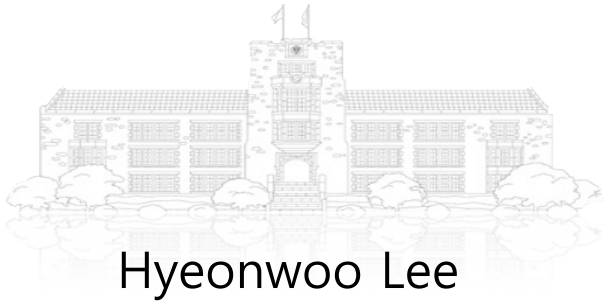
2021 MLP

Robotic Mobile Fulfillment System

Yonsei University
Machine Learning Project Team 09
2021.04.20



Hyeonwoo Lee
Jieun Lee
Segul Roh
Changyong Kim

Contents

1. Project Introduction – RMFS

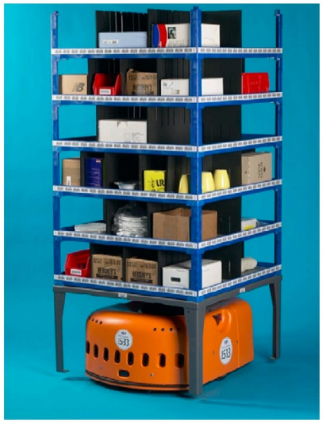
❖ Robotic Mobile Fulfillment System (**RMFS**)

- ✓ Mobile robot in logistics industry (Untact technology)
- ✓ Deliver products from replenishment to pick up station

- ✓ Project Goal: Increase efficiency of the warehouse(RMFS) by analyzing the following:
  - RPS[1]: Efficiency of different **product deployment** by association analysis
  - PSA[2]: Efficiency of **multi-robot path planning** considering pick order, pod's constitution

  ⇒ **Minimize total time & Maximize total throughput**

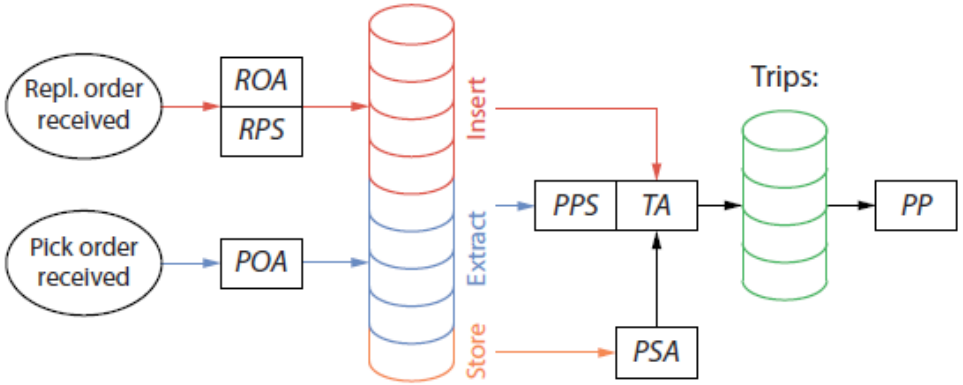
- ✓ Using a recently developed RMFS simulator (RAWSim-O)



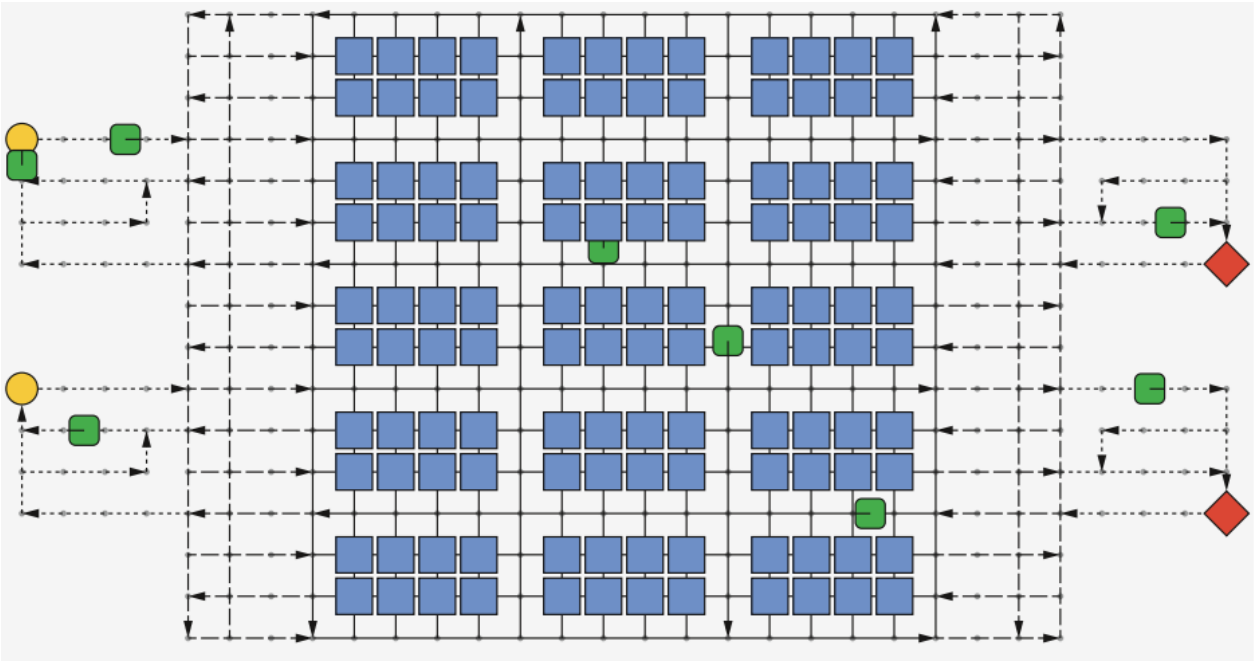
Robot carrying a Pod

1 Replenishment Pod Selection
2 Pod Storage Assignment

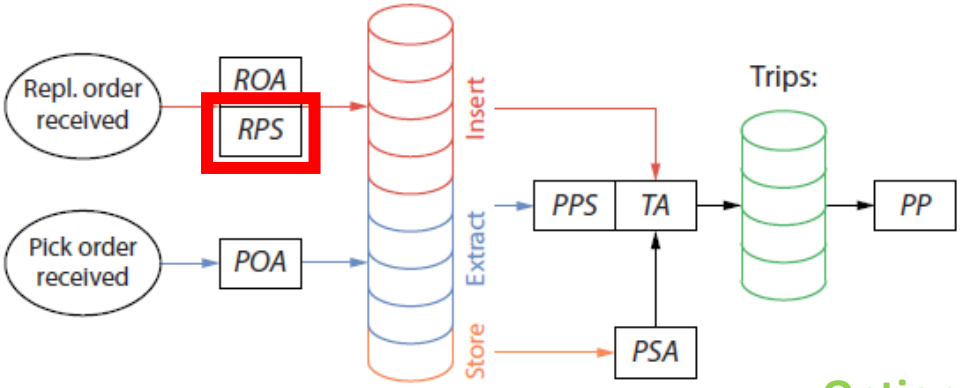
1. Project Introduction – RMFS



ROA : Replenishment Order Assignment
RPS : Replenishment Pod Selection
POA : Pick Order Assignment
PPS : Pick Pod Selection
PSA : Pod Storage Selection
TA : Task Allocation
PP : Path Planning



2. Problem formulation - RPS



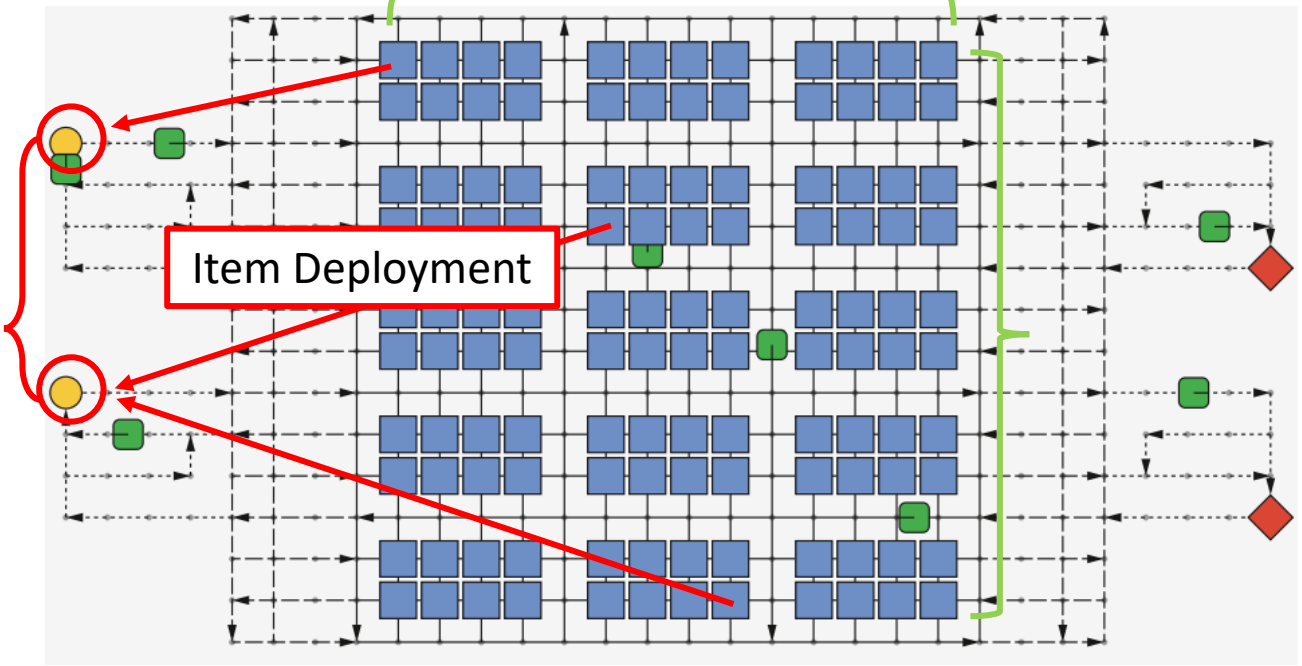
Dataset

↓

Association analysis

Options

m×n matrix



2. Problem formulation - RPS

❖ Replenishment Pod Selection(RPS)

- ✓ Goal: Select an appropriate pod after order assignment
- ✓ Default option : Random, Emptiest, Nearest, Least-Demand, Class

  ⇒ **No RPS decision rule based on item Similarity in RAWSim-O**

- ✓ Existing research[1]
  - Calculate Similarity value of Items from Frequency (Clustering)
  - Maximize the Sum of similarity values

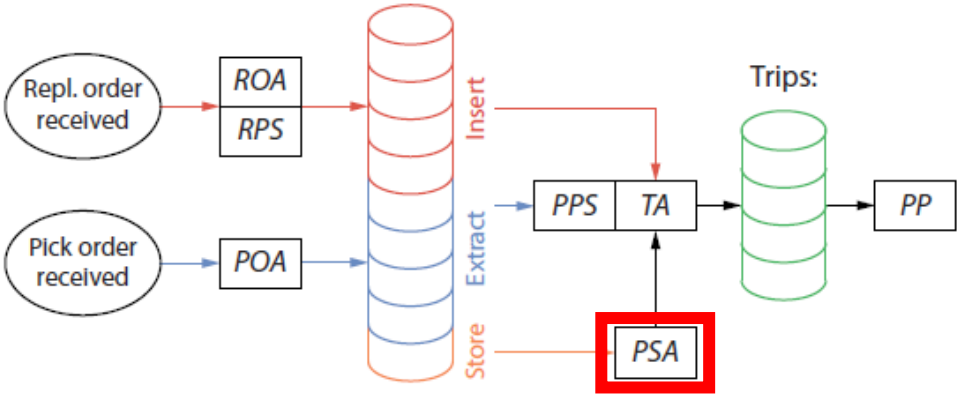
  ⇒ **Utilized only simple cases , Not real-life data**

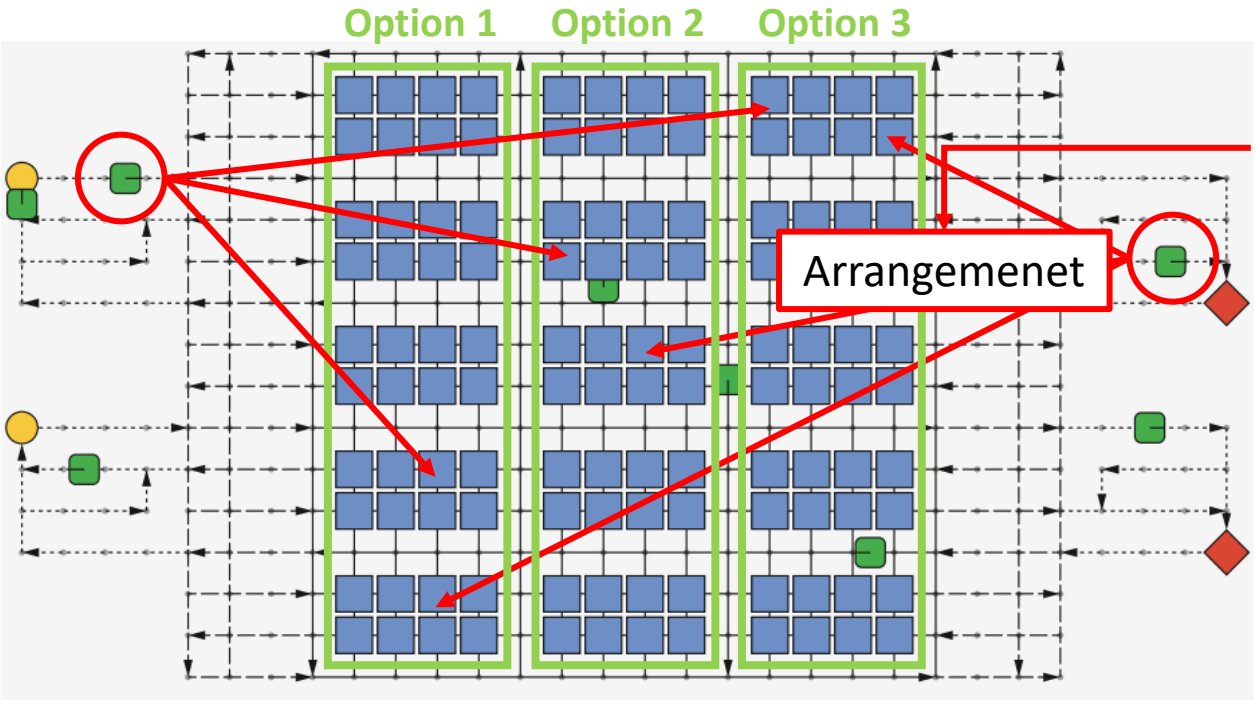
Assignment of five items.

- ✓ Our project
  - Modify item distribution on RAWSim-O
    - SKU size, item distribution, Order distribution
    - Based on Real Dataset
  - Add decision rule of Controllers

1 Kim, H. J., Pais, C., & Shen, Z. J. M. (2020). Item Assignment Problem in a Robotic Mobile Fulfillment System. *IEEE Transactions on Automation Science and Engineering*, *17*(4), 1854-1867.

2. Problem formulation - PSA





Order
&
Pods constitution

2. Problem formulation - PSA

❖ Pod Storage Assignment (PSA)

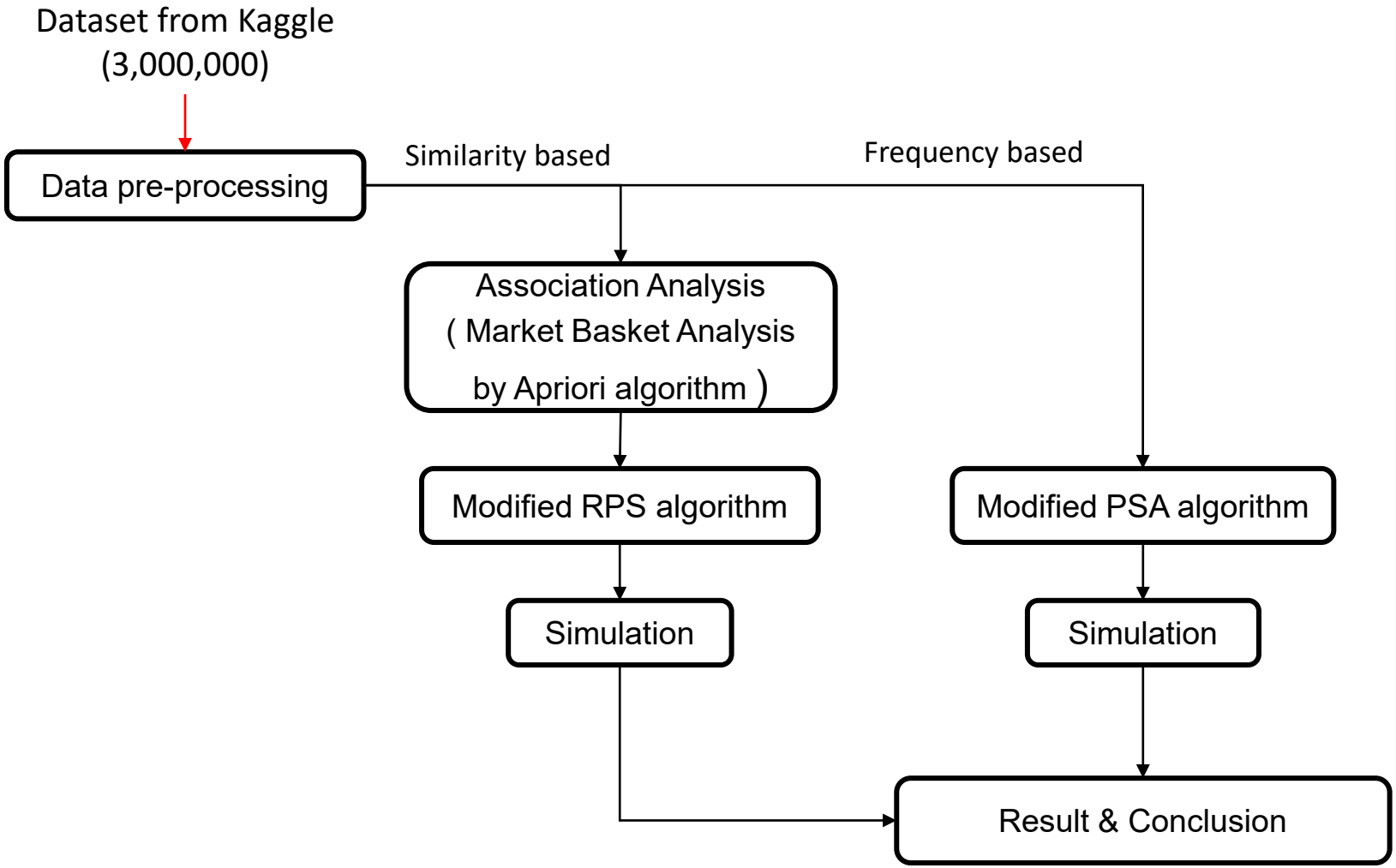
- ✓ Goal: Determine the most efficient storage location when pods go back

- ✓ Default option (Limitation)
  - Random, Fixed, Nearest, Station-Based, Class

- ✓ Our project:
  - Input(Features)
    - Frequency(# of pick order and replenishment order)
    - Remaining items in a Pod
  - Output
    - 3 part of the storage area

2. Problem formulation

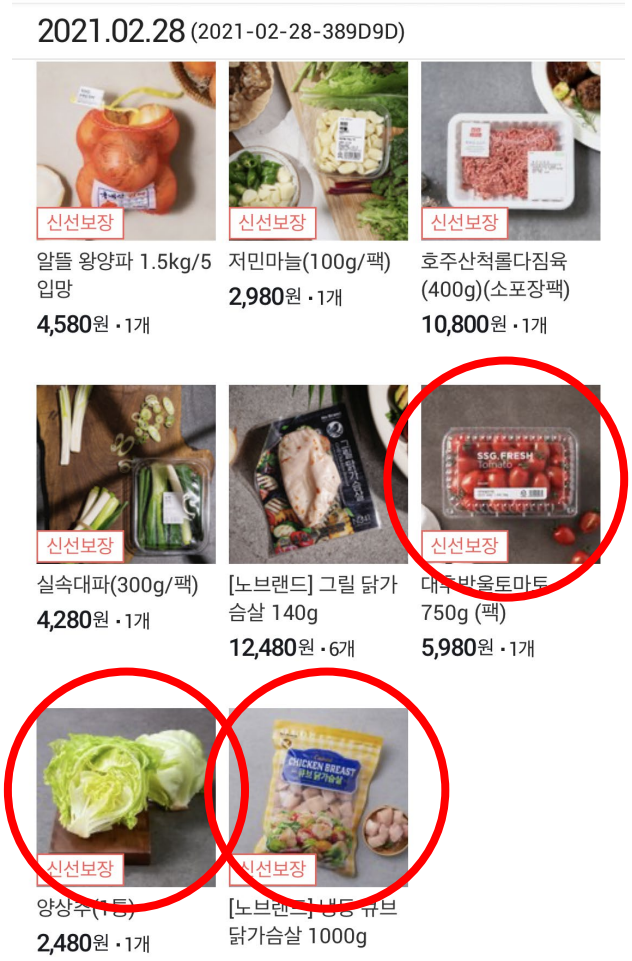
❖ Flow chart of the Project

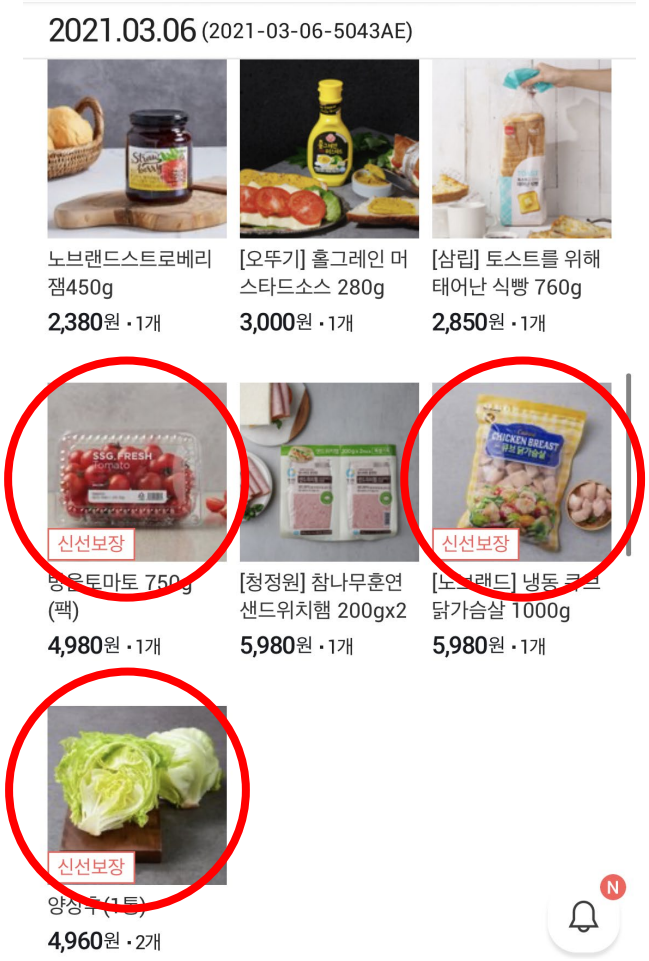


3. Data

❖ Desired Dataset

✓ Market basket Data





3. Data

- Instacart Market Basket Dataset
  - ✓ Instacart: A grocery delivery and pick-up service in the United States
    - A sample of over 3 million grocery orders
    - Contains:
      - User id
      - Order id
      - Product name & id
      - Aisle name & id
      - Department name & id
      - Reordered
      - etc

3. Data

❖ Instacart Market Basket Dataset

aisle_id	aisle	department_id	department
1	prepared soups salads	1	frozen
2	specialty cheeses	2	other
3	energy granola bars	3	bakery
4	instant foods	4	produce
5	marinades meat preparation	5	alcohol
6	other	6	international
7	packaged meat	7	beverages
8	bakery desserts	8	pets
9	pasta sauce	9	dry goods pasta

< Aisle & Department ID >

product_id	product_name	aisle_id	department_id
1	Chocolate Sandwich Cookies	61	19
2	All-Seasons Salt	104	13
3	Robust Golden Unsweetened C	94	7
4	Smart Ones Classic Favorites M	38	1
5	Green Chile Anytime Sauce	5	13
6	Dry Nose Oil	11	11
7	Pure Coconut Water With Oran	98	7
8	Cut Russet Potatoes Steam N' I	116	1
9	Light Strawberry Blueberry Yog	120	16

< Product Name & ID >

order_id	product_id	add_to_cart_order	reordered
1	49302	1	1
1	11109	2	1
1	10246	3	0
1	49683	4	0
1	43633	5	1
1	13176	6	0
1	47209	7	0
1	22035	8	1

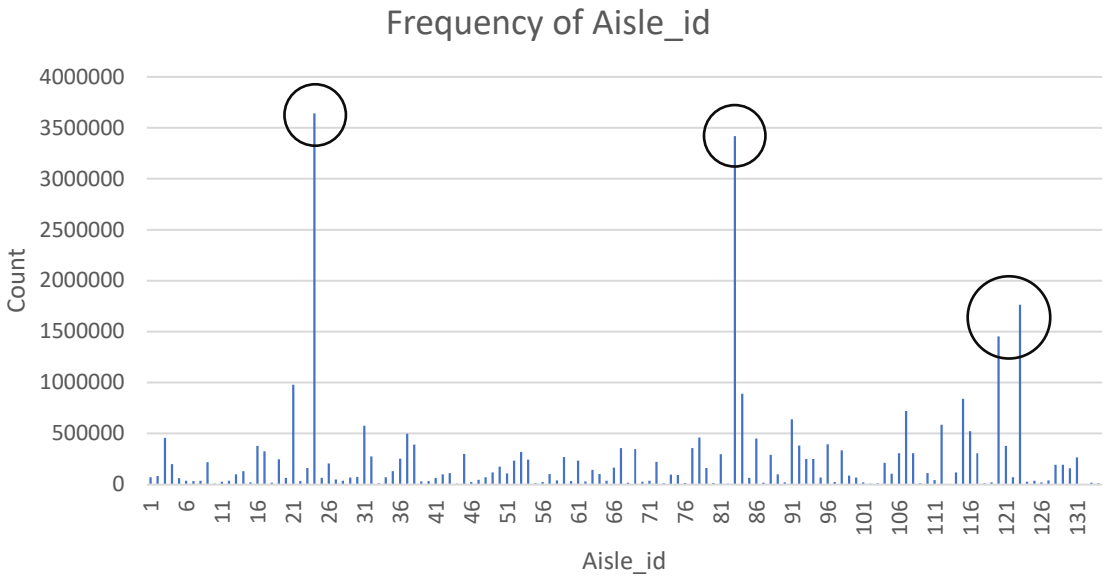
< Orders >

3. Data

- Data Preprocessing
  - ✓ Integrate overlapping products in one group
    - Classify products in same categories called aisle_id
    - Same order_id = Same user's market basket
  - ✓ Expected Problem
    - A few of group(aisle_id) showed too high frequency

One Market Basket

order_id	product_id	aisle_id
431534	13176	24
431534	41787	24
431534	17122	24
431534	10326	24
431534	25133	21
431534	196	77
431534	12427	23
431534	10258	117
473747	30450	88
473747	25133	21



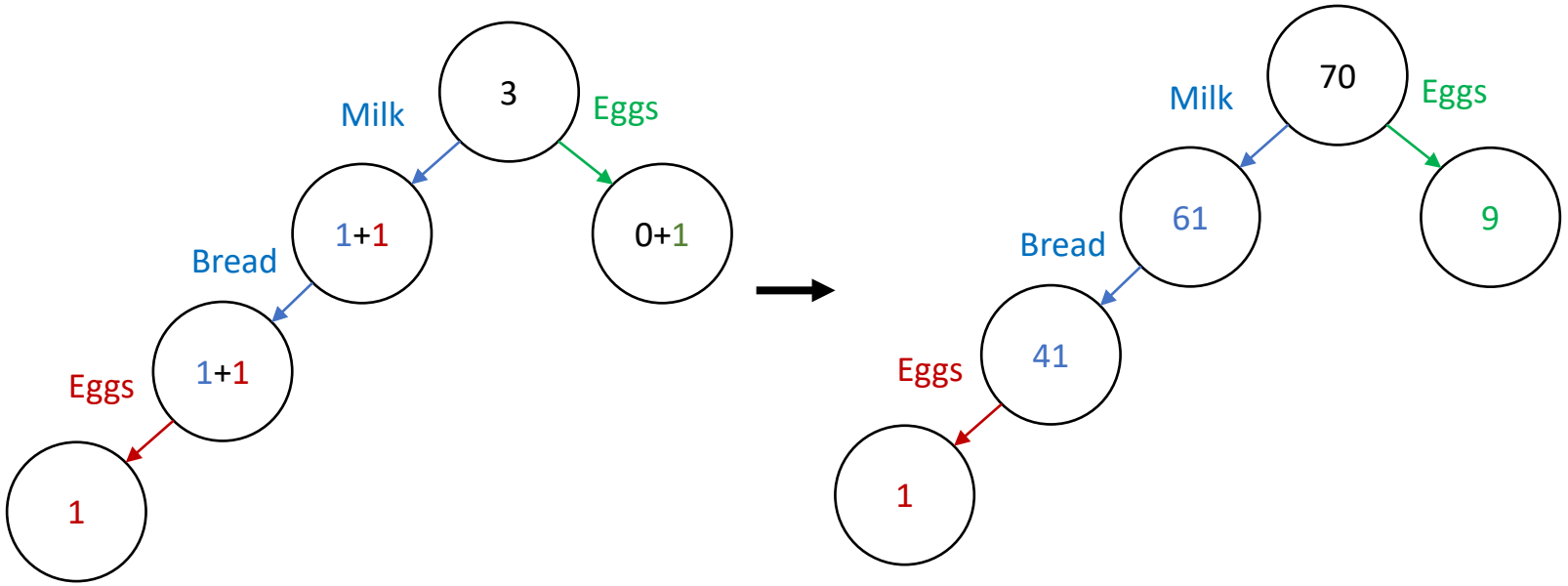
4. Method

❖ Apriori algorithm

- ✓ Frequent item set mining and association rule learning over relational databases
- ✓ Process transactions for frequent items
- ✓ Too many frequent itemsets & Too much time consuming

❖ FP-Growth Algorithm

- ✓ Consider every item
- ✓ Compute the support value easily



4. Method

❖ Improved Apriori Algorithm[1]

- ✓ Reduce computation time than Apriori algorithm
- ✓ The smaller the number of combination, The better

❖ Steps

- ✓ Set minimum support value
- ✓ Evaluate frequent k based on first frequent 1
- ✓ Repeat until there isn't new frequent items

```
//Generate items, items support, their transaction ID
(1) L1 = find_frequent_1_itemsets (T);
(2) For (k = 2; Lk-1 ≠Φ; k++) {
//Generate the Ck from the LK-1
(3) Ck = candidates generated from Lk-1;
//get the item Iw with minimum support in Ck using L1,(1≤w≤k).
(4) x = Get _item_min_sup(Ck, L1);
// get the target transaction IDs that contain item x.
(5) Tgt = get_Transaction_ID(x);
(6) For each transaction t in Tgt Do
(7) Increment the count of all items in Ck that are found in Tgt;
(8) Lk= items in Ck ≥ min_support;
(9) End;
(10) }
```

Items	support	T_IDs	
I_1	6	$T_1, T_4, T_5, T_7, T_8, T_9$	
I_2	7	$T_1, T_2, T_3, T_4, T_6, T_8, T_9$	
I_3	5	T_5, T_6, T_7, T_8, T_9	
I_4	3	T_2, T_3, T_4	
I_5	2	T_1, T_8	deleted

→

Items	support	Min	Found in	
I_1, I_2	4	I_1	$T_1, T_4, T_5, T_7, T_8, T_9$	
I_1, I_3	4	I_3	T_5, T_6, T_7, T_8, T_9	
I_1, I_4	1	I_4	T_2, T_3, T_4	deleted
I_2, I_3	3	I_3	T_5, T_6, T_7, T_8, T_9	
I_2, I_4	3	I_4	T_2, T_3, T_4	
I_3, I_4	0	I_4	T_2, T_3, T_4	deleted

→

Items	support	Min	Found in	
I1, I2 , I3	2	I_3	T_5, T_6, T_7, T_8, T_9	deleted
I1, I2 , I4	1	I_4	T_2, T_3, T_4	deleted
I1, I3 , I4	0	I_4	T_2, T_3, T_4	deleted
I2, I3 , I4	0	I_4	T_2, T_3, T_4	deleted

1 Al-Maolegi, M., & Arkok, B. (2014). An improved Apriori algorithm for association rules. *arXiv preprint arXiv:1403.3948*.

4. Method

❖ Process the datasets using Improved Apriori algorithm

- ✓ Find every itemset(group) of three of 134 items
- ✓ Based on frequency of each items, Delete combinations which have minimum frequency
- ✓ Find maximum sum of group support value $C_{i,j,k}$ $\Rightarrow$ $$argmax \sum_{k}^{n} \sum_{j}^{n} \sum_{i}^{n} C_{i,j,k}$$

❖ TODO list

- ✓ Insert the **combination** and **item distribution** on RAWSim-O
  - Input: Bundle size, Frequency values of group and one item, Weight
- ✓ Add **RPS(Replenishment Pod Selection)** **decision rule** of Controllers
  - Allocate one group in one pod
  - The number of groups is same to that of pods
- ✓ Process deleted item
- ✓ Whether saving computation time is good for us

5. Expected Outcomes

- Expected Outcomes
  - ✓ Increase efficiency compared to the conventional method of RAWSim-O
    - Maximize total **throughput**
      - Related three items will be contained in one pod
      - 1 order, 1 movement of robot
    - Minimize total **time**
      - Reducing unnecessary movement of robots