What we discussed last time?

- What is MASP algorithm?
- Why MASP algorithm?
- How to generate MASP tree?
- How to extract rules from MASP tree?

Problem associated with MASP algorithm

- It depends on the order of items in transactions

Scenario 1

1 2 3 4
2 4 3 1
3 2 4 1
4 2 3 1

Ts = 50%
---------->

ROOT
/
2
/
3

Scenario 2

1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4

Ts = 50%
---------->

- MASP can generate contradictory rules

Examples

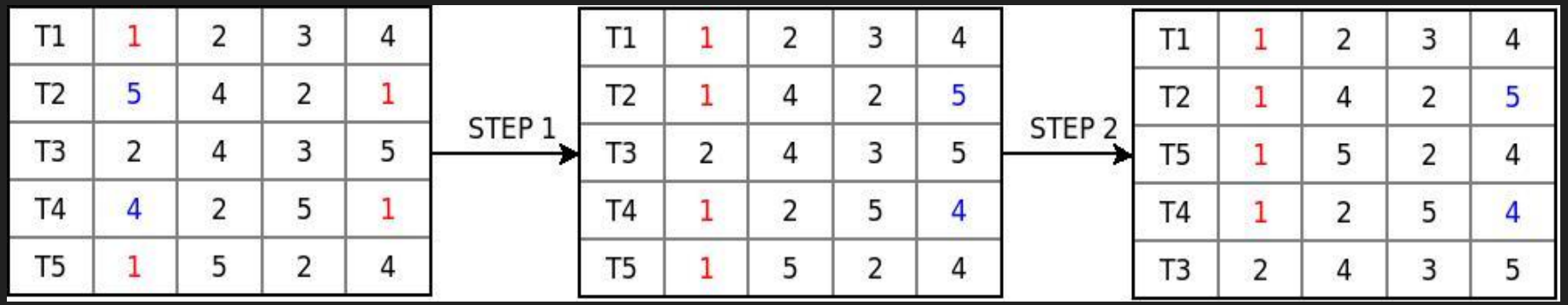
(A, B, ~C, D, E) ===> (~E)

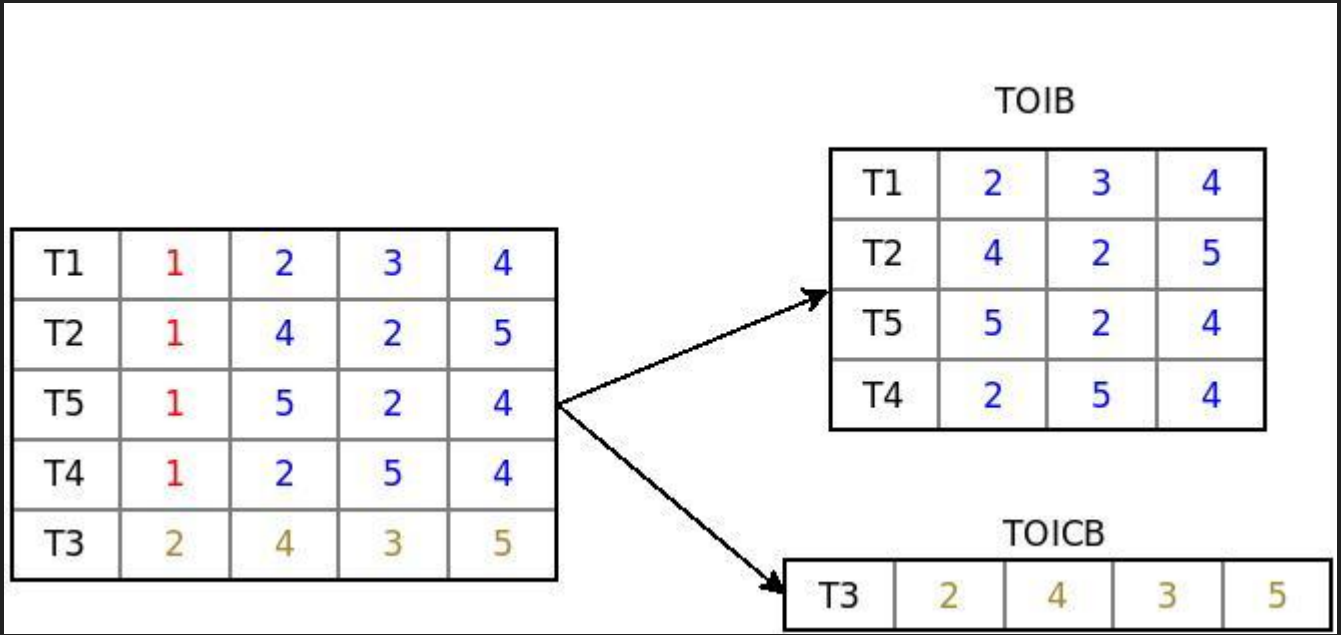
(A, B, ~C, D, E) ===> (C)

OIMASP algorithm

- Order Independent
- Non-contradictory rules

Shuffle : TOIB : TOICB




TOIB
T1 1 2 3 4
T2 1 4 2 5
T5 1 5 2 4
T4 1 2 5 4
T3 2 4 3 5
T1 2 3 4
T2 4 2 5
T5 5 2 4
T4 2 5 4
TOICB
T3 2 4 3 5

Algorithm 1: Algorithm to generate *OIMASP* tree.

1 function OIMASP ($\Gamma_{current}$, *Node*);
  Input : A transaction dataset $\Gamma_{current}$ associated with the *Node* of the *OIMASP* tree
  Output: An *OIMASP* tree
  /* Γ is the input transaction database, $|\Gamma|$ is the number of rows in Γ, τ_s is the threshold support, and τ_c is the threshold confidence. */
2 Obtain the frequency table of $\Gamma_{current}$
3 Find the item I_{max} having maximum frequency f_{max}
4 **if** $(support = \frac{f_{max}}{|\Gamma|}) < \tau_s$ **then**
5 | return *Node*
6 **end**
7 **if** $(confidence = \frac{f_{max}}{|\Gamma_{current}|}) < \tau_c$ **then**
8 | return *Node*
9 **end**
10 Add a node on the left side of *Node* say $Node_{left}$ and store I_{max} in it
11 $\Gamma_{left} = TOIB(\Gamma_{current}, I_{max})$ /* TOIB() and TOICB() functions are explained in Algorithm 2 */
12 $OIMASP(\Gamma_{left}, Node_{left})$

13 **if** $(support = \frac{|\Gamma_{current}| - f_{max}}{|\Gamma|}) < \tau_s$ **then**
14 | return *Node*
15 **end**
16 **if** $(confidence = \frac{|\Gamma_{current}| - f_{max}}{|\Gamma_{current}|}) < \tau_c$ **then**
17 | return *Node*
18 **end**
19 Add a node $Node_{right}$ on the right side of *Node* and store $\sim I_{max}$ in it
20 $\Gamma_{right} = TOICB(\Gamma_{current}, I_{max})$
21 $OIMASP(\Gamma_{right}, Node_{right})$
22 return *Node*

Example [Ts = 0.2 and Tc = 0.3]

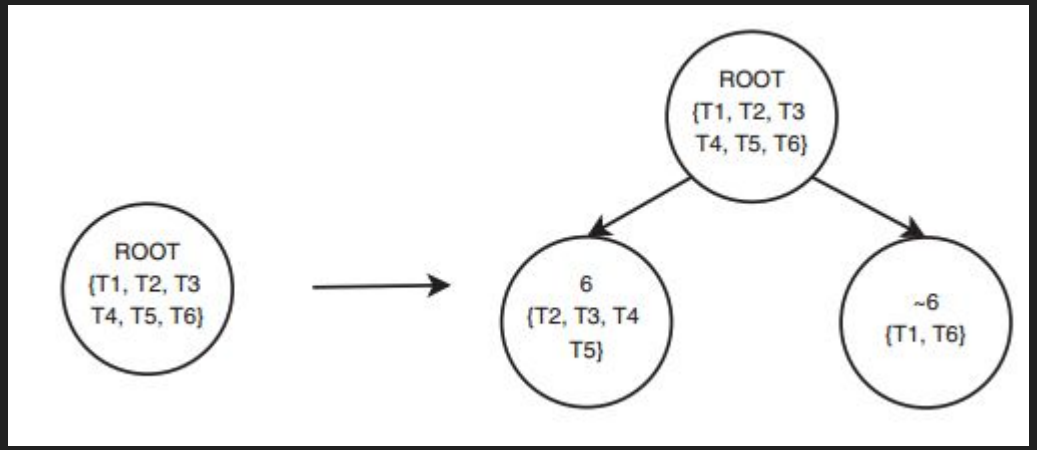
Transaction Table

T1	1	12	3	4	5
T2	1	5	6	4	12
T3	8	6	9	12	5
T4	9	2	3	6	7
T5	6	9	10	8	7
T6	1	8	3	2	7

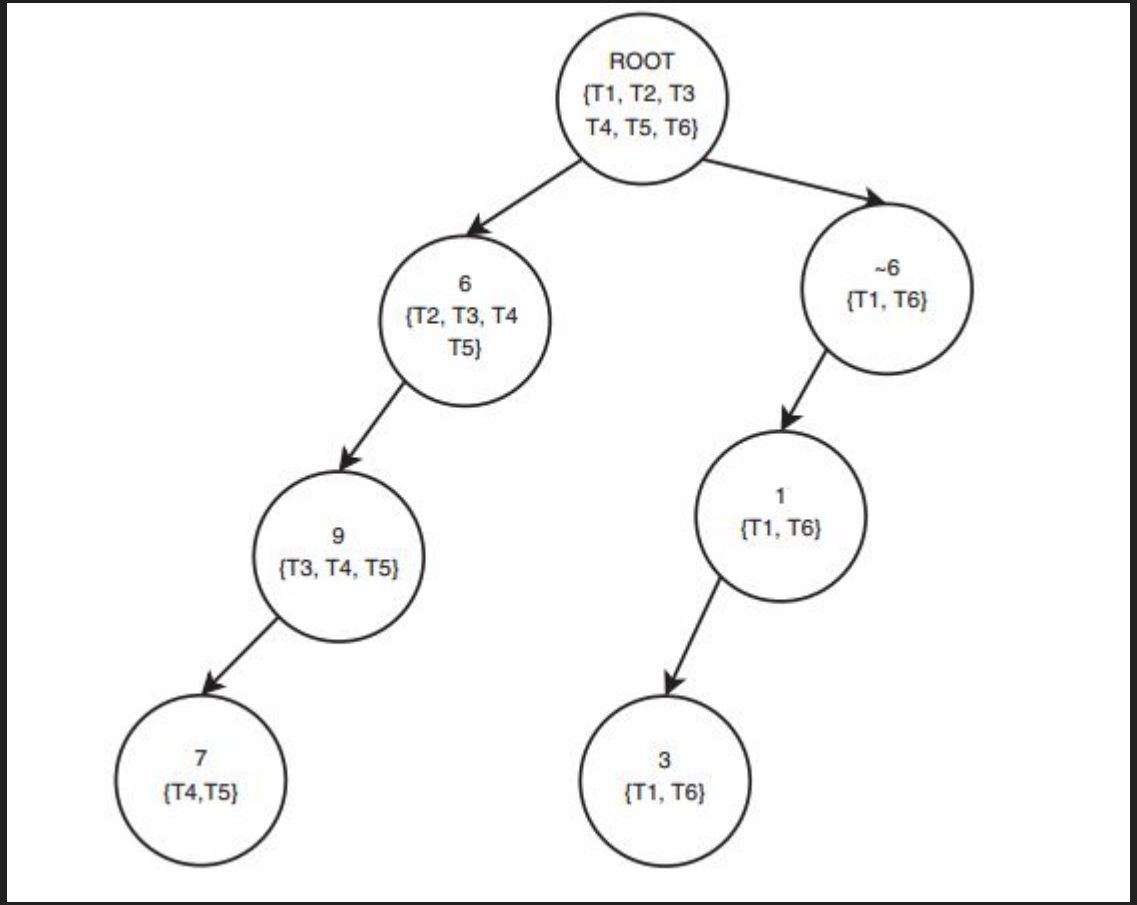
Frequency Table

Item	Count
1	3
10	1
12	3
2	2
3	3
4	2
5	3
6	4
7	3
8	3
9	3

Example



Example


ROOT
{T1, T2, T3
T4, T5, T6}
6
{T2, T3, T4
T5}
~6
{T1, T6}
9
{T3, T4, T5}
1
{T1, T6}
7
{T4,T5}
3
{T1, T6}

OIMASP VS MASP

			MASP		OIMASP	
Dataset	Min-Support	Min-Confidence	total rules	max-rule-size	total rules	max-rule-size
A	0.30	0.60	1	2	2	3
B	0.30	0.60	0	0	9	10
C	0.30	0.60	0	0	15	16
D	0.30	0.60	0	0	50	51
E	0.30	0.60	0	0	131	132

Second Contribution

- Concept of unbiased support and unbiased confidence
- Taking into consideration the origins of items

Second Contribution

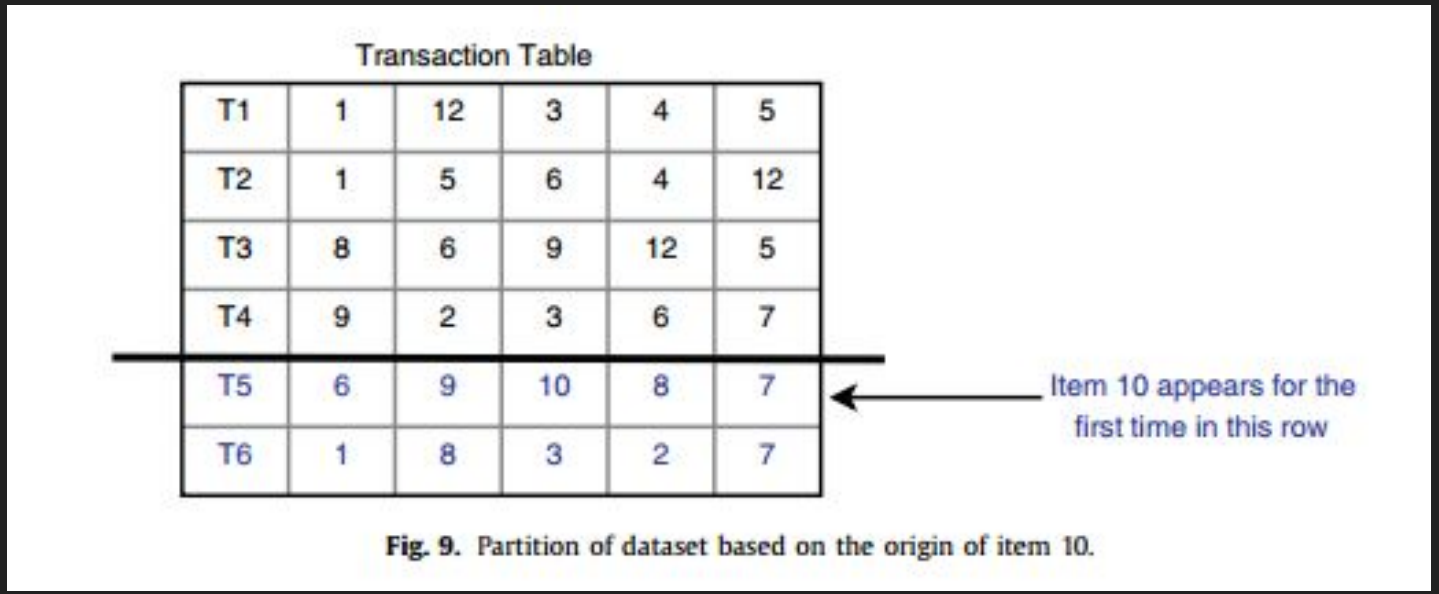
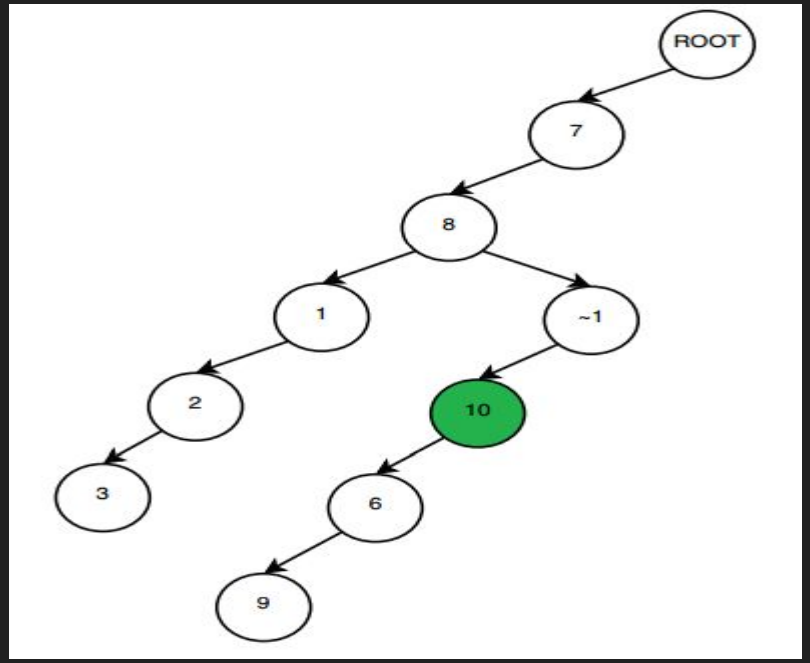


Fig. 9. Partition of dataset based on the origin of item 10.

Second Contribution

OIMASP Tree



(7, 8, ~1) => (10)
(7, 8, ~1, 10) => (6)
(7, 8, ~1, 10, 6) => (9)

OOIMASP [Origin+Order+MASP]

Algorithm 3: OOIMASP algorithm.

```
1  function OOIMASP (Γ, τ_s, τ_c);
   Input  : A transaction dataset Γ[N][M], threshold support τ_s
            andthreshold confidence τ_c
   Output: association rules
2  items[1 : j] ← unique items list
3  origins[1 : j] ← origin of corresponding items in items
4  globalRules ← {}
5  for i in 1 : j do
6  |   currentItem ← items[i]
7  |   itemOrigin ← origins[i]
8  |   if OIMASP tree is not yet generated for origin itemOrigin
   |   then
9  |   |   generate OIMASP tree for dataset = Γ[itemOrigin : N]
   |   |   and given τ_s, τ_c
10 |   end
11 |   generate all association rules from OIMASP tree having
   |   origin = itemOrigin, which contains item currentItem and
   |   add these to globalRules
12 end
13 return globalRules;
```

OOIMASP vs MASP

			MASP		OOIMASP	
A	0.01	0.30	14	7	24	7
B	0.01	0.30	0	0	85	21
C	0.01	0.30	0	0	133	41
D	0.01	0.30	0	0	201	101
E	0.01	0.30	0	0	1690	846
Connect	0.01	0.30	50	38	192	45
Mushroom	0.01	0.30	144	23	282	24
Chess	0.01	0.30	50	33	91	33
A	0.05	0.10	24	7	28	7
B	0.05	0.10	394	27	493	22
C	0.05	0.10	45	12	207	37
D	0.05	0.10	0	0	151	86
E	0.05	0.10	0	0	920	461
Connect	0.05	0.10	56	35	211	45
Mushroom	0.05	0.10	85	19	129	20
Chess	0.05	0.10	53	27	83	27

Conclusions

- OOIMASP generates same association rules irrespective of the order of items
- OOIMASP will not generate contradictory rules
- OOIMASP outperforms MASP in terms of both the metrics i.e. #rules and longest rule size
- OOIMASP requires more computational resources as compared to the MASP algorithm

Conclusions

- Future work : Parallelize OOIMASP algorithm