การกำหนดขนาดออเดอร์และออกแบบลำดับการหยิบที่เหมาะสมต่อการลดระยะทางการหยิบสินค้า

An efficient determination of order batching and design of picking sequence towards order-picking distance

อริสรา ผิวนวล, สุรัติ คงโสม และอัครนันท์ พงศธรวิวัฒน์
สาขาการจัดการโลจิสติกส์ คณะสถิติประยุกต์
สถาบันบัณฑิตพัฒนบริหารศาสตร์

หัวข้อ

1. ลักษณะของธุรกิจ
2. ที่มาและความสำคัญของปัญหา
3. ขอบเขตและวัตถุประสงค์
4. วิธีดำเนินงาน
5. สรุปผลการศึกษา
6. ข้อเสนอแนะ

ลักษณะธุรกิจ III

บริษัทกรณีศึกษาดำเนินธุรกิจจำหน่ายผลิตภัณฑ์จำพวกเนื้อสัตว์ อาหารทะเล และผลิตภัณฑ์จากนม โดยมีคลังสินค้าอยู่ 2 คลัง ได้แก่ คลังสินค้าแช่เย็น และคลังสินค้าแช่แข็ง ปัจจุบันผลิตภัณฑ์ของธุรกิจโดยเฉพาะเนื้อสัตว์มีความต้องการเพิ่มขึ้นอย่างรวดเร็ว เนื่องด้วยการเปิดร้านอาหารที่เน้นไปที่การรับประทานอาหารแบบบริการตนเองอย่างไม่จำกัด อิทธิพลจากสื่อการตลาด และมาตรการผ่อนคลายการควบคุมการแพร่กระจายของโรคระบาดโควิด-19 สนับสนุนให้ประชากรสามารถดำเนินชีวิตภายนอกครัวเรือนได้ ส่งผลให้ยอดขายผลิตภัณฑ์เนื้อสัตว์แช่แข็งของบริษัทกรณีศึกษาที่มีลูกค้าทั้งในประเทศและต่างประเทศมีแนวโน้มเพิ่มสูงขึ้น

ที่มาและความสำคัญของปัญหา

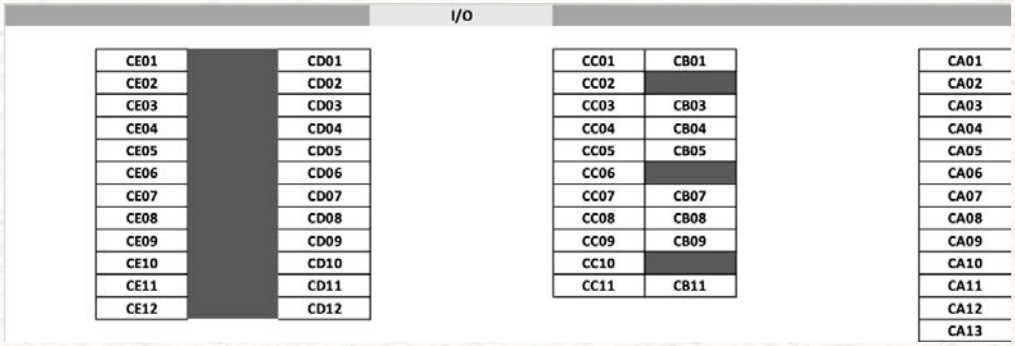
จากอัตราความต้องการที่สูงขึ้น บริษัทกรณีศึกษาประสบปัญหาการส่งสินค้าล่าช้าเนื่องจากการหยิบสินค้าที่ล่าช้า ปัจจุบันการหยิบสินค้าเป็นรูปแบบหยิบทีละพาเลท (Pallet) และทีละเที่ยว (A single unit-load protocol) ทำให้เกิดระยะทางการหยิบสินค้าที่มากเกินความจำเป็น การหยิบสินค้าถือเป็นกิจกรรมที่ใช้ระยะเวลามากที่สุดในคลังสินค้า รวมถึงต้นทุนในการบริหารจัดการคลังสินค้าส่วนใหญ่เป็นต้นทุนที่เกิดจากกระบวนการหยิบสินค้ามากถึง 55%

ดังนั้น การเพิ่มประสิทธิภาพการหยิบสินค้าจึงมีความสำคัญเป็นอย่างมาก ในการลดระยะทางในการเดินหยิบสินค้าเพื่อลดต้นทุนในการจัดการคลังสินค้าและเพิ่มระดับความพึงพอใจของลูกค้ายิ่งขึ้น

ขอบเขตและวัตถุประสงค์ III

ขอบเขตการศึกษา:

ข้อมูลออเดอร์ของลูกค้า 3 เดือนย้อนหลัง โดยเริ่มต้นตั้งแต่ พฤษภาคม - กรกฎาคม 2565 จำนวน 5,712 ออเดอร์ สินค้าทั้งหมด 56 SKUs และสนใจการจัดเก็บที่บริเวณหยิบสินค้าไว (Forward Pick Area) จำนวน 56 locations



วัตถุประสงค์:

1. เพื่อศึกษาผลของรูปแบบการจัดวางที่ส่งผลต่อระยะทางในการหยิบสินค้า
2. เพื่อศึกษารูปแบบขนาดออเดอร์และลำดับการหยิบที่เหมาะสมต่อการลดระยะทางการหยิบสินค้า

วิธีดำเนินงาน III

วิเคราะห์ข้อมูล

จำแนกกลุ่มของสินค้าตามการวิเคราะห์แบบเอบีซี (ABC Analysis)

การปรับตำแหน่งจัดเก็บ

จัดสรรสินค้าเข้าไปในพื้นที่จัดเก็บสินค้า โดยใช้วิธี Class Based Model (Within Aisle Model และ Diagonal Model) และ การประยุกต์ใช้ตัวแบบทางคณิตศาสตร์เชิงเส้น (Storage Location Assignment Problem)

การปรับวิธีการหยิบ

การออกแบบรูปแบบการหยิบแบบแบต (Bath) และลำดับการหยิบ โดยใช้วิธี Roulette Wheel Simulation และ Genetic Algorithm

วิเคราะห์ข้อมูล

ABC Analysis

จำแนกกลุ่มของสินค้าคงคลังตามการวิเคราะห์แบบเอบีซี

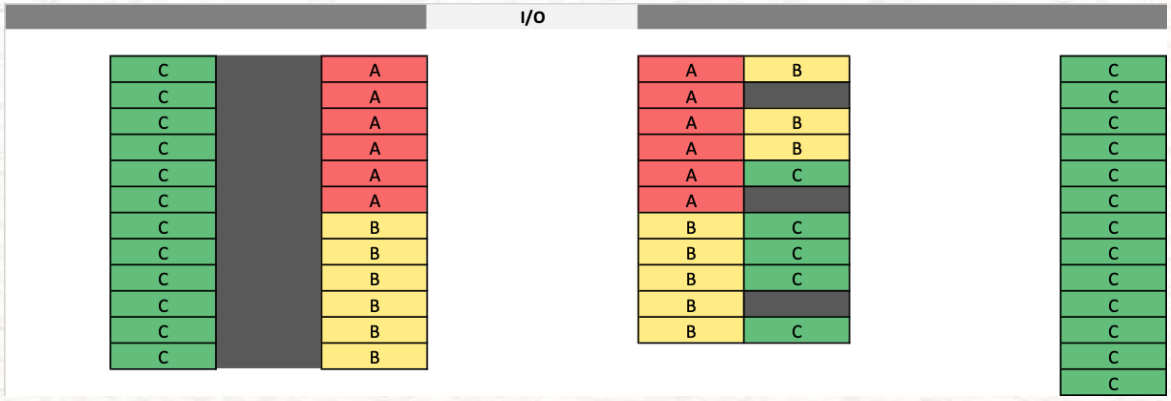
สินค้าจำนวน 56 SKUs นำมาคัดเลือกจาก pick list
เพื่อทำการจัดลำดับและจัดกลุ่ม โดยใช้ความถี่ในการหยิบสินเป็นตัวแบ่ง

From	To	Class	Number of items	%
0%	50%	A	11	20%
50%	80%	B	15	27%
80%	100%	C	30	54%

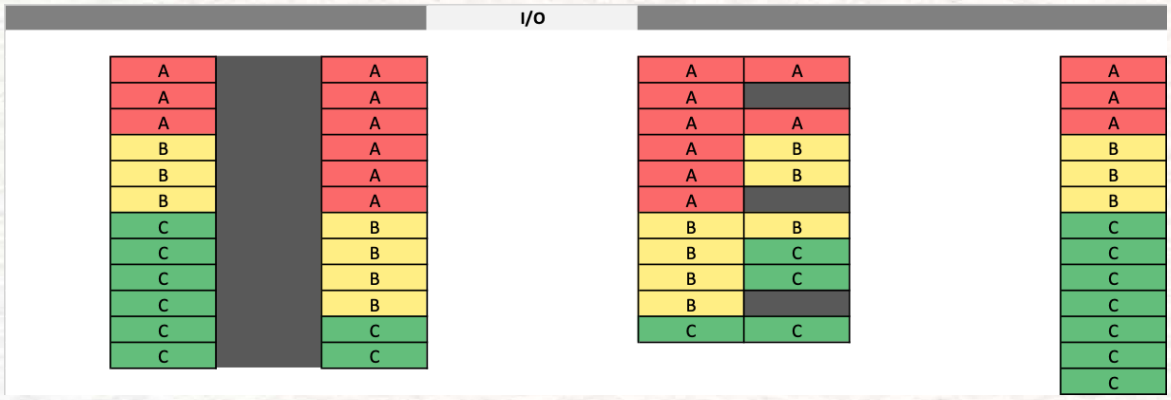
No.	SKUs	Frequency	Prob	Cumulative	Class
1	10400089	682	0.09666903	0.09666903	A
2	10300134	365	0.05173636	0.14840539	A
3	10304243	359	0.0508859	0.19929128	A
4	10300889	324	0.04592488	0.24521616	A
5	10301409	301	0.04266478	0.28788094	A
6	10300208	289	0.04096386	0.32884479	A
7	10300120	278	0.03940468	0.36824947	A
8	10300118	217	0.03075833	0.3990078	A
9	10300125	212	0.03004961	0.42905741	A
10	10301195	194	0.02749823	0.45655563	A
11	10300286	185	0.02622254	0.48277817	A
12	10300076	182	0.02579731	0.50857548	B
13	10300152	176	0.02494685	0.53352232	B
14	10300132	174	0.02466336	0.55818568	B
15	10300061	172	0.02437987	0.58256556	B
16	10300364	163	0.02310418	0.60566974	B
17	20100225	163	0.02310418	0.62877392	B
18	10301160	150	0.02126152	0.65003544	B
19	10300281	138	0.0195606	0.66959603	B
20	10300807	133	0.01885188	0.68844791	B
21	20100051	120	0.01700921	0.70545712	B
22	10304117	118	0.01672573	0.72218285	B
23	20100042	117	0.01658398	0.73876683	B
24	20100037	113	0.01601701	0.75478384	B
25	10300119	110	0.01559178	0.77037562	B
26	10300191	109	0.01545004	0.78582566	B
27	20100048	106	0.01502481	0.80085046	C
28	20100006	103	0.01459957	0.81545004	C
29	10301410	92	0.0130404	0.82849043	C
30	10300172	90	0.01275691	0.84124734	C
31	10301462	72	0.01020553	0.85145287	C
32	20100337	71	0.01006378	0.86151665	C
33	20100050	65	0.00921332	0.87072998	C
34	20100344	65	0.00921332	0.8799433	C
35	10301301	62	0.00878809	0.8887314	C
36	10302942	62	0.00878809	0.89751949	C
37	10303122	56	0.00793763	0.90545712	C
38	20100336	55	0.00779589	0.91325301	C
39	10300349	50	0.00708717	0.92034018	C
40	10302938	50	0.00708717	0.92742736	C
41	10301230	42	0.00595322	0.93338058	C
42	10301913	42	0.00595322	0.93933381	C
43	20100084	39	0.00552799	0.9448618	C
44	10303863	38	0.00538625	0.95024805	C
45	20100053	35	0.00496102	0.95520907	C
46	10300045	33	0.00467753	0.95988661	C
47	10300117	33	0.00467753	0.96456414	C
48	10302853	32	0.00453579	0.96909993	C
49	10300116	31	0.00439405	0.97349398	C
50	10300320	31	0.00439405	0.97788802	C
51	10301297	31	0.00439405	0.98228207	C
52	10300325	29	0.00411056	0.98639263	C
53	20300004	29	0.00411056	0.99050319	C
54	10303657	28	0.00396882	0.99447201	C
55	20300002	21	0.00297661	0.99744862	C
56	10301300	18	0.00255138	1	C

การปรับตำแหน่งจัดเก็บ

Class Based Model



Within Aisle Model



Diagonal Model

การประยุกต์ใช้ตัวแบบทางคณิตศาสตร์เชิงเส้น

$$\text{Minimize } Z = \sum_{i=1}^{n}\sum_{j=1}^{m} F_i d_j X_{ij}$$

Subject to:

$$\sum_{i=1}^{n} X_{ij} = 1 \; ; \; \forall_j$$

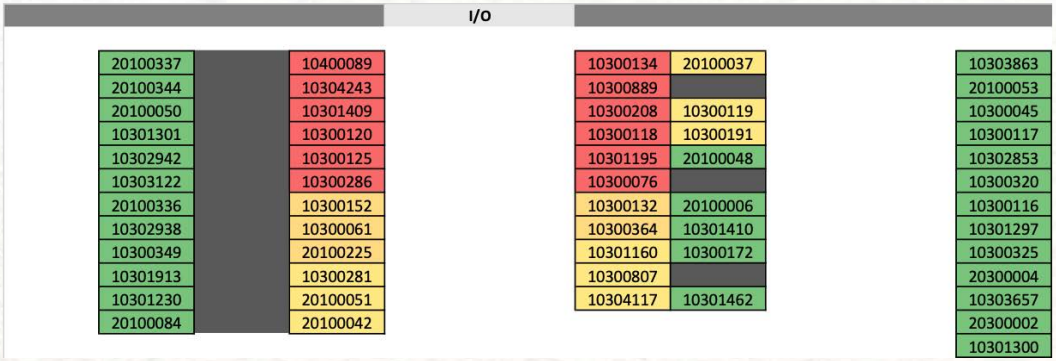
$$\sum_{j=1}^{m} X_{ij} = 1 \; ; \; \forall_i$$

$$X_{ij} = \{0,1\} \; ; \; \forall_i, \forall_j$$

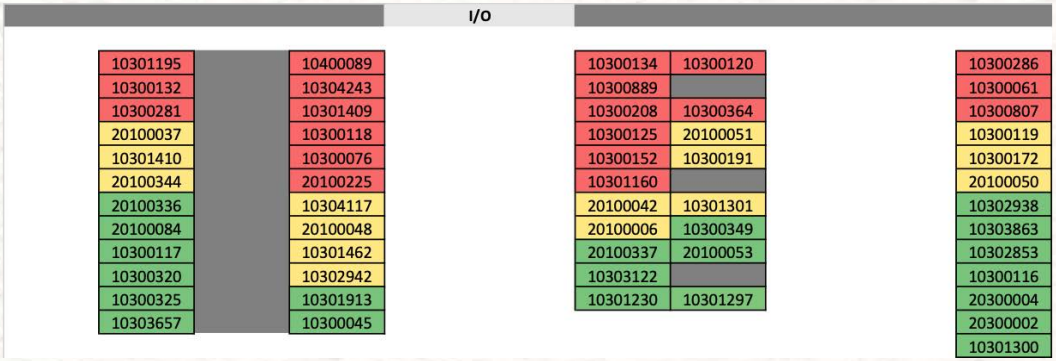
i	คือ สินค้าแต่ละ SKUs
j	คือ ตำแหน่งการจัดเก็บ (Location Numbers)
d_j	คือ ระยะทางของการจัดวางสินค้าแต่ละSKUs (i) ในแต่ละ location การจัดเก็บ (j) ถึงจุดพื้นที่วางสินค้า outbound (I/O)
F_i	คือ ความถี่การสั่งซื้อของสินค้า $SKU\ i$

ตัวแปรตัดสินใจ (Decision variable)

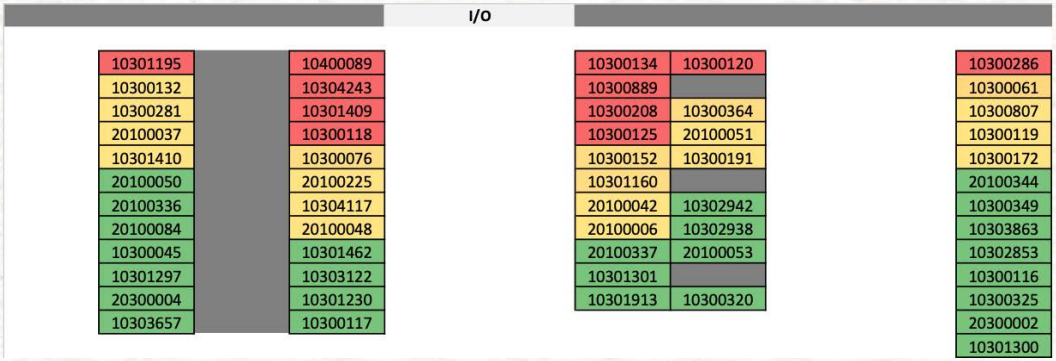
$$X_{ij} = \begin{cases} 1 \text{ , ถ้าสินค้า } i \text{ ถูกจัดเก็บใน } location\ j \\ 0 \text{ , } Otherwise \end{cases}$$



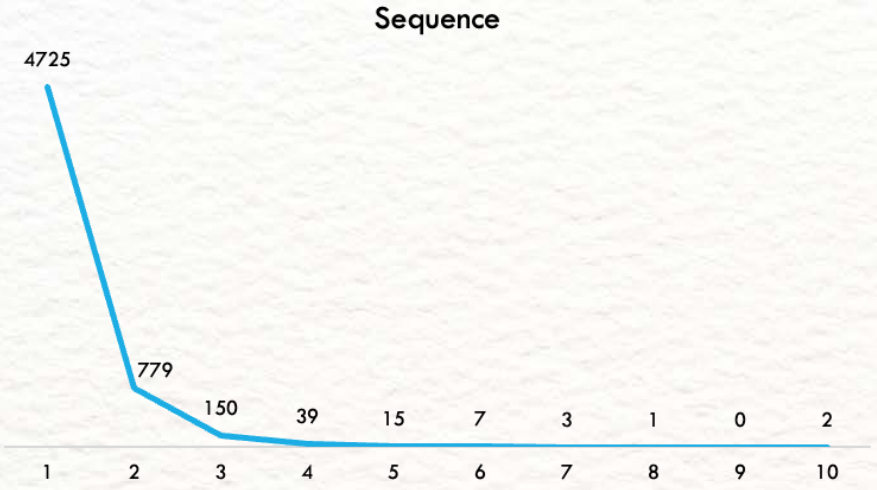
WITHIN AISLE MODEL



DIAGONAL MODEL

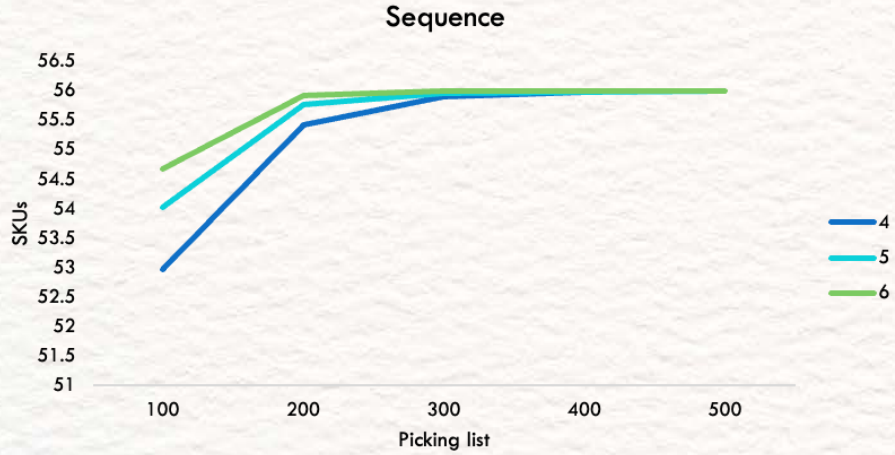


SLAP



Sequecnce	1	2	3	4	5	6	7	8	9	10
Picklist	4,725	779	150	39	15	7	3	1	0	2

จากข้อมูลลักษณะการหยิบสินค้า หยิบ ตั้งแต่ 1 - 10 sequence ต่อรายการหยิบ 1 ใบ ซึ่งพบว่ามีการหยิบ 1 sequence ค่อนข้างมาก ซึ่งส่งผลทำให้ระยะทางในการเดินหยิบสินค้าไม่มีประสิทธิภาพเท่าที่ควร



ทดลองสร้างรายการหยิบตั้งแต่ 4 - 6 sequence จำนวนครั้งที่ทดลอง 100 ครั้ง โดยสร้างรายการหยิบสินค้าตั้งแต่ 100 - 500 ใบ เพิ่มขึ้นทีละ 100 ใบ เพื่อค้นหา sequence ที่เหมาะสมที่ทำให้หยิบสินค้าได้ครบทุก SKUs ก่อน

Order-Batching Problem

$$\text{Minimize } Z = \sum_{i \in I} d_i x_i$$

Subject to:

$$\sum_{i \in I} a_{ij} x_i = 1, \ \forall_j \in J$$

$$X_i = \{0,1\}, \ \forall_i \in I$$

ตัวแปรตัดสินใจ (Decision variable)

$$X_i \begin{cases} 1 = \text{ถ้าเลือก batch } i \\ 0 = \text{Otherwise} \end{cases}$$

J คือ ชุดคำสั่งซื้อของลูกค้า = {1,2,3,...,n}

C_j คือ จำนวนสินค้าในคำสั่งซื้อของลูกค้า j โดยที่ $j \in J$

I คือ ชุดของ *batch* เป็นไปได้ทั้งหมด

d_i คือ ระยะทางในการเลือกสินค้าซึ่งทุกรายการอยู่ใน batch i โดยที่ $i \in I$

a_j คือ ค่าเวกเตอร์ของตัวแปรแบบไบนารี a_{ij} โดย

$$a_j = \begin{cases} 1 \text{ , ถ้าคำสั่งซื้อของลูกค้า } j \text{ ถูกเลือกรวมใน batch } i \\ 0 \text{ , } Otherwise \end{cases}$$

ดังนั้น เซตของ Batch ที่เป็นไปได้ทั้งหมดถูกกำหนดให้อยู่ภายใต้เงื่อนไขของความจุของอุปกรณ์หยิบสินค้า C โดยมีเงื่อนไขดังต่อไปนี้

$$\sum_{j \in J} c_j a_{ij} \leq C \text{ , } \forall_i \in I$$

Simulation Order Batching III

Brk	SKU	Frequency	Prob
0	10400089	682	0.09666903
0.09666903	10300134	365	0.05173636
0.14840539	10304243	359	0.0508859
0.19929128	10300889	324	0.04592488
0.24521616	10301409	301	0.04266478
0.28788094	10300208	289	0.04096386
0.32884479	10300120	278	0.03940468
0.36824947	10300118	217	0.03075833
0.3990078	10300125	212	0.03004961
0.42905741	10301195	194	0.02749823
0.45655563	10300286	185	0.02622254
0.48277817	10300076	182	0.02579731
0.50857548	10300152	176	0.02494685
0.53352232	10300132	174	0.02466336
0.55818568	10300061	172	0.02437987
0.58256556	10300364	163	0.02310418
0.60566974	20100225	163	0.02310418
0.62877392	10301160	150	0.02126152
0.65003544	10300281	138	0.0195606
0.66959603	10300807	133	0.01885188
0.68844791	20100051	120	0.01700921
0.70545712	10304117	118	0.01672573
0.72218285	20100042	117	0.01658398
0.73876683	20100037	113	0.01601701
0.75478384	10300119	110	0.01559178
0.77037562	10300191	109	0.01545004
0.78582566	20100048	106	0.01502481
0.80085046	20100006	103	0.01459957
0.81545004	10301410	92	0.0130404
0.82849043	10300172	90	0.01275691
0.84124734	10301462	72	0.01020553
0.85145287	20100337	71	0.01006378
0.86151665	20100050	65	0.00921332
0.87072998	20100344	65	0.00921332
0.8799433	10301301	62	0.00878809
0.8887314	10302942	62	0.00878809
0.89751949	10303122	56	0.00793763
0.90545712	20100336	55	0.00779589
0.91325301	10300349	50	0.00708717
0.92034018	10302938	50	0.00708717
0.92742736	10301230	42	0.00595322
0.93338058	10301913	42	0.00595322
0.93933381	20100084	39	0.00552799
0.9448618	10303863	38	0.00538625
0.95024805	20100053	35	0.00496102
0.95520907	10300045	33	0.00467753
0.95988661	10300117	33	0.00467753
0.96456414	10302853	32	0.00453579
0.96909993	10300116	31	0.00439405
0.97349398	10300320	31	0.00439405
0.97788802	10301297	31	0.00439405
0.98228207	10300325	29	0.00411056
0.98639263	20300004	29	0.00411056
0.99050319	10303657	28	0.00396882
0.99447201	20300002	21	0.00297661
0.99744862	10301300	18	0.00255138

Sequence	6						
	Item	1	2	3	4	5	6
Picklist	1	20100337	10300120	10400089	10304243	10300116	10300152
Picklist	2	10300076	10300364	10304243	10300125	10301160	20100006
Picklist	3	10300172	10300076	10300076	20100037	10300172	10300076
Picklist	4	10300364	10301409	10301195	10301409	10400089	10400089
Picklist	5	10300076	10301301	10300320	10300134	10300889	10300134
Picklist	6	10300807	10300320	10300889	20100006	20100048	10304243
Picklist	7	10301297	10300208	20100050	10400089	10300125	20300004
Picklist	8	10300118	10300364	20100048	10300125	10301195	10300286
Picklist	9	20100042	20100336	10301195	10300889	10300125	10300076
Picklist	10	10300118	10300076	10300152	10300118	10304117	10300125
Picklist	11	10301409	10304243	10300076	10300125	10300061	10300134
Picklist	12	10304243	10300061	10300325	20100344	10304117	10300208
Picklist	13	10300807	10300889	10300061	20100225	10300119	10300152
Picklist	14	10300286	10300208	10304243	10301195	10300281	10300134
Picklist	15	10400089	10300889	20100225	10300061	10302938	10300152
Picklist	16	10304243	10301301	10304117	10303657	10300125	10301409
Picklist	17	10301409	10300134	10400089	10300125	10300116	10300889
Picklist	18	10300076	10302853	20100336	10301913	10300208	10300191
Picklist	19	20100344	10300208	10301409	10304243	10300120	10300061
Picklist	20	10300076	10400089	10300208	10301301	10300076	20100050
Picklist	21	10300125	10304243	10304117	10300119	10300132	10300120
Picklist	22	10301409	20100225	10300125	10301195	10300134	20100048
Picklist	23	10400089	10302942	10300889	10301409	20300002	10301462
Picklist	24	10300134	10300172	10300061	20100050	10300152	20100037
Picklist	25	10300134	10300132	10304243	10300132	10300118	10304243
Picklist	26	10300134	10300320	10304117	10300134	10300118	20100050
Picklist	27	10301409	10300076	10300134	10300208	20100048	10301301
Picklist	28	10300889	10300061	10302942	20100344	10300286	10304243
Picklist	29	20100042	10301409	10300118	20100042	10301195	10300061
Picklist	30	10301230	10300889	10300889	10400089	10303122	10300120
Picklist	31	10303657	10304243	10300120	10300807	10300172	10304117
Picklist	32	10300191	10302938	10300889	10300807	10300132	10300120
Picklist	33	10400089	20100337	10302942	10301410	10304243	10300281
Picklist	34	10300119	10300116	10300172	10300889	10300120	10300281
Picklist	35	20100225	20100337	10304243	10300281	10300076	10300889
Picklist	36	10300191	10301195	10303657	20100336	10300286	10300286
Picklist	37	10300134	10300061	10300118	10300061	10300364	10300281
Picklist	38	10302853	20100051	10300118	10300889	20100344	10300120
Picklist	39	10300191	10300889	10300134	10400089	20100037	10300076
Picklist	40	10304243	10300208	10301300	10301409	10302853	10300286
Picklist	41	10300045	10300286	10304117	10301409	10300120	10400089
Picklist	42	10400089	10301410	10304243	20100042	10300134	10300286
Picklist	43	10300118	10300118	10300208	10303657	10300120	10300120
Picklist	44	10300134	10300286	10301160	20100037	10300118	10300132
Picklist	45	20100050	10304243	10303863	10300152	10304243	10301409
Picklist	46	20100225	10302853	10300889	20100006	10304243	10300172
Picklist	47	10301301	20100225	10300286	10300286	10301462	10300119
Picklist	48	20100337	20100344	10301195	10300119	10300286	20100037
Picklist	49	10301409	10301409	10302938	10302942	10301409	10300349
Picklist	50	10304243	10300134	10300281	10300134	10300076	10300134
Picklist	51	10301462	10300061	10300118	10300134	20100051	10301160
Picklist	52	20100048	10300152	10300152	10300125	20100042	10300807
Picklist	53	10300208	20100336	10400089	10300125	10301462	10301409
Picklist	54	10304117	10301195	20100042	10300116	10300118	20100051
Picklist	55	10300120	20100037	10400089	20100037	10300120	10300208

Simulation Order Batching 6 Sequence โดยการสุ่มเลือก SKUs ด้วยวิธีการสุ่มแบบวงล้อรูเล็ต (roulette wheel selection) โดย SKUs ไหนที่มีความถี่ในการหยิบสูงมีโอกาสที่จะถูกสุ่มเลือกเขามาใส่ในการสร้าง Order batching มากกว่า SKUs ที่มีความถี่ในการหยิบน้อยกว่า

Genetic Algorithm (GA)

Mutation Probability

โดยทั่วไปค่าความน่าจะเป็นของการปรับเปลี่ยนยีนภายในโครโมโซม จะถูกกำหนดให้อยู่ในช่วง 0-1

Generations

เงื่อนไขในการทำงานที่กำหนดให้วนซ้ำตามจำนวนรอบ

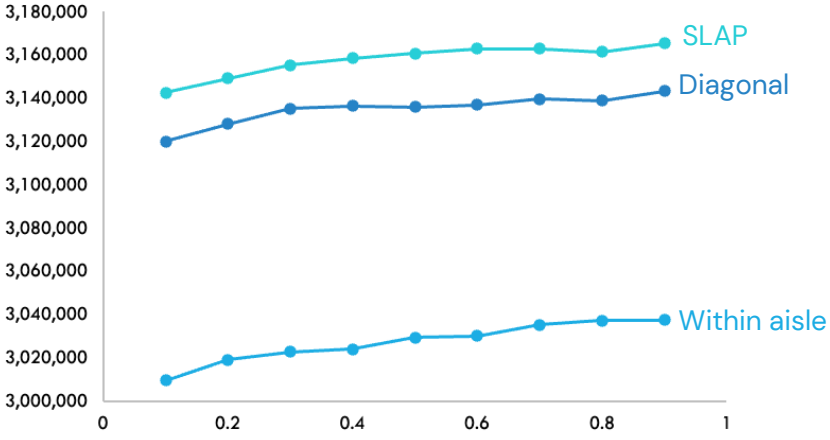
Population size

จำนวนประชากรในแต่ละรุ่นเป็นพารามิเตอร์ที่ต้องกำหนดขึ้นมาก่อน เพื่อสร้างคำตอบตามจำนวนที่ต้องการ หากมีประชากรในแต่ละรุ่นมากจะทำให้คำตอบที่ได้หลากหลายมากขึ้น

```python
import pandas as pd

df = pd.read_excel('C:/Users/Mosla/Desktop/Class Base within aisle.xlsx', sheet_name='Sheet1')
b = list(df['Route'])
results = [eval(x) for x in b]
```

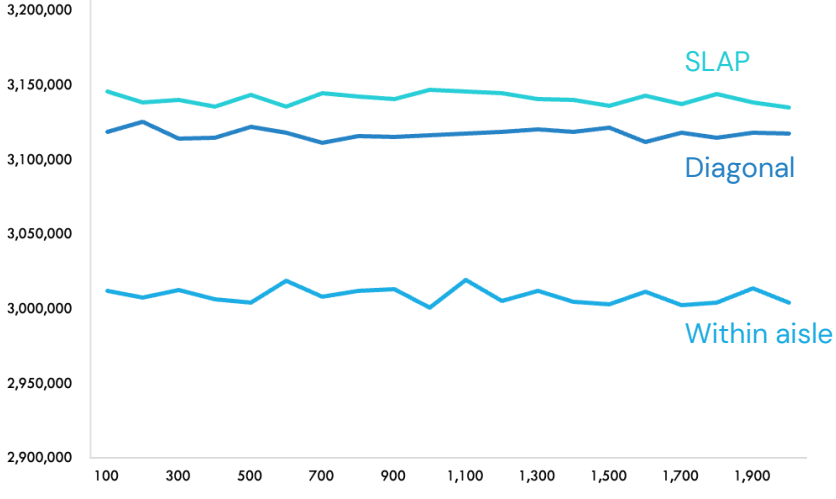
```python
import numpy as np
import random
import time
from datetime import timedelta

a = []
c = 100
for k in range(1, 31):
    b = 0
    start_time = time.monotonic()
    for j in range(len(results)):
        # Example distance matrix and demand matrix (replace these with your data)
        distance_matrix = results[j]

        demand = [0, 1, 1, 1, 1, 1, 1]  # Example demand for each customer

        # Genetic Algorithm Parameters
        POPULATION_SIZE = 100
        GENERATIONS = c
        MUTATION_RATE = 0.1

        # Function to calculate total route distance
        def route_distance(route, distance_matrix):
            dist = 0
            for i in range(len(route) - 1):
                dist += distance_matrix[route[i]][route[i + 1]]
            return dist

        # Generate initial population
        def create_initial_population(num_customers, population_size):
            population = []
            for _ in range(population_size):
                route = list(range(1, num_customers))  # Start with a random permutation of customers
                random.shuffle(route)
                population.append(route)
            return population

        # Crossover operator (ordered crossover)
        def crossover(parent1, parent2):
            start = random.randint(0, len(parent1) - 1)
            end = random.randint(start + 1, len(parent1))
            child = [-1] * len(parent1)
            for i in range(start, end):
                child[i] = parent1[i]

            pointer = 0
            for i in range(len(parent2)):
                if not parent2[i] in child:
                    while child[pointer] != -1:
                        pointer += 1
                    child[pointer] = parent2[i]

            return child

        # Mutation operator (swap mutation)
        def mutate(route):
            idx1, idx2 = random.sample(range(len(route)), 2)
            route[idx1], route[idx2] = route[idx2], route[idx1]
            return route

        # Genetic Algorithm
        def genetic_algorithm(distance_matrix, demand, population_size, generations, mutation_rate):
            num_customers = len(demand)
            population = create_initial_population(num_customers, population_size)

            for generation in range(generations):
                # Evaluate fitness of each route
                fitness_scores = []
                for route in population:
                    total_demand = sum([demand[i] for i in route])
                    if total_demand > 0:  # Ensure the total demand does not exceed vehicle capacity
                        fitness_scores.append(1 / (route_distance([0] + route + [0], distance_matrix) + 1))  # Inverse of route distance as fitness

                # Select parents for crossover using roulette wheel selection
                parents = random.choices(population, weights=fitness_scores, k=population_size)

                # Perform crossover and mutation to create the next generation
                next_generation = []
                for i in range(0, len(parents), 2):
                    child1 = crossover(parents[i], parents[i + 1])
                    child2 = crossover(parents[i + 1], parents[i])
                    if random.random() < mutation_rate:
                        child1 = mutate(child1)
                    if random.random() < mutation_rate:
                        child2 = mutate(child2)
                    next_generation.extend([child1, child2])

                population = next_generation

            # Get the best route
            best_route = max(population, key=lambda x: 1 / (route_distance([0] + x + [0], distance_matrix) + 1))
            best_distance = route_distance([0] + best_route + [0], distance_matrix)
            return best_route, best_distance

        # Solve TSP using Genetic Algorithm
        best_route, best_distance = genetic_algorithm(distance_matrix, demand, POPULATION_SIZE, GENERATIONS, MUTATION_RATE)

        #print("Best Route:", [0] + best_route + [0])
        #print("Best Distance:", best_distance)
        b = b + best_distance
    end_time = time.monotonic()
    print(c ,b , timedelta(seconds=end_time - start_time))
    c = c + 100
    a.append(b)

#df_a_toexcel = pd.DataFrame(a)
#df_a_toexcel.to_excel('C:/Users/Mosla/Desktop/POPTune_SLAP.xlsx')
```

Genetic Algorithm (GA)



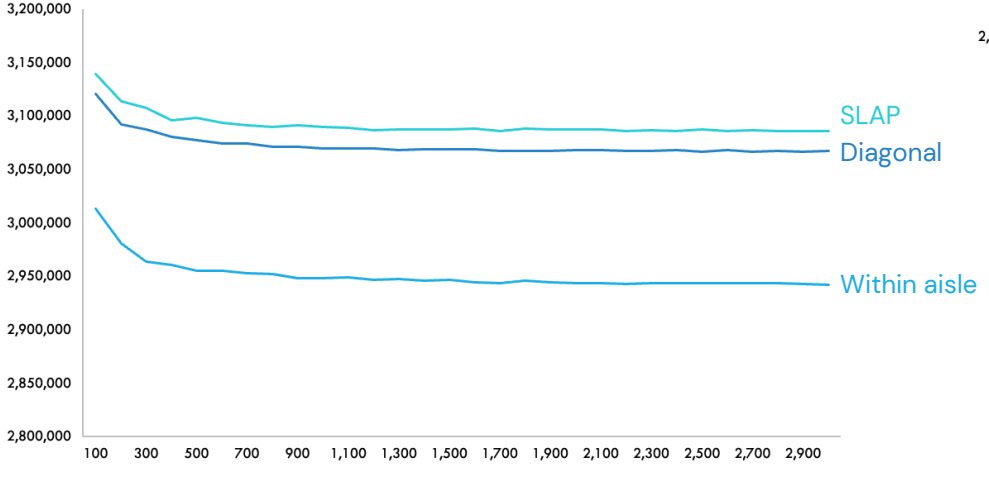
ปรับ Mutation Probability

Population size = 100 , Generations = 100



ปรับ Generation

Mutation Probability = 0.1 , Population size = 100



ปรับ Population Size

Mutation Probability = 0.1 , Generations = 100

สรุปผลการศึกษา III

1: การปรับตำแหน่งจัดเก็บ

Warehouse layout	Total Distance (cm.)	Percent change
Existing Model	9,510,160	
Class-bassed Within Aisle Model	6,823,755	-28.25%
Class-based Diagonal Model	6,598,555	-30.62%
SLAP	6,597,205	**-30.63%**

เพื่อสร้างความน่าเชื่อถือให้กับตัวแบบที่ทำการพัฒนา โดยนำ Order batching ใหม่ จำนวน 500 ใบ มาทดสอบ

Warehouse layout	Total Distance (cm.)	Percent change
Existing Model	4,956,110	
Class-bassed Within Aisle Model	4,222,220	**-14.19%**
Class-based Diagonal Model	4,265,815	-12.32%
SLAP	4,288,825	-12.27%

สรุปผลการศึกษา III

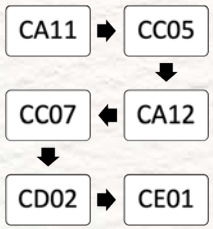
2: การปรับวิธีการหยิบสินค้า

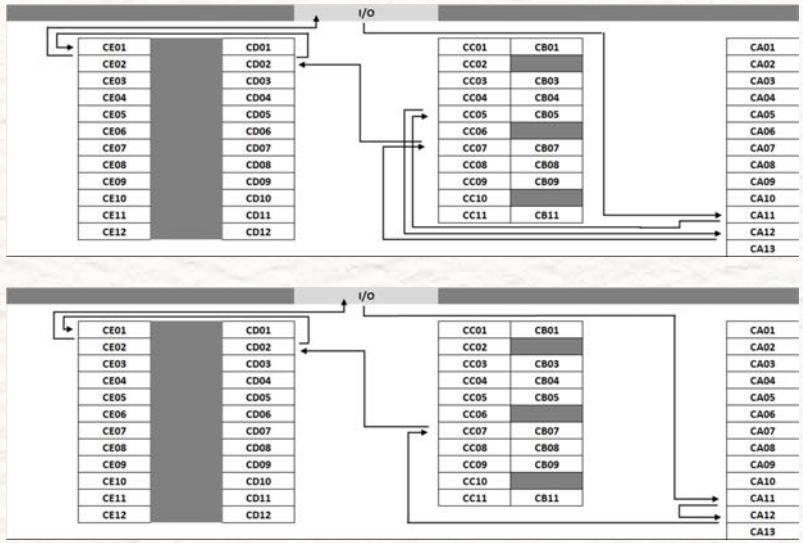
Warehouse layout	Simulation	GA	Percent change
Class-bassed Within Aisle Model	4,222,220	2,948,595	-30.16%
Class-based Diagonal Model	4,265,815	3,069,330	-28.05%
SLAP	4,288,825	3,089,435	-27.97%

Example : Class-Based Within aisle

Population size = 1,000
Generation = 100
Mutation rate = 0.1

Order Batching





Before
9,550 cm

After
7,190 cm.

ข้อเสนอแนะ

1.การกำหนดขนาด Sample size อาจจะต้องมาจากการกำหนดค่า Confidence Level ที่ต้องการ

2.การทำ Bootstrap เพื่อเพิ่มข้อมูลลด Bias ที่อาจจะเกิดขึ้นจากการ generation

3.การทำ batch picking จะต้องมีการกำหนดช่วงเวลา cut-off เพื่อรวบรวม picking order

4.การ Tuning parameter อาจจะลองใช้ DOE ในการปรับตั้งค่าพารามิเตอร์ต่างๆของกระบวนการเพื่อให้กระบวนการสามารถทำงานได้อย่างมีประสิทธิผล

5.เนื่องจากข้อมูลไม่ได้มีจำนวนและความซับซ้อนมากนัก อาจจะทำ วิธี Optimal เพื่อเปรียบเทียบกับวิธี Heuristics

Thank you