# Determining the Shortest Path for Inner Warehouse Transportation

**Noraimi Azlin Mohd Nordin[1], S.Sarifah Radiah Shariff[2], Siti Suzlin Supadi[3], Yosi Pahala[4]**

[1] School of Mathematical Sciences, College of Computing, Informatic and Media, Universiti Teknologi MARA, Cawangan Negeri Sembilan, Kampus Seremban, 70300 Seremban. Malaysia, [2] Malaysia Institute of Transport (MITRANS), Universiti Teknologi MARA, Shah Alam, 40450 Selangor, Malaysia, [3] Institute of Mathematical Sciences, Faculty of Science, University of Malaya, 50603 Kuala Lumpur, Malaysia, [4] Institut Transportasi dan Logistik Trisakti, Jakarta. Indonesia

noraimi@uitm.edu.my, shari990@uitm.edu.my, suzlin@um.edu.my, yosipahala@itltrisakti.ac.id
Tel # 017-2015892:

## Abstract

Always meeting customer needs is the primary goal of logistic service companies specializing in warehousing. However, warehouses are often large, and the movement of goods in the warehouse is problematic because it takes a long time and slows the order-picking process. This problem can be solved if storage assignment and the optimal warehouse movement route are appropriately planned. This study compares the use of Dynamic Programming models for two data in this warehouse in determining the shortest path for the Order Picker in completing and fulfilling customer orders.

Keywords: Order picking, dynamic programming, inner warehouse transportation.

DOI: https://doi.org/10.21834/jabs.v8i25.425

## 1.0 Introduction

Companies producing and manufacturing products look forward to minimizing operation costs, especially in stock-keeping. However, at the same time, they are looking for a value-added process that can maximize the main objective of the companies, which is to satisfy customers (Ngah et al., 2021; Rahmat et al., 2022) and with effective resource utilization and deliver the right product, at the right place and at the right time in good condition (Ma'mun et al., 2018). According to Rushton et al. (2022), the warehouse provides temporary storage, protection of goods, fulfillment of individual customer orders, and packaging of goods.

Despite the normal process in the warehouse, many incidents may increase the operation cost. For example, the incidents that result from inefficient storing assignment, poor architectural design and general layout, long response time for the order processing, long travel distances in the warehouse, and complicated routing of pickers or automated guided vehicles (AGV) (Vrysagotis & Kontis, 2011). These problems can affect the order-picking process and indirectly affect the customer satisfaction due to late delivery to customers (Zakaria & Zurcher, 2023). Hence, the minimization of travel distances in the warehouse can be solved by looking at the shortest path for the points in the warehouse where ordered products are kept (Mirzaie et al., 2021; Shariff et al., 2022). The shortest path can be used to calculate a way to traverse for loading and unloading items and picking orders. The shortest path is defined as the minimum length of distance from starting point to the ending point. It also can be defined for graphs whether undirected, directed, or mixed (Lewis, 2020).

Various types of algorithms are very useful in calculating the shortest path. One of them is the Dynamic programming Method discovered by Bellman (Ferone et al., 2021; Senaras et al., 2021). This method solves some particular types of problems in polynomial time. Furthermore, this method can be applied to other types of vehicles and other types of storage facilities as well. In that way, business processes and productivity can be improved, as well as reduced operational costs. Many studies (Labus & Gajsek, 2018; Anđelković & Radosavljević, 2018) stated that 55% of operating costs in the warehouse can be connected to order picking, and this situation makes inner transportation as an excellent transport to optimize the warehouse. This means that the operating cost is a crucial problem in operating cost since the percentage is more than half. Some warehouses have a limited workforce and insufficiently qualified labor, which can lead to dragging them to finish the work on time (Mc Kinnon et al., 2017; de Leeuw & Wiers, 2010). As a result, workers must do extra work and hours to accomplish all the orders.

This research uses the Dynamic programming method to determine the minimum distance traveling in the warehouse.

## 2.0 Literature Review

Dynamic programming is a combination of mathematical optimization and programming. For the complex problem, this method can be used to simplify the problem by breaking it down into a collection of simpler sub-problems, then solving each sub-problems just once and storing their solutions - ideally, using a memory-based data structure (Gutenschwager et al., 2012). A dynamic programming algorithm will examine the sub-problems and combine the solutions to ensure the ideal solution for the problem. However, during the process of solving sub-problems, it leads to lots of repeated computations which rapidly reduce the operating efficiency of the programs. Generally, it shows that dynamic programming is often used for optimization. Nevertheless, solving a large number of repeated subproblems will be hindered. Furthermore, this method is irrelevant when the three properties, which are optimal sub-structure, non-aftereffect and overlapping sub-problem overlap, are not satisfied.

According to (Lantoine & Russel, 2012), dynamic programming is a well-known technique that can help to tackle a wide range of problems. The objective of this method is to maximize and minimize a function. Therefore, this method is widely used in many fields, such as geotechnical engineering and mathematics. Besides, this method is a systematic procedure to find the optimal decisions. In conclusion, the dynamic programming algorithm is suitable for optimization and can significantly improve efficiency from the exponential to the polynomial level by reducing the time complexity.

## 2.1 Application of Dynamic Programming Method

The dynamic programming method has been widely used in many applications. For example, according to Roodbergen and Koster (2001), dynamic programming can be applied to routing warehouses with multiple cross aisles. This paper considered routing and layout issues for parallel aisle warehouses. In order to pick up all orders from storage, the order pickers can walk or drive along the aisles. However, due to the amount of order handling, time-consuming generally should be considered.

Furthermore, the productivity of the order-picking process can be improved by reducing time handling. In place with the manual picking process, Lee et al., (2015) claimed that travel time often forms the most significant component of total picking time. Masae et al., (2020) developed an efficient algorithm for warehouses with two cross aisles to determine the shortest order-picking routes. Some researchers studied on performance comparisons between optimal routing and heuristics for warehouses with more than two cross aisles and a comparison between the heuristic (Liu & Limere, 2012; Chen et al., 2018).

In addition to that, according to Singhal and Pandey (2016), dynamic programming algorithm was also used to solve the traveling salesman problem. Similar research was done by (Bouman et al., 2016). In the traveling salesman problem, a map of cities is given to the salesman, and they need to visit all the cities. Therefore, they must travel the shortest distance to minimize time handling and wasted costs. However, there are two ways of solving the problem: top-down, called memorization, and bottom-up, called dynamic programming. Dynamic programming was used to solve the problem if the subproblems were cleared before the main problems were fixed.

Meanwhile, memorizing the solution to the problem started with solving it and breaking it downward. If some overlapping of subproblems occur, then it is a big hint to use dynamic programming. For example, the study proposed to use dynamic programming to find an optimal solution for a traveling salesman problem where all parameters are in matrix form.

Next, a dynamic programming method was used in an adaptive traffic signal (Liu et al, 2019). This paper discussed proper timing is required to manage different demands. The traffic signal is one of the causes of the increase in total operation costs. Such plans required manual maintenance and updating or otherwise, the performance declined by about 3% per year. Without human involvement, the controller must operate signals unconcerned of the cycle and stage constraints to optimize the cost over time. Dynamic Programming managed to reduce about 56% of vehicle delays.

Lastly, dynamic programming algorithms also solved the selection of waste disposal ports in cruise shipping. According to Wang et al., (2018), the cruise provided many activities for the passengers. A large cruise ship could have afforded over 6000 passengers and 1500 crew members, leading to many waste ingredients from the cruise ship. There were a few types of waste produced by the ship, such as sewage, graywater, oily bilge water, solid waste, and hazardous waste. The objective was to determine at which port the cruise company needed to dispose of the waste generated by the ship at minimum cost. The study showed that the DP method was very efficient since it only takes less than 0.01s seconds to compute the random static or dynamic problem with 50 ports. DP method performed more perfectly than the greedy method and may solve a significant problem in only a few minutes.

## 3.0 Methodology

Figure 1 describes the dynamic programming method's procedure to solve the problem. The flow of the methodology is described by giving example as follows:'

Phase 1: An introduction of the dynamic programming method, the used-on method, and routes in solving the problem will be explained, and a detailed explanation of the step-by-step approach to solving dynamic programming problems.

Phase 2: A detailed explanation on warehouse layout by Dobrilovic et al. (2012) as an example, and a few assumptions are created to solve the Layout.

Phase 3: An example of how to reconstruct the real solution from the warehouse layout from Figure 3.31. Using the dynamic programming method will be explained in detail, and all the calculations to solve the problem as numerical examples are shown.

Phase 4: The software will be well explained to measure the dynamic programming method's performance.

### 3.1 Phase 1: Brief Description of Dynamic Programming Method

The dynamic Programming method is one of the powerful methods which produce classic algorithms for the diversity of combinations of optimization problems. Dynamic programming methods may solve almost all problems that consist of the overlapping subproblems property (Bhowmik, 2010). This study focuses on the shortest path that has to be passed through by a forklift in a warehouse.

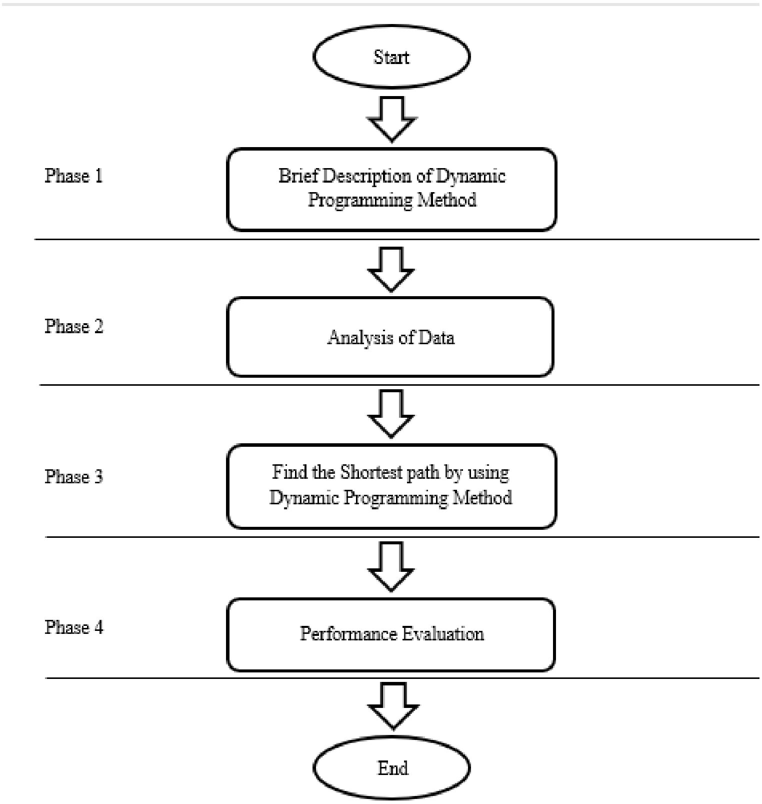

Start
Phase 1
Brief Description of Dynamic Programming Method
Phase 2
Analysis of Data
Phase 3
Find the Shortest path by using Dynamic Programming Method
Phase 4
Performance Evaluation
End

Figure 1: Steps in Applying Dynamic Programming Method

The variables of the Dynamic Programming Method defined in this study were referred on the research paper by (Roodbergen & Koster, 2001) and (Nordin et al., 2018). The variables were specified by the situation of this study which used a warehouse layout as the data.

The variable $k$ denotes the number of blocks in the warehouse layout, while variable $n$ represents the total number of aisles or routes considered to be passed. Then there is some physical position denoted as $a_{ij}$ and $b_{ij}$. $a_{ij}$ stands for the back end of subaisle *j* in block *i* for each block *i*, *i*=1,…..,*k* and for each subaisle *j*, *j*=1,….,*n* and $b_{ij}$ is stands for the front end of subaisle *j* in block *i* for each block *i*, *i*=1,…..,*k* and for each subaisle *j*, *j*=1,….,*n*.

Variable $d$ stands for depot. Depot stands for the starting and end points of the customers' orders that need to be picked up.

Dynamic programming method will be used for this study to manage the order picker's route through a single block *i* (*i*=1,…,*k*). This study's origin and the initial point is at the left-most subaisle, consisting of items (*l*); the terminal and end point are at the right-most subaisle, consisting of items (*r*). However, for this study, the starting point and the endpoint of the forklift are at the same point. This means that the forklift will travel from the starting nodes to collect all of the orders and return back to the first point after completing the order picking. $\text{Therefore}, L_j$ is defined to be the partial route visiting all of the pick points in subaisles *l* through *j* is defined as $L_j$.

Two parts of partial routes are differentiated as follows:

1. $L_j^a$ is a partial route that ends at the back of subaisle $j$, where $j$=1,2,…,n. '$a$' in terms of $L_j^a$ represents that it is part of the back of the subaisle in the warehouse layout.
2. $L_j^b$ is a partial route that ends at the front of subaisle $j$. where $j$=1,2,…,n. '$b$' in terms of $L_j^b$ represents the part of the front of the subaisle in the warehouse layout.

Then, there are two ways to go from subaisle $j-1$ to subaisle $j$, as shown in Figure 2, which is:

1. $_{Ta}$ goes along the back of the block since '$a$' in the term $t_a$ represents that it is part of the back of the subaisle.
2. $t_b$, which goes along the front of the block since '$b$' in the term $t_b$ represents that it is part of the back of the subaisle.

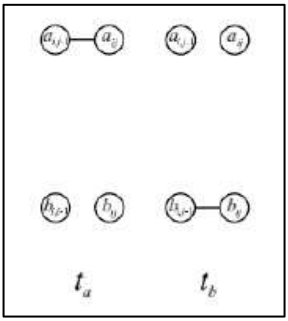


Figure 2: Transition from Side to Side Used in Routing Heuristic. *( Roodbergen & Koster, 2001)*

Besides, from here, the method distinguishes four ways to pick items in subaisle *j*, which are $t_1, t_2, t_3$, and $t_4$. The method of picking items is explained by providing Figure 3 for better understanding. The four ways are:

1. $t_1$ refers that the forklift traverse and passing the entire subaisle. The distance traveled is calculated from the front to the back or vice versa.
2. $t_2$ refers to not entering the subaisle at all. There are no orders to be picked up on the aisle.
3. $t_3$ refers to entering and leaving the subaisle from the front of the block. In this way, the distance traveled calculated from the nodes at the front part of the aisle to the point before the back aisle denotes as $a_{ij}$.
4. $t_4$ refers to entering and leaving the subaisle from the back of the block. In this way, the distance traveled is calculated from the nodes at the back part of the aisle to the point before the front aisle, denoted as $b_{ij}$.

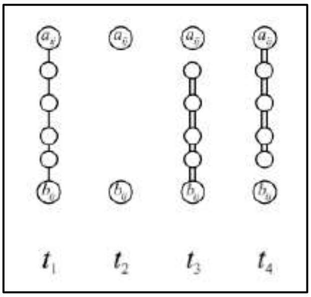


Figure 3: Transition from Back to Front Used in Routing Heuristics.
*(Roodbergen & Koster, 2001)*

If the subaisle does not contain any goods to pick, transition $t_2$ is allowed. The summation of the partial route is $L_j + t_w$ where *w* is a series of transition in Figures 2 and 3 where *w=(1,2,3,4,a,b)* and c(.) denotes the travel time associated with $L_{j+}t_w$. For example, the time needed to walk the partial route $L_j^b$ plus the time needed to walk transition $t_1$ is denoted as $c(L_j^b + t_1)$. Remember that the transition contains information only on how to enter and leave the subaisles. To form the complete order picking route shown in Figure 3.3, which is the warehouse layout, it means $L_r^b$ will be used. The following steps describe the Dynamic Programming Method for one block (Roodbergen & Koster, 2001)

**Step 1**

Let block *i* be the block under consideration, the subaisle from the left will be *l,* and the subaisle from the right will be *r*. Start with the two partial routes below if block *i* is the farthest block from the depot that contains items:

$L_l^a$, which starts at node $b_{il}$ , ends at node $a_{il}$ and consists of transition $t_1$

$L_l^b$ which starts and ends at node $b_{il}$ and consists of transition $t_3$

Otherwise, start with two partial routes given:

$L_l^a$ which starts and ends at node $a_{il}$ and consists of transition $t_4$

$L_l^b$, which starts at node $a_{il}$, ends at node $b_{il}$ and consist of transition $t_1$

**Step 2**
This method determines $L_j^a$ and $L_j^b$ for each consecutive subaisle $j(l < j < r)$ as follows: If there is item(s} in subaisle j, then:

$$L_j^a = \begin{cases} L_{j-1}^a + t_a + t_4 & if\ c\left(L_{j-1}^a + t_a + t_4\right) < c(L_{j-1}^a + t_b + t_1) \\ L_{j-1}^b + t_b + t_1 & otherwise \end{cases} \tag{1}$$

$$L_j^b = \begin{cases} L_{j-1}^b + t_b + t_3 & if\ c\left(L_{j-1}^b + t_b + t_3\right) < c(L_{j-1}^a + t_a + t_1) \\ L_{j-1}^a + t_a + t_1 & otherwise \end{cases} \tag{2}$$

The equations (1) and (2) stated before shows the calculation formula of the Dynamic Programming Method. Based on equation (1), states that in order to get the total distance travel of $L_j^a$, calculate the sum of the back aisle distance based on the formula $L_{j-1}^a + t_a + t_4$ and calculate the sum of front aisle distance according to the formula $L_{j-1}^b + t_b + t_1$. Then, the minimum total distance will be chosen to calculate the next aisle starting $L_1^a$. If there is no item(s) contained in subaisle *j*, then:

$$L_j^a = L_{j-1}^a + t_a \tag{3}$$
$$L_j^b = L_{j-1}^b + t_b \tag{4}$$

**Step 3**
Subaisle *r* is on the right of the depot *r.* For the last subaisle of the block, this method determines that

$$\begin{matrix} L_{r-i}^b + t_b + t_3 & if\ \ c\left(L_{r-i}^b + t_b + t_3\right) < c(L_{r-i}^a + t_a + t_1) \\ L_{r-i}^a + t_a + t_1 & otherwise \end{matrix} \tag{5}$$

One of the essential logistic networks is warehouses (Roodbergen & Koster, 2001). The type of warehouse used for this study is a distribution center. To complete the order-picking route, thus the last partial route will be used, which is $L_r^b$. since all the items in the block have been picked. Therefore $L_r^a$ is no longer needed. However, to complete the picking route, the forklift must go to the front of the block.

**3.2 Phase 2: Data Analyzing**
Data analysis in this research is about analyzing and understanding the data used. This data is adapted from the paper studied by Dobrilovic et al. (2012). This data is chosen because it is suitable to apply Dynamic Programming Method in this project. For this research, this warehouse layout will be contained one forklift as an inner transportation to help order pickers collect the items demanded by customers.

Figure 4 of the warehouse layout shows the forklift is located at the depot of the warehouse and also labeled as node *1*. The meaning of depot is a place representing the start and end points during the collection session while the node is the specific place containing the item that will be picked up by the order picker based on the item demanded. Using the Dynamic Programming Method, the shortest path will be calculated by considering every subaisles with nodes that must visit at least once to collect the items in the warehouse layout. Then, the subaisles are entirely passed through or the order picker enters and leaves the sub aisles from the same side.

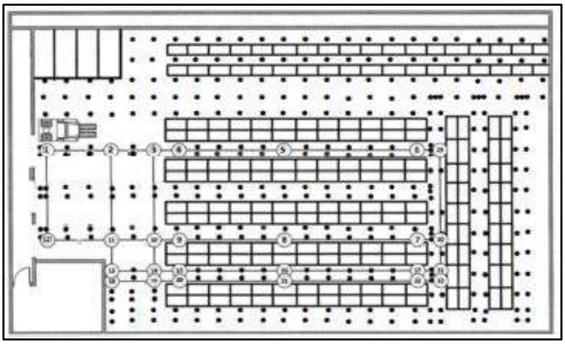


Figure 4: Warehouse Layout with One Forklift

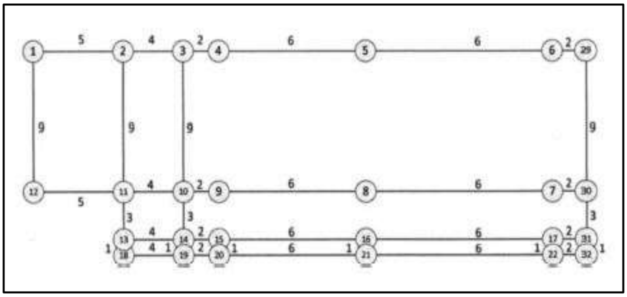


Figure 5: Warehouse Graph with One Forklift

At first, the order pickers must go through to the nearest node with the depot. Every two neighboring nodes have their values at the edge that will present the distance estimations of forklifts with horizontal movements (Dobrilovic et al., 2012). That distance will be used for the calculation to minimize the cost produced during operational hours. For the calculation, it started from the forklift and to the farthest node and went back to the starting point at node *1*. The distance consumption for picking up the items in the study area depends on the distances that the forklift through between two nodes.

Figure 5 shows all possible nodes represented in a graph with path costs (weights) and shows all possible aisles in a warehouse that will be used to pick up the items. For example, if the forklift wants to pick up the item in the storage at node *18*, it has to move along the path 1-12-11-13-18, or some other path because there is more than one route with the exact cost in this study area. Therefore, with the implemented Dynamic Programming Method to find the shortest path, the minimum distance defines at the end of the calculation. However, to calculate this warehouse layout, a few assumptions are made to find the best aisle for the order picker to complete the task given.

The assumptions are as follow:

- Starting point and ending point at the same nodes
- The order pickers are well known for their routes and warehouse picking area.
- All the items are collected based on the demand made by customers.
- Collection sessions happen during normal shift working hours from 8.00 a.m. to 5.00 p.m.
- The items also are well-organized and follow the standard operating procedure (SOP).
- There are no accidents involved during order picking.
- The condition of the forklift that is used is well maintained.
- The forklift cannot carry items at one time.

**3.3 Phase 3: Find the Shortest Path by using Dynamic Programming Method**

Figure 6 shows an example of the types of routes in the warehouse. Each number represents a distance in meters (m). Fourteen items need to be collected. In this situation, $l = 1$ and $r = 4$, and $b_{i1}$ is the starting point (depot). The route starts at the left-most subaisles $(l)$ and represents as the front of the block, which is $b_{i1}$ and $b_{i2}$. While $a_{i1}, a_{i2}, a_{i3}$, and $a_{i4}$ are the nodes from the back of the blocks. Nodes *1* to *14* denote the location of the forklift that needs to stop to take an item.

Firstly, since the block under consideration is block$ed\ i$ (the block closest to the depot and consisting of items), we started with two partial routes, $L_1^a$, and $L_1^b$. $L_1^a$ starts at node $b_{i1}$, and ends at node $a_{i1}$. This process contains transition $t_1$, with associated travel distance c($t_1$)=11. On the other hand, $L_1^b$ starts and ends at node $b_{i1}$ and consists of transition $t_3$, with associated c ($t_3$) = 18.

Next, there are two possibilities of creating $L_2^a$, namely as $L_1^a + t_a + t_4$ (travel distance 11+9+6(2) = 32) or as $L_1^b + t_b + t_1$ (travel distance 18+9+(5+4)(2) = 45). The smallest value between these two will be chosen as the next possible shortest path, which is $L_1^a + t_a + t_4$ . In this situation, the current c ($t_1$)=32. Similarly, to find $L_2^b$, there are two possibilities namely as $L_1^b + t_b + t_3$ (travel distance 18+9+(5+4)(2) = 45 or as $L_1^a + t_a + t_1$ (travel distance 11+9+2+4+5 = 31). Clearly, $L_1^a + t_a + t_1$ will be chosen as the next possible shortest path. Thus, from the findings, c ($t_3$) = 31 is chosen as the next shortest distance. The same procedure will be repeated to find the next possible shortest path, which is $L_3^a$ and $L_3^b$. The calculation can be seen clearer in Figures 7a and b.

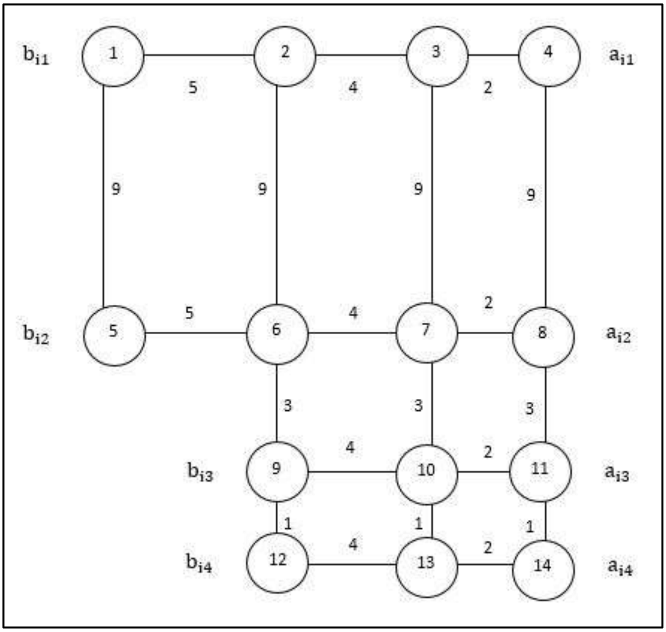


Figure 6: Example Layout

$L_3^a$

$L_2^a + t_a + t_4$

= 24+3+2(2)

= 31

$L_2^b + t_b + t_1$

= 31+5+3+4+2

=45

Figure 7a: Calculation for $\boldsymbol{L_3^a}$

$L_3^b$

$L_2^b + t_b + t_3$

= 31+5+3+4(2)

= 47

$L_2^a + t_a + t_1$

= 24+3+4+2

= 33

Figure 7b: Calculation for $\boldsymbol{L_3^b}$

From the situation in Figure 7 a and b, the current c $(t_1)$ and c $(t_3)$ are 31 and 33 respectively. For the last subaisle r, the next shortest path, $L_r$ will be compared from the previous shortest value of $L_3^a$ and $L_3^b$. In this step, no need to find $L_r^a$ since all the items have been picked and continue to the ending point, which is point $b_{i4}$. The value of *r*=4 represents the right-most aisle in the Layout. $L_4^b$ can be calculated as:

$$L_4^b \left[ \begin{array}{l} L_3^b + t_a + t_3 \\ = 33+1+2(4) \\ = 42 \\ L_3^a + t_b + t_1 \\ = 31+1+4+2 \\ =38 \end{array} \right.$$

Figure 7c: Calculation for $\boldsymbol{L_4^b}$

Based on the value obtained in Figure 7c, it is clearly shown $L_4^b$ can be calculated via $L_3^a + t_b + t_1$. The total distance from $b_{i1}$ to $b_{i4}$ is 42m. This completes the partial route created by the dynamic programming algorithm. The same procedure will be used to calculate the shortest path in the actual data.

## 3.4 Phase 4: Performance Evaluation

This research uses Dynamic Programming Method to calculate the shortest path and distance travel. In Phase 3, few steps are shown on how to calculate using this method, but it may take a long period to complete the calculation manually. Many software tools are available to settle this issue, such as Microsoft Excel, MATLAB, C++ programming, and others. The function of all this software is to help determine the shortest path. The software can reduce the time needed to complete the calculation and help minimize the calculation mistake during calculation manually.

Here, the tools in Microsoft Excel had been adapted to use in order to determine the shortest distance traveled in the warehouse since it is suitable for this research situation which is starting point and ending point at the same node. However, before using the actual data, this paper will try this Microsoft Excel in preliminary data that has been taken quarter than the actual data. With the friendly-user interface, this Microsoft Excel was easy to understand. It was built by using the developer add-in in Figure 8. However, this template had a password already set to prevent other people from editing Microsoft Excel.

This Microsoft Excel spreadsheet contains four steps for users to follow to solve the warehouse problem. Firstly, the users must know the number of nodes containing the warehouse to produce the matrix box according to the number of nodes. Then, after filling in the matrix box with the distance matrix of the data in the warehouse, the user can choose which path they want to start, or the user can use the default setting to execute all the paths that can pass through. As a result, this Microsoft Excel will show the shortest path in the warehouse layout and the total distance needed in order to meet the customer's demand.

With this result, it can save time consumed and help the company decrease operational costs.

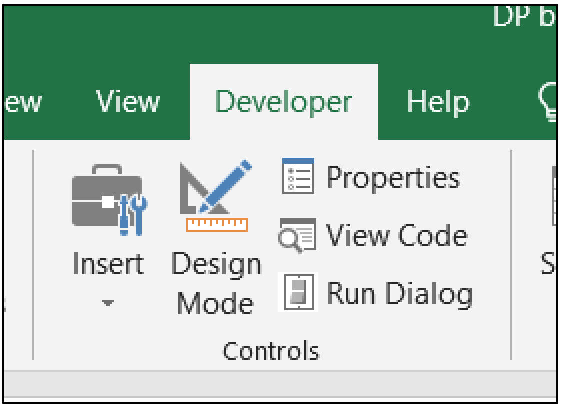


Figure 8: The Developer Used to Build Button

## 4.0 Results

The results are discussed in three sections: understanding the data, calculation of Dynamic Programming Method using interface from Microsoft Excel, and chapter summary. All the process and data in Microsoft Excel were explained to be better understood.

### 4.1 Understanding the Data

The warehouse inner transportation optimization model presents items displacement to achieve its optimal location. Therefore, the data is divided into two categories which are preliminary data and actual data. Each data has its purpose in calculating Dynamic Programming Method.

The first quarter of actual data was labeled as preliminary data. The purpose of having preliminary data was to validate the manual calculation and calculation distance made by Microsoft Excel, as it is in a small area. The preliminary data was enough for model presentation and further testing. Furthermore, actual data may take a more extended period to calculate manually, and whenever having a problem in calculation lead to difficult detection.

Nevertheless, using Microsoft Excel Interface made by Hamdy A. Taha required a proper step to insert the data. For example, node 1, contains all transitions to all nodes. The distance from node 1 to others node is shown in Table 1. The calculation for node *1* to node *2* is 5, to node *3* is 9 and the same calculation will follow to complete in other nodes. However, node *1* to node *14* will be considered the shortest distance to arrive at node *14*. An infinity symbol (∞) represents the nodes that traverse towards itself. The same calculation will be used, and the distance matrix from one node to another shown in Table 2.

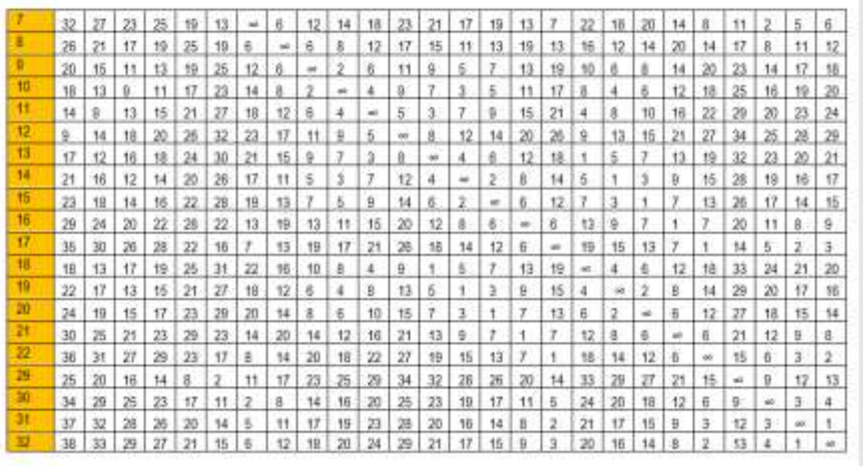
Table 1: Distance Matrix for Node *1*

| Node | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ∞ | 5 | 9 | 11 | 9 | 14 | 18 | 20 | 17 | 21 | 23 | 18 | 22 | 24 |

Table 2: Distance Matrix for Preliminary Data

| Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ∞ | 5 | 9 | 11 | 9 | 14 | 18 | 20 | 17 | 21 | 23 | 18 | 22 | 24 |
| 2 | 5 | ∞ | 4 | 6 | 14 | 9 | 13 | 15 | 12 | 16 | 18 | 13 | 17 | 19 |
| 3 | 9 | 4 | ∞ | 2 | 18 | 13 | 9 | 11 | 16 | 12 | 14 | 17 | 13 | 19 |
| 4 | 11 | 6 | 2 | ∞ | 20 | 15 | 11 | 9 | 18 | 14 | 12 | 19 | 15 | 13 |
| 5 | 9 | 14 | 18 | 20 | ∞ | 5 | 9 | 11 | 8 | 12 | 14 | 9 | 13 | 15 |
| 6 | 14 | 9 | 13 | 15 | 5 | ∞ | 4 | 6 | 3 | 7 | 9 | 4 | 8 | 10 |
| 7 | 18 | 13 | 9 | 11 | 9 | 4 | ∞ | 2 | 7 | 3 | 5 | 8 | 4 | 6 |
| 8 | 20 | 15 | 11 | 9 | 11 | 6 | 2 | ∞ | 9 | 5 | 3 | 10 | 6 | 4 |
| 9 | 17 | 12 | 16 | 18 | 8 | 3 | 7 | 9 | ∞ | 4 | 6 | 1 | 5 | 7 |
| 10 | 21 | 16 | 12 | 14 | 12 | 7 | 3 | 5 | 4 | ∞ | 2 | 5 | 1 | 3 |
| 11 | 23 | 18 | 14 | 12 | 14 | 9 | 5 | 3 | 6 | 2 | ∞ | 7 | 3 | 1 |
| 12 | 18 | 13 | 17 | 19 | 9 | 4 | 8 | 10 | 1 | 5 | 7 | ∞ | 4 | 6 |
| 13 | 22 | 17 | 13 | 15 | 13 | 8 | 4 | 6 | 5 | 1 | 3 | 4 | ∞ | 2 |
| 14 | 24 | 19 | 19 | 13 | 15 | 10 | 6 | 4 | 7 | 3 | 1 | 6 | 2 | ∞ |

Table 3. Distance Matrix for Actual Data



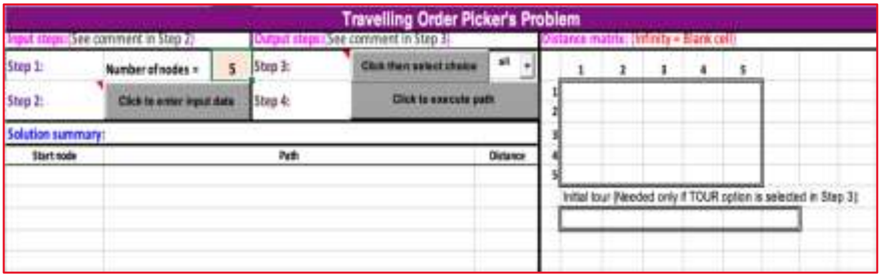
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 32 | 27 | 23 | 25 | 19 | 13 | ∞ | 6 | 12 | 14 | 18 | 23 | 21 | 17 | 19 | 13 | 7 | 22 | 18 | 20 | 14 | 8 | 11 | 2 | 5 | 6 |
| 8 | 26 | 21 | 17 | 19 | 25 | 19 | 6 | ∞ | 6 | 8 | 12 | 17 | 15 | 11 | 13 | 19 | 13 | 16 | 12 | 14 | 20 | 14 | 17 | 8 | 11 | 12 |
| 9 | 20 | 15 | 11 | 13 | 19 | 25 | 12 | 6 | ∞ | 2 | 6 | 11 | 9 | 5 | 7 | 13 | 19 | 10 | 6 | 8 | 14 | 20 | 23 | 14 | 17 | 18 |
| 10 | 18 | 13 | 9 | 11 | 17 | 23 | 14 | 8 | 2 | ∞ | 4 | 9 | 7 | 3 | 5 | 11 | 17 | 8 | 4 | 6 | 12 | 18 | 25 | 16 | 19 | 20 |
| 11 | 14 | 8 | 13 | 15 | 21 | 27 | 18 | 12 | 6 | 4 | ∞ | 5 | 3 | 7 | 9 | 15 | 21 | 4 | 8 | 10 | 16 | 22 | 29 | 20 | 23 | 24 |
| 12 | 9 | 14 | 18 | 20 | 26 | 32 | 23 | 17 | 11 | 9 | 5 | ∞ | 8 | 12 | 14 | 20 | 26 | 9 | 13 | 16 | 21 | 27 | 34 | 25 | 28 | 29 |
| 13 | 17 | 12 | 16 | 18 | 24 | 30 | 21 | 15 | 9 | 7 | 3 | 8 | ∞ | 4 | 6 | 12 | 18 | 1 | 5 | 7 | 13 | 19 | 32 | 23 | 20 | 21 |
| 14 | 21 | 16 | 12 | 14 | 20 | 26 | 17 | 11 | 5 | 3 | 7 | 12 | 4 | ∞ | 2 | 8 | 14 | 5 | 1 | 3 | 9 | 15 | 28 | 19 | 16 | 17 |
| 15 | 23 | 18 | 14 | 16 | 22 | 28 | 19 | 13 | 7 | 5 | 9 | 14 | 6 | 2 | ∞ | 6 | 12 | 7 | 3 | 1 | 7 | 13 | 26 | 17 | 14 | 15 |
| 16 | 29 | 24 | 20 | 22 | 28 | 22 | 13 | 19 | 13 | 11 | 15 | 20 | 12 | 8 | 6 | ∞ | 6 | 13 | 9 | 7 | 1 | 7 | 20 | 11 | 8 | 9 |
| 17 | 35 | 30 | 26 | 28 | 22 | 16 | 7 | 13 | 19 | 17 | 21 | 26 | 18 | 14 | 12 | 6 | ∞ | 19 | 15 | 13 | 7 | 1 | 14 | 5 | 2 | 3 |
| 18 | 18 | 13 | 17 | 19 | 25 | 31 | 22 | 16 | 10 | 8 | 4 | 9 | 1 | 5 | 7 | 13 | 19 | ∞ | 4 | 6 | 12 | 18 | 33 | 24 | 21 | 20 |
| 19 | 22 | 17 | 13 | 15 | 21 | 27 | 18 | 12 | 6 | 4 | 8 | 13 | 5 | 1 | 3 | 9 | 15 | 4 | ∞ | 2 | 8 | 14 | 29 | 20 | 17 | 16 |
| 20 | 24 | 19 | 15 | 17 | 23 | 29 | 20 | 14 | 8 | 6 | 10 | 15 | 7 | 3 | 1 | 7 | 13 | 6 | 2 | ∞ | 6 | 12 | 27 | 18 | 15 | 14 |
| 21 | 30 | 25 | 21 | 23 | 29 | 23 | 14 | 20 | 14 | 12 | 16 | 21 | 13 | 9 | 7 | 1 | 7 | 12 | 8 | 6 | ∞ | 6 | 21 | 12 | 9 | 8 |
| 22 | 36 | 31 | 27 | 29 | 23 | 17 | 8 | 14 | 20 | 18 | 22 | 27 | 19 | 15 | 13 | 7 | 1 | 18 | 14 | 12 | 6 | ∞ | 15 | 6 | 3 | 2 |
| 29 | 25 | 20 | 16 | 14 | 8 | 2 | 11 | 17 | 23 | 25 | 29 | 34 | 32 | 26 | 26 | 20 | 14 | 33 | 29 | 27 | 21 | 15 | ∞ | 9 | 12 | 13 |
| 30 | 34 | 29 | 25 | 23 | 17 | 11 | 2 | 8 | 14 | 16 | 20 | 25 | 23 | 19 | 17 | 11 | 5 | 24 | 20 | 18 | 12 | 6 | 9 | ∞ | 3 | 4 |
| 31 | 37 | 32 | 28 | 26 | 20 | 14 | 5 | 11 | 17 | 19 | 23 | 28 | 20 | 16 | 14 | 8 | 2 | 21 | 17 | 15 | 9 | 3 | 12 | 3 | ∞ | 1 |
| 32 | 38 | 33 | 29 | 27 | 21 | 15 | 6 | 12 | 18 | 20 | 24 | 29 | 21 | 17 | 15 | 9 | 3 | 20 | 16 | 14 | 8 | 2 | 13 | 4 | 1 | ∞ |

As mentioned earlier, the data was adapted from (Dobrilovic et al., 2012). The data consists of 26 nodes, and the forklift is set at node 1 as well as for load and unload items.

Optimization of the data includes the calculation of the shortest path between the forklift and storage position. The route in the data was complex since it has multiple aisles. And all routes consist of all possible transitions. An infinity symbol (∞) represents the nodes that cannot traverse to another node. The same calculation in Table 2 was used to calculate the distance matrix of the actual data as summarized in Table 3.

## 4.2 Applying Dynamic Programming Method to Find the Shortest Path for Forklift in the warehouse

The calculation of the shortest path was done using the adapted Microsoft Excel tools made by Hamdy A. Taha. The manual calculation was not relevant if the data is bigger and may take a long time for its calculation. The system was executed for all the possible routes in the data. The node would be linked to the nearest node in order to get the shortest distance. The process continued until a path was formed. A path provided would pass through all nodes which satisfy the objective. The system requires 4 steps as shown in Figure 9.



Travelling Order Picker's Problem

Input steps: (See comment in Step 2)

Output steps: (See comment in Step 3)

Distance matrix: (Infinity = Blank cell)

Step 1: Number of nodes = 5

Step 2: Click to enter input data

Step 3: Click then select choice all

Step 4: Click to execute path

Solution summary:

| Start node | Path | Distance |
|---|---|---|

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |

Initial tour (Needed only if TOUR option is selected in Step 3):

Figure 9: Microsoft Excel View

Step 1: The number of nodes described the nodes that contain the ordered item. Hence, the order picker needs to collect items from the nodes. It provides a symmetric matrix. For example, the node was set as 5 and the right box shows the 5x5 matrix.

Step 2: This step clears the input data area and may be used only if Step 1 specifies a new number of nodes.   Else, distance matrix may be edited directly as desired.

Step 3: Provide two options which are:

- ALL (default) will display all the possible shortest path for every node. The red colour of font showed the minimum distance travelled, if one is found.
- A specific number will display the path based on the chosen number.

Step 4: A button that executes all possible routes and the shortest distance will be highlighted in red color.

Meanwhile, the right box displayed the distance matrix need to fill with the distance or weightage for the nodes. It linked from one node to another. Lastly, the box at the bottom shows all the possible route.

### 4.2.1 Preliminary Data

After keying in the data into Microsoft Excel Interface, by following the step, it will execute all the possible routes as shown in Table 4. It consists of 14 nodes and set at step 1 with 14, showing the 14x14 matrix. The red color will determine the shortest path and distance. Meanwhile, the first column presented nodes to start and end.

Table 4: Shortest Path and Distance of Preliminary Data in Microsoft Excel

| Start node | Path | Distance |
|---|---|---|
| 1 | 1-2-3-4-8-7-10-13-14-11-9-12-6-5-1 | 54 |
| 2 | 2-3-4-8-7-10-13-14-11-9-12-6-5-1-2 | 54 |
| 3 | 3-4-2-1-5-6-9-12-13-10-11-14-8-7-3 | 54 |
| 4 | 4-3-2-1-5-6-9-12-13-10-11-14-8-7-4 | 54 |
| 5 | 5-6-9-12-13-10-11-14-8-7-3-4-2-1-5 | 54 |
| 6 | 6-9-12-13-10-11-14-8-7-5-1-2-3-4-6 | 62 |
| 7 | 7-8-11-14-13-10-9-12-6-5-1-2-3-4-7 | 54 |
| 8 | 8-7-10-13-14-11-9-12-6-5-1-2-3-4-8 | 54 |
| 9 | 9-12-13-10-11-14-8-7-6-5-1-2-3-4-9 | 62 |
| 10 | 10-13-14-11-8-7-6-9-12-5-1-2-3-4-10 | 60 |
| 11 | 11-14-13-10-7-8-6-9-12-5-1-2-3-4-11 | 60 |
| 12 | 12-9-6-7-8-11-14-13-10-5-1-2-3-4-12 | 68 |
| 13 | 13-10-11-14-8-7-6-9-12-5-1-2-3-4-13 | 62 |
| 14 | 14-11-10-13-12-9-6-7-8-4-3-2-1-5-14 | 62 |

Since the forklift started at node 1, refer to the first column where the nodes begin. The second column shows the resulting tour. As in this situation, **1-2-3-4-8-7-10-13-14-11-9-12-6-5-1** is the best path with a distance 54m. The paths shown in table 4.4 passed all nodes that satisfy the objectives. Order pickers can refer to the system in order to reduce time handling when picking up the goods. The same systems were used to determine the shortest path in actual data as shown in Table 5. For better understanding, the route given in Table 4 will be drawn in Figure 10.

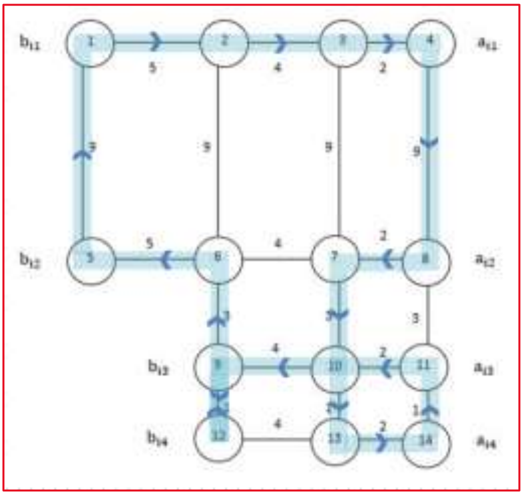


Figure 10: Arrow for Shortest Path in Preliminary Data

### 4.2.2 Actual Data

The same procedure as preliminary data were followed to find the shortest path and distance for actual data. Since the data is big, the shortest route displayed is much longer compared to the preliminary data. It consists of 26 nodes then set at step 1 with 26, the right box will display the 26x26 matrix.

The data actual data consists of 26 nodes, and Table 5 shows the first 17 nodes. Therefore, the shortest path shown in node *17* cannot be considered. In this study, node *17* will not be considered as the shortest path since the starting and ending node will be at node *1*. Referring to Figure 3: Warehouse layout with one forklift, node *17* is located in the middle of the warehouse. Thus, the node was not relevant for the forklift to start operation. This is also mentioned in the assumption part made earlier. Therefore, for this case, **1-2-3-4-5-6-23-24-7-25-26-22-17-16-21-20-15-14-19-18-13-11-10-9-8-12-1** is the only shortest path for node *1* with 108m. For better understanding, the shortest route given in Table 5 were drawn in Figure 11.

Table 5: Shortest Path and Distance of Actual Data in Microsoft Excel

| Start node | Path | Distance |
|---|---|---|
| 1 | 1-2-3-4-5-6-23-24-7-25-26-22-17-16-21-20-15-14-19-18-13-11-10-9-8-12-1 | 108 |
| 2 | 2-3-4-5-6-23-24-7-25-26-22-17-16-21-20-15-14-19-18-13-11-10-9-8-12-1-2 | 108 |
| 3 | 3-4-5-6-23-24-7-25-26-22-17-16-21-20-15-14-19-18-13-11-10-9-8-12-1-2-3 | 108 |
| 4 | 4-3-2-1-12-11-13-18-19-14-15-20-21-16-17-22-26-25-24-7-8-9-10-5-6-23-4 | 112 |
| 5 | 5-6-23-24-7-25-26-22-17-16-21-20-15-14-19-18-13-11-10-9-8-12-1-2-3-4-5 | 108 |
| 6 | 6-23-5-4-3-2-1-12-11-13-18-19-14-15-20-21-16-17-22-26-25-24-7-8-9-10-6 | 112 |

| | | |
|---|---|---|
| 7 | 7-24-25-26-22-17-16-21-20-15-14-19-18-13-11-10-9-8-23-6-5-4-3-2-1-12-7 | 120 |
| 8 | 8-9-10-14-19-20-15-16-21-22-17-25-26-24-7-23-6-5-4-3-2-1-12-11-13-18-8 | 108 |
| 9 | 9-10-14-19-20-15-16-21-22-17-25-26-24-7-8-11-13-18-12-1-2-3-4-5-6-23-9 | 120 |
| 10 | 10-9-14-19-20-15-16-21-22-17-25-26-24-7-8-11-13-18-12-1-2-3-4-5-6-23-10 | 124 |
| 11 | 11-13-18-19-14-15-20-21-16-17-22-26-25-24-7-8-9-10-12-1-2-3-4-5-6-23-11 | 120 |
| 12 | 12-11-13-18-19-14-15-20-21-16-17-22-26-25-24-7-8-9-10-3-4-5-6-23-2-1-12 | 112 |
| 13 | 13-18-19-14-15-20-21-16-17-22-26-25-24-7-8-9-10-11-12-1-2-3-4-5-6-23-13 | 120 |
| 14 | 14-19-20-15-10-9-11-13-18-12-1-2-3-4-5-6-23-24-7-25-26-22-17-16-21-8-14 | 122 |
| 15 | 15-20-19-14-10-9-11-13-18-12-1-2-3-4-5-6-23-24-7-25-26-22-17-16-21-8-15 | 122 |
| 16 | 16-21-22-17-25-26-24-7-8-9-10-14-19-20-15-13-18-11-12-1-2-3-4-5-6-23-16 | 108 |
| 17 | 17-22-26-25-24-7-8-9-10-14-19-20-15-16-21-18-13-11-12-1-2-3-4-5-6-23-17 | 106 |

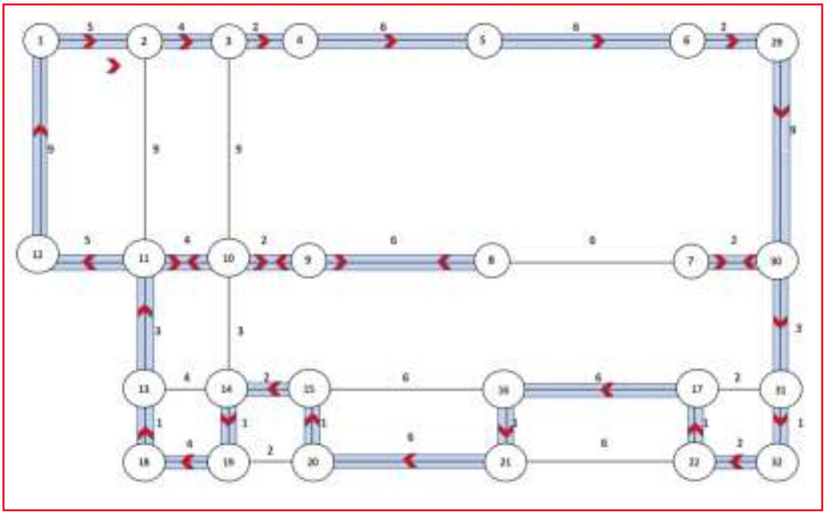


Figure 11: Shortest Path in Actual Data

## 5.0 Discussion and Conclusion

This study is about applying the dynamic programming method in the warehouse. Dynamic programming method minimizes the path to be passed by the order picker in collecting the ordered items. Determining the shortest route for this study is to save warehouse costs such as transportation costs and time is taken. Microsoft Excel is the platform used in this study to calculate the shortest distance.

The first objective, which is to calculate the optimal distance traveled and determine the shortest path for order pickers in the warehouse. The result was obtained using Microsoft Excel, showing that the minimum total distance for the order picker to collect all orders in the warehouse studied is 112 meters which are for the actual data. The second objective is to determine the shortest path for the whole area under study using the interface by Hamdy A. Taha. The system was adapted to suit the warehouse designs.

Minimizing the distance traveled of the forklift helps to reduce the handling time of ordered items and waste of the forklift services. For example, when the distance traveled by the forklift is reduced, the least used forklift can reduce maintenance and service costs.The recommendations for this study and future research are to add the number of forklifts to collect customers' demands. Next, consider weekends and holiday as some staff works overtime because this paper only considers regular working hours during the day.

## Acknowledgement

The authors wish to thank ReNeU UiTM and ILD UiTM for facilitating the writing and publication workshop. Special thanks to Ungku Sofi Dayana Bt Ungku Iskandar Zukarnine ,Nurdiana Bt Mohd Razif and Nurul Najiha Bt Mohammed Hairul for running the data on Excel and compiling this report.

## Article Contribution to Related Field of Study

This article is contributing to the field of logistics management in which every process has its own cost. The output of the study can help the practitioner to plan their process and for the researchers to enhance their knowledge.

## References

Anđelković, A., & Radosavljević, M. (2018). Improving the order-picking process through implementation of the warehouse management system. *Strategic Management-International Journal of Strategic Management and Decision Support Systems in Strategic Management*, *23*(1).

Bhowmik, B. (2010). Dynamic Programming - Its Principles, Applications, Strengths, and Limitations. *International Journal of Engineering Science and Technology, 2*(9), 4822-4826.

Bouman, P., Agatz, N., & Schmidt, M. (2016). <154419746.pdf>. *9*(1), 61-63. doi:10.1145?321105.321111

Chen, F., Xu, G., & Wei, Y. (2018). Heuristic routing methods in multiple-block warehouses with ultra-narrow aisles and access restriction. *International Journal of Production Research, 57*(1), 228-249. doi:10.1080/00207543.2018.1473657

de las Casas, P. M., Sedeno-Noda, A., & Borndörfer, R. (2021). An improved multiobjective shortest path algorithm. Computers & Operations Research, 135, 105424.

de Leeuw, S. & Wiers, V. C. S. (2010). Warehouse manpower planning strategies in times of financial crisis: evidence from logistics service providers and retailers in the Netherlands. *Production Planning and Control, 2*, 1-16.

Dobrilovic, D., Jevtic, V., Beker, I., & Stojanov, Z. (2012). Shortest-path based Model for Warehouse Inner Transportation Optimization. *7th IEEE International Symposium on Applied Computational Intelligence and Informatics*, 63-68. doi:978-1-4673-1014-7

Ferone, D., Festa, P., Fugaro, S., & Pastore, T. (2021). A dynamic programming algorithm for solving the k-Color Shortest Path Problem. *Optimization Letters*, *15*(6), 1973-1992.

Ghani, N. E. A., Shariff, S. S. R., & Zahari, S. M. (2016). An alternative algorithm for vehicle routing problems with time windows for daily deliveries. *Advances in Pure Mathematics*, 6(5), 342-350.

Gutenschwager, K., Radtke, A., Volker, S., & Zeller, G. (2012). The Shortest Path: Comparison of Different Approaches and Implementation for the Automatic Routing of Vehicles. *Proceedings of the 2012 Winter Simulation Conference*, 3312-3323. doi:978-4673-4782-2

Heragu, S. S., Du, L., Mantel, R. J., & Schuur, P. C. (2005). A mathematical model for warehouse design and product allocation. *International Journal of Production Research, 43*(2), 327-338. doi:10.1080/00207540412331285841

Labus, N., & Gajsek, B. (2018). Use of Ergonomic Principles in Manual Order Picking Systems. *Logistics & Sustainable Transport, 9*(1), 11-22.

Lantoine, G., & Russell, R. P. (2012). A hybrid differential dynamic programming algorithm for constrained optimal control problems. Part 1: Theory. *Journal of Optimization Theory and Applications*, *154*, 382-417.

Lee, J. A., Chang, Y. S., Shim, H.-J., & Cho, S.-J. (2015). A Study on the Picking Process Time. *Procedia Manufacturing, 3*, 731-738. doi:10.1016/j.promfg.2015.07.316

Lewis, R. (2020). Algorithms for finding shortest paths in networks with vertex transfer penalties. *Algorithms*, *13*(11), 269.

Liu, J & Limere, V. (2012). An Improved Largest Gap Routing Heuristic For Order Picking. *Industrial Management*, 1–5.

Liu, D., Yu, W., Baldi, S., Cao, J., & Huang, W. (2019). A switching-based adaptive dynamic programming method to optimal traffic signaling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(11), 4160-4170.

Ma'mun, S., Kusrini, E., Indah Asmarawati, C., Masita Sari, G., Nurjanah, A., Kisanjani, A., . . . Purnomo, M. R. A. (2018). Warehousing performance improvement using Frazelle Model and per group benchmarking: A case study in the retail warehouse in Yogyakarta and Central Java. *MATEC Web of Conferences, 154*, 01091. doi:10.1051/matecconf/201815401091

Masae, M., Glock, C. H., & Vichitkunakorn, P. (2020). Optimal order picker routing in a conventional warehouse with two blocks and arbitrary starting and ending points of a tour. *International Journal of Production Research*, *58*(17), 5337-5358.

McKinnon, A., Flöthmann, C., Hoberg, K., & Busch, C. (2017). *Logistics competencies*, *skills, and training: a global overview*. World Bank Publication

Mirzaei, M., Zaerpour, N., & De Koster, R. (2021). The impact of integrated cluster-based storage allocation on parts-to-picker warehouse performance. *Transportation Research Part E: Logistics and Transportation Review*, *146*, 102207.

Ngah, R., Abdullah, J., Khalique , M. ., & Goyipnazarov, S. B. . (2021). The Influence of Socio-Economics on

Travel Behavior of Public Transportation in Malaysia. *Environment-Behaviour Proceedings Journal,* 6(17), 269-275.

Nordin, N. A. M., Omar, M., & Shariff, S. S. R. (2018). The Application of Dynamic Programming Method in Finding Shortest Path for Order Picker with Limited Picking Capacity. *An International Journal, 13*, 20-48.

Queirolo, F., Tonelli, F., Schenone, M., Nan, P. & Zunino, I. (2002). Warehouse Layout Design Minimizing Travel Time with a Genetic and Simulative Approach. *Simulation Symposium, 14th Edition*, 2-6.

Rahmat, A. K., Razak, A. H. A., Yusak, N. A. M., & Pahim, K. M. (2022). Refining Electronic Hailing Service Quality on Customer Satisfaction and Impact on Electronic Word of Mouth. *Environment-Behaviour Proceedings Journal*, 7(SI8), 61-67.

Roodbergen, K. J., & Koster, R. (2001). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research, 39*(9), 1865-1883. doi:10.1080/00207540110028128

Rushton, A., Croucher, P., & Baker, P. (2022). *The handbook of logistics and distribution management: Understanding the supply chain*. Kogan Page Publishers.

Şenaras, A. E., İnanç, Ş., Sezen, H. K., & Şenaras, O. M. (2021). Shortest Route Application via Dynamic Programming in the Transportation Networks. In *Handbook of Research on Decision Sciences and Applications in the Transportation Sector* (pp. 362-371). IGI Global.

Shariff, S. R. ., Mohd Nordin, N. A. ., Omar, M. ., & Supadi, S. S. . (2022). Modeling the Inner Warehouse Shortest Route Planning using Dynamic Programming Block . *Environment-Behaviour Proceedings Journal*, 7(SI9), 611-617.

Singhal, D. A., & Pandey, P. (2016). Travelling Salesman Problems by Dynamic Programming Algorithm. *Scientific Engineering and Applied Science (IJSEAS), 2*(1), 263-267.

Vrysagotis, V., & Kontis, P. A. (2011). Warehouse layout problems : Types of problems and solution algorithms. *Journal of Computations & Modelling, 1*(1), 131-152.

Wang, S., Zhen, L. & Zhuge, D. (2018). Dynamic programming algorithms for the selection of waste disposal ports in cruise shipping. doi:10.1016/j.trb.2017.12.016

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research, 95*, 139-150. doi:10.1016/j.cor.2018.03.012

Zakaria, A., & Zurcher, S. A. (2023). An Analysis of Sustainability Approaches of Malaysian Public Universities. *Environment-Behaviour Proceedings Journal*, *8*(23)