Simulating Fulfillment Center

INMAS, February 2024

Ben Isaac Sultan Vlassis Yantao

Overview

Introduction

Problem Setup

Data/Visuals

Conclusions

Part I

Introduction

Introduction

We develop a simulator to get a distribution of travel distances to fulfil a day’s worth of orders. Here are the assumptions:

1. You have N products
2. On a given day, you have M orders.
3. The i-th order is list of n_i products, i = 1, …, M.
4. Some products are more popular than others.
5. Assume that the shape of the warehouse is square, and all N products are lined up along the four walls (same number per wall).
6. The picker, a robot, receives an order at a (Southeast) corner and picks all the products in a given order, and returns to the same corner.

Introduction

Task:

- Measure the distance travelled by the robot to fulfill all the orders for the day.
- Perform many simulations to get a distribution of distances.
- Play with the parameters N and M to see if the distribution is sensitive to them.

Does it matter how the N products are arranged in the warehouse?

Part II

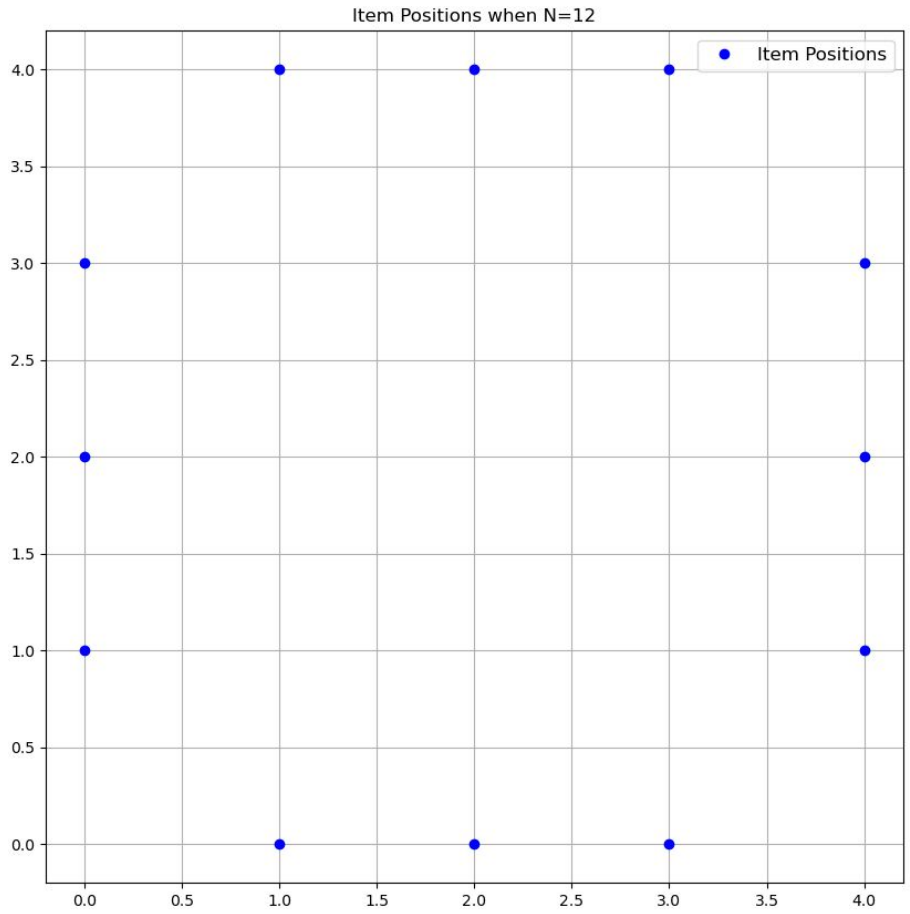
Problem Set-Up

Problem Set-Up

We assume that the N items are distributed in the following way:

Note that when N gets bigger, so does the warehouse, and so does the square.

We consider two cases: when item popularity does not affect the item positions on the square, and when item popularity does affect item positions.



Problem Set-Up

Before collecting any data we may try to optimize over two parameters:

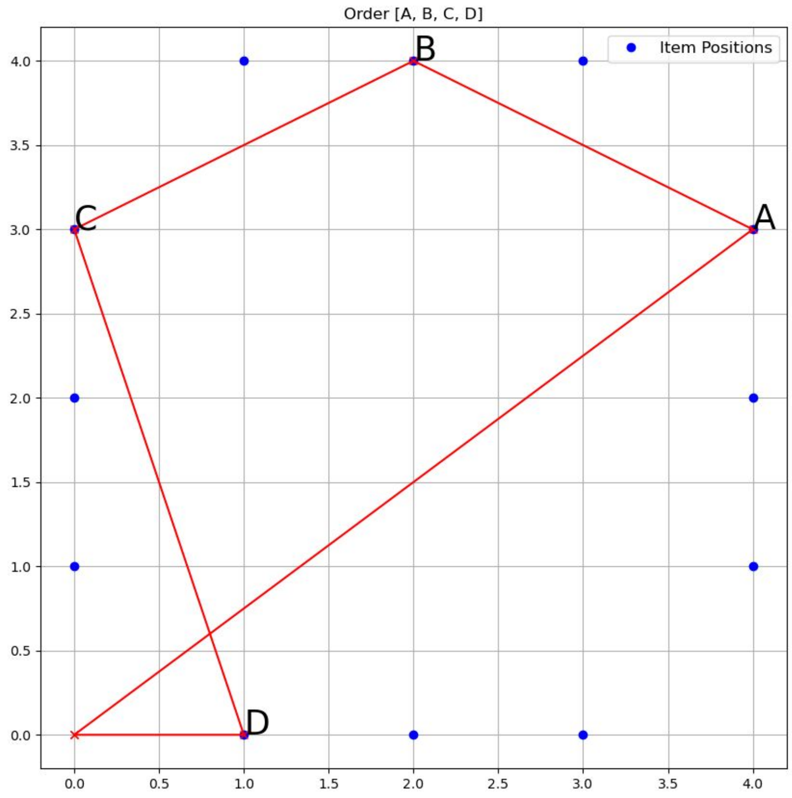
1. The movement of the robot

2. The placement of the items in the warehouse

Movement of the robot

We decided to choose a greedy algorithm for the movement of the robot: it collects the item closest to its position.

The Travelling Salesman problem is NP-hard. Greedy Algorithm is near optimal: on average it produces a solution within 25% deviation. (Per Wikipedia)



Placement of the items

For the placement of the items in the warehouse we had two approaches:

1. Place the items arbitrarily in the warehouse

2. Place the most popular items closest to the origin

Surprisingly, the second approach did not really make the movement of the robot more efficient.

Generating Data

To generate the data we did the following:

Given the number of items N and the number of orders in the day M:

1. Fix random numbers $\omega_j \sim N(0,1)$ for j from 1 to N, corresponding to the popularity of j-th product in the given day
2. Fix $n_{i,j}$ the number of j-th product in i-th order

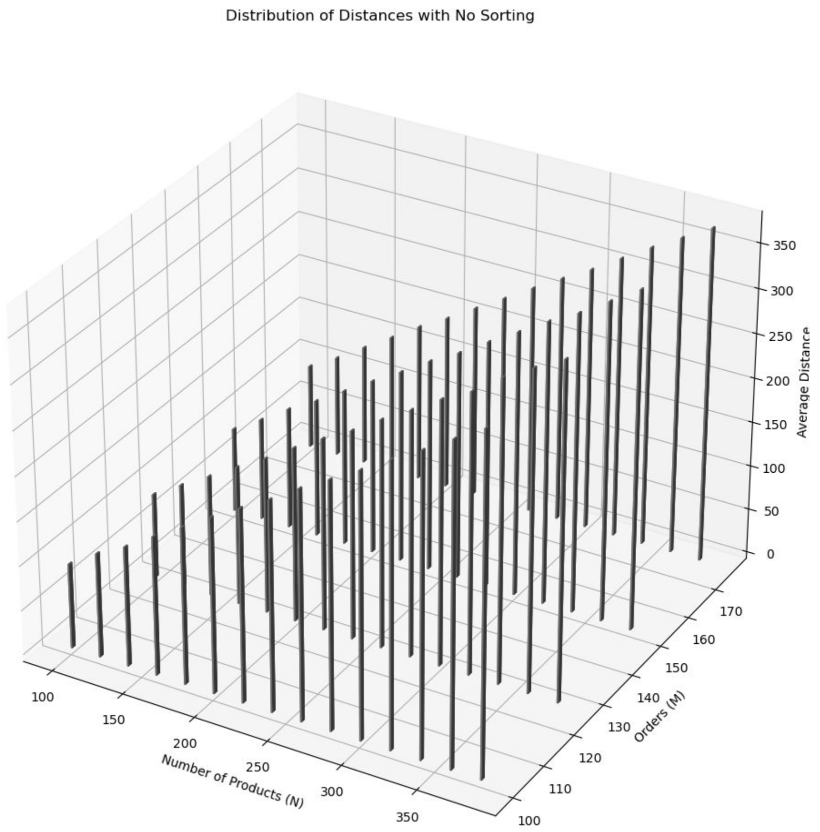
$$n_{i,j} \sim Geom\left(\frac{1}{1+|w_j|}\right) - 1$$

3. For each order, compute the distance travelled by the robot to gather the corresponding products (given the previously described algorithm)

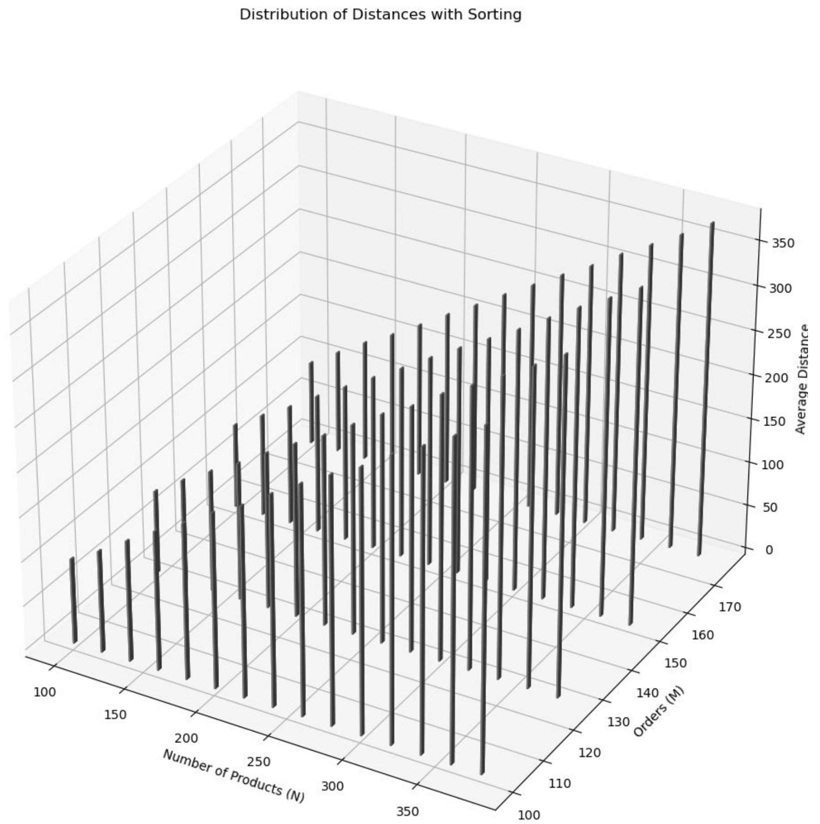
Part III

Generating Data and Finding Distance Distribution

Distance Distribution when item positions **do not** matter

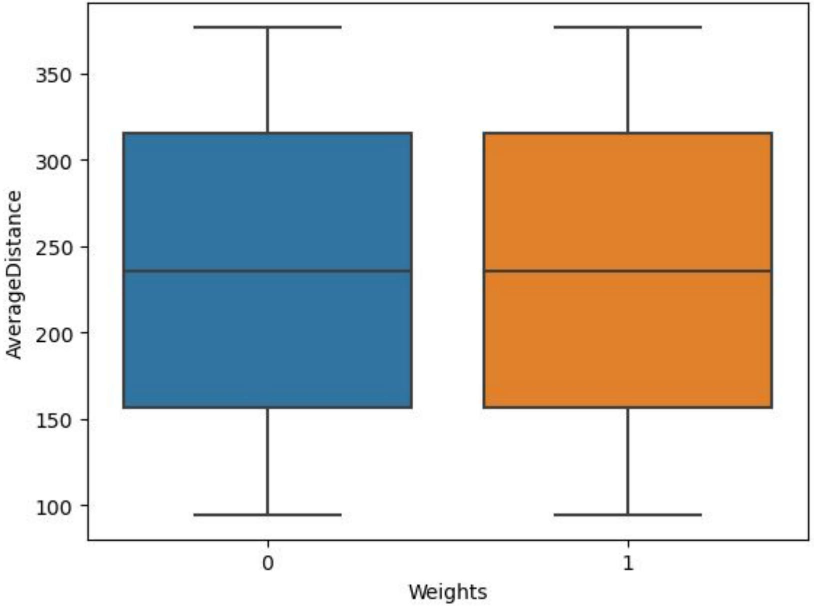


Distance Distribution when item positions **do** matter



Analysis

Eye-test seems to say that sorting by popularity has virtually zero effect.




```
                           Logit Regression Results
==============================================================================
Dep. Variable:                Weights   No. Observations:                  300
Model:                          Logit   Df Residuals:                      298
Method:                           MLE   Df Model:                            1
Date:                Sat, 03 Feb 2024   Pseudo R-squ.:               1.041e-07
Time:                        19:08:06   Log-Likelihood:                -207.94
converged:                       True   LL-Null:                       -207.94
Covariance Type:            nonrobust   LLR p-value:                    0.9948
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept          -0.0021      0.336     -0.006      0.995      -0.660       0.656
AverageDistance  8.795e-06      0.001      0.007      0.995      -0.003       0.003
===================================================================================
```

Part IV

Conclusions

Conclusions

Several avenues to explore:

1. Change the pick-up algorithm to dynamic programming or machine learning.
2. Consider different distribution for weights (Beta/Gamma/Dirichlet/etc).

any questions

feel free to ask

Bibliography

- Wikipedia contributors. (2024, February 1). Travelling salesman problem. In *Wikipedia, The Free Encyclopedia*. Retrieved 23:43, February 3, 2024, from https://en.wikipedia.org/w/index.php?title=Travelling_salesman_problem&oldid=1201894762