Title	Model-based design of a forklift fleet management system to realize intelligent warehouse : joint optimization of storage location assignment and vehicle routing
Sub Title	
Author	冒, 嘉駒(Mao, Jiaju) 西村, 秀和(Nishimura, Hidekazu)
Publisher	慶應義塾大学大学院システムデザイン・マネジメント研究科
Publication year	2020
Jtitle	
JaLC DOI	
Abstract	
Notes	修士学位論文. 2020年度システムエンジニアリング学 第314号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40002001-00002020-0006

Master’s Dissertation 2020

Model-Based Design of a Forklift Fleet Management System to Realize Intelligent Warehouse

Joint Optimization of Storage Location Assignment and Vehicle Routing

Mao Jiaju
（Student ID Number: 81834576）

Supervisor Prof. Hidekazu Nishimura

September 2020

Graduate School of System Design and Management,
Keio University
Major in System Design and Management

SUMMARY OF MASTER'S DISSERTATION

Student Identification Number	81834576	Name	Mao Jiaju

Title

Model-Based Design of a Forklift Fleet Management System to Realize Intelligent Warehouse: Joint Optimization of Storage Location Assignment and Vehicle Routing

Abstract

Intelligence in warehouses is developing rapidly in various industries, with warehouse management system and internet of things deployed in more and more warehouses.

However, despite the fully automated or half-automated warehouses, there are still many traditional forklift warehouses operating in supply chains around the world. The intelligence in forklift warehouses is low compared to automated warehouses. Limited by the automation level, the forklift fleet is not cooperating with warehouse management systems as well as the automated equipment.

The purpose of this thesis is to propose a model-based process to design and optimize the forklift fleet management system. In the proposed model, the forklift fleet management system is studied in a broader picture, with warehouse management system and warehouse design in the context. The task list from the warehouse management system and warehouse layout from the warehouse design are utilized as input to the forklift fleet management system. Different from existing solutions, the new forklift fleet management system gathers and considers more information from the external systems. Therefore, it can perform a joint optimization of storage location assignment and vehicle routing.

Based on the context analysis, we design a new warehouse operation strategy which jointly optimizes storage location assignment and routing of material handling vehicles in the warehouse. the new strategy aims to minimize the traveling distance of the forklifts without changing its current layout. This new strategy will help warehouses to make better use of warehouse management system. It also provides better routing solutions to improve warehouse efficiency and handle growing storage and order picking tasks from the supply chain.

We also develop a novel algorithm within the forklift management system to process the information from task list and warehouse layout. With the algorithm, the information processing and decision making become automatic. Several functions in the algorithm ensures the quality of joint optimization.

We utilize a label function to index the storage locations in the warehouse with a structured grid. A distance function is also developed to calculate route length. The novelty of the algorithm is that it calculates both random assignment solutions and fixed assignment solutions. Therefore, more candidate routes are generated for further selection. Theoretically there is higher possibility to achieve optimal solution. The computational experiment proves that, compared to fixed assignment solutions, which is commonly used by traditional systems, the random assignment achieves better solutions in all 16 cases. In addition, the bigger the task size is, the more distance can be saved by the new algorithm.

Experiment proves the effectiveness of the new algorithm. The new algorithm has both economic significance and technical innovativeness.

The economic value is obvious: the new algorithm considers more information than previous solutions to make better management decisions. It provides an automatic comparison between random and fixed assignment methods to select optimal solution from a larger candidate group.

The technical value comes from compatibility, and scalability.

Compatibility means the algorithm will various scenarios and warehouse layouts. The scenarios can be at different automation level with requests of different complexity. Compatibility ensures that the fleet management system can adapt to external environments.

Scalability means the algorithm allows new vehicles to join the fleet. Even heterogeneous vehicles from external systems can join and leave the fleet flexibly under certain agreements. Scalability ensures that the system can manage fleets extending from small-scale to large-scale.

Key Word (5 words)
Model-Based Systems Engineering, Fleet Management System, Intelligent Vehicle, Storage Location Assignment, Vehicle Routing

Table of Contents

List of figures

List of tables

1 Introduction

1.1 Intelligent warehouse

With customers requiring shorter delivery time, the warehouse, as an important component of the supply chain, is faced with many new challenges. On the other hand, new information technologies and warehouse management systems “provide new opportunities to improve warehouse operations (Gu, J. et al., 2007)” [1].

Intelligence in warehouses is developing rapidly in various industries, with warehouse management system and internet of things deployed in more and more warehouses.

However, despite the fully automated or half-automated warehouses, there are still many traditional forklift warehouses operating in supply chains around the world. The intelligence in forklift warehouses is low compared to automated warehouses. Limited by the automation level, the forklift fleet is not cooperating with warehouse management systems as well as the automated equipment.

The purpose of this thesis is to propose a model-based process to design and optimize the forklift fleet management system. In the proposed model, the forklift fleet management system is studied in a broader picture, with warehouse management system and warehouse design in the context. The task list from the warehouse management system and warehouse layout from the warehouse design are utilized as input to the forklift fleet management system. Different from existing solutions, the new forklift fleet management system gathers and considers more information from the external systems. Therefore, it can perform a joint optimization of storage location assignment and vehicle routing. As summarized in Speranza, M. G, 2018, “better solutions to problems can be identified when broader parts of the supply chain are jointly modeled and optimized” [2].

1.2 Model-based systems engineering and warehouse operation

The design and operation of a warehouse is a complicated system. Decisions made in design phase, especially those on warehouse layout and operation strategies have high impact on efficiency of operations. Although layout has been proved to influence warehouse efficiency in a significant degree, the cost of changing layout is usually too much compared with the resulting benefits, making it not economical. That is why this thesis decides to focus on the design of strategies based on a determined typical layout. In the proposed strategy, the data from warehouse management system will be utilized to support both strategies. In SLAP, that means a data-driven location assignment. In VRP, that means a data-driven route which tells the fleet of forklifts how to move along the aisles and visit storage locations and docks. A solution to actualize such strategy is joint optimization of SLAP and VRP.

The joint optimization in warehouse operations, or logistics engineering activities, is also encouraged by INCOSE. “Logistics should be addressed from a life cycle perspective and be considered in all stages of a program and especially as an inherent part of system concept definition and development.” [3]

2 Forklift fleet management in literature

Preliminary studies include algorithms and models proposed fork different fleet management systems. Simao Hugo P et al [4] developed an approximate dynamic programming algorithm for large scale fleet management, which simulated the movements of over 6,000 drivers. Nair Rahul, and Elise Miller-Hooks [5] developed a model to manage fleet with vehicle sharing operations and used a novel divide-and-conquer algorithm. Bsaybes Sahar et al [6] proposed a framework to manage a fleet of individual public autonomous vehicles. Salazar-Cabrera et al [7] proposed a fleet management and control systems using intelligent transportation systems (ITS) services. Rudyk Tomasz et al [8] developed a model that considers safety and ecology in a sustainable fleet management model.

There are also efforts to apply latest techniques to fleet management systems. Ruan Yichen et al [9] used deep reinforcement learning to solve the dynamic vehicle dispatching problem for ride-hailing fleet network. Lin Kaixiang et al [10] used a multi-agent reinforcement learning framework to tackle the large-scale fleet management system.

3 A mathematic model for the forklift fleet joint optimization

The problem of forklift fleet joint optimization can be descripted as the following example.

At the beginning of a storage operation, a volume of stock keeping units (SKU) A is unloaded onto the dock. A fleet of homogeneous material handling vehicles (in this thesis, forklifts) is required to handle these SKU A to unoccupied storage locations. The problem is how to decide the storage locations and routes of the fleet so that the total travel distance can be minimized.

To simplify and formulate the problem, we make hypothesis as follows.

(1) the delivery and pick-up tasks of each forklift are given

(2) each forklift can handle both delivery and pick-up tasks simultaneously

(3) every two storage locations can be connected with at least one route and the length of the route is given

(4) each task must be handled by one and only forklift

(5) delivery and pick-up tasks can be handled within one route

(6) the weight, volume, and size capacity of forklifts are given and homogeneous

(7) each forklift departs from the depot and go back to the depot

Notation of the model is as Table 3.1.

Table 3.1 Notation

Symbol	Definition
V	Set of vertices, $V = \{0,1,\dots,n\}$
A	Set of arcs, $A = \{(i,j) \mid i,j \in V, i \neq j\}$
G	Graph composed by vertices and arcs, $G = (V,A)$
H	Set of routes, $H = \{H_1, H_2, \dots, H_m\}$
d_k	Traveling distance of route H_k
a_{kl}	A binary variable to represent whether a certain route visits a certain vertex or not, $a_{kl} = \begin{cases} 1, & H_k\ visits\ vertex\ l \\ 0, & H_k\ does\ not\ visit\ vertex\ l \end{cases}$
x_k	A binary variable to represent whether the routing plan contains a certain route or not, if the routing plan contains H_k, then $x_k = 1$; otherwise, $x_k = 0$.
Cw	Weight capacity of the forklift
Cv	Volume capacity of the forklift
cw_l	Total weight of the task at vertex l
cv_l	Total volume of the task at vertex l
K	Number of routes in the routing plan

We formulate the model as follows.

$$\min \sum_{k=1}^{m} x_k d_k \tag{1}$$

This is to calculate the total travel distance of the whole forklift fleet and ensure that the solution is optimal among all candidate solutions.

$$\sum_{k=1}^{m} x_k a_{kl} = 1, \forall \mathrm{l} \in \mathrm{V} \tag{2}$$

This is to ensure the related storage locations are all serviced and serviced only once. When it comes to forklift fleet management, it means the fleet visit the storage location only once, by only one forklift.

$$\sum_{k=1}^{m} x_k = K \tag{3}$$

Equation (3) ensures the routing plan contains K routes. In industrial warehouses, a forklift usually handles one item within one tour. In this case, K routes generates K tours, meaning there are K items contained in the storage task of the fleet.

$$\sum_{l=0}^{n} x_k a_{kl} cw_l < C_w k, \forall \mathrm{k} \tag{4}$$

This is to ensure the total weight of items in the task does not go beyond the capacity of the fleet.

$$\sum_{l=0}^{n} x_k a_{kl} cv_l < C_v k, \forall \mathrm{k} \tag{5}$$

This is to ensure the total volume of items in the task does not go beyond the capacity of the fleet.

4 A joint optimization algorithm

In Chapter 3.5, we formulated the optimization problem of forklift fleet management as a joint optimization of vehicle routing and storage location assignment. In Chapter 4, we will propose a novel and systematic algorithm to solve the joint optimization problem. The algorithm is programmed with MATLAB. Therefore, we will explain the detailed procedure of the algorithm step by step, with MATLAB code attached. Although the main code is written as one project, it can be divided into two parts: vehicle routing and storage location assignment. Each part can be furtherly divided into several functions. However, for a better explanation, we will first introduce an example case before the functions. The example case will illustrate how each function works and how the functions work with each other in the algorithm.

4.1 Example case

4.1.1 Example warehouse layout

We select one warehouse case from literatures as an example to demonstrate the joint optimization algorithm. In existing literatures, there have been many cases for benchmark calculation and comparison. The case we select is a classic industrial warehouse compatible with many types of material handling devices, including forklifts, therefore is suitable as an example for forklift fleet optimization algorithm. The case is proposed and used firstly by De Koster et al [11] in 1999.

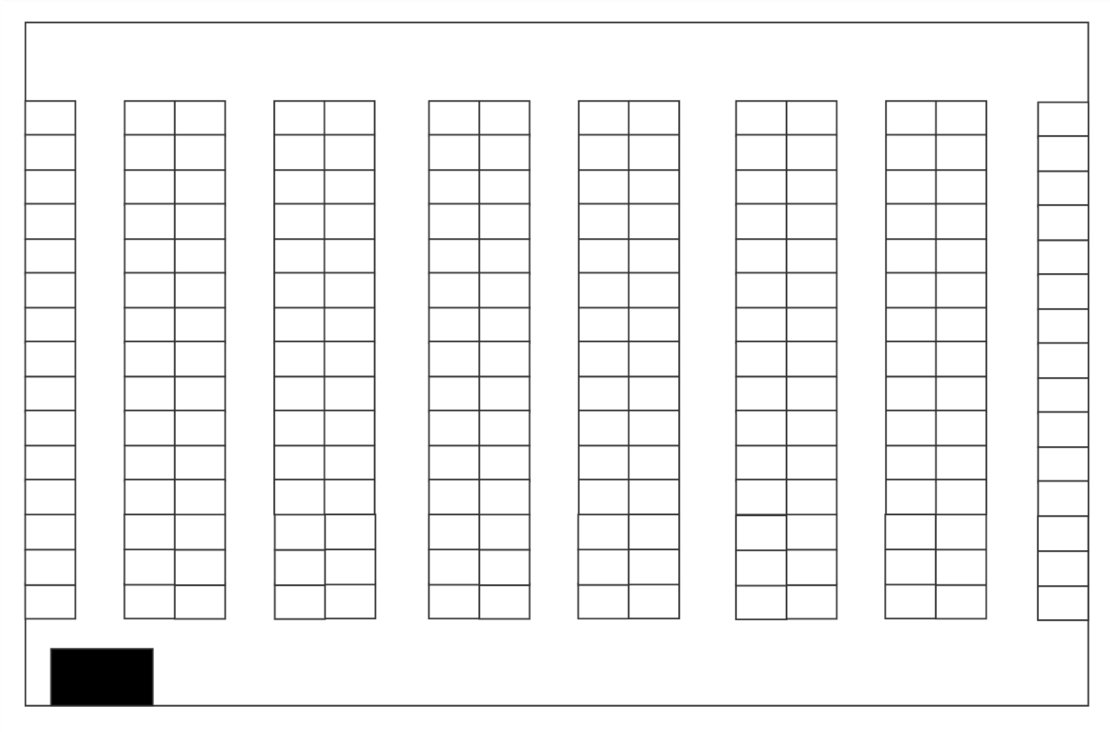


Figure 4.1 Example warehouse layout, including depot, shelves, and aisles

The case specifies the complete layout of a warehouse, including the number and location of depots, shelves, and aisles, as shown in Figure 4.1. The black block at the bottom represents for the depot in the warehouse, the blank blocks in the figure for shelves, and other space for the aisles. The shelves are divided into 14 racks by 7 aisles, with 15 item locations per rack. That is, 210 storage locations in total. Locations for all shelves and the depot are fixed. The depot is located at the bottom of the first aisle (first aisle counting from left to right).

4.1.2 Example task list

Aside from the warehouse layout, storage tasks should be specified as inputs to the algorithm. To study the performance of the algorithm under various situations, a task list is designed as Table 4.1. A task specifies a certain group of items to be handled by the forklift fleet. The list considers both SKU type and task size. For example, task 1 contains 10 items and they are categorized into 2 SKU types. The task list requires the forklift fleet to complete 16 tasks one by one. Therefore, the purpose of joint optimization algorithm is to generate executable orders from the task list and give theses orders to the forklift fleet.

Table 4.1 Example task list

task number	SKU types	task size
1	2	10
2	2	20
3	2	30
4	2	40
5	3	10
6	3	20
7	3	30
8	3	40
9	4	10
10	4	20
11	4	30
12	4	40
13	7	10
14	7	20
15	7	30
16	7	40

4.2 Vehicle routing strategies

The vehicle routing part of the algorithms is responsible to give routes to forklift fleet based on the warehouse layout. Routes include storage routes, order picking routes and emergency routes, designed for different tasks or situations.

4.2.1 storage routes

Based on the example case, a routing plan is generated as Figure 4.2. The routing plan utilize all 7 vertical aisles and 1 horizontal aisle to ensure each storage location can be accessed by the forklift fleet from the depot. The arrow on the routing plan means it is a routing plan for storage tasks.

We give 3 example routes in Figure 4.3 to show how the forklifts drive from the depot to 3 storage locations in different racks, following the routing plan, to operate storage tasks.

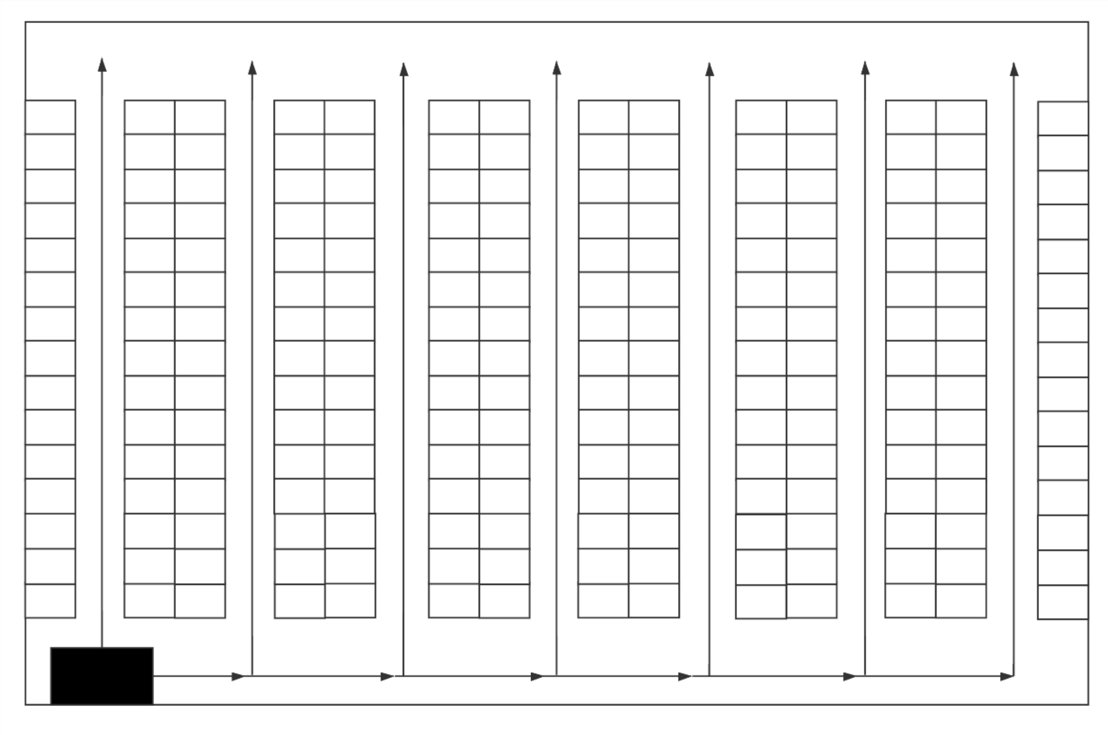


Figure 4.2 Routing plan for storage tasks in the example warehouse layout, 7 vertical aisles and 1 horizontal aisle

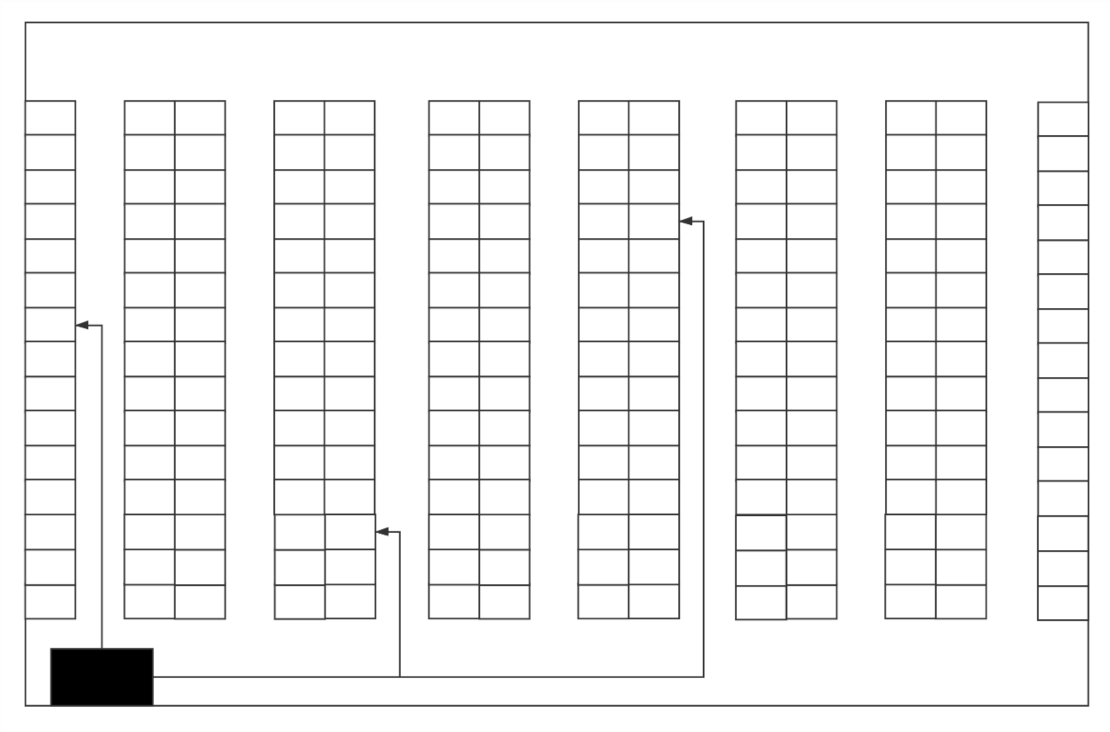


Figure 4.3 Three example routes of forklifts operating storage tasks

4.2.2 Order picking routes

According to the routing plan, routes for order picking tasks utilize the same aisles as storage tasks, only in opposite direction, as Figure 4.4 shows.

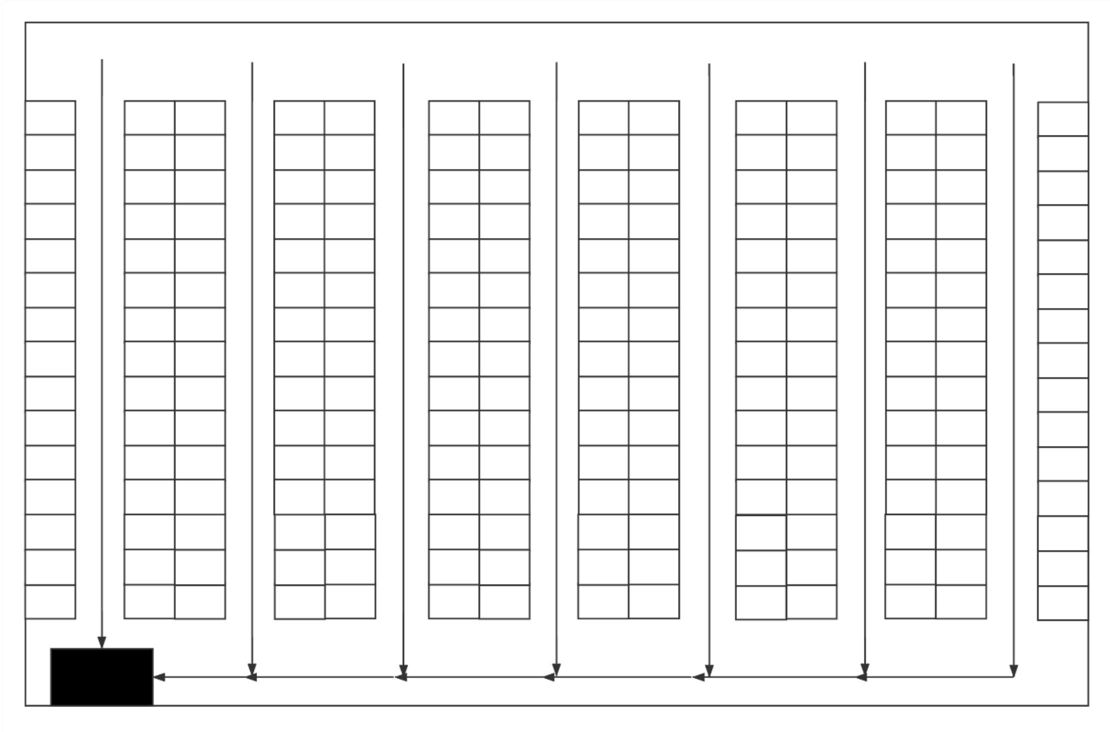


Figure 4.4 Routing plan for order picking tasks in the example warehouse layout, 7 vertical aisles and 1 horizontal aisle

We also give 3 order picking route examples in Figure 4.5. The forklifts need to follow the storage route in Figure 4.2 to arrive at 3 certain storage locations. Then after picking the items required by the task list, the forklifts follow the order picking routes and drive back to the depot.

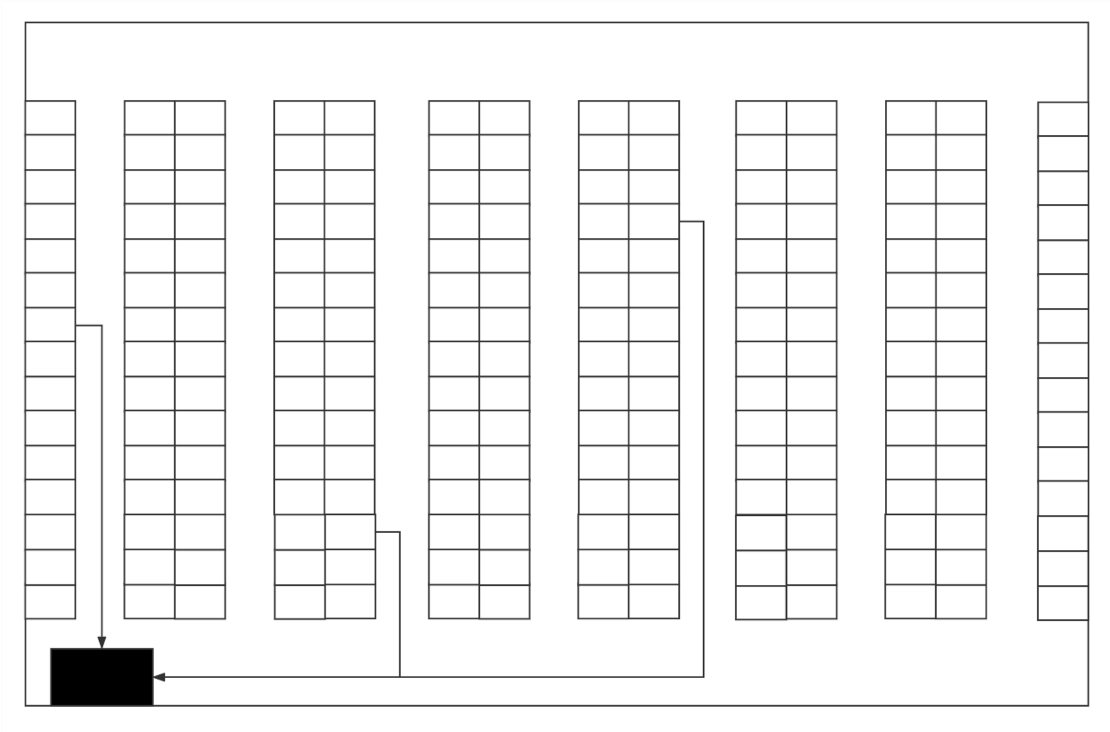


Figure 4.5 Three example routes of forklifts operating order picking tasks

Comparing Figure 4.3 and Figure 4.5, we see that the routing plan is composed of two information: the links between depot and each storage location, and the direction in which forklifts drive on the links. The links are consistent for both storage tasks and order picking tasks. For each location in the warehouse, the routing plan gives one and only route and ensures the given route is the shortest path from the depot to the location.

To complete storage tasks, the forklifts drive on the links in the forward direction, that is, from the depot to locations. When arriving at the target location, forklifts store the items and the task is completed. At last, forklifts drive on the links in the backward direction, that is, from locations to the depot to go back to the depot and get ready for the next task.

To complete order picking tasks, the forklifts drive in the forward direction and backward direction, the same as storage tasks. However, in storage tasks the useful tour is forward tour while in order picking tasks the useful tour is backward tour. Only in useful tour, the capacity of forklifts is utilized to handle items.

4.2.3 Emergency routes

Aside from storage routes and order picking routes, the routing plan also gives routes under emergencies. The routing plan define emergency as the situation that the usual (storage and order picking) route for a certain task is not accessible. To deal with such situation, the routes will be given as Figure 4.6.

The routing plan under emergency utilizes all 7 vertical aisles and 2 horizontal aisles and allows both directions in the aisles.

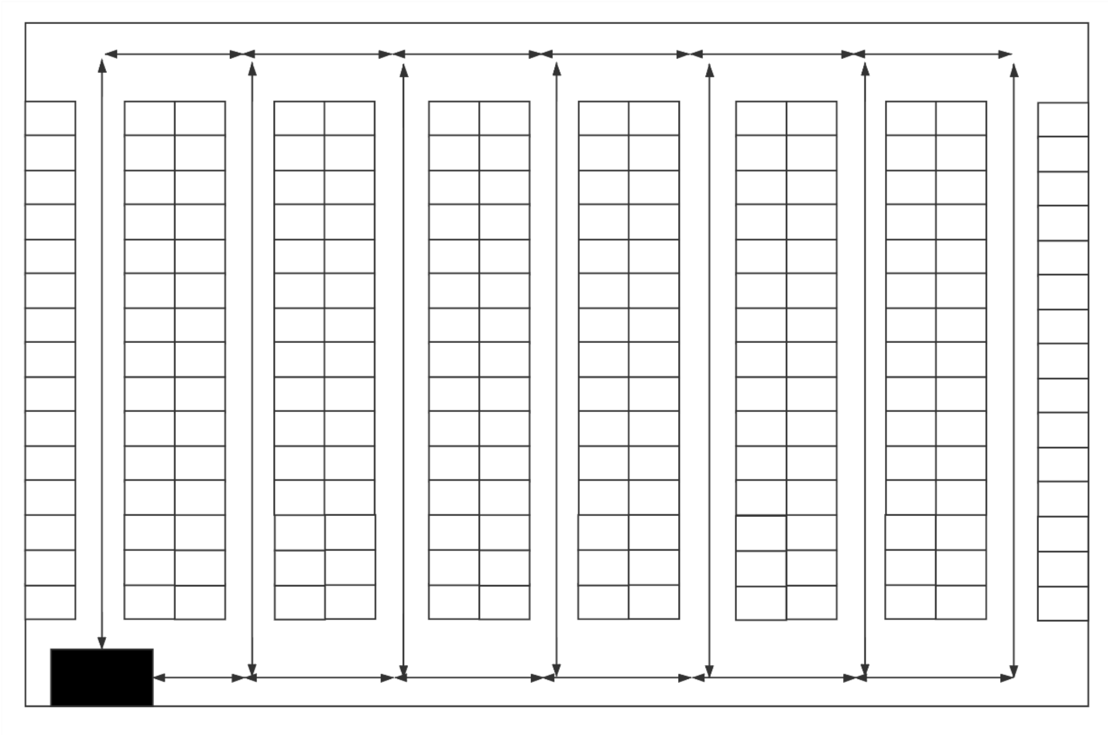


Figure 4.6 Routing plan under emergency in the example warehouse layout, 7 vertical aisles and 2 horizontal aisles

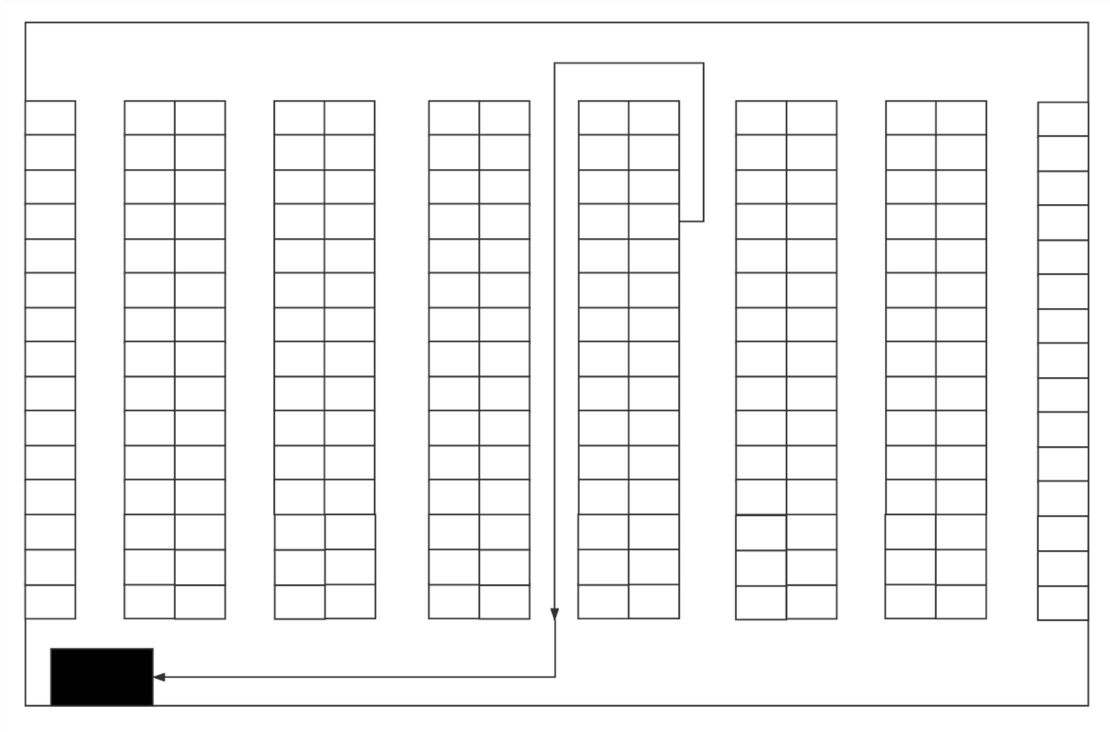


Figure 4.7 An example route under emergency

We give an example of emergency route in Figure 4.7. Comparing the route with usual routes in Figure 4.3 or Figure 4.5, we can notice there exists obvious detour in the route. Therefore the route is only used under emergency and never in usual tasks so that the total travel distance in usual cases can be controlled as minimal.

4.3 Storage location assignment algorithms

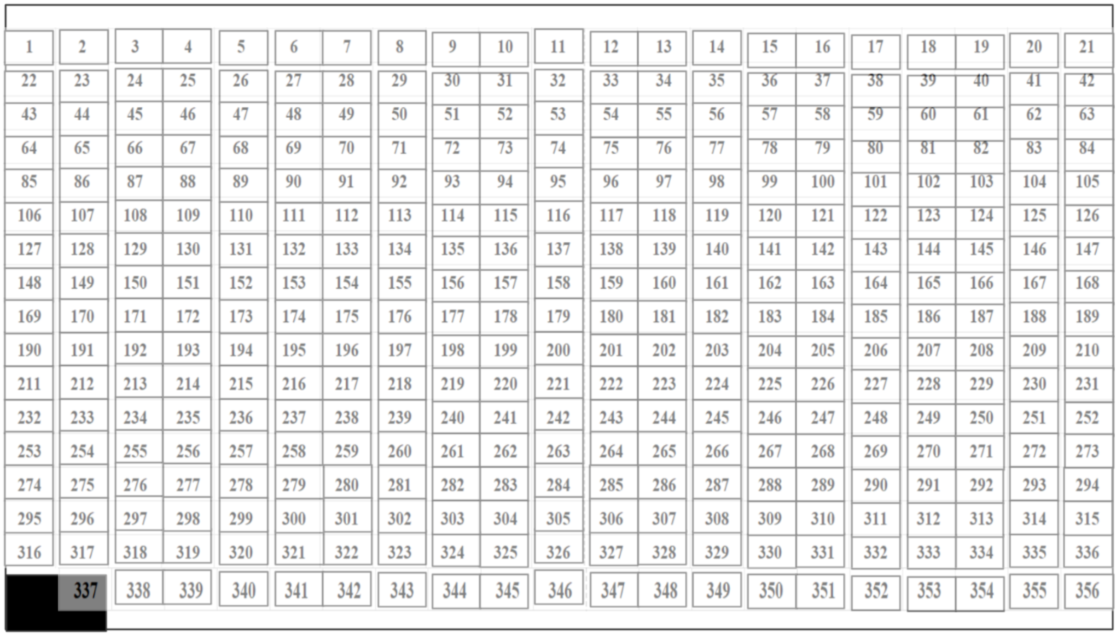
In the joint optimization algorithm, the vehicle routing part decides how the forklifts get to a certain storage location. However, the forklift fleet also needs to know which storage location to go to, which is decided by the storage location assignment algorithms.

When it comes to storage location assignment, two different assignment strategies are used in real practice: random storage location assignment and fixed storage location assignment. The algorithm we proposed considers both strategies and compare the results to achieve optimal solutions.

To assign the storage location, the algorithm needs to do several preliminary processing to the warehouse layout: building a grid for the warehouse layout, calculating the distance, and initializing the storage locations.

4.3.1 Building a grid for the warehouse layout

The algorithm needs to index all storage locations in the warehouse for further calculation and comparison. We build a grid for the warehouse layout and label each of the block in the grid to index all locations in the warehouse, both storage locations and aisle locations. The grid and labels are shown in Figure 4.8. With the labels, the algorithm can index any location inside the warehouse.

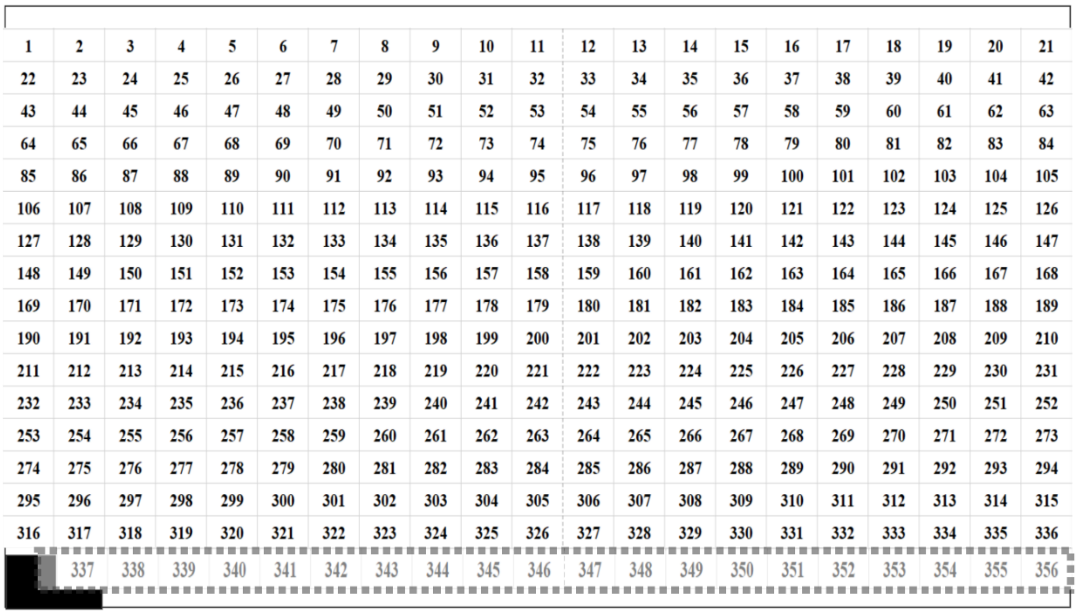


1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147
148	149	150	151	152	153	154	155	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	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273
274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294
295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315
316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356

Figure 4.8 Building a grid for all locations in the warehouse, label from 1 to 356

All locations labeled in Figure 4.8 can be categorized into two types: aisle locations and storage locations.

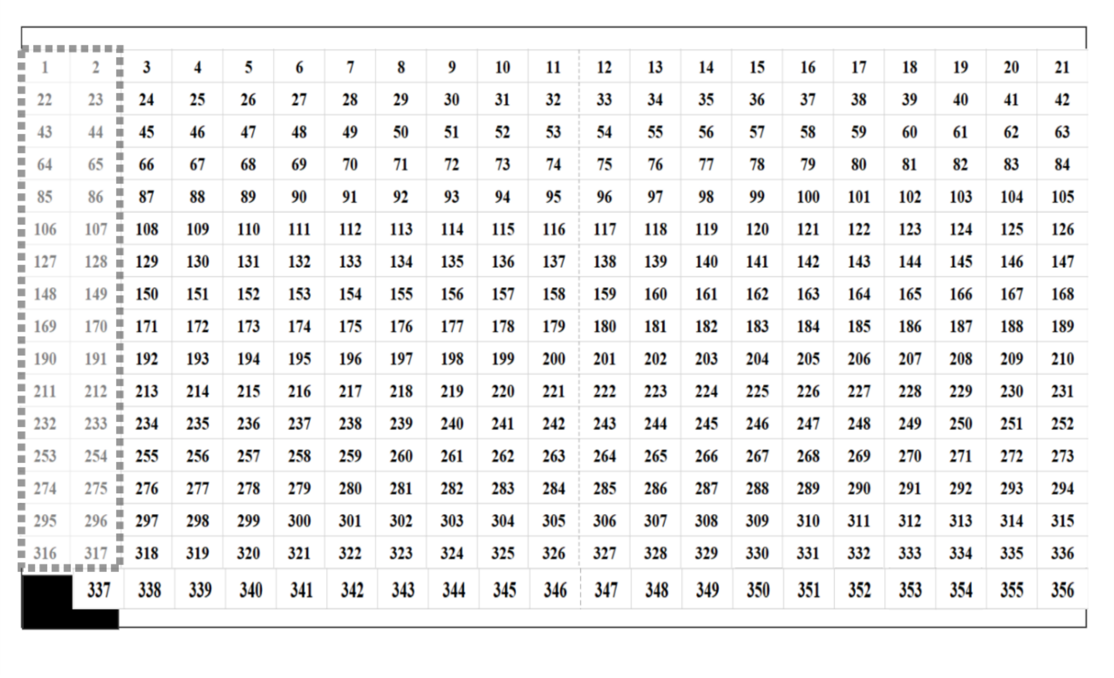
Figure 4.9 shows examples of aisle locations. The blocks circled by dash line represents for locations on the bottom horizontal aisle. As the figure shows, the aisle is divided into 19 blocks and labeled from 337 to 356.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147
148	149	150	151	152	153	154	155	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	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273
274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294
295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315
316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356

Figure 4.9 Exampled of aisle locations in the warehouse, circled by dash line

Figure 4.10 shows examples of storage locations and aisle locations next to storage location. The blocks circled by dash line are arranged into two columns. Blocks labeled 22, 43, 64, 85, 106, 127, 148, 169, 190, 211, 232, 253, 274, 295 and 316 represent for storage locations, while the other blocks represent for aisles.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147
148	149	150	151	152	153	154	155	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	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273
274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294
295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315
316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356

Figure 4.10 Exampled of storage locations and aisle locations in the warehouse, circled by dash line

The grid building and labeling is an important preparation for distance calculation and location assignment. Any mistake in this part will influence the results. Therefore, we add a quick verification in the MATLAB code to ensure the grid and labels are correct. The code is attached as appendix. The verification will plot the labels as Figure 4.11. In the figure, the height and color of the plot represent for the value of each label. If the plot surface is as even and smooth as Figure 4.11 shows, the grid building and labeling should be successful.

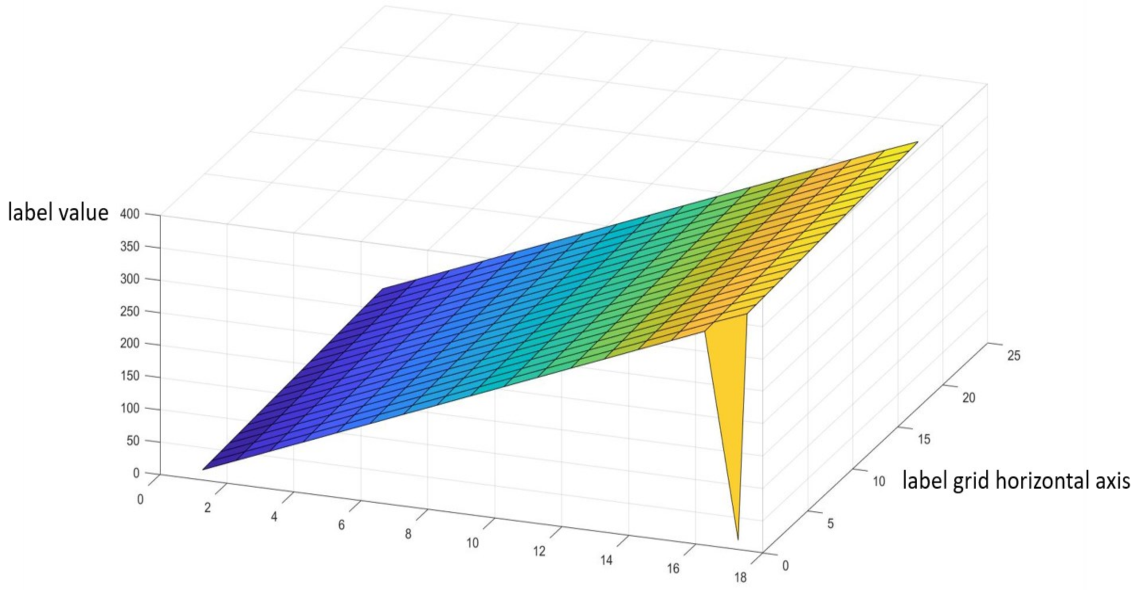
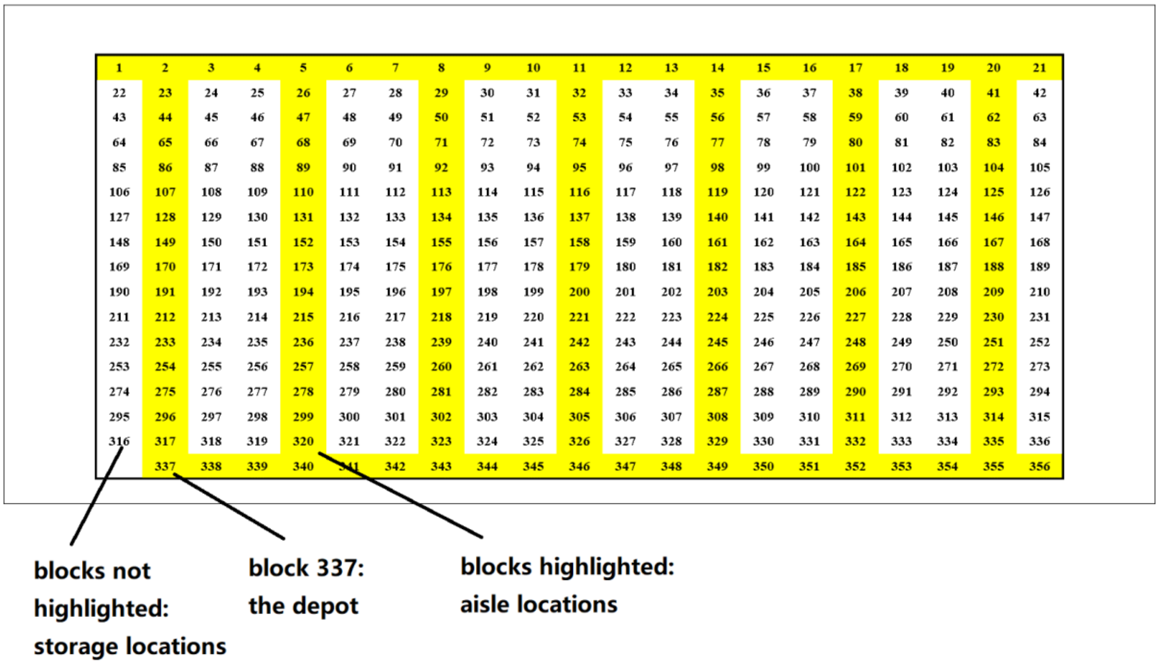


Figure 4.11 Quick verification of the grid and labels, results showing in the plot

4.3.2 Distance algorithm

The distance from depot to each storage location is essential for location assignment problems. In the algorithm, we propose a novel distance calculation method. To calculate the distance, we first build a graph to express the accessibility between every two locations in Figure 4.12. The blocks highlighted yellow are locations in aisles and the depot (block 337). Other blocks are storage locations.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147
148	149	150	151	152	153	154	155	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	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273
274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294
295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315
316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336
	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356

blocks not highlighted: storage locations

block 337: the depot

blocks highlighted: aisle locations

Figure 4.12 labeled locations in the warehouse grid

The graph generated from the algorithm is as Figure 4.13. Each node in the graph represent for a storage location or the depot in the warehouse, and the edges between every two nodes represent for the routes between storage locations. The distance of each route is also calculated as the graph is built, using the “dist” function in appendix.

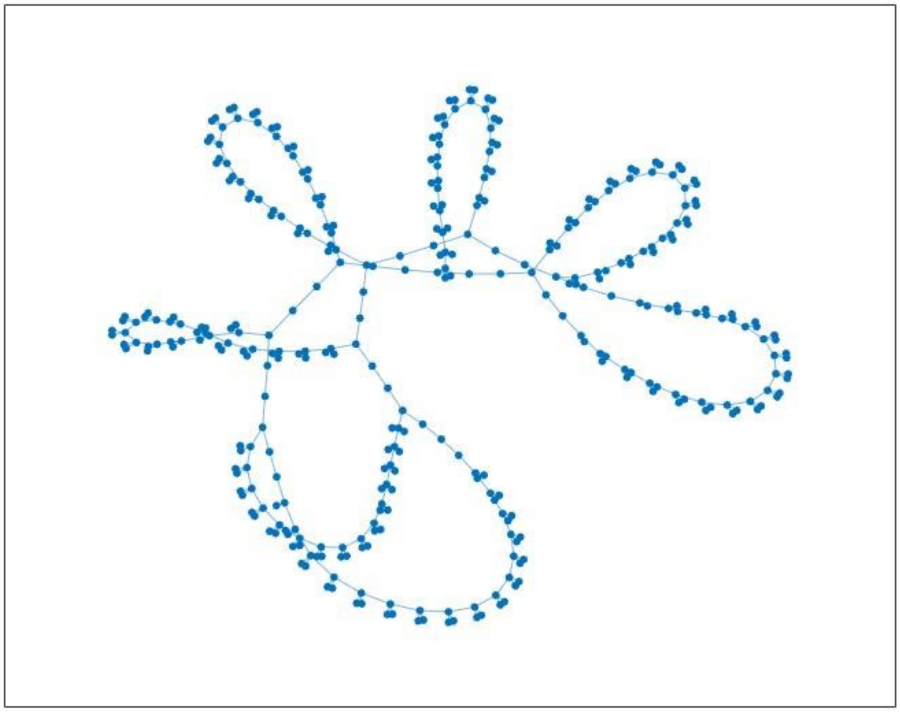


Figure 4.13 Graph generated from the algorithm

4.3.3 Initialization of the storage location

Since the storage locations are labeled, the algorithm can initialize the warehouse. By initialization, the algorithm registers the status of each location in the warehouse as occupied or unoccupied. As shown in Figure 4.14, shadow blocks mean occupied locations while blank blocks mean unoccupied. In the algorithm, we use “randperm” function to decide which location to be occupied. Code of the function is attached as appendix.

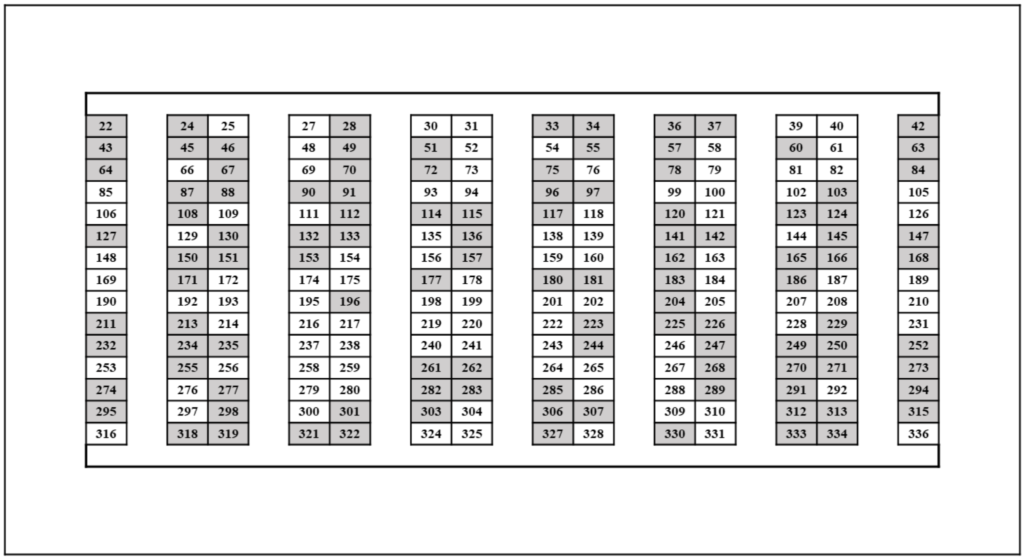


Figure 4.14 Initialized storage locations in the warehouse

4.3.4 Random storage location assignment algorithm

In the random storage location assignment algorithm, we select the nearest unoccupied location for the coming tasks, without considering the SKU types contained in the task. Therefore, the only difference between tasks is the task size and the results of case 1, 5, 9 and 13 should be the same. It also applies for case 2, 3 and 4. The results under random assignment are summarized in Table 4.2, 4.3, 4.4 and 4.5, with the warehouse status visualized as Figure 4.15, 4.16, 4.17 and 4.18.

Table 4.2 Case 1, 5, 9 and 13, random assignment results

label	distance
316	2
297	3
276	4
253	5
300	6
279	7
192	8
256	8
258	8
324	8

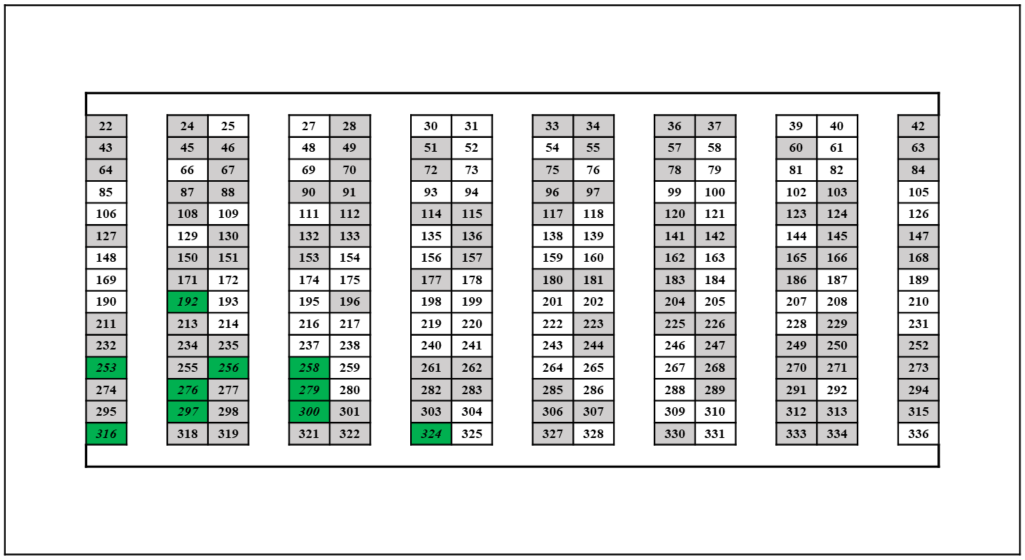


Figure 4.15 Case 1, 5, 9 and 13, random assignment results visualized in the warehouse

Table 4.3 Case 2, 6, 10 and 14, random assignment results

label	distance
316	2
297	3
276	4
253	5
300	6
279	7
192	8
256	8
258	8
324	8
190	8
169	9
237	9
216	10
214	10
148	10
280	10
259	11
129	11
195	11

Figure 4.16 Case 2, 6, 10 and 14, random assignment results visualized in the warehouse

Table 4.4 Case 3, 7, 11 and 15, random assignment results

label	distance
316	2
297	3
276	4
253	5
300	6
279	7
192	8
256	8
258	8
324	8
190	8
169	9
237	9
216	10
214	10
148	10
280	10
259	11
129	11
195	11
325	11
193	11
238	12
174	12
106	12
172	12
240	12
304	12
85	13
217	13

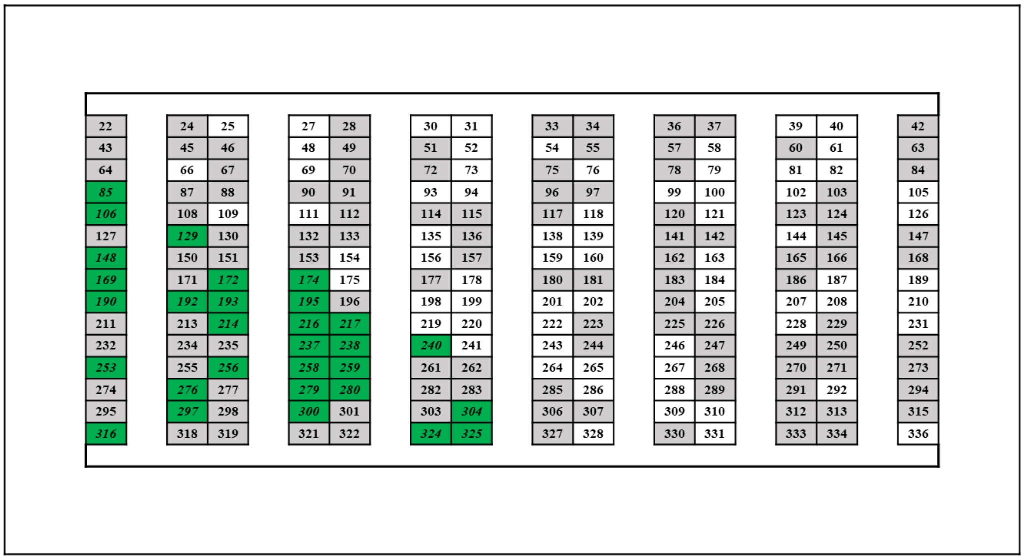


Figure 4.17 Case 3, 7, 11 and 15, random assignment results visualized in the warehouse

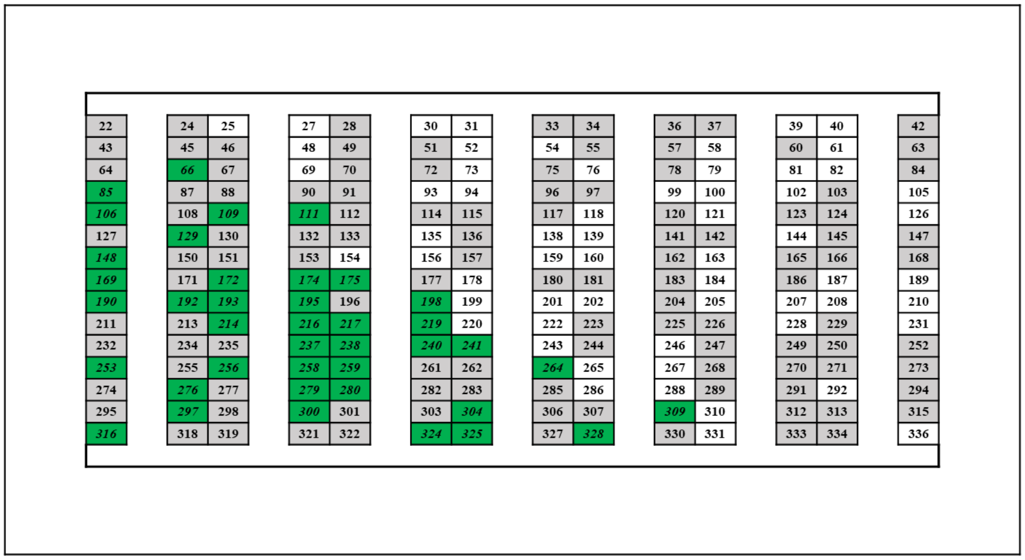


Figure 4.18 Case 4, 8, 12 and 16, random assignment results visualized in the warehouse

Table 4.5 Case 4, 8, 12 and 16, random assignment results

label	distance
316	2
297	3
276	4
253	5
300	6
279	7
192	8
256	8
258	8
324	8
190	8
169	9
237	9
216	10
214	10
148	10
280	10
259	11
129	11
195	11
325	11
193	11
238	12
174	12
106	12
172	12
240	12
304	12
85	13
217	13
219	13
328	14
264	14
66	14
198	14
175	15
241	15
309	15
111	15
109	15

4.3.5 Fixed storage location assignment algorithm

The algorithm also considers fixed assignment method. By “fixed”, we mean the storage locations in the warehouse are divided into several departments. Each department can store only one type of item. The number of departments is consistent with the number of SKU types in each storage task. The results of Case 1 to 16 are summarized as Table 4.6 to 4.21 and visualized as Figure 4.19 to 4.34.

Table 4.6 Case 1, fixed assignment results

label	distance
316	2
253	5
297	3
276	4
300	6
264	14
243	15
222	16
328	14
309	15

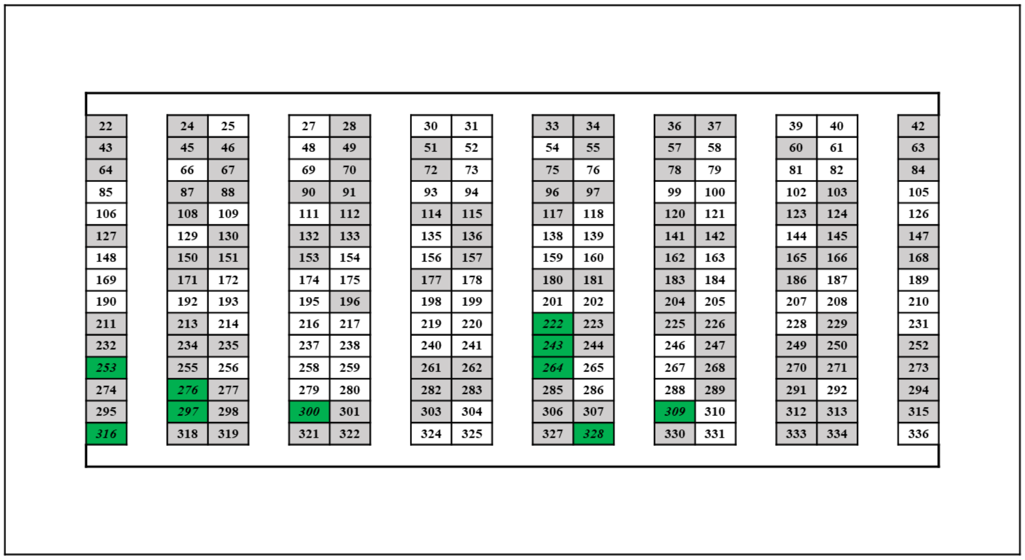


Figure 4.19 Case 1, fixed assignment results visualized in the warehouse

Table 4.7 Case 2, fixed assignment results

label	distance
316	2
253	5
190	8
297	3
276	4
192	8
256	8
300	6
279	7
258	8
264	14
243	15
222	16
201	17
328	14
286	16
265	17
309	15
288	16
267	17

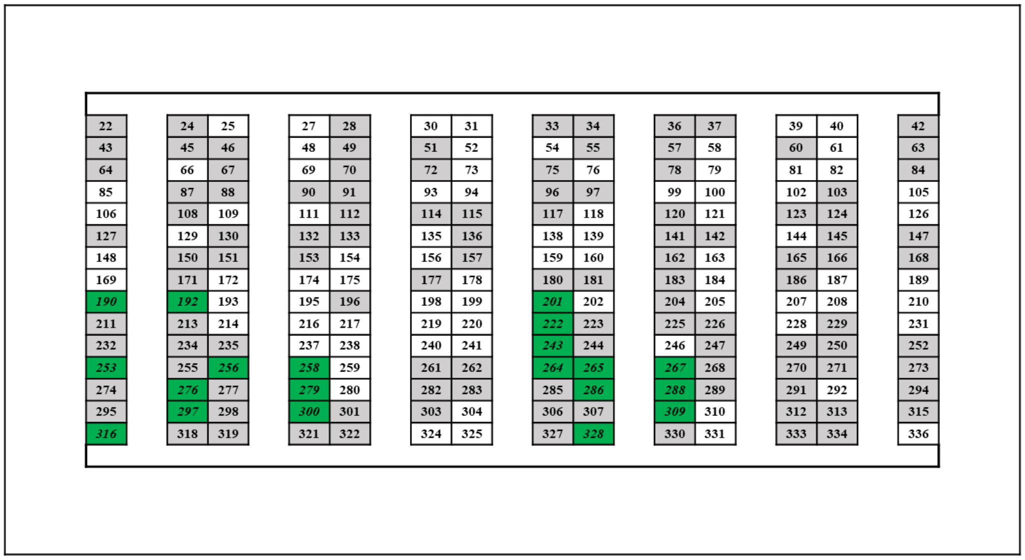


Figure 4.20 Case 2, fixed assignment results visualized in the warehouse

Table 4.8 Case 3, fixed assignment results

label	distance
316	2
253	5
190	8
169	9
148	10
297	3
276	4
192	8
256	8
214	10
300	6
279	7
258	8
280	10
324	8
264	14
243	15
222	16
201	17
159	19
138	20
328	14
286	16
265	17
309	15
288	16
267	17
246	18
331	17
310	18

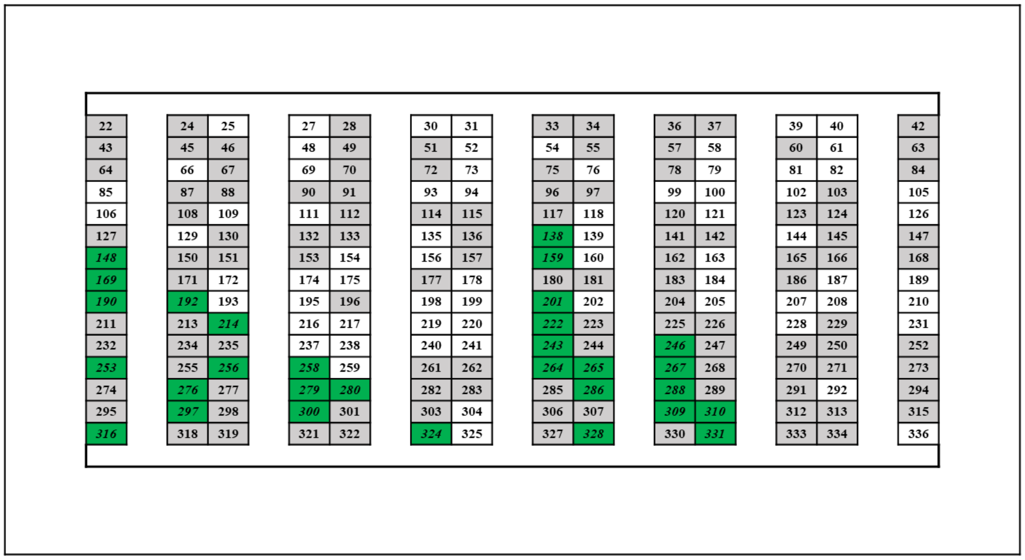


Figure 4.21 Case 3, fixed assignment results visualized in the warehouse

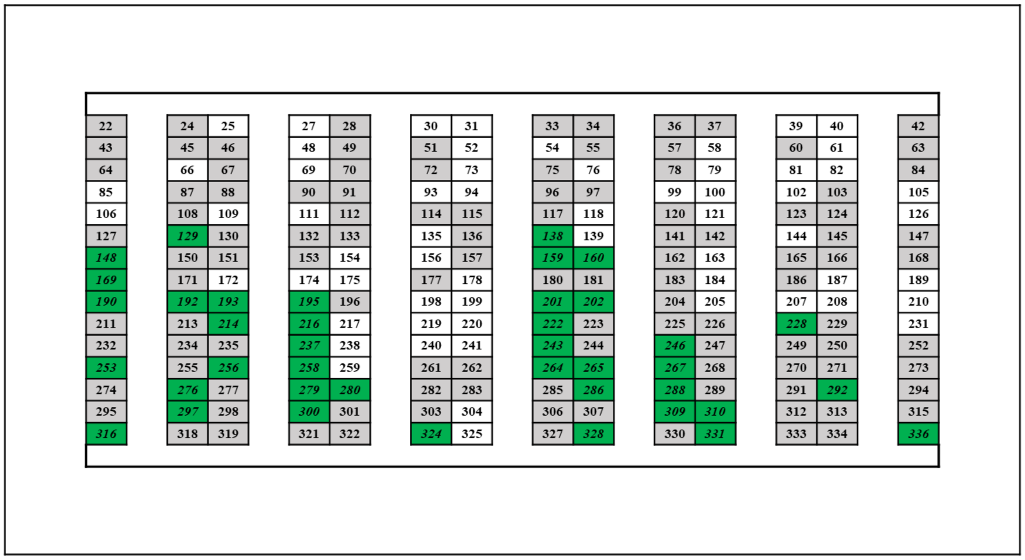


Figure 4.22 Case 4, fixed assignment results visualized in the warehouse

Table 4.9 Case 4, fixed assignment results

label	distance
316	2
253	5
190	8
169	9
148	10
297	3
276	4
192	8
129	11
256	8
214	10
193	11
300	6
279	7
258	8
237	9
216	10
195	11
280	10
324	8
264	14
243	15
222	16
201	17
159	19
138	20
328	14
286	16
265	17
202	20
160	22
309	15
288	16
267	17
246	18
331	17
310	18
228	22
292	22
336	20

Table 4.10 Case 5, fixed assignment results

label	distance
316	2
253	5
297	3
276	4
324	8
240	12
325	11
309	15
288	16
267	17

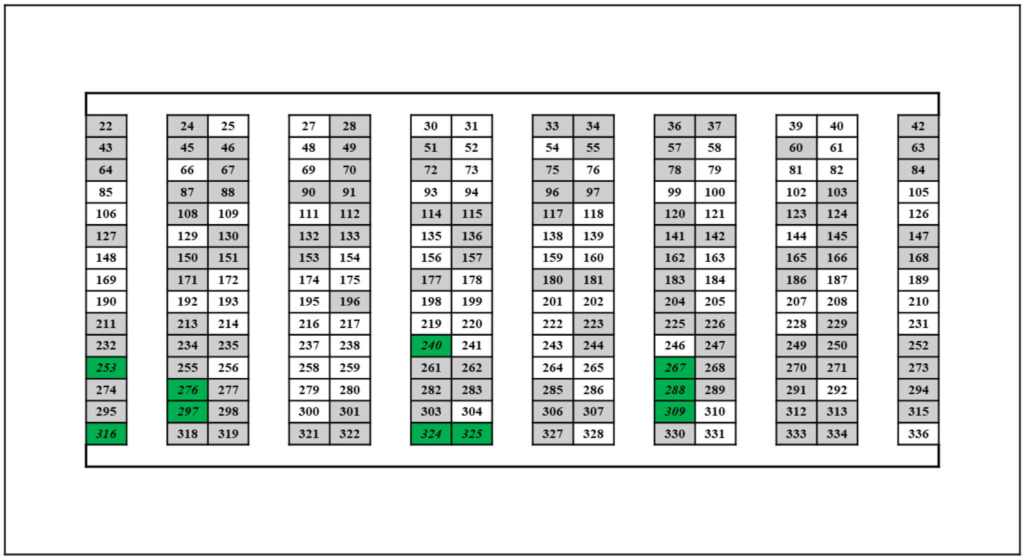


Figure 4.23 Case 5, fixed assignment results visualized in the warehouse

Table 4.11 Case 6, fixed assignment results

label	distance
316	2
253	5
190	8
297	3
276	4
300	6
279	7
324	8
240	12
219	13
198	14
325	11
304	12
264	14
309	15
288	16
267	17
246	18
331	17
310	18

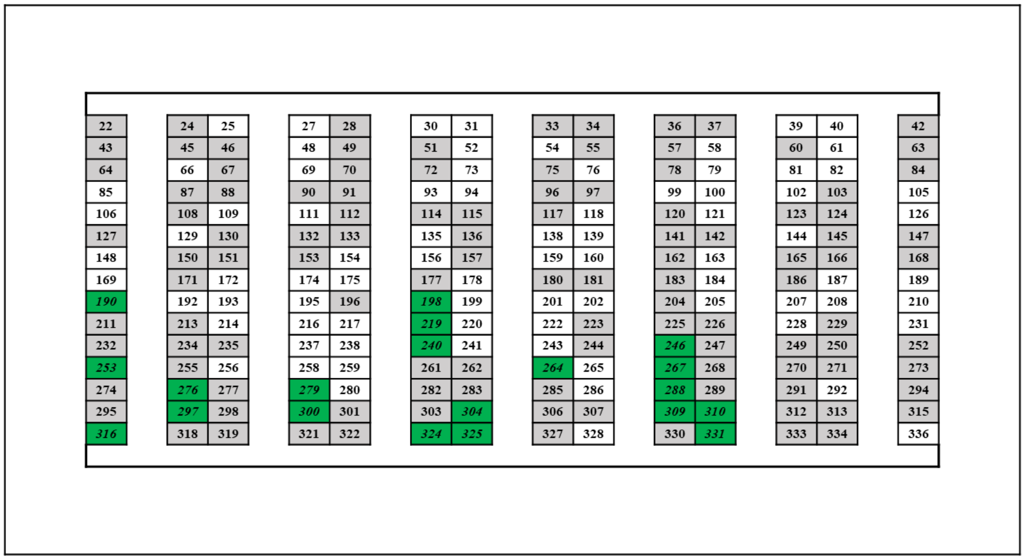


Figure 4.24 Case 6, fixed assignment results visualized in the warehouse

Table 4.12 Case 7, fixed assignment results

label	distance
316	2
253	5
190	8
297	3
276	4
192	8
256	8
300	6
279	7
258	8
324	8
240	12
219	13
198	14
325	11
304	12
241	15
264	14
243	15
328	14
309	15
288	16
267	17
246	18
331	17
310	18
205	23
228	22
292	22
336	20

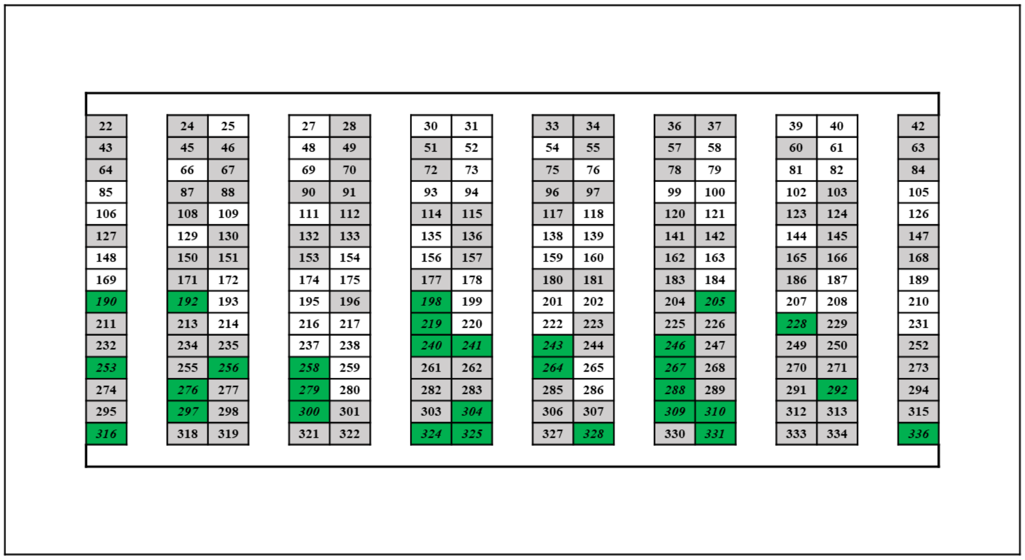


Figure 4.25 Case 7, fixed assignment results visualized in the warehouse

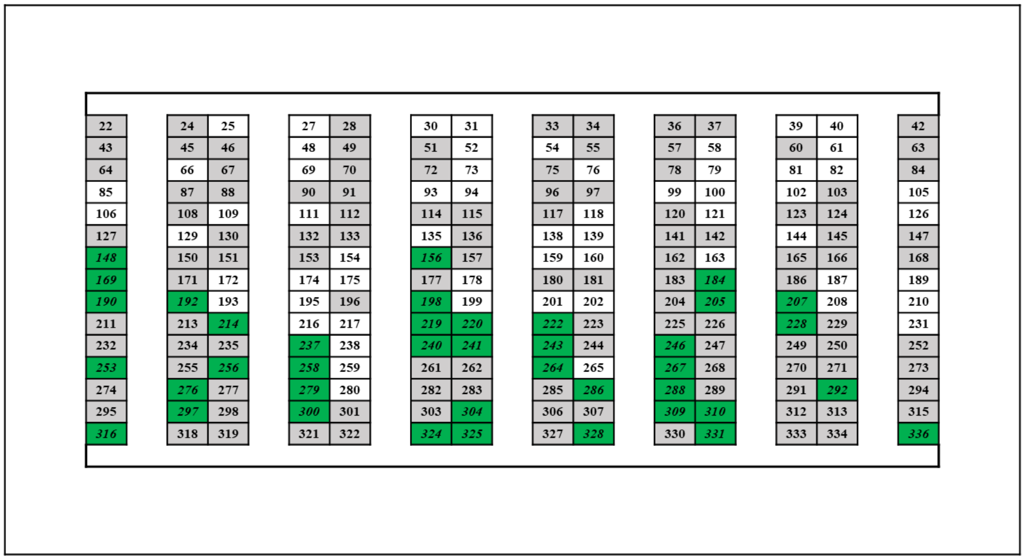


Figure 4.26 Case 8, fixed assignment results visualized in the warehouse

Table 4.13 Case 8, fixed assignment results

label	distance
316	2
253	5
190	8
169	9
148	10
297	3
276	4
192	8
256	8
214	10
300	6
279	7
258	8
237	9
324	8
240	12
219	13
198	14
156	16
325	11
304	12
241	15
220	16
264	14
243	15
222	16
328	14
286	16
309	15
288	16
267	17
246	18
331	17
310	18
205	23
184	24
228	22
207	23
292	22
336	20

Table 4.14 Case 9, fixed assignment results

label	distance
316	2
297	3
276	4
300	6
279	7
258	8
264	14
328	14
228	22
336	20

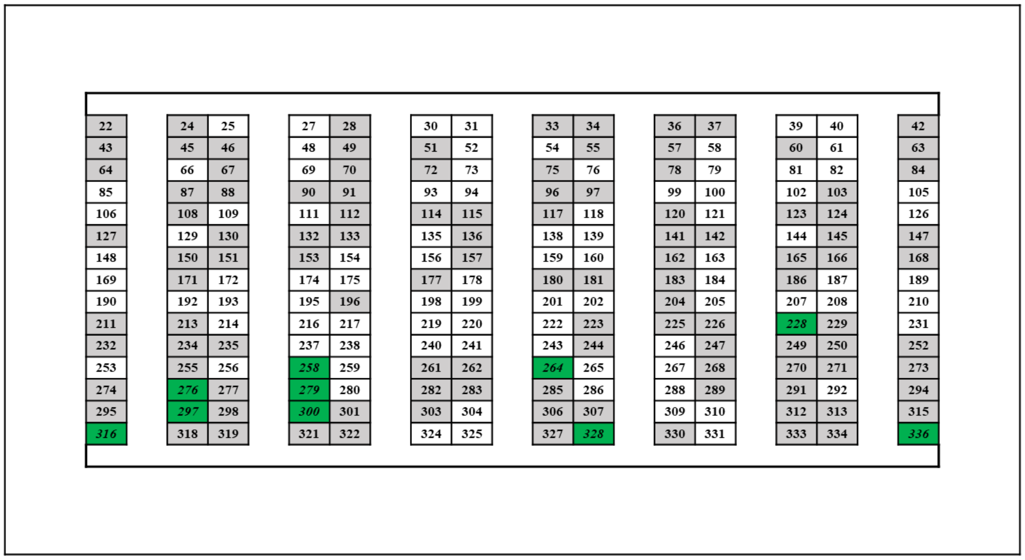
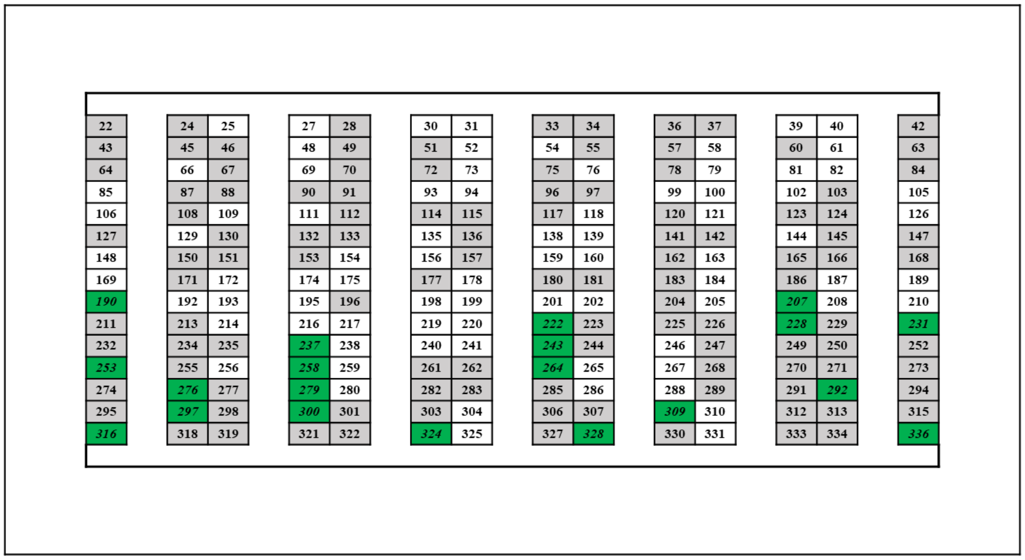


Figure 4.27 Case 9, fixed assignment results visualized in the warehouse

Table 4.15 Case 10, fixed assignment results

label	distance
316	2
253	5
190	8
297	3
276	4
300	6
279	7
258	8
237	9
324	8
264	14
243	15
222	16
328	14
309	15
228	22
207	23
292	22
336	20
231	25

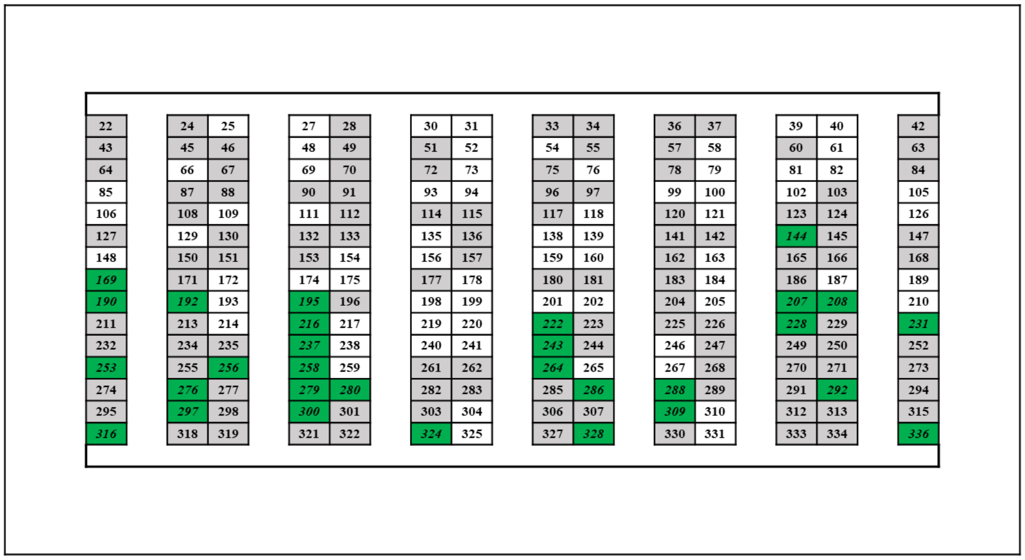


22	24	25	27	28	30	31	33	34	36	37	39	40	42
43	45	46	48	49	51	52	54	55	57	58	60	61	63
64	66	67	69	70	72	73	75	76	78	79	81	82	84
85	87	88	90	91	93	94	96	97	99	100	102	103	105
106	108	109	111	112	114	115	117	118	120	121	123	124	126
127	129	130	132	133	135	136	138	139	141	142	144	145	147
148	150	151	153	154	156	157	159	160	162	163	165	166	168
169	171	172	174	175	177	178	180	181	183	184	186	187	189
190	192	193	195	196	198	199	201	202	204	205	*207*	208	210
211	213	214	216	217	219	220	*222*	223	225	226	*228*	229	*231*
232	234	235	*237*	238	240	241	*243*	244	246	247	249	250	252
253	255	256	*258*	259	261	262	*264*	265	267	268	270	271	273
274	*276*	277	*279*	280	282	283	285	286	288	289	291	*292*	294
295	*297*	298	*300*	301	303	304	306	307	*309*	310	312	313	315
316	318	319	321	322	*324*	325	327	*328*	330	331	333	334	*336*

Figure 4.28 Case 10, fixed assignment results visualized in the warehouse

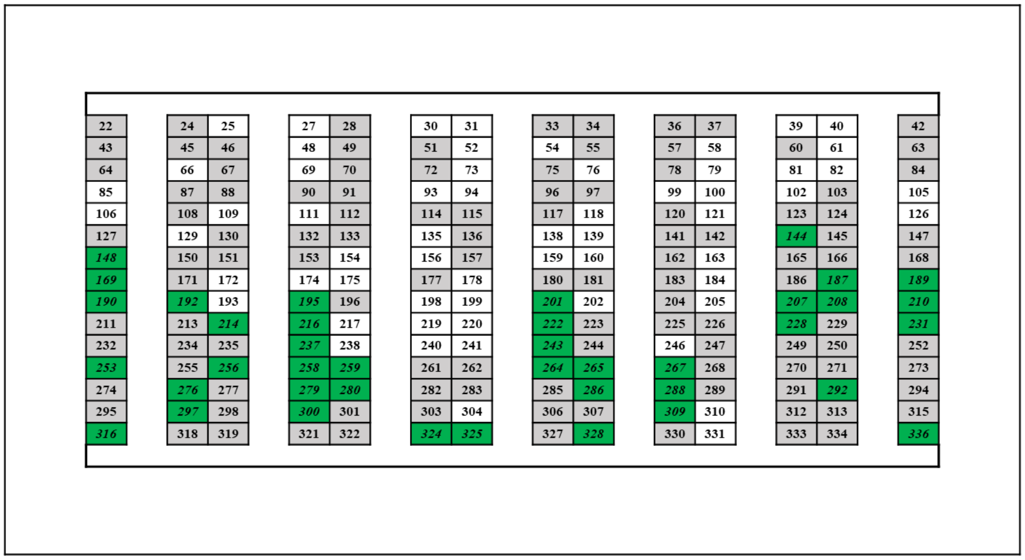
Table 4.16 Case 11, fixed assignment results

label	distance
316	2
253	5
190	8
169	9
297	3
276	4
192	8
256	8
300	6
279	7
258	8
237	9
216	10
195	11
280	10
324	8
264	14
243	15
222	16
328	14
286	16
309	15
288	16
228	22
207	23
144	26
292	22
208	26
336	20
231	25



22	24	25	27	28	30	31	33	34	36	37	39	40	42
43	45	46	48	49	51	52	54	55	57	58	60	61	63
64	66	67	69	70	72	73	75	76	78	79	81	82	84
85	87	88	90	91	93	94	96	97	99	100	102	103	105
106	108	109	111	112	114	115	117	118	120	121	123	124	126
127	129	130	132	133	135	136	138	139	141	142	*144*	145	147
148	150	151	153	154	156	157	159	160	162	163	165	166	168
169	171	172	174	175	177	178	180	181	183	184	186	187	189
190	*192*	193	*195*	196	198	199	201	202	204	205	*207*	*208*	210
211	213	214	*216*	217	219	220	*222*	223	225	226	*228*	229	*231*
232	234	235	*237*	238	240	241	*243*	244	246	247	249	250	252
253	255	*256*	*258*	259	261	262	*264*	265	267	268	270	271	273
274	*276*	277	*279*	*280*	282	283	285	*286*	*288*	289	291	*292*	294
295	*297*	298	*300*	301	303	304	306	307	*309*	310	312	313	315
316	318	319	321	322	*324*	325	327	*328*	330	331	333	334	*336*

Figure 4.29 Case 11, fixed assignment results visualized in the warehouse



22	24	25	27	28	30	31	33	34	36	37	39	40	42
43	45	46	48	49	51	52	54	55	57	58	60	61	63
64	66	67	69	70	72	73	75	76	78	79	81	82	84
85	87	88	90	91	93	94	96	97	99	100	102	103	105
106	108	109	111	112	114	115	117	118	120	121	123	124	126
127	129	130	132	133	135	136	138	139	141	142	*144*	145	147
148	150	151	153	154	156	157	159	160	162	163	165	166	168
169	171	172	174	175	177	178	180	181	183	184	186	*187*	*189*
190	*192*	193	*195*	196	198	199	*201*	202	204	205	*207*	*208*	*210*
211	213	*214*	*216*	217	219	220	*222*	223	225	226	*228*	229	*231*
232	234	235	*237*	238	240	241	*243*	244	246	247	249	250	252
253	255	*256*	*258*	*259*	261	262	*264*	*265*	*267*	268	270	271	273
274	*276*	277	*279*	*280*	282	283	285	*286*	*288*	289	291	*292*	294
295	*297*	298	*300*	301	303	304	306	307	*309*	310	312	313	315
316	318	319	321	322	*324*	*325*	327	*328*	330	331	333	334	*336*

Figure 4.30 Case 12, fixed assignment results visualized in the warehouse

Table 4.17 Case 12, fixed assignment results

label	distance
316	2
253	5
190	8
169	9
148	10
297	3
276	4
192	8
256	8
214	10
300	6
279	7
258	8
237	9
216	10
195	11
280	10
259	11
324	8
325	11
264	14
243	15
222	16
201	17
328	14
286	16
265	17
309	15
288	16
267	17
228	22
207	23
144	26
292	22
208	26
187	27
336	20
231	25
210	26
189	27

Table 4.18 Case 13, fixed assignment results

label	distance
316	2
297	3
300	6
279	7
280	10
324	8
325	11
328	14
331	17
336	20

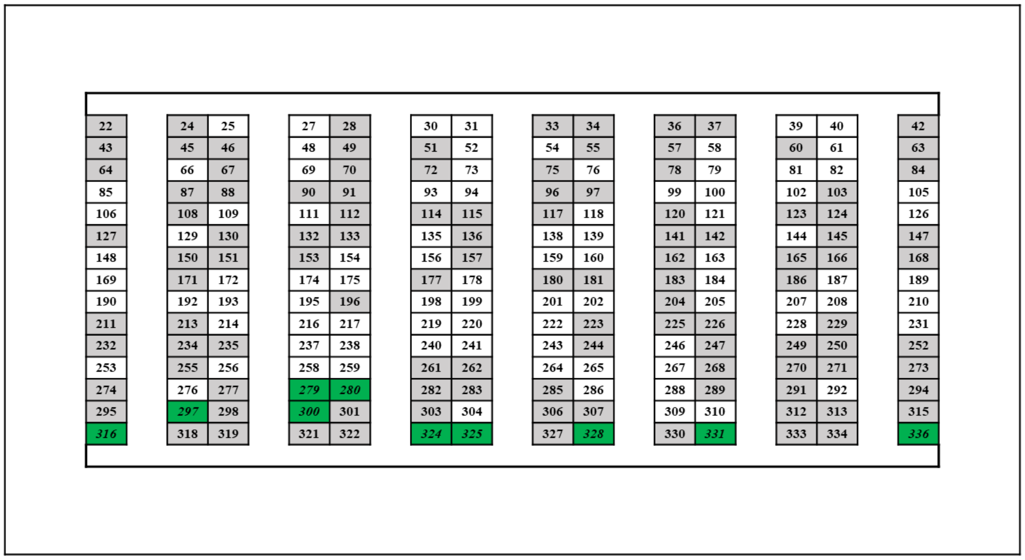


Figure 4.31 Case 13, fixed assignment results visualized in the warehouse

Table 4.19 Case 14, fixed assignment results

label	distance
316	2
297	3
276	4
256	8
300	6
279	7
280	10
259	11
324	8
325	11
304	12
264	14
328	14
286	16
309	15
331	17
310	18
228	22
292	22
336	20

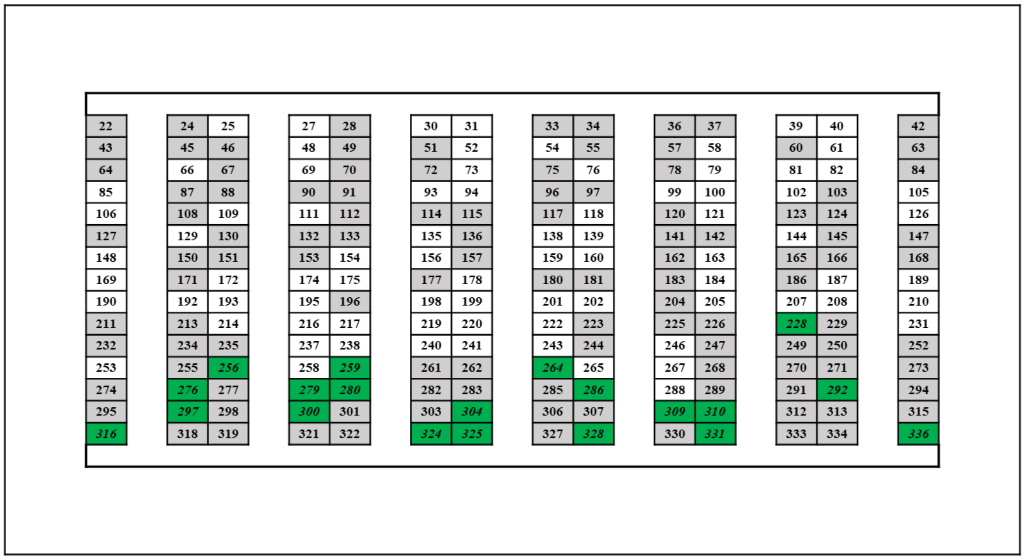


Figure 4.32 Case 14, fixed assignment results visualized in the warehouse

Table 4.20 Case 15, fixed assignment results

label	distance
316	2
253	5
190	8
297	3
276	4
256	8
300	6
279	7
258	8
237	9
280	10
259	11
238	12
324	8
325	11
304	12
241	15
264	14
328	14
286	16
309	15
288	16
331	17
310	18
205	23
228	22
292	22
208	26
336	20
231	25

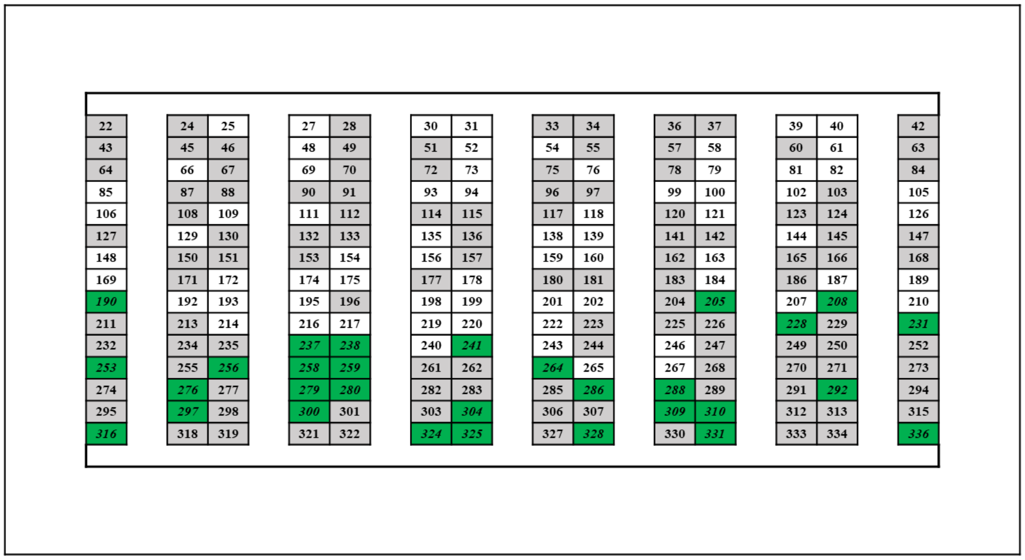


Figure 4.33 Case 15, fixed assignment results visualized in the warehouse

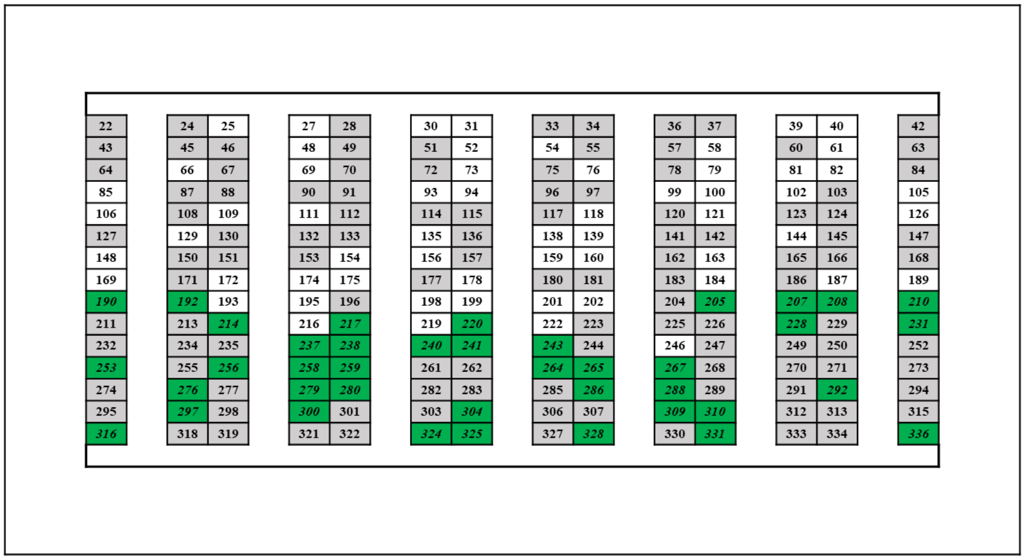


Figure 4.34 Case 16, fixed assignment results visualized in the warehouse

Table 4.21 Case 16, fixed assignment results

label	distance
316	2
253	5
190	8
297	3
276	4
192	8
256	8
214	10
300	6
279	7
258	8
237	9
280	10
259	11
238	12
217	13
324	8
240	12
325	11
304	12
241	15
220	16
264	14
243	15
328	14
286	16
265	17
309	15
288	16
267	17
331	17
310	18
205	23
228	22
207	23
292	22
208	26
336	20
231	25
210	26

5 Results and analysis

We summarized the results of all 16 cases under both random assignment and fixed assignment to compare total traveling distance of the forklift fleet, as Table 5.1 shows. The total distance is commonly used to evaluate the performance of vehicle routing algorithms.

From the results we learn that the number of SKU types in the tasks does not influence the random assignment results. The distance under random assignment method is only related with task size, that is the number of items to be handled in a task.

Table 5.1 Results of all cases, comparing the total distance

task number	number of SKU types	task size	total distance by random assignment	total distance by fixed assignment
1	2	10	59	94
2	2	20	158	216
3	2	30	278	355
4	2	40	422	513
5	3	10	59	93
6	3	20	158	220
7	3	30	278	375
8	3	40	422	524
9	4	10	59	100
10	4	20	158	246
11	4	30	278	386
12	4	40	422	559
13	7	10	59	98
14	7	20	158	240
15	7	30	278	387
16	7	40	422	544

To study the performance of two different storage location assignment algorithms, we plot two curves for the total distance under two assignment strategies, as Figure 5.1 shows.

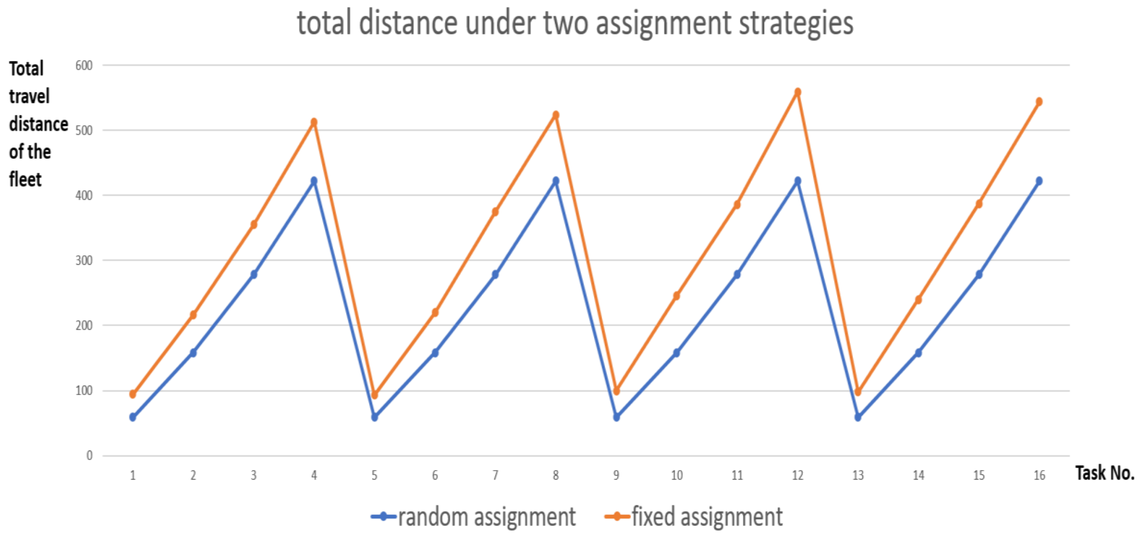


Figure 5.1 Total traveling distance under two assignment strategies, 16 cases

We can see from the figure the task size also influences the gap between two strategies. The bigger the task size, the more traveling distance can be saved from random assignment algorithms, compared with fixed assignment algorithms.

6 Conclusion and discussion

The new algorithm has both economic significance and technical innovativeness.

The economic value is obvious: the new algorithm considers more information than previous solutions to make better management decisions. It provides an automatic comparison between random and fixed assignment methods to select optimal solution from a larger candidate group.

The technical value comes from compatibility, and scalability.

Compatibility means the algorithm will various scenarios and warehouse layouts. The scenarios can be at different automation level with requests of different complexity. Compatibility ensures that the fleet management system can adapt to external environments.

Scalability means the algorithm allows new vehicles to join the fleet. Even heterogeneous vehicles from external systems can join and leave the fleet flexibly under certain agreements. Scalability ensures that the system can manage fleets extending from small-scale to large-scale.

In the future research, there are several potential directions to extend the algorithm.

First is to extend the operation scenarios. The layout and traffic of warehouses are similar to scenarios such as ports, city blocks and parking facilities. The system model and mathematic model can be modified to adapt to these scenarios.

Second is to include more decision-making in the algorithm to cover more operations, for example, order batching, order picker routes, mixed storage and order picking tasks, etc.

Acknowledgement

First, completion of this thesis relies heavily on the support from Professor Hidekazu Nishimura. As my supervisor, Professor Nishimura has been guiding me in studies, research and life without great kindness and patience. It is his encouragement that makes me start my study in Graduate School of System Design and Management. During the two-year master's program in SDM, I have learnt a lot from the discussion with Professor Nishimura.

I also owe a huge debt of gratitude to my parents. All through my stay in Japan, they have supported me both financially and emotionally. Thanks to them for giving me the courage to overcome any difficulty or obstacle.

Reference

[1] Gu, Jinxiang, Marc Goetschalckx, and Leon F. McGinnis. "Research on warehouse operation: A comprehensive review." *European journal of operational research* 177.1 (2007): 1-21.

[2] Speranza, M. Grazia. "Trends in transportation and logistics." *European Journal of Operational Research* 264.3 (2018): 830-836.

[3] Walden, David D., et al. *Systems engineering handbook: A guide for system life cycle processes and activities*. John Wiley & Sons, 2015: 222.

[4] Simao, Hugo P., Jeff Day, Abraham P. George, Ted Gifford, John Nienow, and Warren B. Powell. "An approximate dynamic programming algorithm for large-scale fleet management: A case application." Transportation Science 43, no. 2 (2009): 178-197.

[5] Nair, Rahul, and Elise Miller-Hooks. "Fleet management for vehicle sharing operations." Transportation Science 45, no. 4 (2011): 524-540.

[6] Bsaybes, Sahar, Alain Quilliot, and Annegret K. Wagler. "Fleet management for autonomous vehicles using flows in time-expanded networks." TOP 27, no. 2 (2019): 288-311.

[7] Salazar-Cabrera, Ricardo, Álvaro Pachón De La Cruz, and Juan Manuel Madrid Molina. "Fleet Management and Control System from Intelligent Transportation Systems perspective." In 2019 2nd Latin American Conference on Intelligent Transportation Systems (ITS LATAM), pp. 1-7. IEEE, 2019.

[8] Rudyk, Tomasz, Emilian Szczepański, and Marianna Jacyna. "Safety factor in the sustainable fleet management model." Archives of Transport 49 (2019).

[9] Ruan, Yichen, Jinhang Zuo, Jakob Cassiman, and Xiaoyi Duan. "DYNAMIC FLEET MANAGEMENT VIA DEEP REIN-FORCEMENT LEARNING."

[10] Lin, Kaixiang, Renyu Zhao, Zhe Xu, and Jiayu Zhou. "Efficient large-scale fleet management via multi-agent deep reinforcement learning." In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1774-1783. ACM, 2018.

[11] De Koster, M. B. M., Edo S. Van der Poort, and MJIJoPR Wolters. "Efficient orderbatching methods in warehouses." *International Journal of Production Research* 37.7 (1999): 1479-1504.

Appendix

MATLAB code

```
clc
%create the grid as 21 by 17
x = 1:21;
y = 1:17;
[X,Y] = meshgrid(x,y);%the grid
grid_size = size(X);
%check grid size
%F = zeros(grid_size)

%label the grid
label_grid = (Y-1)*21+X;
label_grid(17,1) = 0;
for i = 2:21
   label_grid(17, i) = label_grid(17, i)-1;
end
%check label
%surf(X,Y,label_grid)

%create the graph composed of storage location and aisle
s = [s1; s2];
t = [t1; t2];
G = graph(s,t);
plot(G)
%plot(G,'Layout','layered')
h = plot(G,'Layout','force');
```

```
%find the shortest path distances from depot (node 337) to all other nodes in the graph
d = distances(G,337);

%expel aisle nodes from candidate storage locations
d_storage = d;
for j = 1:21
   d_storage(j) = 999;%let aisle nodes distance be big enough to expel
end
for k = 337:356
      d_storage(k) = 999;
end
for l = 23:3:335
      d_storage(l) = 999;
end
%check modified distance vector (correct answer is 19)
%check_node_225 = d_storage(225)

%create submatrix of label_grid for storage locations
label_storage = zeros(210,1);
m = 1;
for n = 1:356
   if d_storage(n) ~= 999
      label_storage(m) = n;%label_storage stores the label of all storage locations
      m = m+1;
   end
end
storage_size = m-1;

%create submatrix of label_storage for several departments
v = 1;
```

```
% create random unoccupied storage location
unoccupied_size=100;%the value of unoccupied_size should be from 0 to storage_size
label_random = randperm(storage_size,unoccupied_size);
label_unoccupied = zeros(unoccupied_size,1);
q = 1;
for o = 1:unoccupied_size
   p = label_random(o);
   label_unoccupied(q) = label_storage(p);%label_unoccupied stores the label of all unoccupied
storage locations
   q = q+1;
end

%give the distance from the depot to all unoccupied storage location
tt = 1;
tt1 = 1;
tt2 = 1;
tt3 = 1;
tt4 = 1;
tt5 = 1;
tt6 = 1;
tt7 = 1;
tt8 = 1;
tt9 = 1;
tt10 = 1;
tt11 = 1;
tt12 = 1;
tt13 = 1;
tt14 = 1;
tt15 = 1;
d_unoccupied = zeros(unoccupied_size,1);
for r = 1:unoccupied_size
   u = label_unoccupied(r);
```

```
d_unoccupied(tt) = d_storage(u);%d_unoccupied stores the distance

if rem(u,21) == 1
   label_unoccupied_col1(tt1) = u;
   d_unoccupied_col1(tt1) = d_storage(u);
   tt1 = tt1+1;

elseif rem(u,21) == 3
   label_unoccupied_col2(tt2) = u;
   d_unoccupied_col2(tt2) = d_storage(u);
   tt2 = tt2+1;

elseif rem(u,21) == 4
   label_unoccupied_col3(tt3) = u;
   d_unoccupied_col3(tt3) = d_storage(u);
   tt3 = tt3+1;
elseif rem(u,21) == 6
   label_unoccupied_col4(tt4) = u;
   d_unoccupied_col4(tt4) = d_storage(u);
   tt4 = tt4+1;
elseif rem(u,21) == 7
   label_unoccupied_col5(tt5) = u;
   d_unoccupied_col5(tt5) = d_storage(u);
   tt5 = tt5+1;
elseif rem(u,21) == 9
   label_unoccupied_col6(tt6) = u;
   d_unoccupied_col6(tt6) = d_storage(u);
   tt6 = tt6+1;
elseif rem(u,21) == 10
   label_unoccupied_col7(tt7) = u;
   d_unoccupied_col7(tt7) = d_storage(u);
```

```
    tt7 = tt7+1;
elseif rem(u,21) == 12
    label_unoccupied_col8(tt8) = u;
    d_unoccupied_col8(tt8) = d_storage(u);
    tt8 = tt8+1;
elseif rem(u,21) == 13
    label_unoccupied_col9(tt9) = u;
    d_unoccupied_col9(tt9) = d_storage(u);
    tt9 = tt9+1;
elseif rem(u,21) == 15
    label_unoccupied_col10(tt10) = u;
    d_unoccupied_col10(tt10) = d_storage(u);
    tt10 = tt10+1;
elseif rem(u,21) == 16
    label_unoccupied_col11(tt11) = u;
    d_unoccupied_col11(tt11) = d_storage(u);
    tt11 = tt11+1;
elseif rem(u,21) == 18
    label_unoccupied_col12(tt12) = u;
    d_unoccupied_col12(tt12) = d_storage(u);
    tt12 = tt12+1;
elseif rem(u,21) == 19
    label_unoccupied_col13(tt13) = u;
    d_unoccupied_col13(tt13) = d_storage(u);
    tt13 = tt13+1;
elseif rem(u,21) == 0
    label_unoccupied_col14(tt14) = u;
    d_unoccupied_col14(tt14) = d_storage(u);
    tt14 = tt14+1;
    end
```

```
    tt = tt+1;
end

%sort the distance of unoccupied locations
%combine label_unoccupied and d_unoccupied
mat_unoccupied(1,:) = label_unoccupied;
mat_unoccupied(2,:) = d_unoccupied;
%[d_sort,seq] = sort(mat_unoccupied);
mat_unoccupied_col1(:,1) = label_unoccupied_col1;
mat_unoccupied_col1(:,2) = d_unoccupied_col1;
mat_unoccupied_col2(:,1) = label_unoccupied_col2;
mat_unoccupied_col2(:,2) = d_unoccupied_col2;
mat_unoccupied_col3(:,1) = label_unoccupied_col3;
mat_unoccupied_col3(:,2) = d_unoccupied_col3;
mat_unoccupied_col4(:,1) = label_unoccupied_col4;
mat_unoccupied_col4(:,2) = d_unoccupied_col4;
mat_unoccupied_col5(:,1) = label_unoccupied_col5;
mat_unoccupied_col5(:,2) = d_unoccupied_col5;
mat_unoccupied_col6(:,1) = label_unoccupied_col6;
mat_unoccupied_col6(:,2) = d_unoccupied_col6;
mat_unoccupied_col7(:,1) = label_unoccupied_col7;
mat_unoccupied_col7(:,2) = d_unoccupied_col7;
mat_unoccupied_col8(:,1) = label_unoccupied_col8;
mat_unoccupied_col8(:,2) = d_unoccupied_col8;
mat_unoccupied_col9(:,1) = label_unoccupied_col9;
mat_unoccupied_col9(:,2) = d_unoccupied_col9;
mat_unoccupied_col10(:,1) = label_unoccupied_col10;
mat_unoccupied_col10(:,2) = d_unoccupied_col10;
mat_unoccupied_col11(:,1) = label_unoccupied_col11;
mat_unoccupied_col11(:,2) = d_unoccupied_col11;
```

```
mat_unoccupied_col12(:,1) = label_unoccupied_col12;
mat_unoccupied_col12(:,2) = d_unoccupied_col12;
mat_unoccupied_col13(:,1) = label_unoccupied_col13;
mat_unoccupied_col13(:,2) = d_unoccupied_col13;
mat_unoccupied_col14(:,1) = label_unoccupied_col14;
mat_unoccupied_col14(:,2) = d_unoccupied_col14;
```