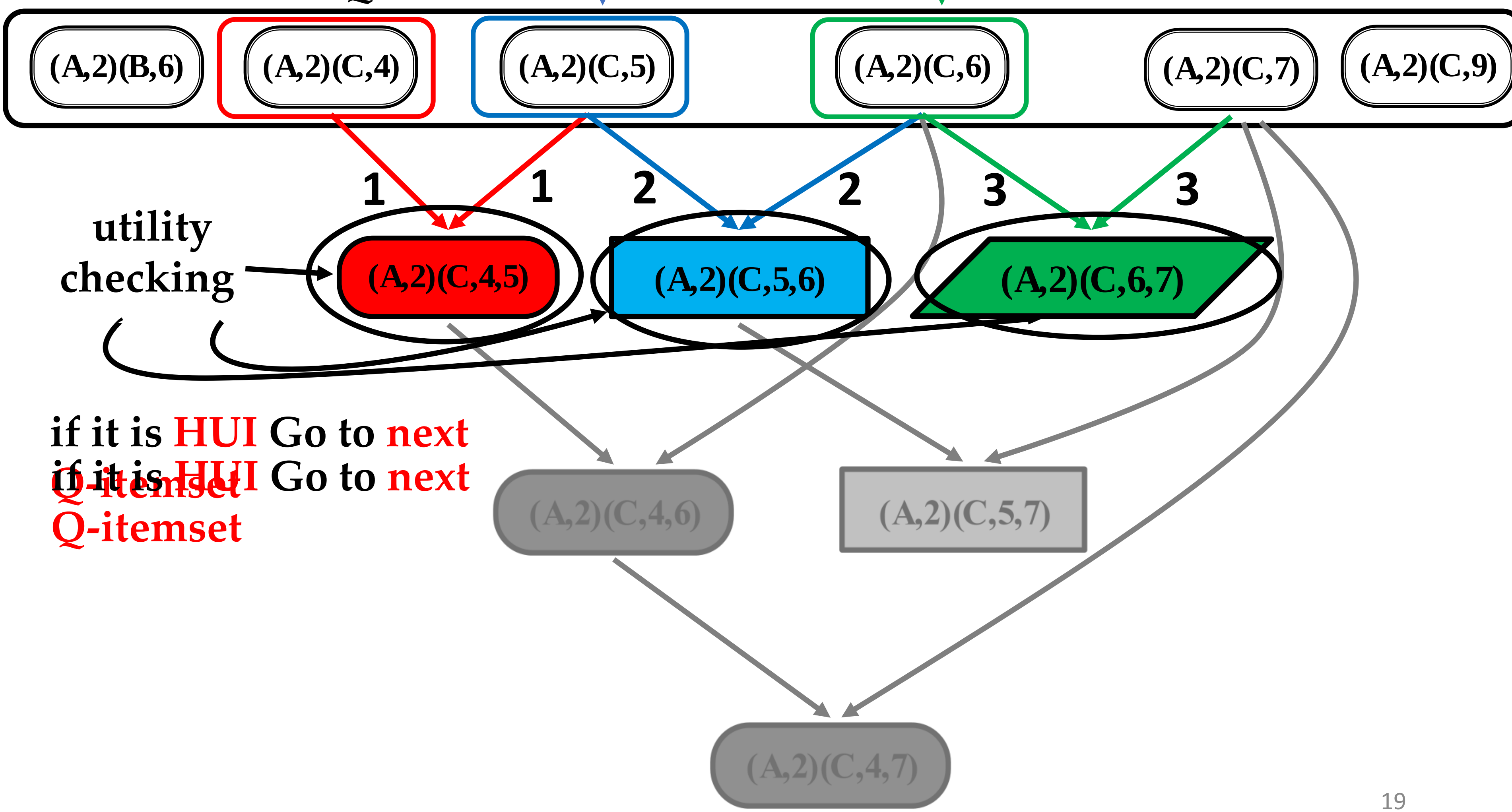
  {[(A,2)(C,4,5)], [(A,2)(C,4,6)], [(A,2)(C,4,7)], [(A,2)(C,5,6)], [(A,2)(C,5,7)], [(A,2)(C,6,7)]}.

Q-itemsets Combination

Combine-Min

Set of candidate Q-itemsets

qrc=4



if it is HUI Go to next Q-itemset

if it is HUI Go to next Q-itemset

Q-itemsets Combination

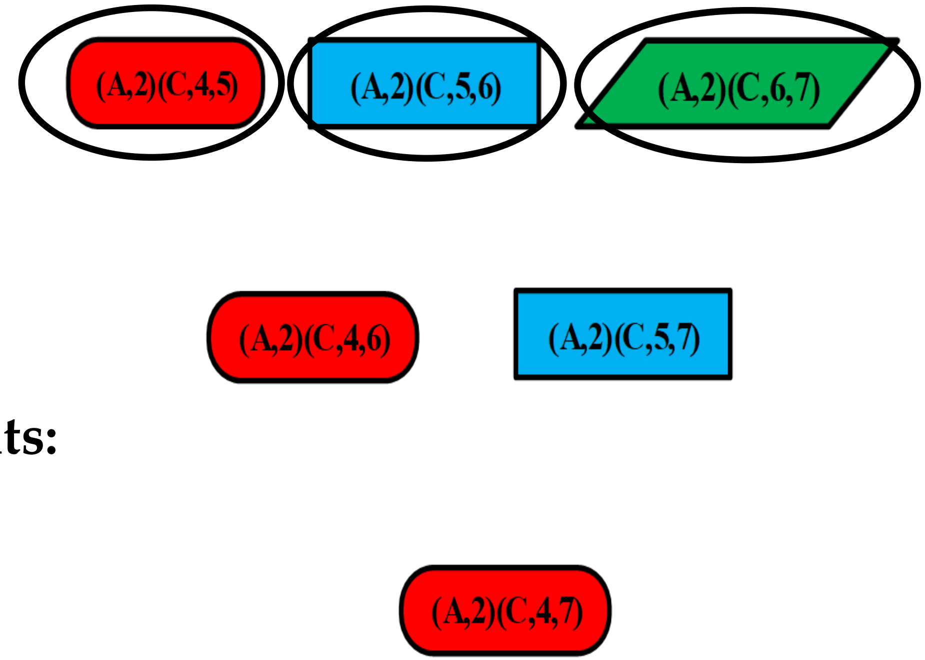
Combine-Min

- **Combine-min** method keeps **high utility range Q-itemsets** from colored Q-itemsets with minimal Q-intervals.

- If all generated Q-itemset are HUIs, then Combine-Min outputs:

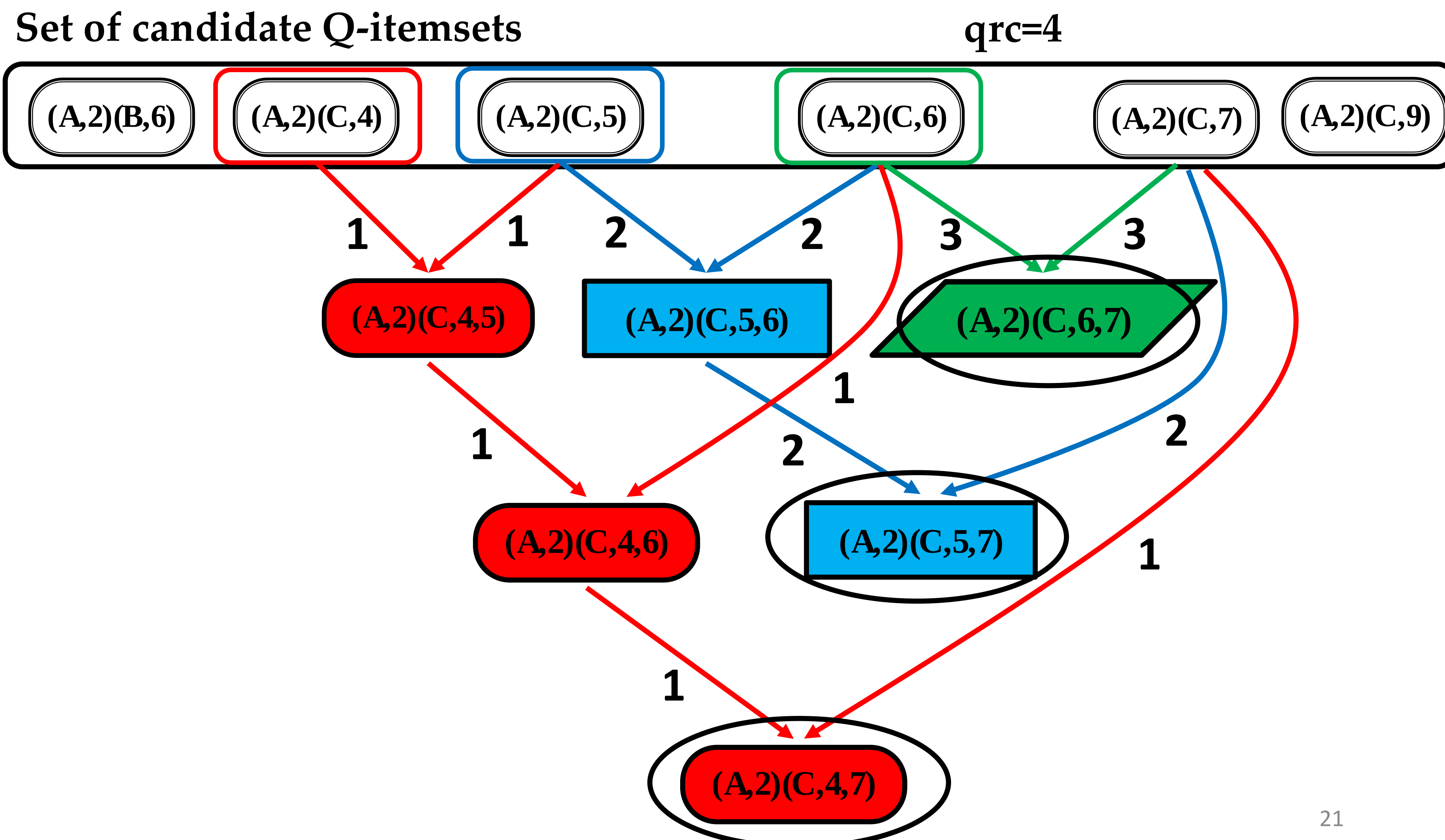
  {[(A,2)(C,4,5)], [(A,2)(C,5,6)], [(A,2)(C,6,7)]}.

Minimal Q-intervals



Q-itemsets Combination

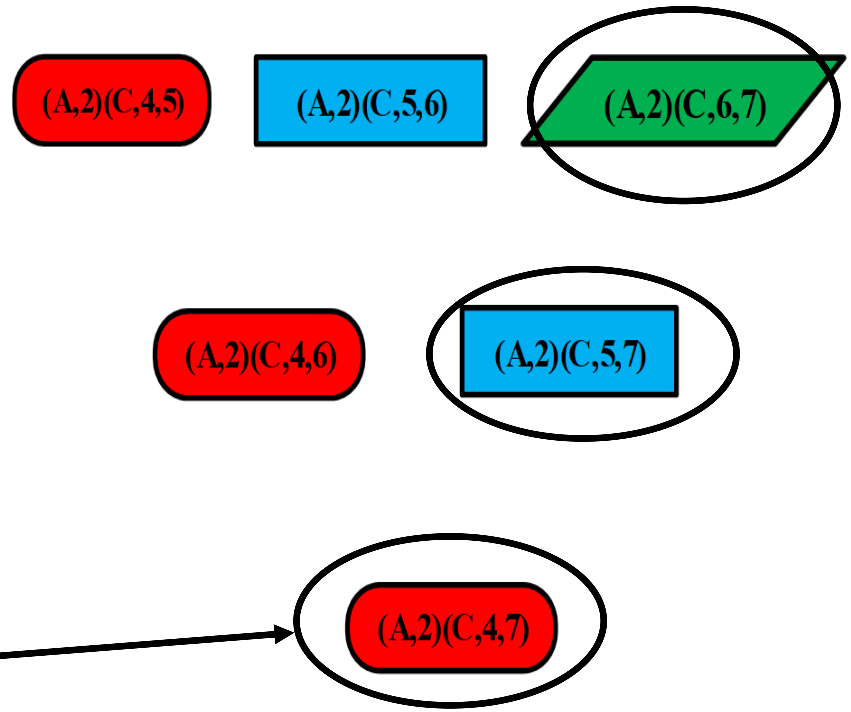
Combine-Max



Q-itemsets Combination

Combine-Max

- **Combine-max method keeps high utility range Q-itemsets with maximal Q-intervals.**
- **If all generated Q-itemset are HUIs, then Combine-Min outputs only {[(A,2)(C,4,7)]} and remove {[(A,2)(C,5,7)]} and {[(A,2)(C,6,7)]}.**



Maximal Q-interval

Problem Definition

Input: Transaction Database + Profit table + Minimum Utility Threshold (minutil) + Combining method+ Quantitative related coefficient (qrc).

A transaction database

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

External utility values

Item	Unit profit
a	42$
b	87$
c	94$
d	97$

a *minutil* threshold

***minutil* = 25%=1327**

Combine_All method

***qrc* = 5**

(d,4,6):1455$	(c,9) (b,8) (d,6) (a,7):2418$
(c,9) (b,8):1542$	(c,9) (b,8) (d,6) (a,7):2418$
(c,9) (d,6):1428$	(b,8) (d,6) (a,7):1572$
(c,9) (b,8) (d,6):2124$	(d,4,6) (a,7):1558$$
(c,9) (b,8) (a,7):1836$	

Output: High-utility quantitative itemsets (including both range and exact Q-itemsets).

Q-itemsets Utility-lists

- **An utility-list** is created for each Q-itemset.

Utility-list of (A,2)		
TID	*iutil*	*rutil*
T_1	6	11
T_3	6	4
T_4	6	8
Sum	18	23

Utility of itemset in a transaction

Remaining-utility

Q-itemsets Utility-lists

- The **utility-lists** of **single Q-items** can be constructed by **scanning the database.**
- For other itemsets, it can be obtained by **joining their child Q-itemset's utility-lists (Join operation).**

Utility-list of (A,2)		
TID	*iutil*	*rutil*
T_1	6	11
T_3	6	4
T_4	6	8
Sum	18	23

\+

Utility-list of (D,1)		
TID	*iutil*	*rutil*
T_1	2	0
T_4	2	0
Sum	4	0

→

Utility-list of [(A,2),(D,1)]		
TID	*iutil*	*rutil*
T_1	8	0
T_4	8	0
Sum	16	0

Q-itemsets Utility-lists

Construction of Utility-Lists for k-Q-itemsets (Join operation).

Utility-list of [(A,2),(C,2)]		
TID	*iutil*	*rutil*
T_1	10	2
T_3	10	0
Sum	20	2

\+

Utility-list of [(A,2),(D,1)]		
TID	*iutil*	*rutil*
T_1	8	0
T_4	8	0
Sum	16	0

→

Utility-list of [(A,2),(C,2),(D,1)]		
TID	*iutil*	*rutil*
T_1	12	0
Sum	12	0

Utility-list of [(A,2)]		
TID	*iutil*	*rutil*
T_1	6	11
T_3	6	4
T_4	6	8
Sum	18	23

- **Join operations** are very costly in terms of **execution time.**

Q-itemsets Utility-lists

Construction of the Utility-List of a range Q-itemsets (Merge operation).

Utility-list of (B,4)		
TID	*Iutil*	*rutil*
T_2	4	6
Sum	4	6

Utility-list of (B,5)		
TID	*iutil*	*rutil*
T_1	5	6
Sum	5	6

Utility-list of (B,4,5)		
TID	*iutil*	*rutil*
T_1	5	6
T_1	4	6
Sum	9	12

How to reduce the search space?

TWU of a Q-itemset X

$$TWU(X) = \sum_{T_{\mathrm{d}} \in \mathrm{OCC(x)}} TU(T_{\mathrm{d}}), where\ TU(T_{\mathrm{d}}) = \sum_{x \in T_{\mathrm{d}}} u(x, T_{\mathrm{d}})$$

Promising Q-itemsets

$$TWU(X) \geq minutil/qrc$$

TWU pruning strategy

- If a Q-itemset X is unpromising, then X is low utility Q-itemset as well as all its extensions.

How to reduce the search space?

Remaining Utility of a Q-itemset X

$$Rutil(X, T_d) = \sum_{X \in T_d / X} u(X, T_d)$$

$$Rutil(X) = \sum_{T_d \in OCC(X)} REU(X, T_d)$$

Remaining utility pruning strategy

- Given an itemset X, if $(u(X) + REU(X)) < minutil$, prune X with all its extensions.

How to reduce the search space?

TQCS Structure

- TQCS structure stores TWU of all pairs of Q-items that co-occur in the database.
- TQCS structure is used to reduce the frequency of performing join operations.

TID	Items
T_1	(A,2), (C,7), (H,4), (I,9)
T_2	(A,3), (C,8)
T_3	(A,2), (B,1), (C,7), (G,7), (H,4)
T_4	(B,2), (C,9), (G,8), (H,5)
T_5	(A,2), (D,5), (E,1), (F,1)

Item	Unit profit
A	20$
B	15$
C	70$
D	54$
E	11$
F	100$
G	75$
H	47$
I	96$

How to reduce the search space?

TQCS Structure

a	b	c	a	b	c
(C,7)	(A,2)	2840	(D,5)	(E,1)	421
	(B,1)	1258		(A,2)	421
	(H,4)	2840		(F,1)	421
(A,2)	(E,1)	421	(G,8)	(B,2)	1495
	(B,1)	1258		(H,5)	1495
(I,9)	(A,2)	1582	(G,7)	(A,2)	1258
	(C,7)	1582		(C,7)	1258
	(H,4)	1582		(B,1)	1258
(H,5)	(B,2)	1495		(H,4)	1258
(H,4)	(A,2)	2840	(C,9)	(G,8)	1495
	(B,1)	1258		(B,2)	1495
(F,1)	(E,1)	421		(B,5)	1495
	(A,2)	421	(C,8)	(A,3)	620

How to reduce the search space?

Exact Q-items Co-occurrence Pruning Strategy (EQCPS)

- Given 2 Q-items x and y, if there is no tuple (a,b,c) such that $x = a, y = b \; and \; c \geq minutil/qrc$, then Q-itemset [xy] is pruned with all its extensions.

Range Q-items Co-occurrence Pruning Strategy (RQCPS)

- Given a range Q-item $x = (i, l, u)$ and exact Q-item y. If $\sum_{i=l}^{u} c_i < \frac{minutil}{qrc}$ where c_i is TWU between x_i and y, then Q-itemset [xy] should be pruned with all its extensions.

How to reduce the search space?

EQCPS

minutil=1400
qrc=2
x=(C,8)
y=(A,3)

$$\boldsymbol{TWU}([(\boldsymbol{C},\mathbf{8})(\boldsymbol{A},\mathbf{3})]) = \mathbf{620}$$

$$\frac{\boldsymbol{minutil}}{\boldsymbol{qrc}} = \mathbf{700}$$

$$\boldsymbol{TWU}([(\boldsymbol{C},\mathbf{8})(\boldsymbol{A},\mathbf{3})]) < \frac{\boldsymbol{minutil}}{\boldsymbol{qrc}}$$

Q-itemset [(C,8)(A,3)] is pruned with all its extensions.

a	b	c	a	b	c
(C,7)	(A,2)	2840	(D,5)	(E,1)	421
	(B,1)	1258		(A,2)	421
	(H,4)	2840		(F,1)	421
(A,2)	(E,1)	421	(G,8)	(B,2)	1495
	(B,1)	1258		(H,5)	1495
(I,9)	(A,2)	1582	(G,7)	(A,2)	1258
	(C,7)	1582		(C,7)	1258
	(H,4)	1582		(B,1)	1258
(H,5)	(B,2)	1495		(H,4)	1258
(H,4)	(A,2)	2840	(C,9)	(G,8)	1495
	(B,1)	1258		(B,2)	1495
(F,1)	(E,1)	421		(B,5)	1495
	(A,2)	421	(C,8)	(A,3)	620

How to reduce the search space?

RQCPS (1)

minutil=1400
qrc=2
x=(C,7,8)
y=(H,4)

a=(C,7) b=(H,4)

$$\boldsymbol{TWU}([(\boldsymbol{C},\mathbf{7})(\boldsymbol{H},\mathbf{4})]) = \mathbf{2840}$$

a=(C,8) b=(H,4)

$$\boldsymbol{TWU}([(\boldsymbol{C},\mathbf{8})(\boldsymbol{H},\mathbf{4})]) = \mathbf{0}$$

$$\boldsymbol{TWU}([(\boldsymbol{C},\mathbf{7},\mathbf{8})(\boldsymbol{A},\mathbf{3})]) \geq \frac{\boldsymbol{minutil}}{\boldsymbol{qrc}}$$

Q-itemset [(C,7,8)(A,3)] is not pruned.

a	b	c	a	b	c
(C,7)	(A,2)	2840	(D,5)	(E,1)	421
	(B,1)	1258		(A,2)	421
	(H,4)	2840		(F,1)	421
(A,2)	(E,1)	421	(G,8)	(B,2)	1495
	(B,1)	1258		(H,5)	1495
(I,9)	(A,2)	1582	(G,7)	(A,2)	1258
	(C,7)	1582		(C,7)	1258
	(H,4)	1582		(B,1)	1258
(H,5)	(B,2)	1495		(H,4)	1258
(H,4)	(A,2)	2840	(C,9)	(G,8)	1495
	(B,1)	1258		(B,2)	1495
(F,1)	(E,1)	421		(B,5)	1495
	(A,2)	421	(C,8)	(A,3)	620

How to reduce the search space?

RQCPS (2)

minutil=1400
qrc=2
x=(A,2,3)
y=(E,1)

a=(A,2) b=(E,1)

$$\boldsymbol{TWU}([(\boldsymbol{A},\boldsymbol{2})(\boldsymbol{E},\boldsymbol{1})]) = \boldsymbol{421}$$

a=(A,3) b=(E,1)

$$\boldsymbol{TWU}([(\boldsymbol{A},\boldsymbol{3})(\boldsymbol{E},\boldsymbol{1})]) = \boldsymbol{0}$$

$$\boldsymbol{TWU}([(\boldsymbol{A},\boldsymbol{2},\boldsymbol{3})(\boldsymbol{E},\boldsymbol{1})]) < \frac{\boldsymbol{minutil}}{\boldsymbol{qrc}}$$

Q-itemset [(A,2,3)(E,1)] is pruned with all its extensions.

a	b	c	a	b	c
(C,7)	(A,2)	2840	(D,5)	(E,1)	421
	(B,1)	1258		(A,2)	421
	(H,4)	2840		(F,1)	421
$\boxed{(A,2)}$	$\boxed{(E,1) \quad 421}$		(G,8)	(B,2)	1495
	(B,1)	1258		(H,5)	1495
(I,9)	(A,2)	1582	(G,7)	(A,2)	1258
	(C,7)	1582		(C,7)	1258
	(H,4)	1582		(B,1)	1258
(H,5)	(B,2)	1495		(H,4)	1258
(H,4)	(A,2)	2840	(C,9)	(G,8)	1495
	(B,1)	1258		(B,2)	1495
(F,1)	(E,1)	421		(B,5)	1495
	(A,2)	421	(C,8)	(A,3)	620

Pseudocode

Algorithm 1: The FHUQI-Miner algorithm

Input : D: The quantitative transaction database, θ: The user-defined minimum utility threshold, CM: The combining method ($Combine\_Min$, $Combine\_Max$ or $Combine\_All$), qrc: The quantitative related coefficient.

Output: The complete set of HUQIs.

First database scan to calculate the TWU of each Q-item;
Create initial set of promising Q-items P^* such that $\forall x \in P^* : TWU(x) \geq \frac{\theta}{qrc}$;
Second database scan to create utility-lists of promising Q-items $ULs(P^*)$ and build the TQCS structure;
foreach $x \in P^*$ **do**
    if $UL(x).SumEutil \geq \theta$ **then**
        $H = H \cup x$;
        $Output\ x$;
    end
    else
        if $UL(x).SumEutil + UL(x).SumRutil \geq \theta$ **then**
            $E = E \cup x$;
        end
        if $\frac{\theta}{qrc} \leq UL(x).SumEutil \leq \theta$ **then**
            $C = C \cup x$;
        end
    end
end
Discover High Utility range Q-itemsets (HR) using CM and C;
$QI_s \leftarrow sort(H \cup E \cup HR)$;
Recursive_Mining_Search($\emptyset$, QI_s, $UL_s(QI_s)$, P^*, qrc, CM, θ);

TWU pruning strategy

TQCS structure

Utility checking

Remaining utility pruning strategy

Candidate Q-itemsets

Combination process

Recursive mining search

Pseudocode

Algorithm 2: *Recursive_Mining_SearchAlgorithm*

Input : P: The prefix Q-itemset, QI_s: The Q-itemsets list, $UL_s(QI_s)$: Utility lists of Q-itemsets, P^*: The list of promising Q-itemsets, qrc: The quantitative related coefficient, CM: The combining method, θ: The pre-defined minimum utility threshold

Output: The set of HUQIs with respect to prefix P.

```
foreach [Px] such that x ∈ QI_s do
    QI_s ← ∅; P* ← ∅
    foreach [Py] such that y ∈ P* and y ≻ x do
        if [Px] is an exact Q − itemset then
            c ← TQCS(x, y);
            if (c == Null Or c < θ/qrc) then
                Go to next [Py];
            end
        end
        else
            c ← Σ_{q=l}^{u} TQCS(x_i, y);
            if c < θ/qrc then
                Go to next [Py];
            end
        end
        Z ← [Pxy]; UL(Z) = Construct(x, y, P);
        if UL(Z)! = Null and TWU(Z) >= θ/qrc then
            P* = P* ∪ Z; if UL(Z).SumEutil ≥ θ then
                H = H ∪ Z; Output Z;
            end
            else
                if UL(Z).SumEutil + UL(Z).SumRutil ≥ θ then
                    E = E ∪ Z;
                end
                if θ/qrc ≤ UL(Z).SumEutil ≤ θ then
                    C = C ∪ Z;
                end
            end
        end
    end
    Discover High Utility range Q-itemsets HR using CM and C;
    QI_s ← (H ∪ E ∪ HR);
    Recursive_Mining_Search(Px,QI_s,UL_s(QI_s),P*,qrc,CM,θ);
end
```

TQCS & RQCS pruning strategies

Join operation

Utility checking

Remaining utility pruning strategy

Candidate Q-itemsets

Combination process

Recursive call

Experimental evaluation

Datasets' characterictics

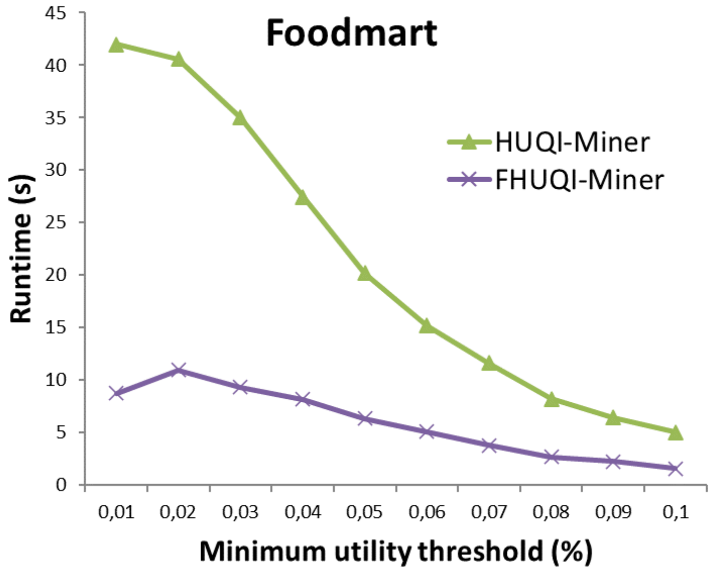
Dataset	M	N	Q	Type
Foodmart	4141	1559	1-10	Sparse
Retail	88162	16470	1-10	Sparse
BMSWebView2	77512	3340	1-10	Sparse
Mushroom	8416	128	1-10	Dense
Connect	67557	129	1-5	Dense
Pumsb	49046	2113	1-10	Dense

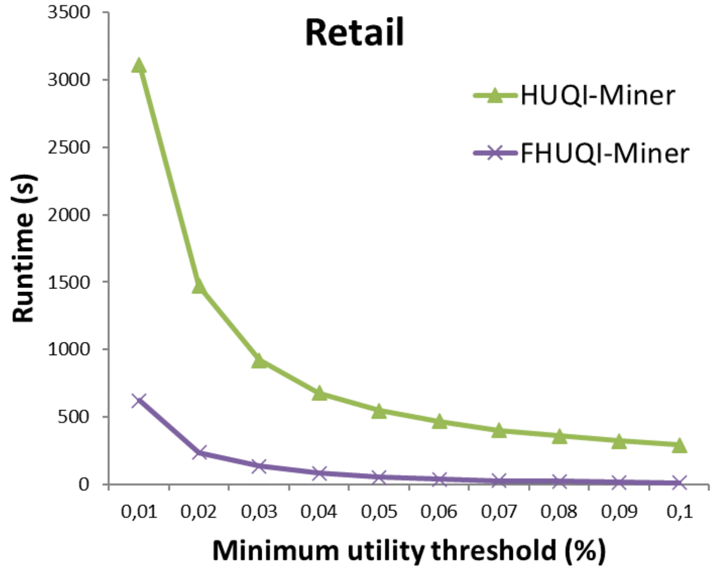
- D: Number of transactions.
- N: Number of distinct items.
- Q: Quantities range.

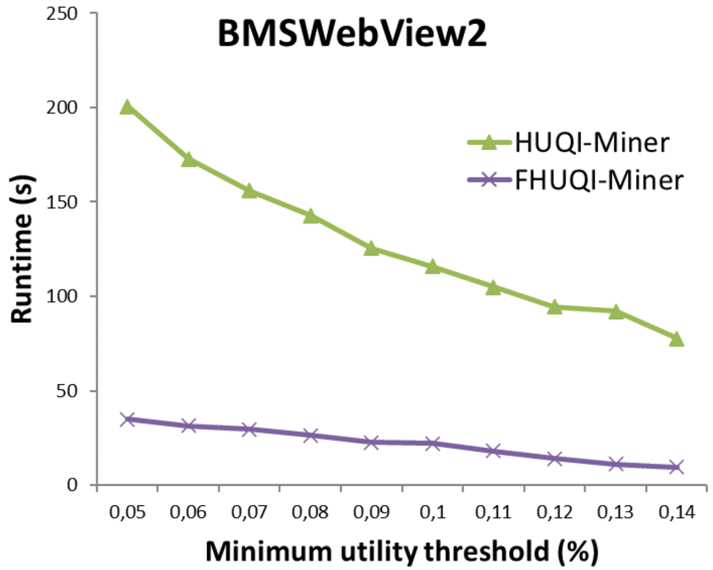
FHUQI-Miner is compared with the current state-of-art HUQI-Miner.

Experimental evaluation

Runtimes with sparse datasets



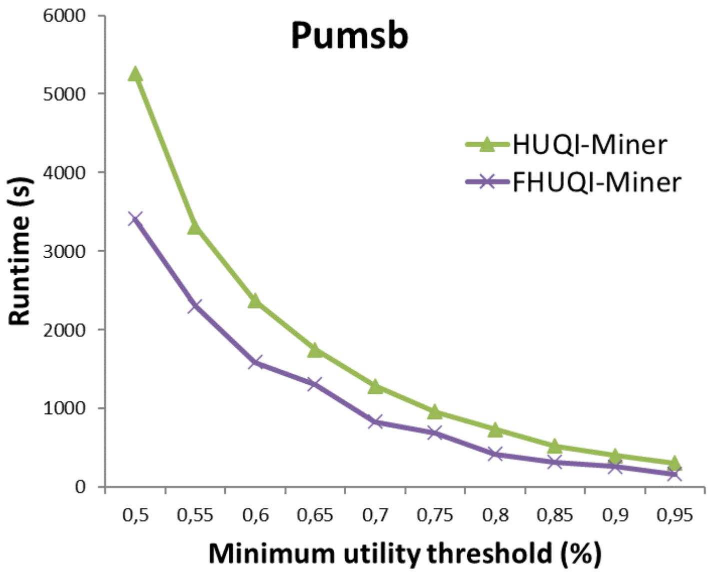


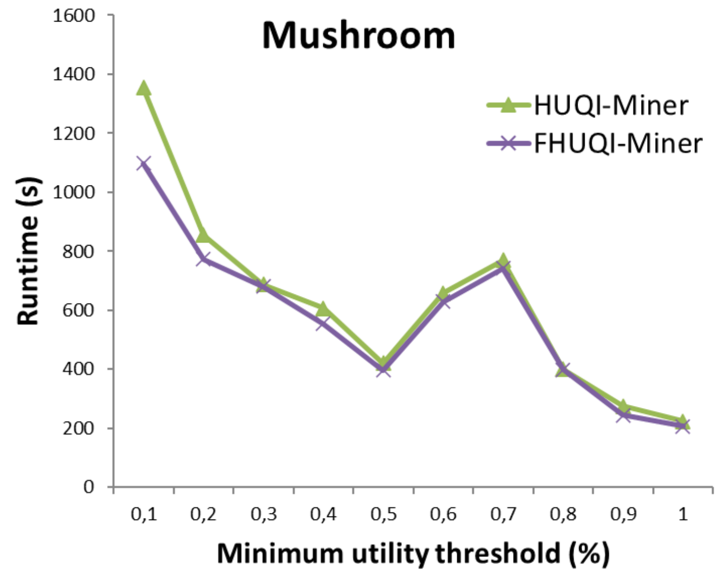


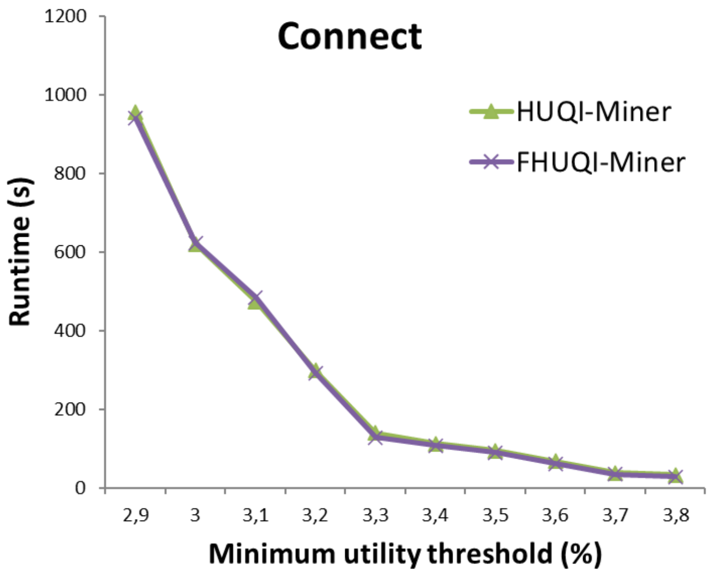
- FHUQI-Miner **clearly outperforms** HUQI-Miner.
- FHUQI-Miner is up **22 time faster** than HUQI-Miner in Retail dataset.

Experimental evaluation

Runtimes with dense datasets



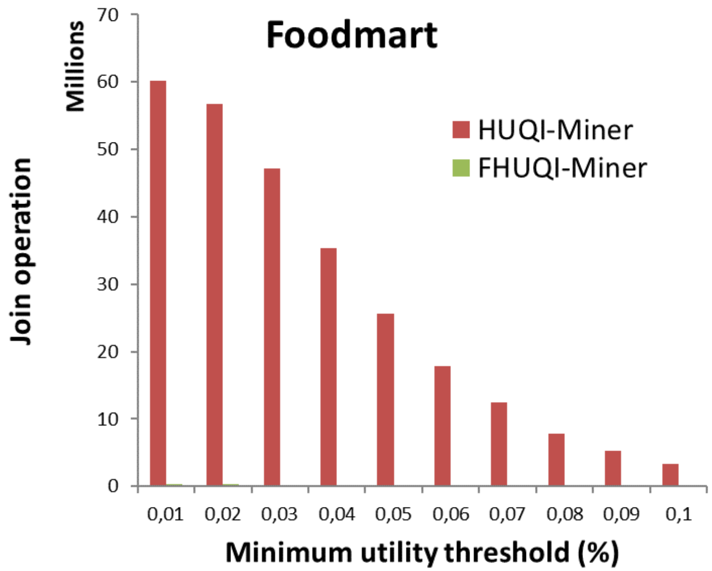


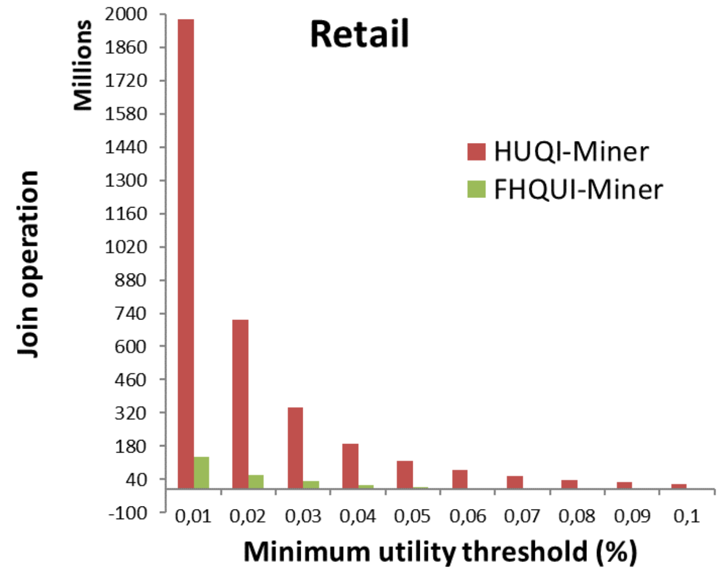


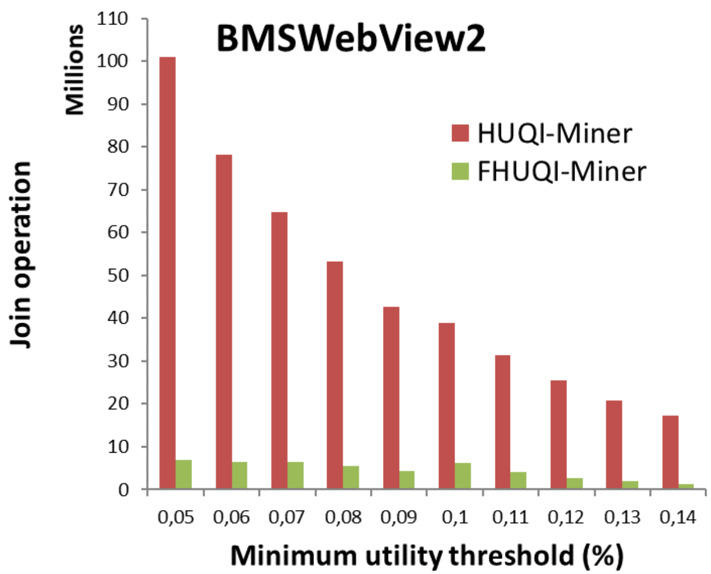
- For Mushroom and Connect datasets, the two algorithms have **similar running time**. However, FHUQI-Miner **still outperforms HUQI Miner** in most of cases.

Experimental evaluation

Join operations count with sparse datasets



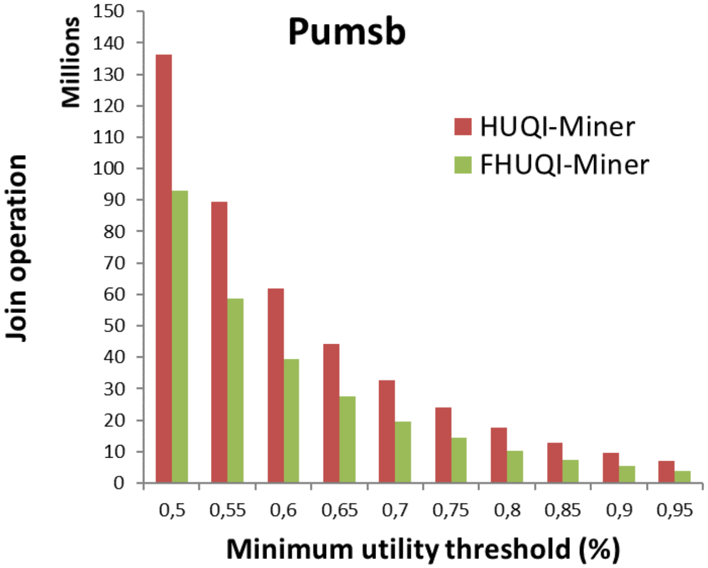


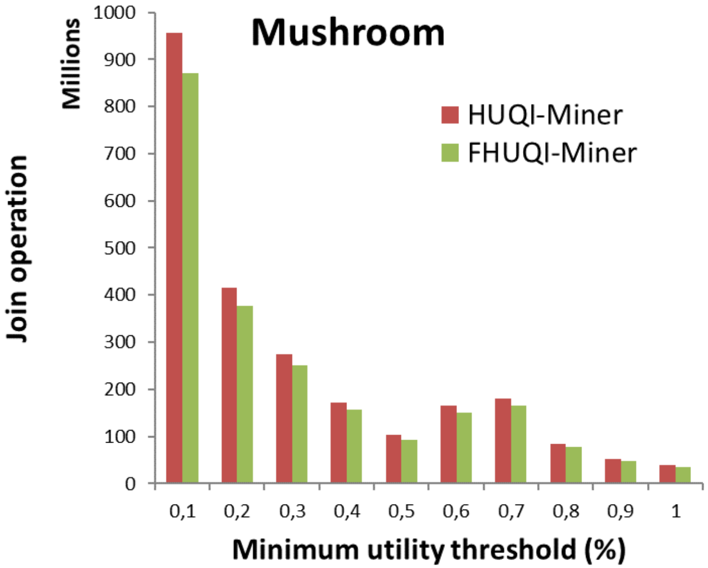


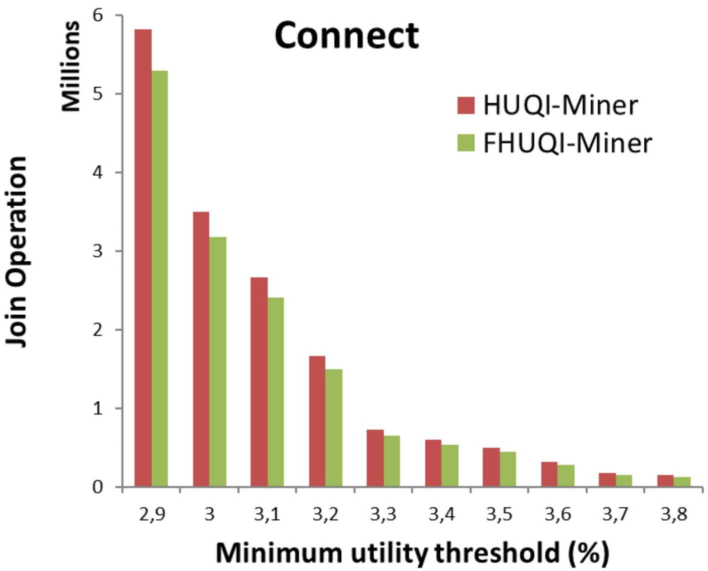
- FHUQI-Miner performs **much less join operations** than HUQI-Miner for all datasets.
- The diffidence between FHUQI-Miner and HUQI-Miner is **huge.**

Experimental evaluation

Join operations count with dense datasets







- The performance of FHUQI-Miner on **Pumsb** is clearly better than its performance on the two other datasets, **Mushroom** and **Connect**.

Experimental evaluation

Pruning rates on sparse datasets

Foodmart				Retail				BMSWebView2			
θ (%)	All (%)	Min (%)	Max (%)	θ (%)	All (%)	Min (%)	Max (%)	θ (%)	All (%)	Min (%)	Max (%)
0.01	99.55	99.54	99.52	0.01	93.15	98.35	98.88	0.05	93.15	98.11	98.59
0.02	99.50	99.52	99.48	0.02	91.58	98.30	98.83	0.06	91.84	98.05	98.55
0.03	99.55	99.58	99.54	0.03	90.92	98.43	98.89	0.07	89.88	97.99	98.53
0.04	99.61	99.64	99.63	0.04	91.49	98.59	98.99	0.08	89.52	97.97	98.55
0.05	99.67	99.71	99.71	0.05	92.94	98.76	99.11	0.09	89.72	97.99	98.57
0.06	99.70	99.75	99.76	0.06	93.27	98.86	99.18	0.1	84.33	98.01	98.61
0.07	99.70	99.76	99.80	0.07	94.36	98.98	99.29	0.11	87.00	98.04	98.67
0.08	99.71	99.76	99.81	0.08	94.86	99.03	99.34	0.12	89.09	98.12	98.74
0.09	99.72	99.77	99.82	0.09	95.82	99.05	99.40	0.13	91.05	98.15	98.80
0.1	99.72	99.76	99.82	0.1	96.17	99.07	99.43	0.14	92.18	98.22	98.88
Avg	99,64	99,68	99,69	Avg	93,46	98,74	99,13	Avg	89,78	98,07	98,65

- FHUQI-Miner prunes up to 99.72%, 96.17% and 93.15% of unpromising Q-itemsets for the **Foodmart**, **Retail** and **BMSWebView2** datasets, respectively.

Experimental evaluation

Pruning rates on dense datasets

Pumsb				Connect				Mushroom			
θ (%)	All (%)	Min (%)	Max (%)	θ (%)	All (%)	Min (%)	Max (%)	θ (%)	All (%)	Min (%)	Max (%)
0.5	31.75	32.60	40.30	2.9	9.14	15.03	30.10	0.1	8.89	18.61	22.90
0.55	34.47	18.91	8.63	3	9.18	15.80	32.70	0.2	9.08	18.50	28.96
0.6	36.31	23.23	7.88	3.1	9.36	16.63	33.82	0.3	8.46	20.81	36.93
0.65	37.92	38.27	45.56	3.2	9.55	17.30	36.12	0.4	8.86	24.11	44.78
0.7	39.44	40.57	54.74	3.3	10.59	17.37	37.14	0.5	8.73	27.65	51.12
0.75	40.53	42.18	62.11	3.4	11.15	17.01	38.08	0.6	8.68	29.85	52.71
0.8	41.89	44.08	67.42	3.5	11.85	17.62	39.46	0.7	8.43	31.43	55.55
0.85	43.18	45.40	70.83	3.6	13.29	20.06	40.01	0.8	8.35	35.67	60.14
0.9	44.51	47.04	72.71	3.7	16.30	24.30	40.44	0.9	8.38	39.00	63.15
0.95	46.52	48.92	73.35	3.8	18.28	26.22	41.42	1	8.55	41.73	65.05
Avg	40,43	39,23	52,45	Avg	11,87	18,73	36,93	Avg	8,64	28,74	48,13

- FHUQI-Miner prunes up to 46.52%, 18.28% and 9.08% of unpromising Q-itemsets for the **Pumsb**, **Connect** and **Mushroom** datasets, respectively.

Outline

- Introduction
  - Frequent itemset mining
  - High utility itemset mining
- Limitation of HUIM
- Solution: FHUQI-Miner
- Conclusion

Conclusion

Contributions

- In this research work, we have proposed new algorithm FHUQI-Miner to solve one important extensions of HUIM.
- In FHUQI-Miner, two new pruning strategies (EQCPS end RQCPS) are adopted to improve the efficiency of HUQIM in terms of execution time.

Conclusion

Future work

- Propose new efficient Pattern-growth based algorithms.
- Propose new tighter upper bounds that can further improve the mining search process.
- Consider other types of patterns such as sequential patterns and episodes.
- Perform HUQ patterns in other domains such as social network mining and computer networks.

Thank You for Your Attention

Questions or comments?