Solving the Joint Order Batching and Picker Routing Problem for Large Instances

Final Presentation

Khoong Wei Hao

Department of Mathematics
National University of Singapore

4th April 2019

Introduction (About Us I)

Wei Hao Khoong

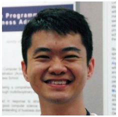
National University of Singapore

A final year undergraduate majoring in Applied Mathematics, with a specialization in operations research and financial mathematics.

Dr Melvin Zhang[1]

Co-founder + CTO Cosmiqo International Pte Ltd

An avid programmer who enjoys designing and implementing novel algorithms.



[1] Retrieved from: `https://cosmiqo.com/index.html`

Outline

1. The Project (Recap)
    - Objectives

2. Literature Review & The Working Model
    - Preliminaries
    - The Order Batching Problem
    - The Picker Routing Problem
    - The Joint Order Batching and Picker Routing Problem

3. Solving the JOBPRP for Large Instances
    - Heuristics
    - Implementation
    - Experiments & Results
    - Conclusion & Future Work

Outline

1. The Project (Recap)
   - Objectives
2. Literature Review & The Working Model
   - Preliminaries
   - The Order Batching Problem
   - The Picker Routing Problem
   - The Joint Order Batching and Picker Routing Problem
3. Solving the JOBPRP for Large Instances
   - Heuristics
   - Implementation
   - Experiments & Results
   - Conclusion & Future Work

The Project

Objectives

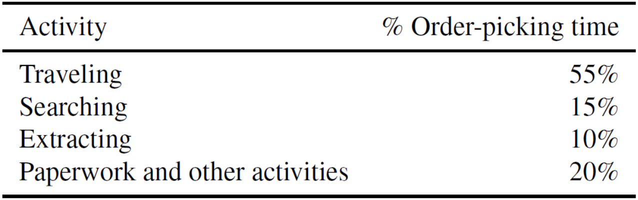
- Investigate methods to optimize order picking
  - by batching several orders together
  - by planning a good route to **minimize the distance** required to pick up all the items
- Proposed method should scale up to typical warehouse sizes

What is order picking?

- Retrieving products in an order from storage blocks in a warehouse
- Accounts for as much as 60% of all labor activities[2]

[2] Noud Gademann & Van De SteeF Velde, 2005. Order batching to minimize total travel time in a parallel-aisle warehouse.

What is order picking?



Activity	% Order-picking time
Traveling	55%
Searching	15%
Extracting	10%
Paperwork and other activities	20%

Breakdown of order-picking costs[3]

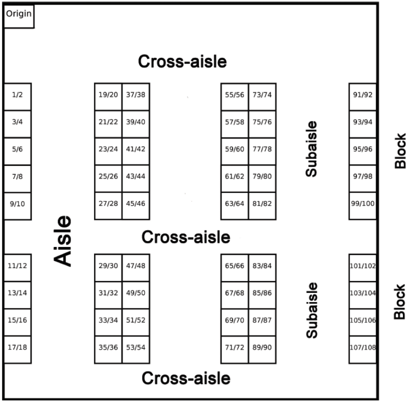
[3] John J Bartholdi and Steven T Hackman, 2017. Warehouse & distribution science release 0.98.

Outline

1. The Project (Recap)
   - Objectives
2. Literature Review & The Working Model
   - Preliminaries
   - The Order Batching Problem
   - The Picker Routing Problem
   - The Joint Order Batching and Picker Routing Problem
3. Solving the JOBPRP for Large Instances
   - Heuristics
   - Implementation
   - Experiments & Results
   - Conclusion & Future Work

Important Definitions I

The Warehouse

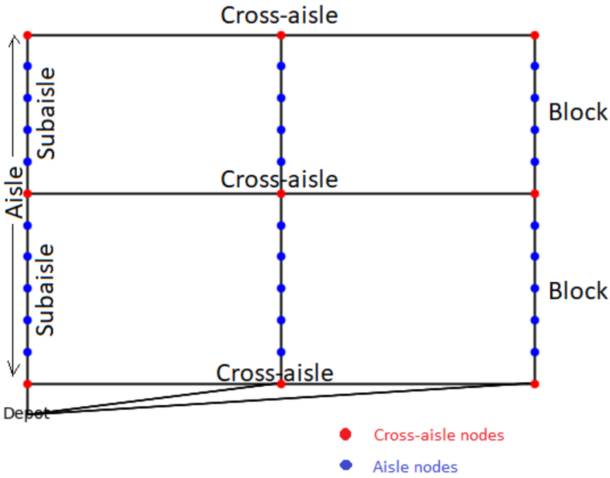


A warehouse structure[4]

[4] Cristiano Arbex Valle, John E. Beasley, and Alexandre Salles da Cunha, 2017. Optimally solving the joint order batching and picker routing problem.

Important Definitions II

The Warehouse



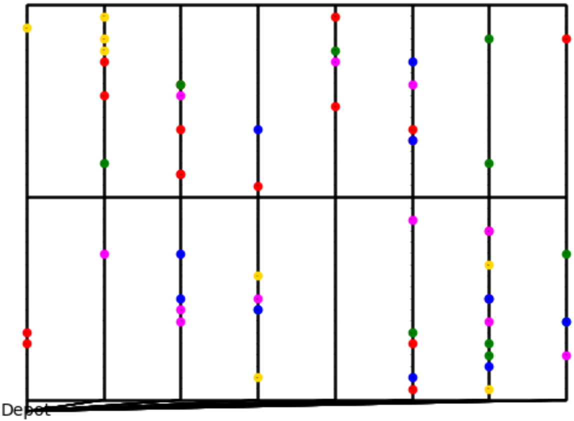
Outline

1. The Project (Recap)
   - Objectives
2. Literature Review & The Working Model
   - Preliminaries
   - The Order Batching Problem
   - The Picker Routing Problem
   - The Joint Order Batching and Picker Routing Problem
3. Solving the JOBPRP for Large Instances
   - Heuristics
   - Implementation
   - Experiments & Results
   - Conclusion & Future Work

Problem Description

The Order Batching Problem - Input

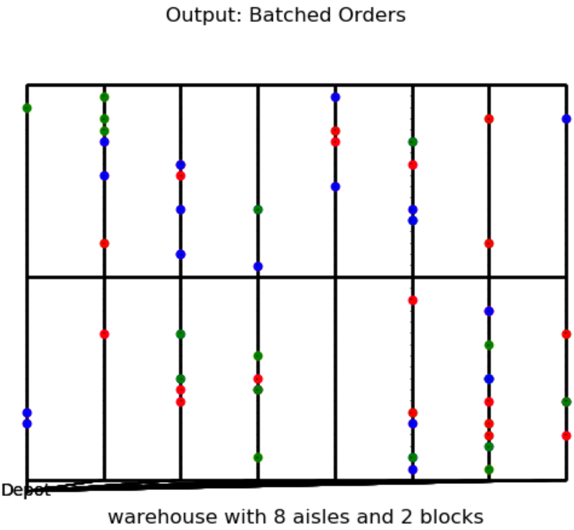
Input: Orders before batching



warehouse with 8 aisles and 2 blocks

Problem Description

The Order Batching Problem - Output



Problem Description

The Order Batching Problem

- Group orders received into a set of batches
- Time required to collect all the orders is minimized
- Problem is $\mathcal{NP}$-hard[5]

[5] Noud Gademann & Van De SteeF Velde, 2005. Order batching to minimize total travel time in a parallel-aisle warehouse.

Literature Review

The Order Batching Problem - Exact Solving Methods

- Branch-and-price algorithm to solve modest sized warehouse instances to optimality[6]
- Iterated descent approximation algorithm was suggested for larger warehouse instances
- Joint Order Batching & Picker Routing Problem (JOBPRP) was formulated and solved with a branch-and-cut algorithm[7]

[6] Noud Gademann & Van De SteeF Velde, 2005. Order batching to minimize total travel time in a parallel-aisle warehouse.

[7] Cristiano Arbex Valle, John E. Beasley, and Alexandre Salles da Cunha, 2017. Optimally solving the joint order batching and picker routing problem.

Literature Review

The Order Batching Problem - Heuristic Solving Methods

- Cluster analysis of orders[8]
- Seed-order selection rules[9]
- Time Savings[10]

[8] H. Hwang & D. G. Kim, 2005. Order-batching heuristics based on cluster analysis in a lowlevel picker-to-part warehousing system.

[9] Ying-Chin Ho, Teng-Sheng Su & Zhi-Bin Shi, 2008. Order-batching methods for an order-picking warehouse with two cross aisles.

[10] M. B. M. De Koster , E.S. Van der Poort & M. Wolters, 1999. Efficient orderbatching methods in warehouses.

Outline

1. The Project (Recap)
   - Objectives

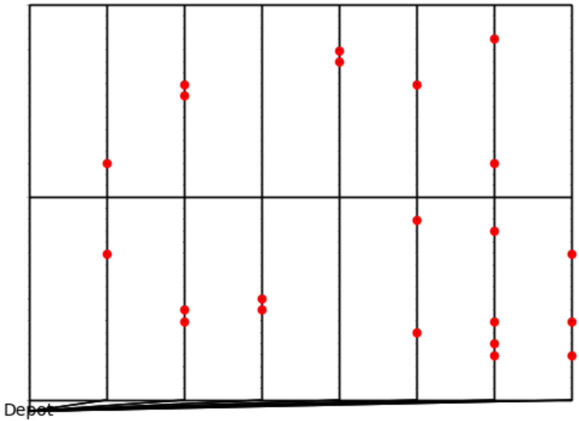
2. Literature Review & The Working Model
   - Preliminaries
   - The Order Batching Problem
   - The Picker Routing Problem
   - The Joint Order Batching and Picker Routing Problem

3. Solving the JOBPRP for Large Instances
   - Heuristics
   - Implementation
   - Experiments & Results
   - Conclusion & Future Work

Problem Description

The Picker Routing Problem - Input

Input 1: Pick Locations of Picks In Batch 1

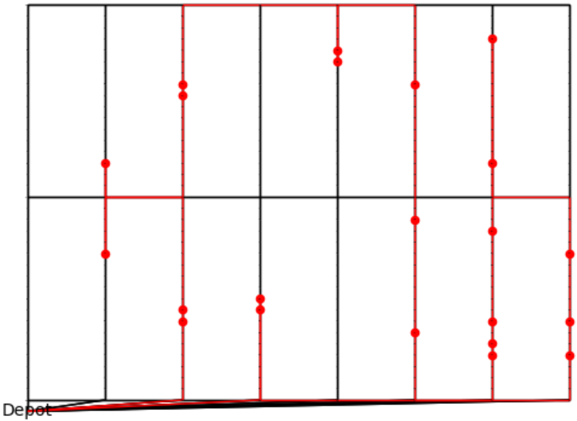


warehouse with 8 aisles and 2 blocks

Problem Description

The Picker Routing Problem - Output

Output 1: Route Traversed By Picker To Pick All Items In Batch 1



warehouse with 8 aisles and 2 blocks

Problem Description

The Picker Routing Problem

- Each picker is assigned a batched order
- Determine an optimal path to pick the items in each batch
- Problem is equivalent to the Travelling Salesman Problem (TSP)
- Problem is also $\mathcal{NP}$-hard[11]

[11] Wolsey L. A., 1998. Integer Programming, Wiley New York.

Literature Review

The Picker Routing Problem - Exact Solving Methods

- Concorde[12] TSP Solver
- Solved polynomially via a dynamic programming approach[13]

[12] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J, 2007. Cook. The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics).

[13] Cristiano Arbex Valle, John E. Beasley, and Alexandre Salles da Cunha, 2017. Optimally solving the joint order batching and picker routing problem.

Literature Review

The Picker Routing Problem - Heuristic Solving Methods

- S-shape, Largest Gap, Combined+ routing heuristics[14]

[14] Kees Jan Roodbergen and RenÉde Koster, 2001. Routing methods for warehouses with multiple cross aisles.

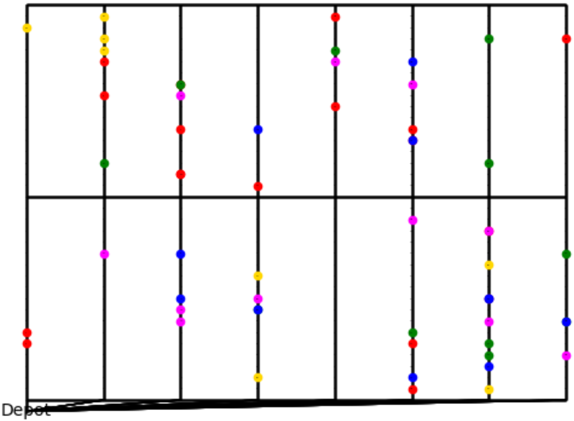
Outline

1. The Project (Recap)
   - Objectives
2. Literature Review & The Working Model
   - Preliminaries
   - The Order Batching Problem
   - The Picker Routing Problem
   - The Joint Order Batching and Picker Routing Problem
3. Solving the JOBPRP for Large Instances
   - Heuristics
   - Implementation
   - Experiments & Results
   - Conclusion & Future Work

The Problem

JOBPRP - Input

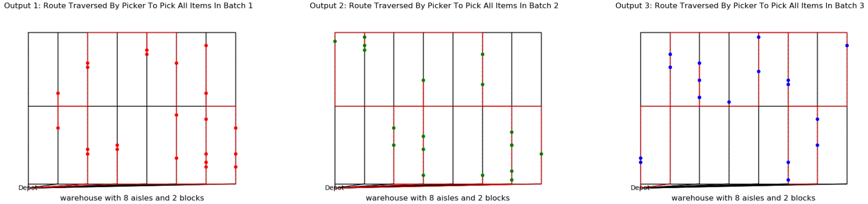
Input: Orders before batching



warehouse with 8 aisles and 2 blocks

The Problem

JOBPRP - Outputs



Output 1: Route Traversed By Picker To Pick All Items In Batch 1

Output 2: Route Traversed By Picker To Pick All Items In Batch 2

Output 3: Route Traversed By Picker To Pick All Items In Batch 3

Depot
warehouse with 8 aisles and 2 blocks
Depot
warehouse with 8 aisles and 2 blocks
Depot
warehouse with 8 aisles and 2 blocks

Outputs - 3 routed batched orders

Cost Function & Main Constraints[15]

- **Cost function**: To minimize total distance traversed by all the pickers
- **Capacity constraint**: To ensure picker capacity is not exceeded
- **Order assignment constraint**: To ensure each order is picked by only one picker

[15] Cristiano Arbex Valle, John E. Beasley, and Alexandre Salles da Cunha, 2017. Optimally solving the joint order batching and picker routing problem.

Outline

1 The Project (Recap)
- Objectives

2 Literature Review & The Working Model
- Preliminaries
- The Order Batching Problem
- The Picker Routing Problem
- The Joint Order Batching and Picker Routing Problem

3 Solving the JOBPRP for Large Instances
- Heuristics
- Implementation
- Experiments & Results
- Conclusion & Future Work

Heuristics

Overview and Trade-offs

- The JOBPRP is $\mathcal{NP}$-hard[16]
- We use heuristics to achieve "good" enough feasible solutions in a short time

[16] Cristiano Arbex Valle, John E. Beasley, and Alexandre Salles da Cunha, 2017. Optimally solving the joint order batching and picker routing problem.

Heuristics Used

Overview

Picker Routing Heuristics[17]**:** S-shape, Largest Gap

Batching Heuristic(s): Time Savings[18]

[17] Kees Jan Roodbergen & RenÉde Koster, 2001. Routing methods for warehouses with multiple cross aisles.

[18] M. B. M. De Koster , E.S. Van der Poort & M. Wolters, 1999. Efficient orderbatching methods in warehouses.

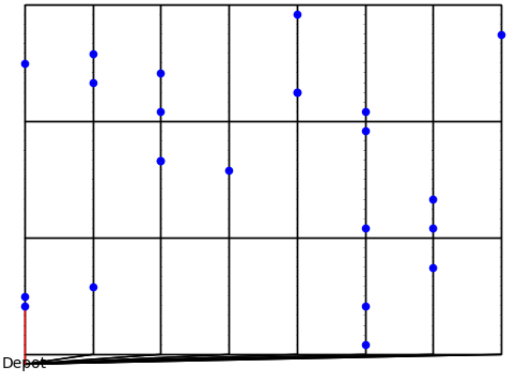
Heuristics Used I

S-shape

Here's an example to illustrate how the S-shape heuristic works:

Heuristics Used II

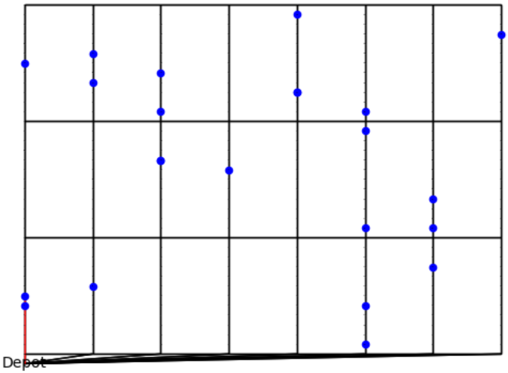
S-shape



Total Distance = 336.6m

Heuristics Used III

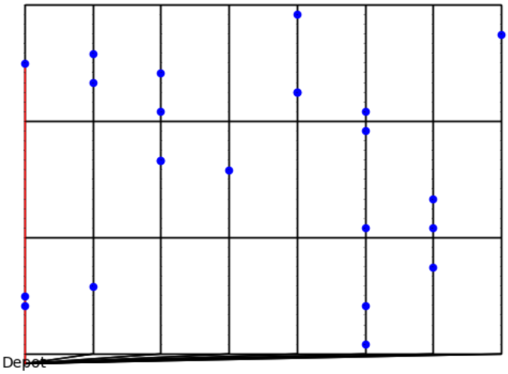
S-shape



Total Distance = 336.6m

Heuristics Used IV

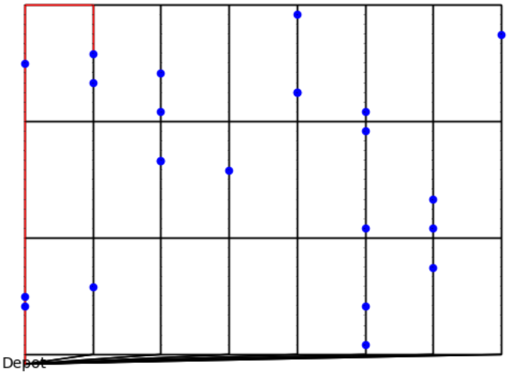
S-shape



Total Distance = 336.6m

Heuristics Used V

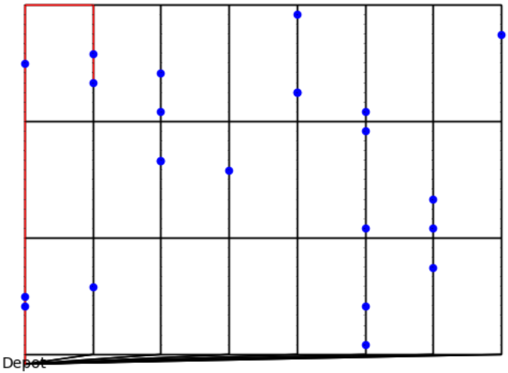
S-shape



Total Distance = 336.6m

Heuristics Used VI

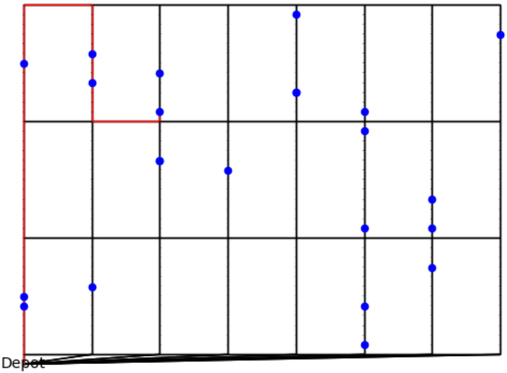
S-shape



Total Distance = 336.6m

Heuristics Used VII

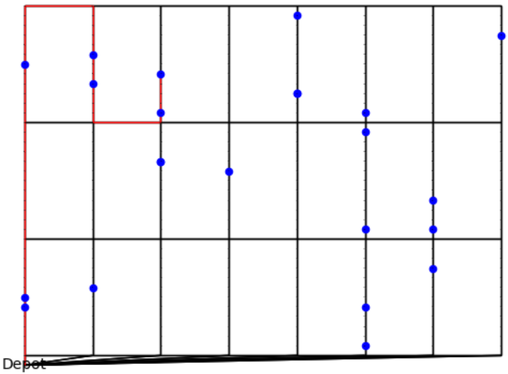
S-shape



Total Distance = 336.6m

Heuristics Used VIII

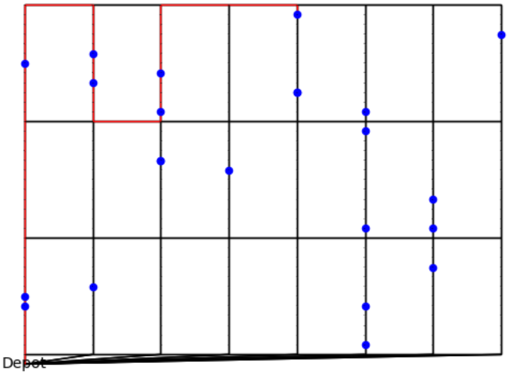
S-shape



Total Distance = 336.6m

Heuristics Used IX

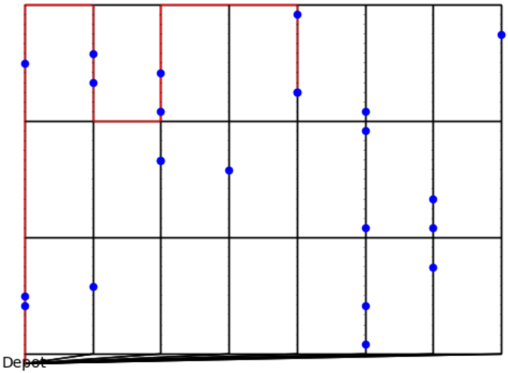
S-shape



Total Distance = 336.6m

Heuristics Used X

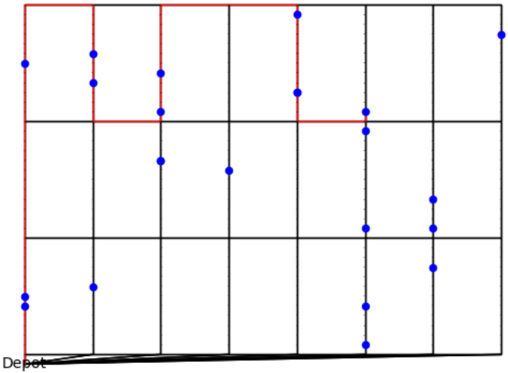
S-shape



Total Distance = 336.6m

Heuristics Used XI

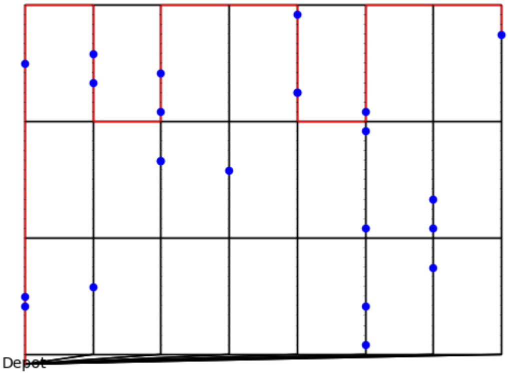
S-shape



Total Distance = 336.6m

Heuristics Used XII

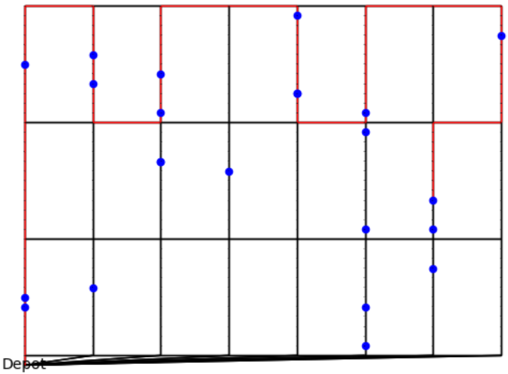
S-shape



Total Distance = 336.6m

Heuristics Used XIII

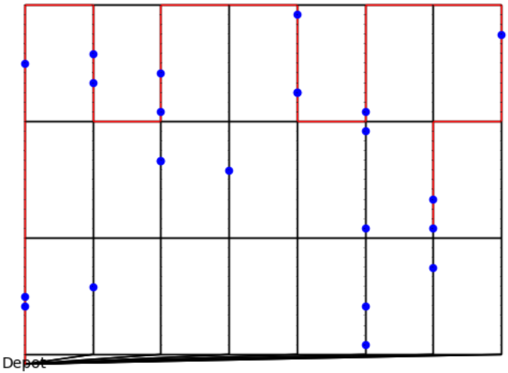
S-shape



Total Distance = 336.6m

Heuristics Used XIV

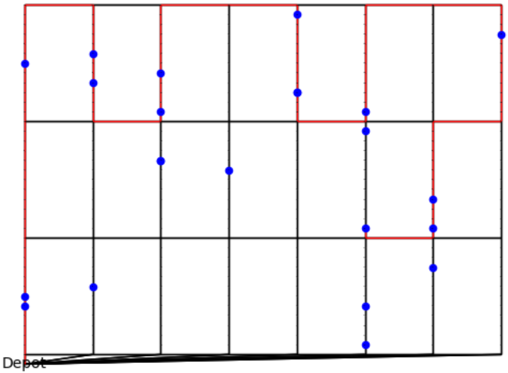
S-shape



Total Distance = 336.6m

Heuristics Used XV

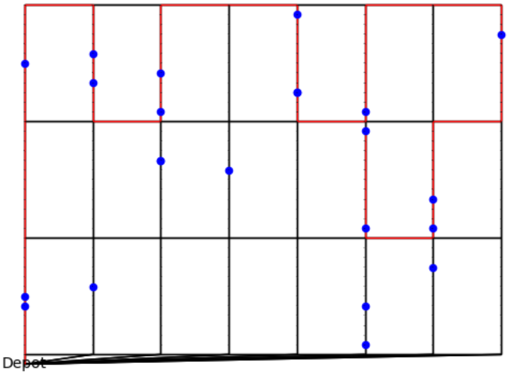
S-shape



Total Distance = 336.6m

Heuristics Used XVI

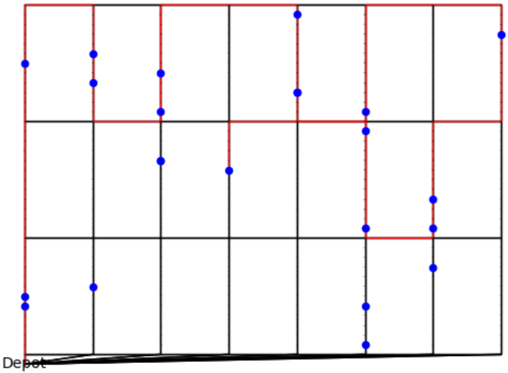
S-shape



Total Distance = 336.6m

Heuristics Used XVII

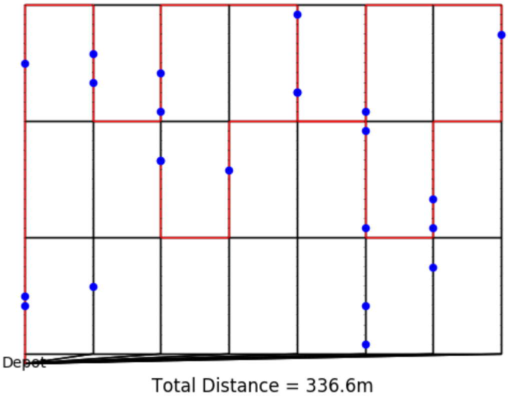
S-shape



Total Distance = 336.6m

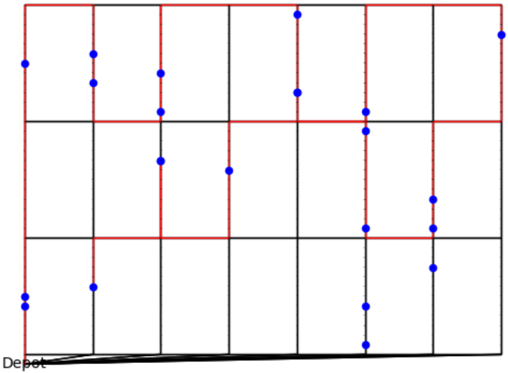
Heuristics Used XVIII

S-shape



Heuristics Used XIX

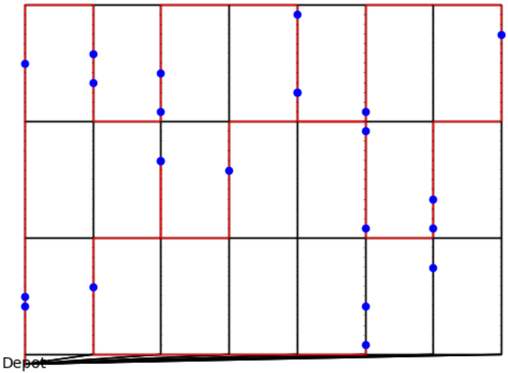
S-shape



Total Distance = 336.6m

Heuristics Used XX

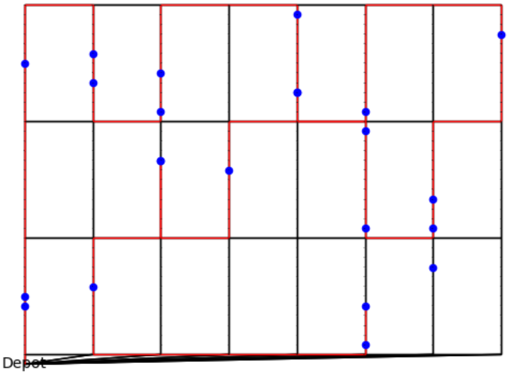
S-shape



Total Distance = 336.6m

Heuristics Used XXI

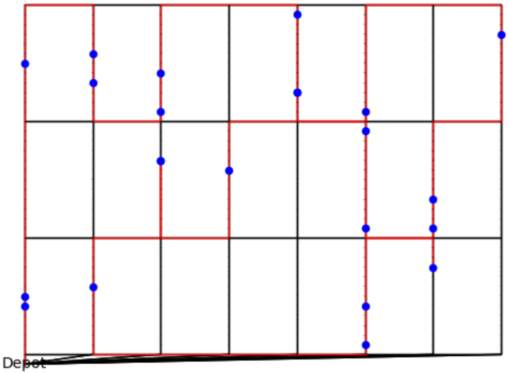
S-shape



Total Distance = 336.6m

Heuristics Used XXII

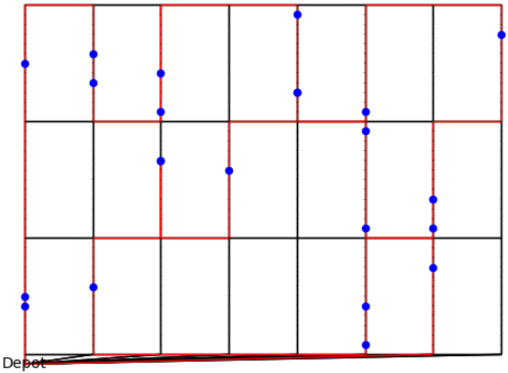
S-shape



Total Distance = 336.6m

Heuristics Used XXIII

S-shape



Total Distance = 336.6m

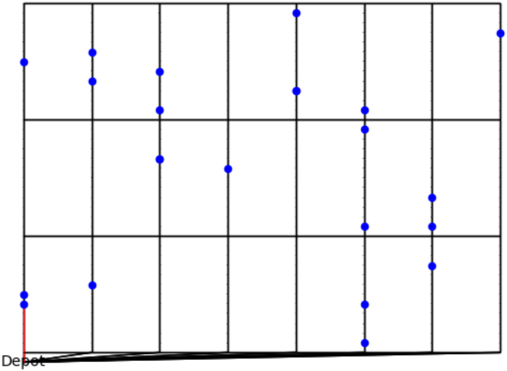
Heuristics Used I

Largest Gap

Here's an example to illustrate how the Largest Gap heuristic works:

Heuristics Used II

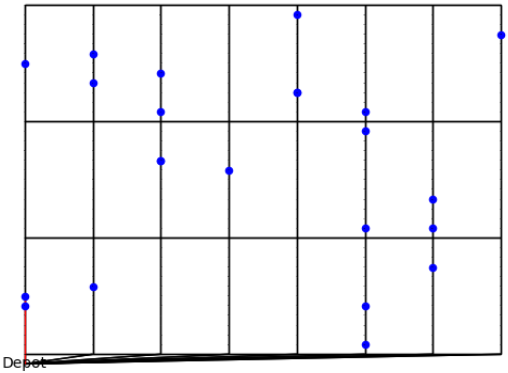
Largest Gap



Total Distance = 380.62m

Heuristics Used III

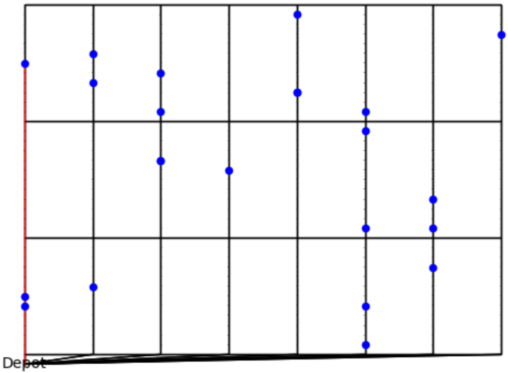
Largest Gap



Total Distance = 380.62m

Heuristics Used IV

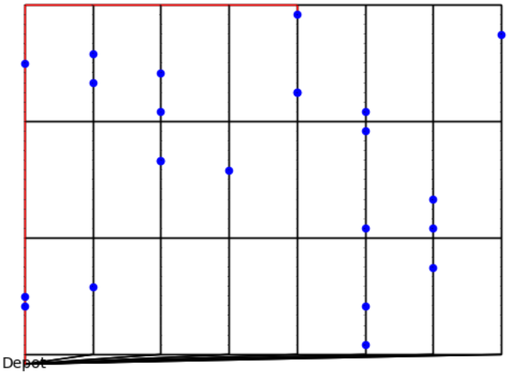
Largest Gap



Total Distance = 380.62m

Heuristics Used V

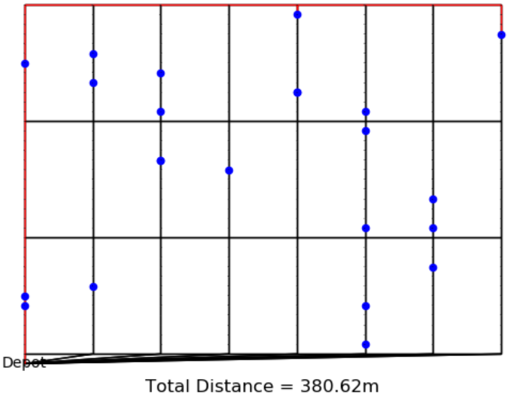
Largest Gap



Total Distance = 380.62m

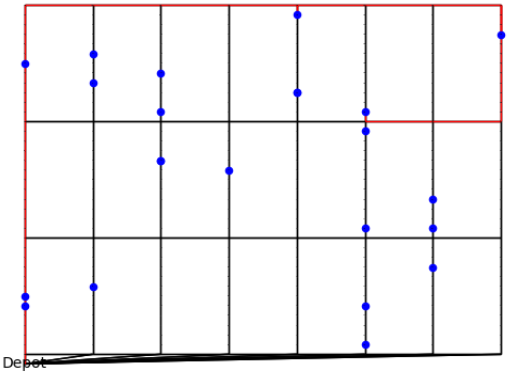
Heuristics Used VI

Largest Gap



Heuristics Used VII

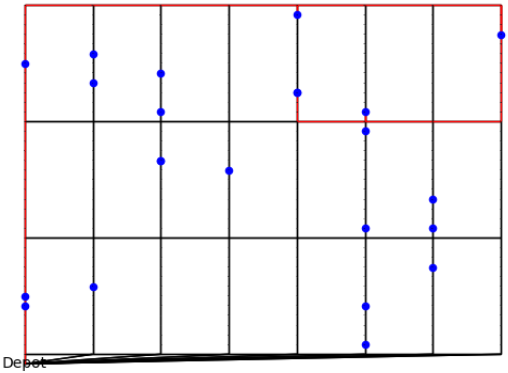
Largest Gap



Total Distance = 380.62m

Heuristics Used VIII

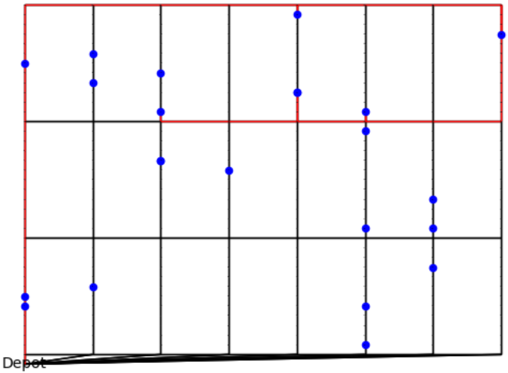
Largest Gap



Total Distance = 380.62m

Heuristics Used IX

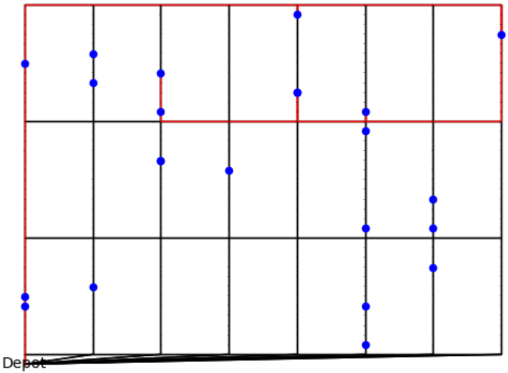
Largest Gap



Total Distance = 380.62m

Heuristics Used X

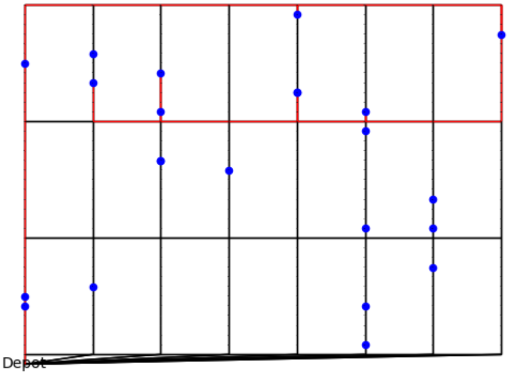
Largest Gap



Total Distance = 380.62m

Heuristics Used XI

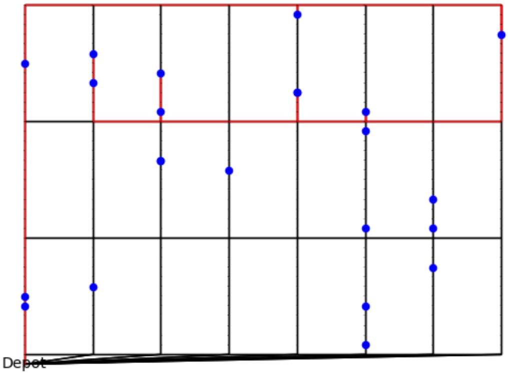
Largest Gap



Total Distance = 380.62m

Heuristics Used XII

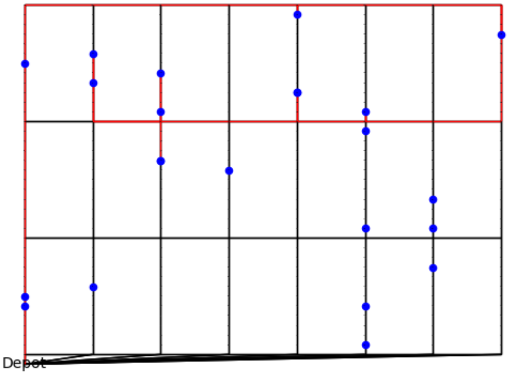
Largest Gap



Total Distance = 380.62m

Heuristics Used XIII

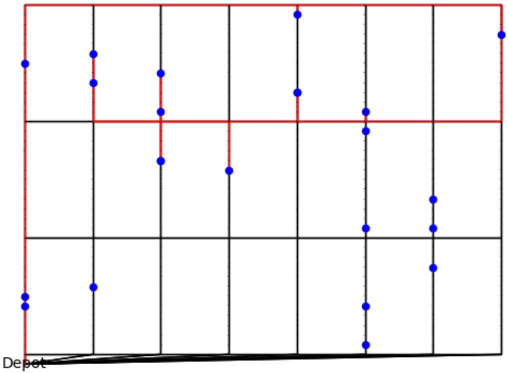
Largest Gap



Total Distance = 380.62m

Heuristics Used XIV

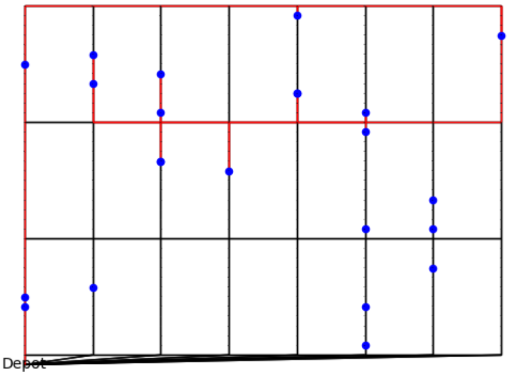
Largest Gap



Total Distance = 380.62m

Heuristics Used XV

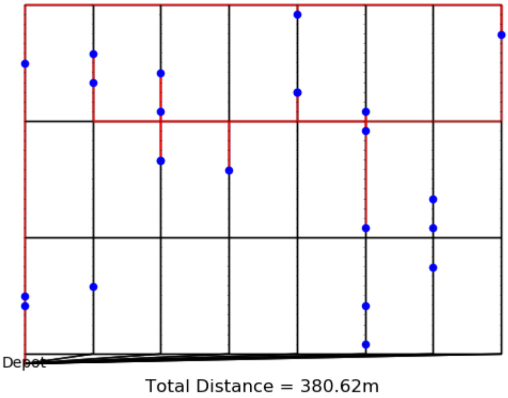
Largest Gap



Total Distance = 380.62m

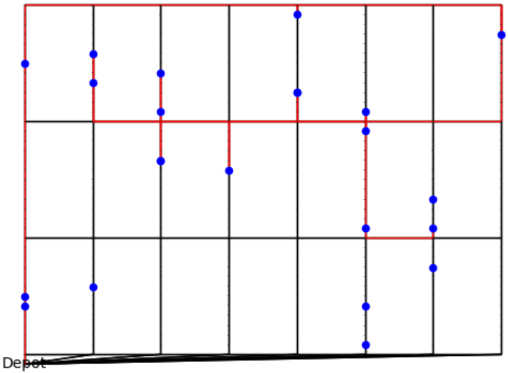
Heuristics Used XVI

Largest Gap



Heuristics Used XVII

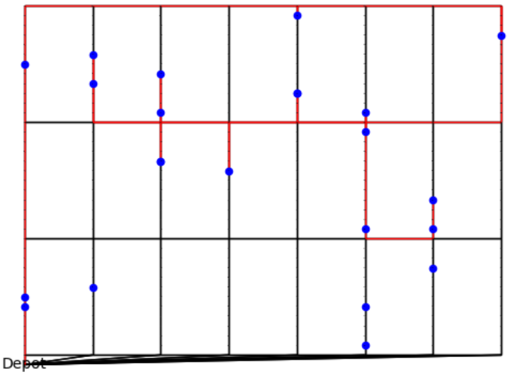
Largest Gap



Total Distance = 380.62m

Heuristics Used XVIII

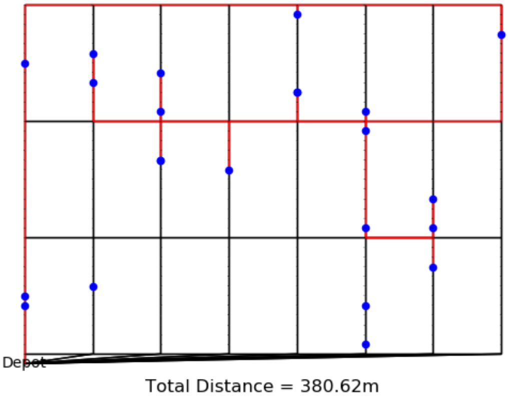
Largest Gap



Total Distance = 380.62m

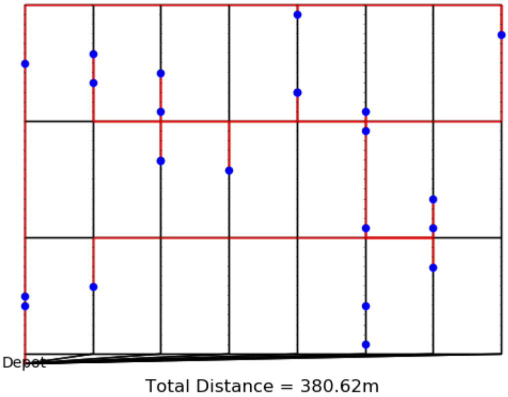
Heuristics Used XIX

Largest Gap



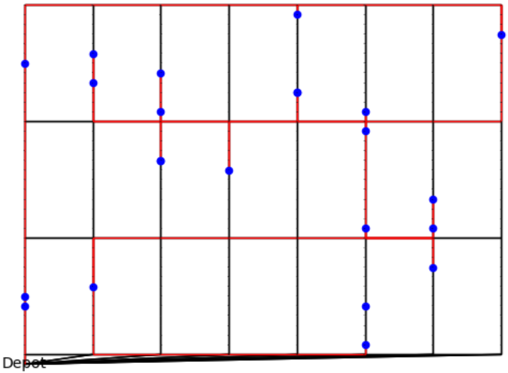
Heuristics Used XX

Largest Gap



Heuristics Used XXI

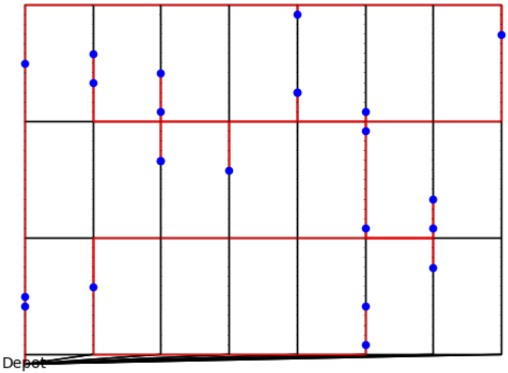
Largest Gap



Total Distance = 380.62m

Heuristics Used XXII

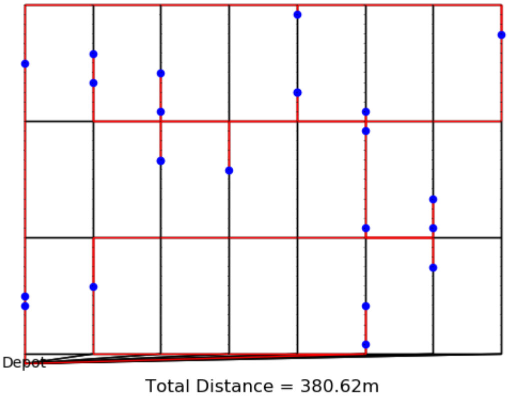
Largest Gap



Total Distance = 380.62m

Heuristics Used XXIII

Largest Gap



Heuristics Used I

Time Savings - Preliminaries

Picker Routing Heuristics: S-shape, Largest Gap

Batching Heuristic(s): Time Savings

Heuristics Used I

Time Savings - Purpose

- Merge orders in a greedy manner according to savings saved[19]
- Fewer batches of orders results in less distance travelled

Heuristics Used II

Time Savings - Purpose

Definition 1 (Savings)

For any 2 orders o_i and o_j, we have for a savings router sr,

$$\text{savings}_{ij} = sr(o_i) + sr(o_j) - sr(o_i \cup o_j)$$

where $sr(o_i)$ is the distance travelled to pick all items in order o_i.

[19] Clarke, G. and Wright, J.R. (1964) Scheduling of Vehicle Routing Problem from a Central Depot to a Number of Delivery Points.

Heuristics Used

Time Savings - An Example

Order i	1	2	3	4	5	6	7
1	-						
2	X	-					
3	74	X	-				
4	64	58	74	-			
5	68	X	94	54	-		
6	X	X	X	78	68	-	
7	-2	10	10	-4	10	-10	-

An example of a savings matrix[20]

Order i	Weight w_i
1	4
2	6
3	4
4	2
5	3
6	5
7	1

Weights of the orders

[20] M. B. M. De Koster , E.S. Van der Poort & M. Wolters, 1999. Efficient orderbatching methods in warehouses.

Heuristics Used I

Time Savings - An Example

Order i	1	2	3	4	5	6	7
1	-						
2	X	-					
3	74	X	-				
4	64	58	74	-			
5	68	X	94	54	-		
6	X	X	X	78	68	-	
7	-2	10	10	-4	10	-10	-

Order i	Weight w_i
1	4
2	6
3	4
4	2
5	3
6	5
7	1

Picker capacity $= 8$, Batches $= \{ \}$

1. Select pair with highest savings

$$\max \text{savings}_{ij} \rightarrow \{3, 5\}$$

$$\text{Batches} \rightarrow \text{Batches} + \{3, 5\}$$

Heuristics Used II

Time Savings - An Example

Order i	1	2	3	4	5	6	7
1	-						
2	X	-					
3	74	X	-				
4	64	58	74	-			
5	68	X	94	54	-		
6	X	X	X	78	68	-	
7	-2	10	10	-4	10	-10	-

Order i	Weight w_i
1	4
2	6
3	4
4	2
5	3
6	5
7	1

$$\text{Picker capacity} = 8, \text{ Batches} = \Big\{\{3,5\}\Big\}$$

2. Select next largest savings $\{4,6\}$. Since $w_4 + w_6 = 7 < 8$; Batches $\rightarrow$ Batches $+ \{4,6\}$

Heuristics Used III

Time Savings - An Example

Order i	1	2	3	4	5	6	7
1	-						
2	X	-					
3	74	X	-				
4	64	58	74	-			
5	68	X	94	54	-		
6	X	X	X	78	68	-	
7	-2	10	10	-4	10	-10	-

Order i	Weight w_i
1	4
2	6
3	4
4	2
5	3
6	5
7	1

$$\text{Picker capacity} = 8, \text{ Batches} = \Big\{\{3,5\},\{4,6\}\Big\}$$

3. Next largest savings pair(s): $\{1,3\},\{3,4\}$. Select $\{1,3\}$ as 3 & 4 already in a route.

   BUT $w_{1,3,5} = 11 > 8$ and $\{3,4\}$ not possible!

Heuristics Used IV

Time Savings - An Example

Order i	1	2	3	4	5	6	7
1	-						
2	X	-					
3	74	X	-				
4	64	58	74	-			
5	68	X	94	54	-		
6	X	X	X	78	68	-	
7	-2	10	10	-4	10	-10	-

Order i	Weight w_i
1	4
2	6
3	4
4	2
5	3
6	5
7	1

Picker capacity $= 8$, Batches $= \Big\{\{3,5\},\{4,6\}\Big\}$

4. Next possible pair: $\{2,7\}$

   Since $w_2 + w_7 = 7 < 8$, 2 and 7 are not in any route;

   Batches $\rightarrow$ Batches $+ \{2,7\}$

Heuristics Used V

Time Savings - An Example

Order i	1	2	3	4	5	6	7
1	-						
2	X	-					
3	74	X	-				
4	64	58	74	-			
5	68	X	94	54	-		
6	X	X	X	78	68	-	
7	-2	10	10	-4	10	-10	-

Order i	Weight w_i
1	4
2	6
3	4
4	2
5	3
6	5
7	1

$$\text{Picker capacity} = 8, \text{ Batches} = \Big\{\{3,5\},\{4,6\},\{2,7\}\Big\}$$

5. Remaining order: 1

   Since 1 cannot be added to any Batch;

   Batches $\rightarrow$ Batches $+ \{1\}$

   $\therefore$ Batches $= \{\{3,5\},\{4,6\},\{2,7\},\{1\}\}$.

Heuristics Used VI

Time Savings - An Example

Order i	1	2	3	4	5	6	7
1	-						
2	X	-					
3	74	X	-				
4	64	58	74	-			
5	68	X	94	54	-		
6	X	X	X	78	68	-	
7	-2	10	10	-4	10	-10	-

Order i	Weight w_i
1	4
2	6
3	4
4	2
5	3
6	5
7	1

$$\text{Picker capacity} = 8,\ \text{Batches} = \Big\{\{3,5\},\{4,6\},\{2,7\},\{1\}\Big\}$$

6. Now that batches of orders have been computed, use the **batch router** to compute the route of each batch.

Heuristics Used VII

Time Savings - An Example

```
savings = [ ];
for orders $o_i, o_j, o_{ij}$, $i = 1, ..., n, j = 1, ..., n, i \neq j$ do
    if $weight(i) + weight(j) \leq$ picker capacity then
        $s_{ij} = sr(o_i) + sr(o_j) - sr(o_{ij})$;
        savings$+ = [s_{ij}]$;
    end
end
Sort savings in decreasing sequence;
Batches = [ ];
for $s_{ij}$ in savings do
    if $i$ and $j \notin$ Batches then
        Batches $\leftarrow$ Batches $+ (i, j)$;
    end
    if $i \in$ Batches and $j \notin$ Batches and
     $weight(Batch\ i) + weight(j) \leq$ picker capacity
     then
        $\text{Batch}_i \leftarrow \text{Batch}_i + j$;
    end
    if $i \notin$ Batches and $j \in$ Batches and
     $weight(i) + weight(Batch\ j) \leq$ picker capacity
     then
        $\text{Batch}_i \leftarrow \text{Batch}_j + i$;
    end
end
Routes = [ ];
for Batch in Batches do
    Routes $\leftarrow br(Batch)$;
end
```

Outline

1. The Project (Recap)
   - Objectives
2. Literature Review & The Working Model
   - Preliminaries
   - The Order Batching Problem
   - The Picker Routing Problem
   - The Joint Order Batching and Picker Routing Problem
3. Solving the JOBPRP for Large Instances
   - Heuristics
   - Implementation
   - Experiments & Results
   - Conclusion & Future Work

Implementation

Routing Heuristics Overview

- The following were implemented in Python 3:
  - Shortest paths computed with NetworkX[21] library
  - Time Savings, S-shape and Largest Gap Heuristics
  - Preprocessing of warehouse parameters and raw order file data

[21] See documentation at: https://networkx.github.io/documentation/stable/

Implementation

Routing Heuristics - S-shape Code Snippet

```
node_sequence = [curr_pos]
for b in reduced_pick_blocks:
    last_pos = node_sequence[-1]
    closest_sa_and_path =
               self.get_closest_sa_from_pick(b, last_pos)
    closest_sa = closest_sa_and_path.closest_sa
    updated_b = self.reverse_subaisles_order(b, closest_sa)
    last_sa_in_b = updated_b[-1]
    for sa in updated_b:
        if updated_b.index(sa) % 2 == 0:
            node_sequence += list(reversed(sa.sa))
        elif updated_b.index(sa) % 2 != 0 and sa !=
                                      last_sa_in_b:
            node_sequence += sa.sa
        else:
            last_pick = sa.picks[-1]
            node_sequence += sa.sa[:last_pick.index+1]
```

Implementation

Setting Up A Workflow In Snakemake[22]

- A workflow management system described in Python based language
- To run experiments locally on our PCs
- To submit jobs to the cluster in NUS HPC

[22] See https://snakemake.readthedocs.io/en/stable/ for documentation.

Implementation

Setting Up A Workflow In Snakemake - Code Snippet

```
rule method1_nn_nn_d10ord10:
    output:
        "method1_d10ord10_nn_nn_8_1_3_1560.csv"
    shell:
        "python3 time_savings_heuristic.py "
        "-g ../data/foodmart/warehouseGenerator.pl "
        "-l ../data/foodmart/productsDB_1560_locations.txt "
        "-o ../data/foodmart/order/instances_d10_ord10.txt "
        "-csv {output} -c 320 -na 8 -nc 1 -ns 3 -np 1560 "
        "-sr nn -br nn"
```

Implementation

Submitting Jobs To NUS High Performance Computing (HPC)

- Experiments involving large warehouse instances were memory consuming
- Some instances took over a day to run with Method 2
- HPC made it possible to run multiple jobs at the same time
- HPC has large number of CPUs and memory at our disposal

Implementation

Submitting Jobs To NUS High Performance Computing (HPC) - Job Script Snippet

```
#!/bin/bash

## -P Exact_HPCTMP: Job project name
#PBS -P Exact_HPCTMP

## -q Queue_Name: which queue to sbmit the job to in HPC
## Note: parallel12 has wall time of 720 hours.
#PBS -q parallel12

## -l reserves 1 units of 1 cpus, 5GB memory for this job
#PBS -l select=1:ncpus=12:mem=5GB

## -j oe states to join output and error files together
#PBS -j oe

## -N Exact_Solving_Job_Output: set filename for standard output/error message.
#PBS -N Exact_HPCTMP_Job_Output

## Change to the working dir in the exec host
cd $PBS_O_WORKDIR;

##--- Put your exec/application commands below ---
## source /etc/profile.d/rec_modules.sh gets path where modules are installed
source /etc/profile.d/rec_modules.sh
module load python3.6.4
## permanently have MiniZinc driver on environment PATH in HPC
export PATH="$PATH:/home/svu/e0004335/minizinc/bin"
## exec_job is where snakemake inserts the command
{exec_job}
```

Outline

1. The Project (Recap)
    - Objectives
2. Literature Review & The Working Model
    - Preliminaries
    - The Order Batching Problem
    - The Picker Routing Problem
    - The Joint Order Batching and Picker Routing Problem
3. Solving the JOBPRP for Large Instances
    - Heuristics
    - Implementation
    - Experiments & Results
    - Conclusion & Future Work

Our Approach to Solving the JOBPRP - Heuristics I

Idea: Instantiate Time Savings[23] Heuristic (TSH) with 2 different routers

- **Savings router**: used to compute routing estimates in computing savings
- **Batch router**: used to route each picker after assignment of batched order

[23] M. B. M. De Koster , E.S. Van der Poort & M. Wolters, 1999. Efficient orderbatching methods in warehouses.

Our Approach to Solving the JOBPRP - Heuristics II

- Time Savings Heuristic has 2 parameters: savings router & batch router

	Method 1	**Method 2**	**Method 3**
Savings Router	S-shape, Largest Gap	S-shape, Largest Gap	Optimal
Batch Router	S-shape, Largest Gap	Optimal	Optimal

Breakdown of combinations of routers used in each method[24]

- PyConcorde[25], a Python wrap-around for the Concorde[26] TSP solver was used for optimal routing

[24] Cristiano Arbex Valle, John E. Beasley, and Alexandre Salles da Cunha, 2017. Optimally solving the joint order batching and picker routing problem.

[25] See PyConcorde documentation at: https://github.com/jvkersch/pyconcorde

[26] See Concorde TSP Solver documentation at: http://www.math.uwaterloo.ca/tsp/concorde.html

Our Approach to Solving the JOBPRP - Optimal Solving

- Used the JOBPRP ILP model in MiniZinc[27]
- In MiniZinc, we use the Coin-or Branch and Cut[28] (CBC) solver

[27] Based on the work of a previous intern. MiniZinc is a free and open-source constraint modeling language. See https://www.minizinc.org/

[28] CBC is an open-source mixed integer programming solver written in C++. See CBC documentation at: https://projects.coin-or.org/Cbc

Results & Analysis I

Definitions

Definition 2

$$\text{quality of solution} = \frac{\text{total distance without batching} - \text{objective value}}{\text{total distance without batching}}$$

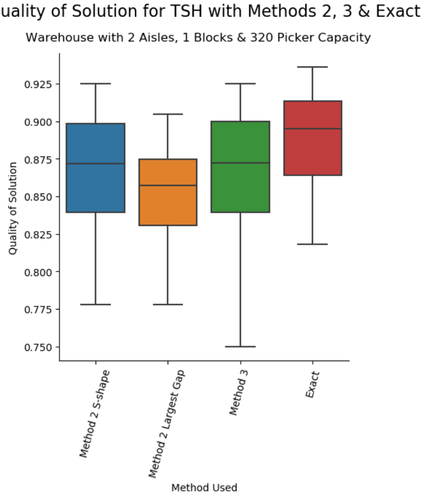
Example 3

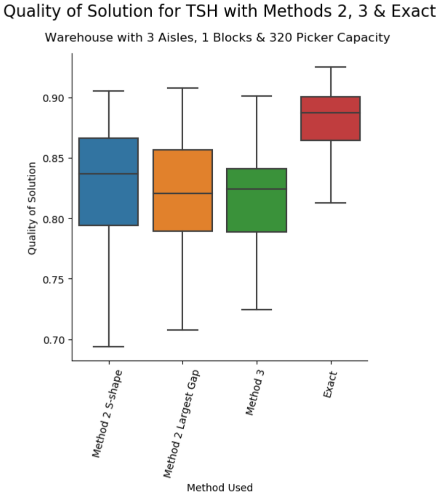
Suppose that the total distance without batching is 330, and the objective value from Method 3 is 170. Then we have

$$\text{quality of solution} = \frac{330 - 170}{330} = 0.4848$$

Results & Analysis - Exact Solving

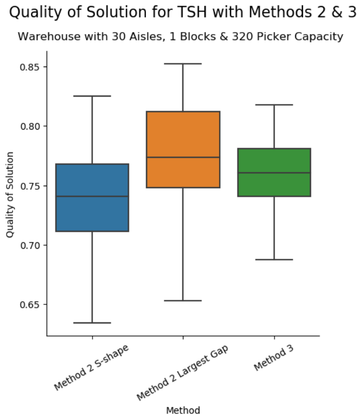
Results of Solution Quality with Methods 2 & 3, Exact Solving





Results & Analysis - Heuristic Solving I

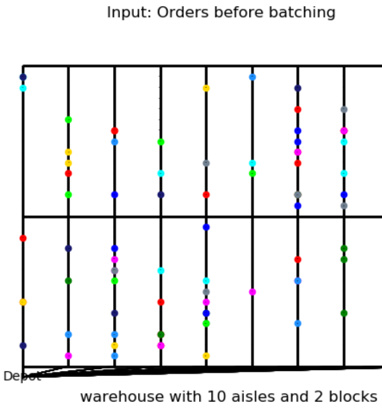
Results of Solution Quality with Methods 2 & 3



- Method 3 did not have the highest median quality of solution for most large warehouse instances

Results & Analysis - Heuristic Solving II

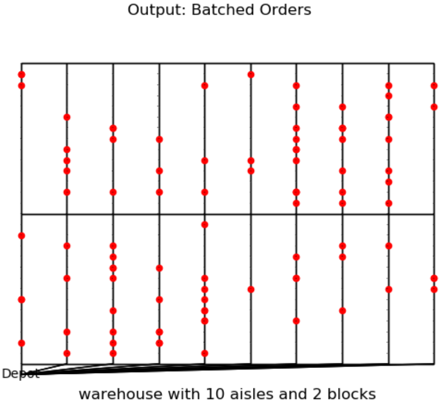
Results of Solution Quality with Methods 2 & 3


Input: Orders before batching
Depot
warehouse with 10 aisles and 2 blocks

Pick locations for each order (by color) before batching

Results & Analysis - Heuristic Solving III

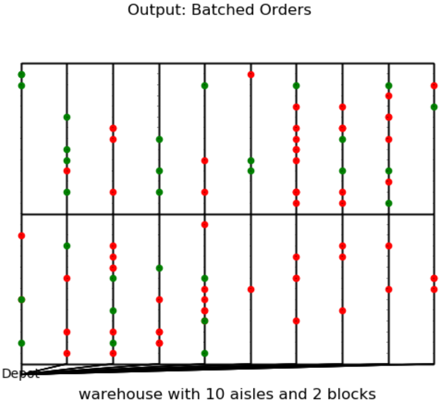
Results of Solution Quality with Methods 2 & 3



Pick locations for each order (by color) after batching with Method 2 - S-shape

Results & Analysis - Heuristic Solving IV

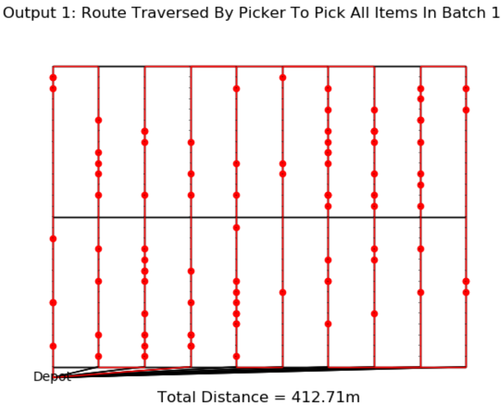
Results of Solution Quality with Methods 2 & 3



Pick locations for each order (by color) after batching with Method 3

Results & Analysis - Heuristic Solving V

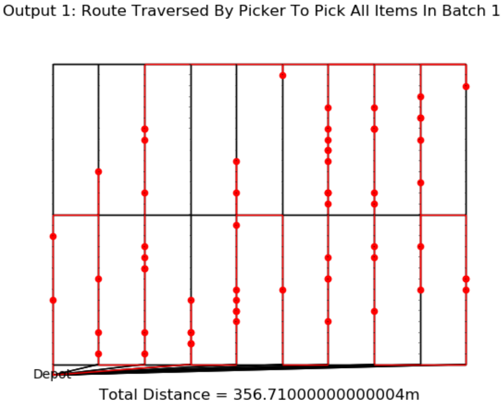
Results of Solution Quality with Methods 2 & 3


Output 1: Route Traversed By Picker To Pick All Items In Batch 1
Depot
Total Distance = 412.71m

Route with total distance for single batched order with Method 2 S-shape

Results & Analysis - Heuristic Solving VI

Results of Solution Quality with Methods 2 & 3

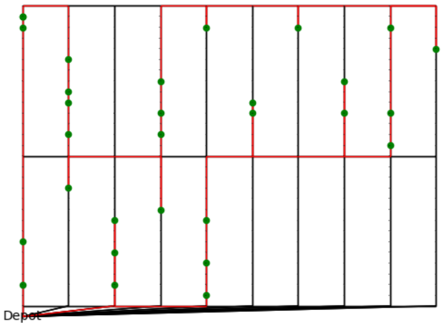

Output 1: Route Traversed By Picker To Pick All Items In Batch 1
Depot
Total Distance = 356.71000000000004m

Route with total distance for first batched order with Method 3

Results & Analysis - Heuristic Solving VII

Results of Solution Quality with Methods 2 & 3

Output 2: Route Traversed By Picker To Pick All Items In Batch 2



Total Distance = 276.81m

Route with total distance for second batched order with Method 3

Results & Analysis - Heuristic Solving VIII

Results of Solution Quality with Methods 2 & 3

- Using S-shape as savings router results in only one batched order as output
- Method 3 resulted in two batched orders as outputs
- Overall objective value is thus higher for Method 3

Results & Analysis - Heuristic Solving IX

Results of Solution Quality with Methods 2 & 3

Order *i*	1	2	3	4	5	6	7	8	9	10
1	-									
2	146.69	-								
3	156.47	156.26	-							
4	174.69	150.69	192.26	-						
5	173.09	130.69	152.47	154.69	-					
6	151.09	112.31	106.37	142.31	124.99	-				
7	100.9	86.69	100.47	100.69	86.9	72.52	-			
8	124.47	108.41	136.47	148.41	124.47	112.37	68.47	-		
9	147.09	128.69	140.47	144.69	165.09	124.71	92.9	116.47	-	
10	140.9	124.69	136.47	132.69	156.9	140.8	96.9	100.47	132.9	-

Method 2 S-shape Savings Matrix

$$\text{Batches} = \{\{3, 4, 1, 5, 9, 10, 2, 6, 8, 7\}\}$$

Results & Analysis - Heuristic Solving X

Results of Solution Quality with Methods 2 & 3

Order i	1	2	3	4	5	6	7	8	9	10
1	-									
2	96.8	-								
3	128.47	130.47	-							
4	142.42	112.8	144.37	-						
5	126.71	94.52	110.47	114.52	-					
6	112.71	92.71	92.52	104.42	100.71	-				
7	100.31	78.59	88.41	108.59	100.41	84.31	-			
8	104.47	92.47	118.47	126.47	100.47	104.37	88.41	-		
9	124.71	90.52	112.47	112.52	124.81	120.43	92.69	90.19	-	
10	126.47	110.37	122.47	120.47	132.47	128.71	92.26	112.47	124.19	-

Method 3 Savings Matrix

$$\text{Batches} = \{\{3, 4, 1, 2, 8, 7\}, \{5, 10, 6, 9\}\}$$

Results & Analysis - Heuristic Solving XI

Results of Solution Quality with Methods 2 & 3

So what happened during the batching?

Results & Analysis - Heuristic Solving XII

Results of Solution Quality with Methods 2 & 3

- Method 2 S-shape Batches $= \{\{3, 4, 1, 5, 9, 10, 2, 6, 8, 7\}\}$
- Method 3 Batches $= \{\{3, 4, 1, 2, 8, 7\}, \{5, 10, 6, 9\}\}$
- In Method 3's batching, Order 5 does not get added to the first batch $\{3, 4, 1\}$

Results & Analysis - Heuristic Solving XIII

Results of Solution Quality with Methods 2 & 3

Step 1 : $\{\{3, 4\}\}$
Step 2 : $\{\{3, 4, 1\}\}$
Step 3 : $\{\{3, 4, 1, 5\}\}$

Steps of Method 2's batching

Step 1 : $\{\{3, 4\}\}$
Step 2 : $\{\{3, 4, 1\}\}$
Step 3 : $\{\{3, 4, 1\}, \{5, 10\}\}$

Steps of Method 3's batching

Results & Analysis - Heuristic Solving XIV

Results of Solution Quality with Methods 2 & 3

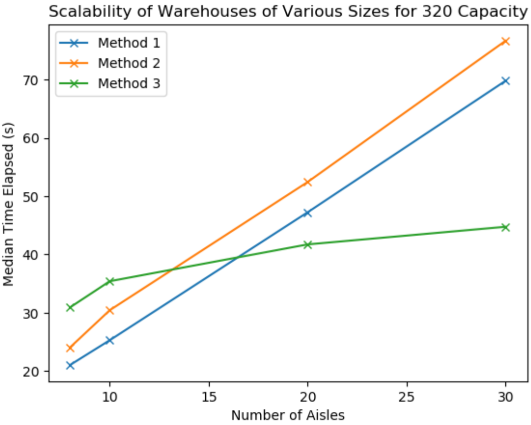
Order *i*	1	2	3	4	5	6	7	8	9	10
1	-									
2	96.8	-								
3	128.47	130.47	-							
4	142.42	112.8	144.37	-						
5	126.71	94.52	110.47	114.52	-					
6	112.71	92.71	92.52	104.42	100.71	-				
7	100.31	78.59	88.41	108.59	100.41	84.31	-			
8	104.47	92.47	118.47	126.47	100.47	104.37	88.41	-		
9	124.71	90.52	112.47	112.52	124.81	120.43	92.69	90.19	-	
10	126.47	110.37	122.47	120.47	132.47	128.71	92.26	112.47	124.19	-

Method 3 Savings Matrix

- For Method 3, order pair $(5, 10)$ had a higher savings than $(1, 5)$ in the third step
  - $s_{5,10} = 132.47 \geq 126.71 = s_{1,5}$

Results & Analysis - Heuristic Solving XV

Results of Solution Quality with Methods 2 & 3

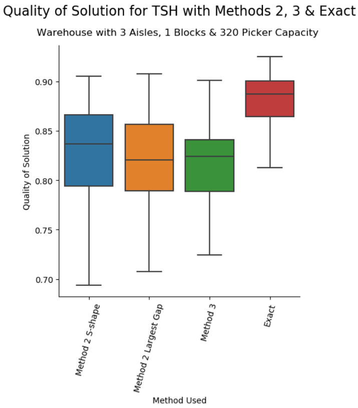


- Method 3 had the lowest median run-time than Methods 1 & 2 for 20 and 30 aisles

Outline

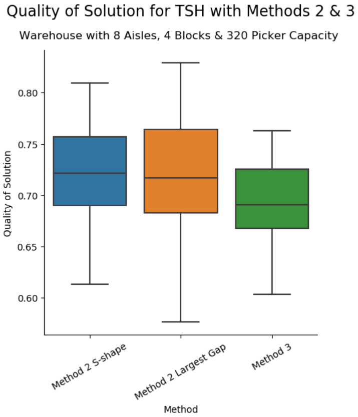
1. The Project (Recap)
   - Objectives
2. Literature Review & The Working Model
   - Preliminaries
   - The Order Batching Problem
   - The Picker Routing Problem
   - The Joint Order Batching and Picker Routing Problem
3. Solving the JOBPRP for Large Instances
   - Heuristics
   - Implementation
   - Experiments & Results
   - Conclusion & Future Work

Conclusion I



- We investigated the trade-offs of using heuristics in-place of exact solving

Conclusion II

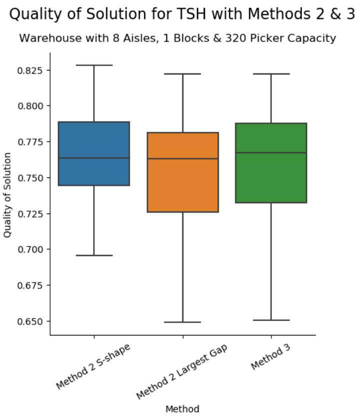


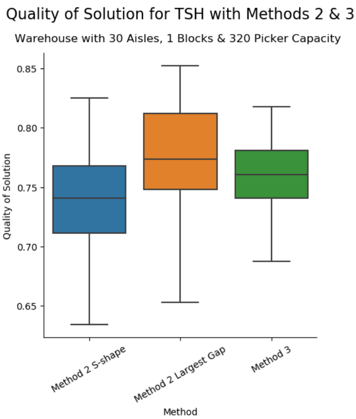
- Routing heuristics do indeed give a better quality of solution than optimal routing in most cases

Conclusion III

- ILP formulation takes a long time to solve, heuristics are faster

Conclusion IV





- Different methods and heuristics work better for certain warehouses

Future Work

- Improve the run-times for Method 2
- Batching heuristic using optimal routes of all items to cluster orders