1. Describe the machine you are running your tests on:

CPU: AMD 1700X @3.9GHz

OS: Windows10 64bit

Memory: 32GB DDR4 @3066MHz

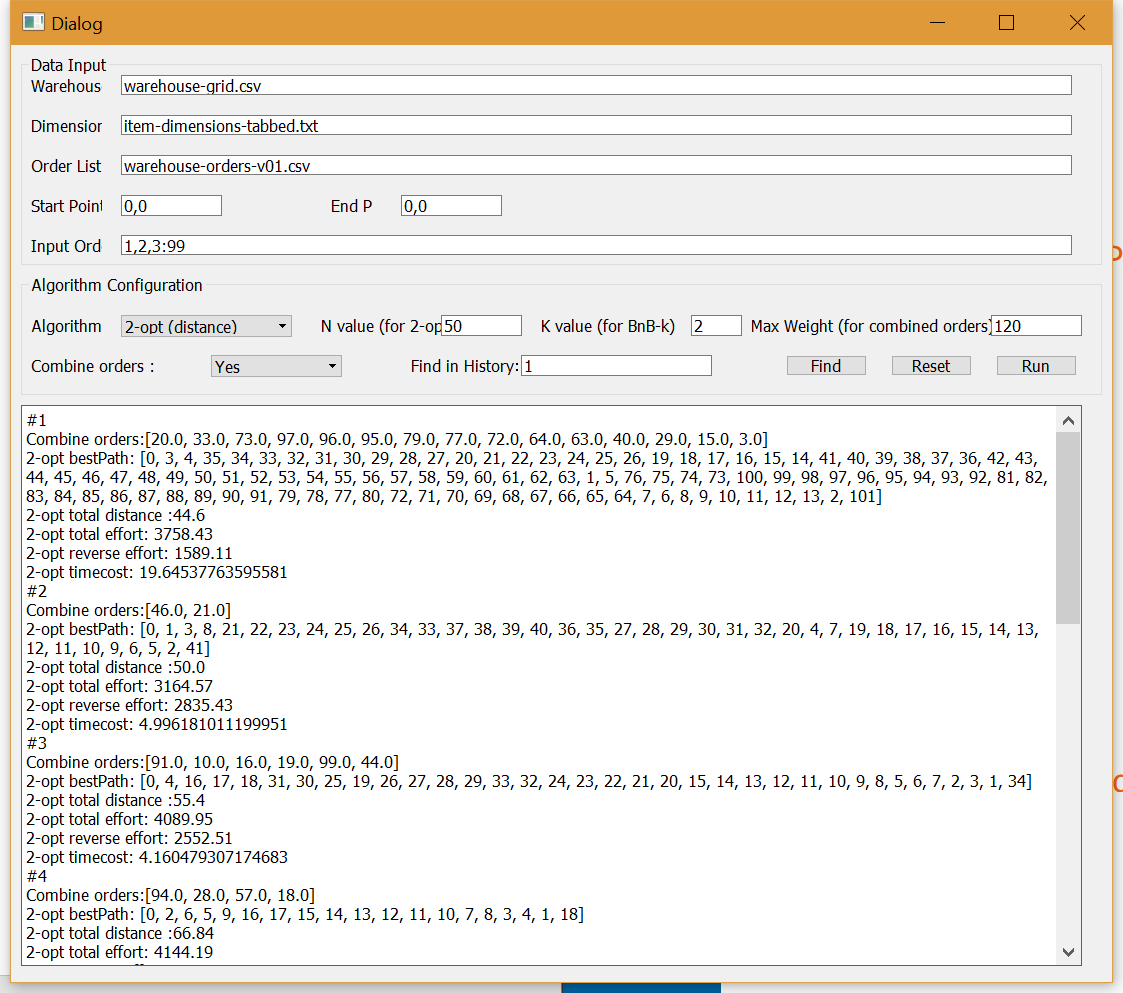
#### 2. Minimum Requirements (80%):

Be able to batch process all orders

If a single order is too large for your program, terminate after 30 seconds (or 1 minute w/ justification) & provide the best result you can.

Make sure batch processed order is saved in a format you can read back into the program (for next section)

In the “Input Order” text box, user can choose input multiple (continuous or discontinuous) orders (format: 3,5,12:20,33).

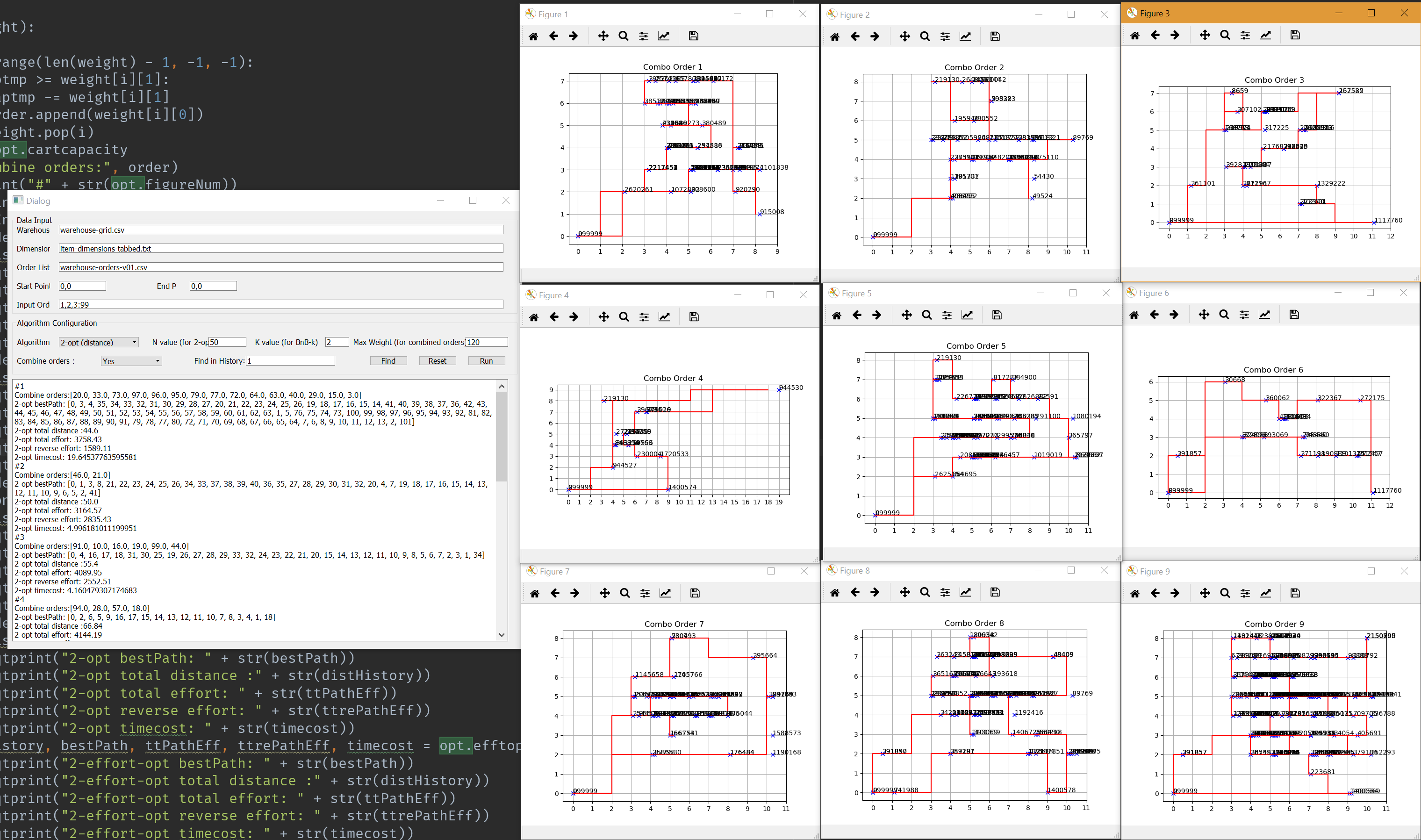


#### 3. Preferred Requirements (90%)

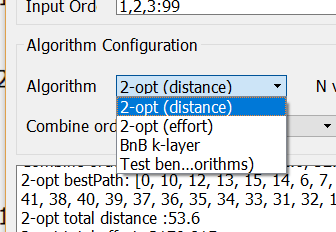
Have a GUI interface

* that can show the path for a current order (user putting in order list)
* that can show any order from the batch file you generated (aka, batch file that has already been processed, the user should have a menu option where they can display one of the orders, example: order #1 or order #2500). If an order was combined/split, simply notify the user & show that combined/split information.
  + If you already have a GUI interface, you must update it to support the above tasks.
  + If you do not have a GUI interface, you now must have one.
  + Text directions for the paths, as well as "effort" calculations need to be displayed to the user as well (think Google Maps).

The GUI is shown in following figure. It shows the running results of 99 orders in “warehouse-orders-v01.csv”. The program merges all 99 orders into 9 trips with the max weight of 120. The total running time is less than 150 seconds.



Users could also choose one specific algorithm among 2-optimization (optimize total distance), 2-optimization (optimize total effort), branch and bound with k-depth in the GUI. Note that 2-optimization (optimize total effort) is an algorithm designed for optimize total effort. It can significantly reduce the effort by 20% to 60%, compared with 2-opt (distance) and BnB-k.



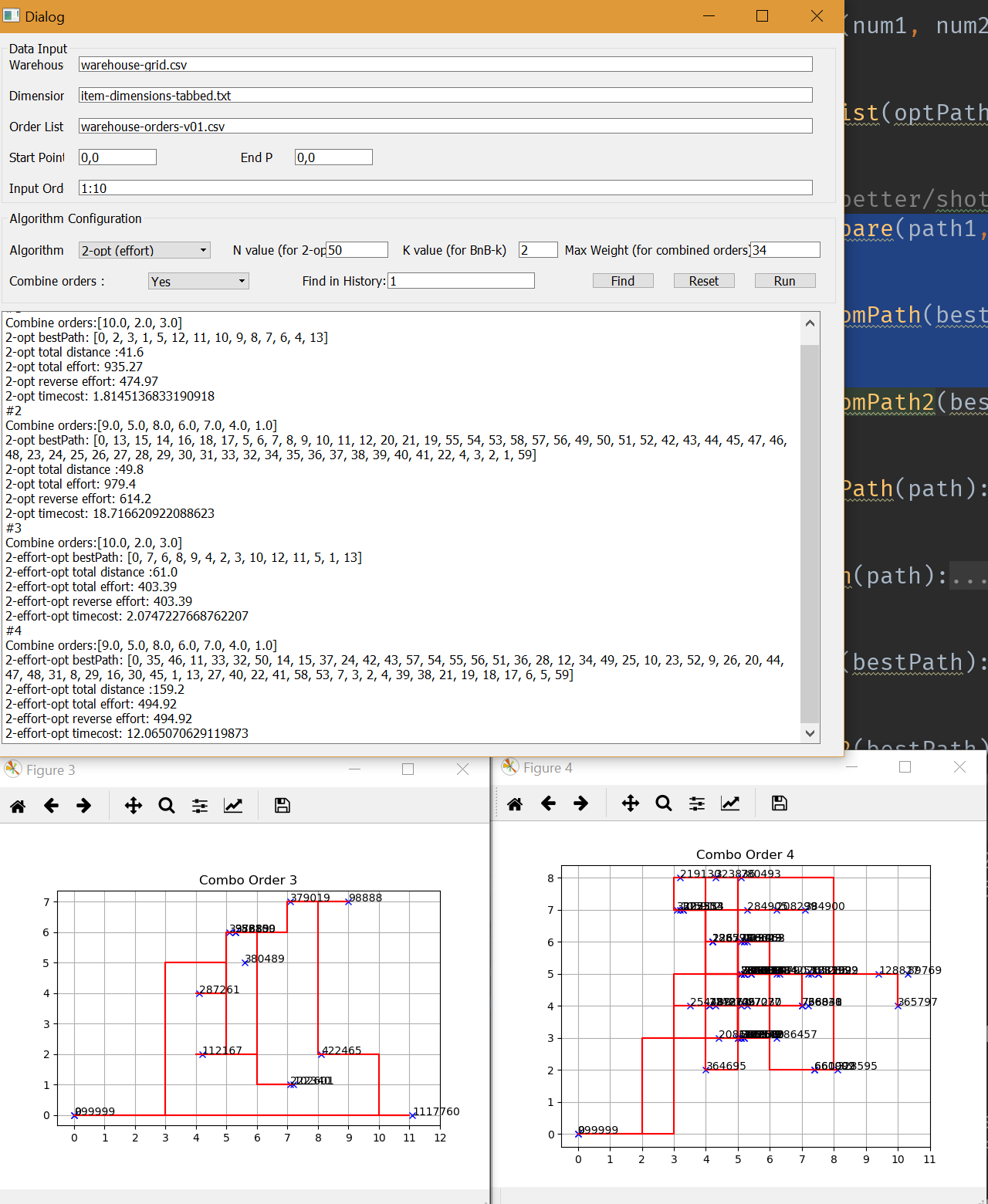
#### 4. Ideal Requirements (100%)

Allow user/program input of the max weight a worker can push (bonus from Part 5 now mandatory)

* For a single order, if this weight is reached, the order should be split into multiple trips; this should be reflected in program output
* If this weight limit is not reached with a single order, add on additional orders
  + It is OKAY to limit the # of items combined (aka, max out at XX items so your algorithm can process it in a reasonable time)
  + If is OKAY to not reach the weight limit as long as there is justification (limiting # of items as above, or preventing splitting too many orders)
  + This will need to be reflected in the program output

|  |  |  |  |
| --- | --- | --- | --- |
| Orders: 1~10 | | | |
| Max weight: 34 | | | |
| Merged into 2 trips: [10.0, 2.0, 3.0] [9.0, 5.0, 8.0, 6.0, 7.0, 4.0, 1.0] | | | |
|  | Total distance | Total effort | Running Time |
| Opt-2(distance) | 91.4 | 1089.2 | 20.5 |
| Opt-2(effort) | 220.2 | 898.3 | 14.1 |
| BnB-2 | 111.6 | 1563.7 | 2.8 |
| BnB-3 | 100 | 1109.1 | 321 |

#### 



#### 5. Bonus +30%

Based on the new order set, you are now allowed to rearrange any items to optimize the existing orders as well as predicting where to put items for better future orders.

* For any item moved, it must be swapped with another item from the destination shelf. You can swap any item that is within 30 cubic inches with each other (refer to dimensions list). If the item doesn't have dimensions, you cannot swap it. If the item is greater than 30 cubic inches, multiple items must be swapped.
* To show that your improvements worked, you must show the new distances for the existing 2500 orders comparing the old & new warehouse setup.
* (+0% to 20%) The final test of this will be another new order list (~2500 more orders) where you can test the original warehouse arrangement & your current warehouse arrangement. This list will be made available once someone is ready and can demonstrate the above tasks (notify me when ready via email, discussion/office hours, or in lecture).

Solution:

Greedily swap the high frequency items and high weight items close to the ending point. (Here we ignore the relationship between orders. If two items have coherent relationship, they should have similar show-up frequency, thus the greedy algorithm should be able to put them close enough.)

#### Following testing case shows 100 orders merged running results.

#### For all 2500 orders, still running.......

#### #1

Combine orders:[20.0, 33.0, 73.0, 97.0, 96.0, 95.0, 79.0, 77.0, 72.0, 64.0, 63.0, 40.0, 29.0, 15.0, 3.0]

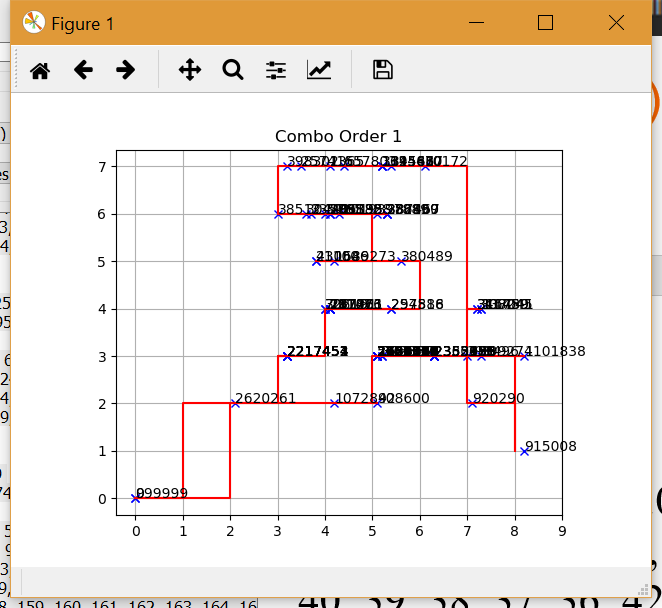
2-opt bestPath: [0, 3, 4, 35, 34, 33, 32, 31, 30, 29, 28, 27, 20, 21, 22, 23, 24, 25, 26, 19, 18, 17, 16, 15, 14, 41, 40, 39, 38, 37, 36, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 1, 5, 76, 75, 74, 73, 100, 99, 98, 97, 96, 95, 94, 93, 92, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 79, 78, 77, 80, 72, 71, 70, 69, 68, 67, 66, 65, 64, 7, 6, 8, 9, 10, 11, 12, 13, 2, 101]

2-opt total distance :44.6

2-opt total effort: 3758.43

2-opt reverse effort: 1589.11

2-opt timecost: 19.64537763595581



#2

Combine orders:[46.0, 21.0]

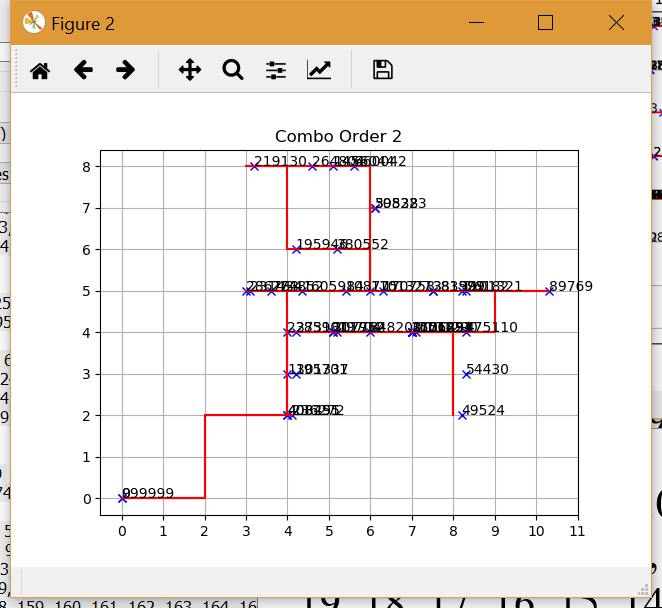
2-opt bestPath: [0, 1, 3, 8, 21, 22, 23, 24, 25, 26, 34, 33, 37, 38, 39, 40, 36, 35, 27, 28, 29, 30, 31, 32, 20, 4, 7, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 6, 5, 2, 41]

2-opt total distance :50.0

2-opt total effort: 3164.57

2-opt reverse effort: 2835.43

2-opt timecost: 4.996181011199951



#3

Combine orders:[91.0, 10.0, 16.0, 19.0, 99.0, 44.0]

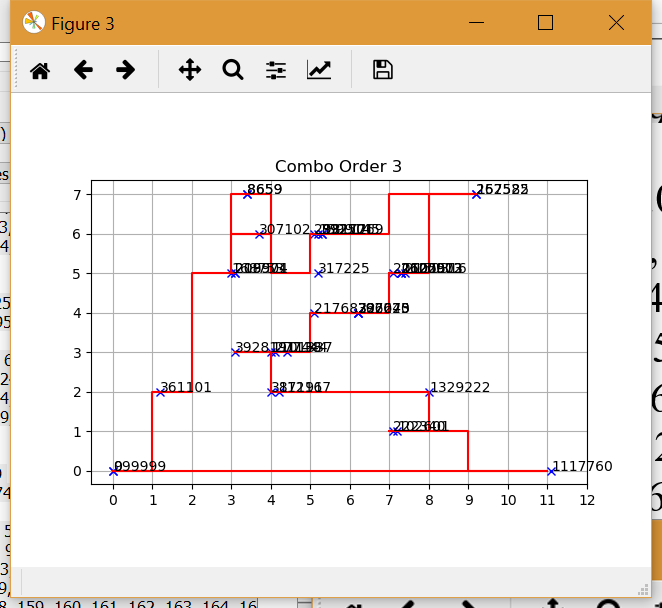
2-opt bestPath: [0, 4, 16, 17, 18, 31, 30, 25, 19, 26, 27, 28, 29, 33, 32, 24, 23, 22, 21, 20, 15, 14, 13, 12, 11, 10, 9, 8, 5, 6, 7, 2, 3, 1, 34]

2-opt total distance :55.4

2-opt total effort: 4089.95

2-opt reverse effort: 2552.51

2-opt timecost: 4.160479307174683



#4

Combine orders:[94.0, 28.0, 57.0, 18.0]

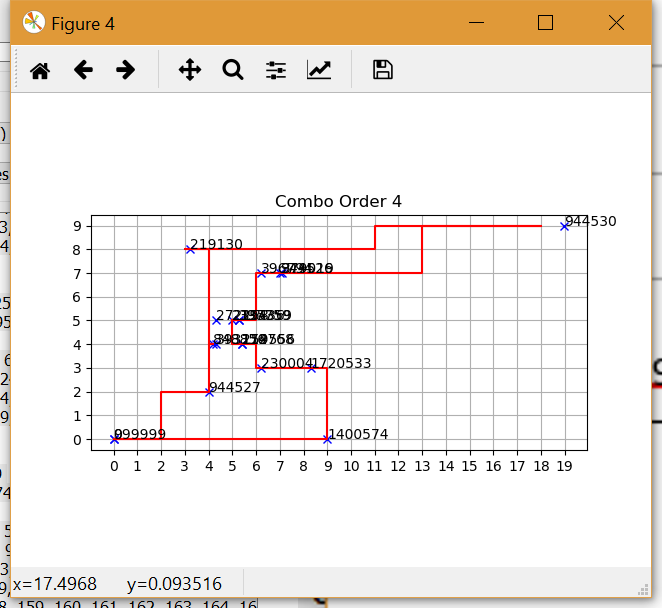
2-opt bestPath: [0, 2, 6, 5, 9, 16, 17, 15, 14, 13, 12, 11, 10, 7, 8, 3, 4, 1, 18]

2-opt total distance :66.84

2-opt total effort: 4144.19

2-opt reverse effort: 3876.61

2-opt timecost: 2.708266019821167



#5

Combine orders:[47.0, 90.0, 9.0, 62.0, 4.0, 1.0]

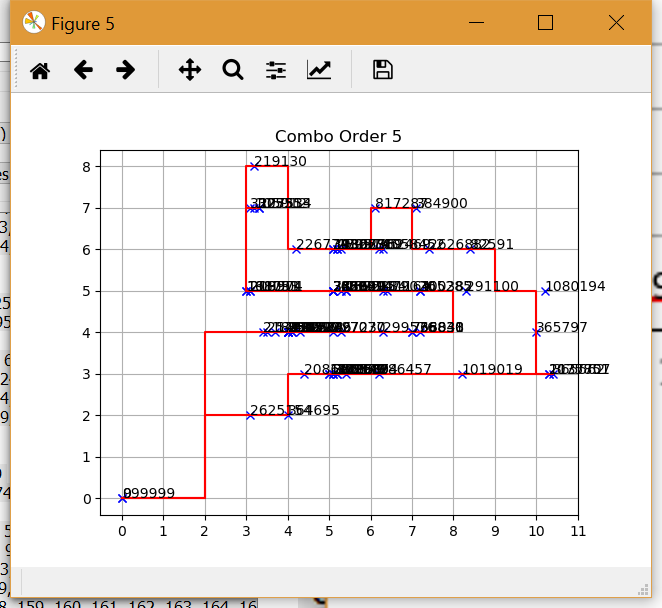
2-opt bestPath: [0, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 61, 62, 63, 64, 67, 51, 52, 53, 54, 55, 56, 57, 58, 65, 66, 59, 60, 50, 33, 13, 14, 15, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 68]

2-opt total distance :51.2

2-opt total effort: 3209.31

2-opt reverse effort: 2924.45

2-opt timecost: 13.002836465835571



#6

Combine orders:[27.0, 81.0, 23.0, 24.0]

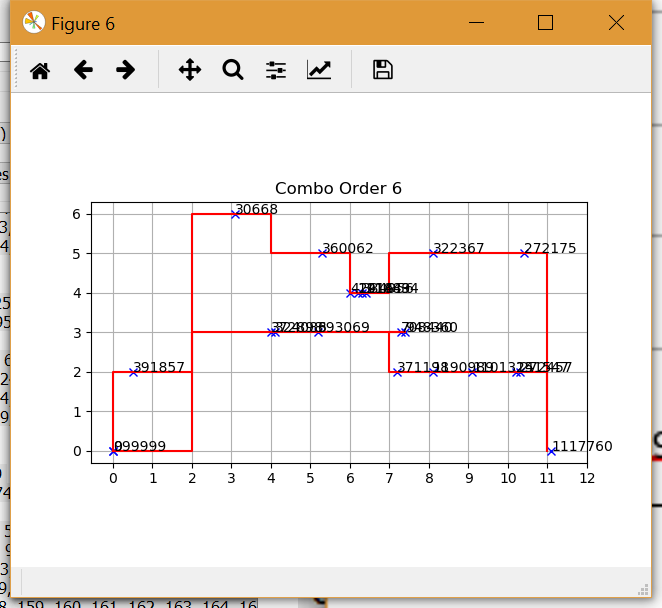
2-opt bestPath: [0, 8, 9, 10, 11, 12, 3, 4, 5, 6, 7, 1, 19, 18, 16, 15, 14, 13, 17, 20, 2, 21]

2-opt total distance :43.0

2-opt total effort: 2719.42

2-opt reverse effort: 2419.08

2-opt timecost: 3.8642287254333496



#7

Combine orders:[68.0, 66.0, 75.0, 12.0, 52.0, 84.0, 92.0]

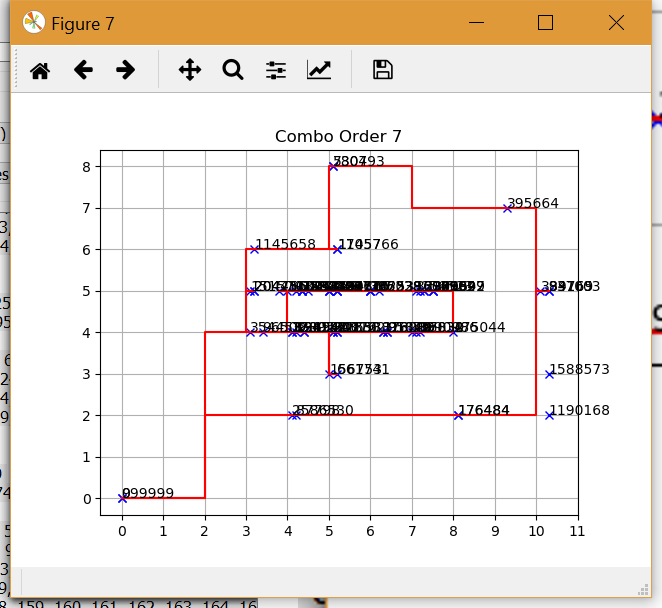
2-opt bestPath: [0, 10, 12, 13, 15, 14, 6, 7, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 38, 40, 39, 37, 36, 35, 34, 33, 31, 32, 11, 9, 30, 29, 28, 58, 59, 60, 63, 62, 61, 57, 56, 55, 8, 5, 4, 3, 2, 1, 64]

2-opt total distance :53.6

2-opt total effort: 3179.815

2-opt reverse effort: 3246.825

2-opt timecost: 11.186330795288086



#8

Combine orders:[41.0, 31.0, 65.0, 51.0, 13.0, 59.0, 58.0, 67.0, 11.0, 30.0]

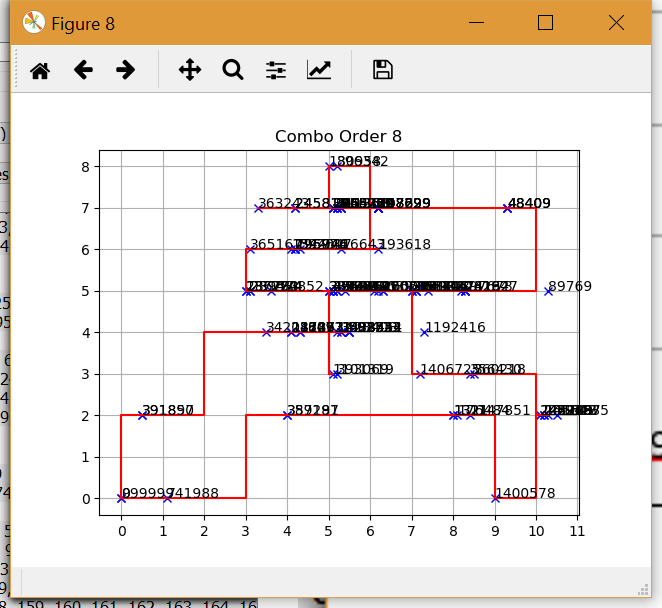
2-opt bestPath: [0, 4, 3, 22, 24, 23, 26, 25, 27, 28, 29, 30, 31, 32, 18, 17, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 62, 63, 64, 65, 66, 67, 68, 69, 92, 91, 72, 71, 70, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 61, 60, 59, 58, 57, 56, 53, 54, 55, 50, 49, 48, 47, 46, 45, 44, 52, 51, 33, 19, 20, 21, 16, 15, 14, 13, 12, 11, 2, 10, 9, 7, 8, 6, 5, 1, 93]

2-opt total distance :65.2

2-opt total effort: 4918.95

2-opt reverse effort: 2885.49

2-opt timecost: 17.7017719745636



#9

Combine orders:[50.0, 88.0, 5.0, 87.0, 43.0, 49.0, 34.0, 98.0, 83.0, 71.0, 42.0, 8.0, 70.0, 38.0, 17.0, 2.0, 69.0, 78.0, 45.0, 80.0, 76.0, 56.0, 25.0, 36.0, 60.0, 22.0, 82.0, 93.0, 55.0, 48.0, 14.0, 89.0, 74.0, 61.0, 85.0, 53.0, 32.0, 6.0, 37.0, 26.0, 54.0, 7.0, 39.0, 86.0, 35.0]

2-opt bestPath: [0, 1, 2, 3, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 55, 56, 81, 80, 144, 143, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 211, 210, 209, 200, 199, 198, 197, 196, 194, 195, 193, 192, 191, 190, 189, 188, 186, 187, 208, 207, 206, 205, 204, 203, 202, 201, 185, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 112, 113, 114, 115, 116, 117, 71, 72, 73, 74, 75, 76, 77, 78, 79, 142, 141, 140, 139, 138, 137, 136, 135, 134, 133, 132, 131, 130, 129, 128, 127, 126, 125, 122, 123, 124, 119, 120, 121, 118, 109, 110, 111, 107, 108, 106, 105, 104, 100, 99, 96, 97, 98, 103, 101, 102, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 22, 14, 15, 16, 17, 18, 19, 20, 21, 13, 12, 11, 10, 9, 8, 7, 35, 4, 5, 6, 212]

2-opt total distance :82.0

2-opt total effort: 3890.788

2-opt reverse effort: 4694.612

#### 2-opt timecost: 65.40654969215393

#### 