Scientific Research Publishing

Open Journal of Applied Sciences, 2021, 11, 946-965
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

Controllers Design for the Multi-Shuttle and Multi-Station Transportation System

Tien Dong Ha, Minh Tien Trinh, Tran Thanh Cong Vu, Tuong Quan Vo*

Faculty of Mechanical Engineering, Ho Chi Minh City University of Technology, VNU-HCM, Viet Nam
Email: *vtquan@hcmut.edu.vn

How to cite this paper: Ha, T.D., Trinh, M.T., Vu, T.T.C. and Vo, T.Q. (2021) Controllers Design for the Multi-Shuttle and Multi-Station Transportation System. *Open Journal of Applied Sciences*, **11**, 946-965.
https://doi.org/10.4236/ojapps.2021.118069

Received: July 28, 2021
Accepted: August 28, 2021
Published: August 31, 2021

Abstract

Nowadays, the Multi-Shuttle and Multi-Station Transportation System (MMTS) is one of the most interesting research topics in many fields of industries. It is an effective solution to reduce unexpected accidents that occur during transportation as well as increase productivity in manufacturing. The aim of this paper is to introduce the controller design for the MMTS which is built in our BK-Recme BioMech Lab at Ho Chi Minh City University of Technology (VNU-HCM), Viet Nam. Based on the design of this system, the control algorithms will be conducted to check the operation of the whole system. To evaluate the feasibility and effectiveness of this model, we design a series of random instances for different quantities of nodes as well as the different quantities of shuttles. Our system includes 4 stations and 6 shuttles which are assembled in the serial chain system. However, the number of stations and number of shuttles can be expanded to any desired ones which are based on the requirement of the industries. In this paper, we mainly focus on the controller design of this system to make it operate in an effective way that the goods will be transported and delivered to the target station as fast as possible. In order to solve the large-scale instances and realistic transport problems, we propose three algorithms for three progresses as shuttles calling, path reading and shuttles communicating. The shuttles calling is to decide which shuttle should be called to the start-node. Path reading to determine the shortest way to go from start-node to end-node. Finally, shuttles communicating, which allow one shuttle to interact with the next shuttles so we have a loop of orders (shuttle 1 to shuttle 2; shuttle 2 to shuttle 3; etc.; shuttle n-1 to shuttle n). This proposes solution can help us to solve the huge numbers of shuttles and stations in the system. The specific result of this study is applying Dijkstra's algorithm to propose an algorithm that allows handling a transportation system without caring about the number of shuttles as well as the number of stations for the closed-loop path. Several test problems are carried out in order to check the feasibility and the effectiveness of our purposed control algorithm.

Keywords

Multi-Station, Multi-Shuttle, Transportation, Shuttle, Calling, RFID, Communicating, Start-Node, End-Node

1. Introduction

This paper introduces the Multi-Shuttle and Multi-Station Transportation System (MMTS) design for general manufacturing facilities and warehouse operation. This proposed system is suitable to apply in industries to do the task of moving material from one location to other locations in the manufacturing environments or warehouses. The ideas of this system have been proved to be so necessary for industries to improve productivity. In industries, we can see that most of the operations are related to the transport of discrete parts or scrap.

Many kinds of research about multi-station system have been carried out by many researchers around the world. Prerna Tiwari and Manoj K. Tiwari proposed a novel methodology for sensor placement for the multi-station manufacturing processes so that the dimensional variation in the manufactured product [1]. Frank L. Hitchcock did research about desiring the least costly manner of distribution products from factories in a number of cities [2]. J. Ashayeri and L.F. Gelders examined the literature of warehouse design optimization and conducted a survey of the literature related to the design problem [3]. In recent years, there are many research focus on the queueing theory, which is a powerful tool in modelling and performance analysis of many complicated systems, such as communication between network nodes, telecommunication systems, automatic shuttle systems, manufacturing systems. Visschers *et al.* [4] considered a memoryless single station service system with many servers, and the research found out that there existed assignment probabilities under which the system had a product from stationary distribution and obtained explicit expressions for it; the waiting time distributions in steady state had been derived. Mather *et al.* [5] developed some multi-class queueing networks that time-dependent distributions for the multi-class queue can have a factored form which reduces the problem of computing, such distributions to a similar problem for related single-class queueing networks. Kim and Morrison [6] presented some equilibrium probabilities in a class of two stations closed queueing network. Jung and Morrison [7] gave closed-form solutions for the equilibrium probability distribution in the closed Lu-Kumar network under two buffer priority policies. In general, we can see that there are many studies related to multi-station transportation systems, however, just a few of them concern about algorithms to handle the movement of vehicles in the system. And, the most important thing is that the general algorithm can handle the situation by changing the number of shuttles and the stations in closed-loop transportation.

Re-entrant lines, which is described by Harrison [8], are a special case of

queueing models related to systems composed of some machines or stations, in which customers are processed several times by the same server. These schemes are used to model a variety of real-life systems, including service centers, production or manufacturing systems, computer and communication networks. Much attention has been devoted to obtaining stability conditions for this kind of network, Adan and Weiss (2005, 2006) [9] [10] did research about two-node Jackson network with infinite supply of work and gave an analysis of a simple Markvian re-entrant line with infinite supply of work under the LBFS policy, Nazarathy and Weiss [11], Weiss [12] [13] and Nazarathy [14].

Because of the complexity of the queueing algorithm (re-entrant line), we propose the queueing algorithm that can apply for the serial multi-shuttle and multi-station system, which is easier to reach and allows to solve the multi-station system with n number of stations and m number of shuttles. In this first period, our proposed algorithm is just applied to closed-loop path and all the shuttles are called at the same time.

The remaining of this paper is arranged into five sections as follows: Section 2 introduces the general concepts of controller design. The detail of the controller design for two main modules is presented in Sections 3 and 4. The simulation results of the algorithm are described in Section 5. Finally, the conclusions and direction of future work are provided in Section 6.

2. General Concepts

An intuitive demonstration, including the outside design of multi-station transportation model and the layout of electrical elements, is introduced in **Figure 1**. This model has four stations and six shuttles which are arranged on two floors operating at the same time. Therefore, the discrete parts can be transported from floor 1 to floor 2 easily with the help of two cylinders. The RFID cards are arranged evenly along the way and the RFID reader is set up on each shuttle so that the server can grasp the real-time operation of the system. The algorithms which are used in this paper are introduced in the below sections.

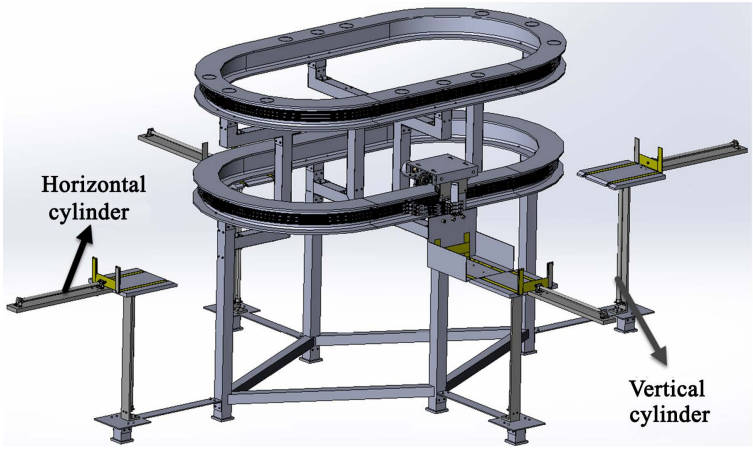

Figure 1. The proposed structure of the multi-station transportation system model.

2.1. Path Reading Algorithm

To find out the shortest way from start-node to end-node, there are many algorithms which are extremely popular, can be chosen to use, such as Dijkstra algorithm, Bellman-Ford algorithm, Floyd-Warshall algorithm, Johnson algorithm or TSP algorithm. In this research, Dijkstra algorithm is chosen to apply for our proposed system. The reason why Dijkstra algorithm is selected to apply in our research is because this is a popular algorithm, easy to apply in many types of controllers, simple, fast computation, etc. Besides, this algorithm is still used in routing protocols for solving many single-source shortest path problems without negative edge weight in the graphs. Overview of this algorithm let the node at which we start to be called a start-node and the node at which we finished be called end-node. The distance from the start-node to end-node is called briefly the distance of end-node. The proposed design of our serial multi-station system is introduced in Figure 1.

Djikstra algorithm will assign some initial distance values and will try to improve them step by step. Each step of the algorithm will be described as the following steps:

Step 1: Mark all nodes as unvisited. Create a set of all the unvisited nodes called the unvisited set.

Step 2: Assign to every node a tentative distance value: set it to zero for our start-node and to infinity to all other nodes. Set the start-node at the current position.

Step 3: For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the new calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node X is marked with a distance of 3, and the edge connecting it with a neighbor Y has the length 2, then the distance from Y through X will be 3 + 2 = 5. If Y was previously marked with a greater distance than 5 then change it to 5. Otherwise, the current value will be kept.

Step 4: When we finish considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.

Step 5: If the destination node has been marked visited or if the smallest tentative distance among the nodes in the unvisited set is infinity, then stop. The algorithm is finished.

Step 6: Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node" and get back to step 3.

An example of the Dijkstra algorithm applying in our proposed MMTS

In this example, as in Figure 2, we will calculate the shortest path from node A and other nodes in our graph. During the algorithm execution, we will mark every node with its minimum distance to node A (selected node). For node A, this distance is 0. For the rest, as we still don't know the minimum distance, we suppose these values are infinity (∞). We also have a current node. Initially, we

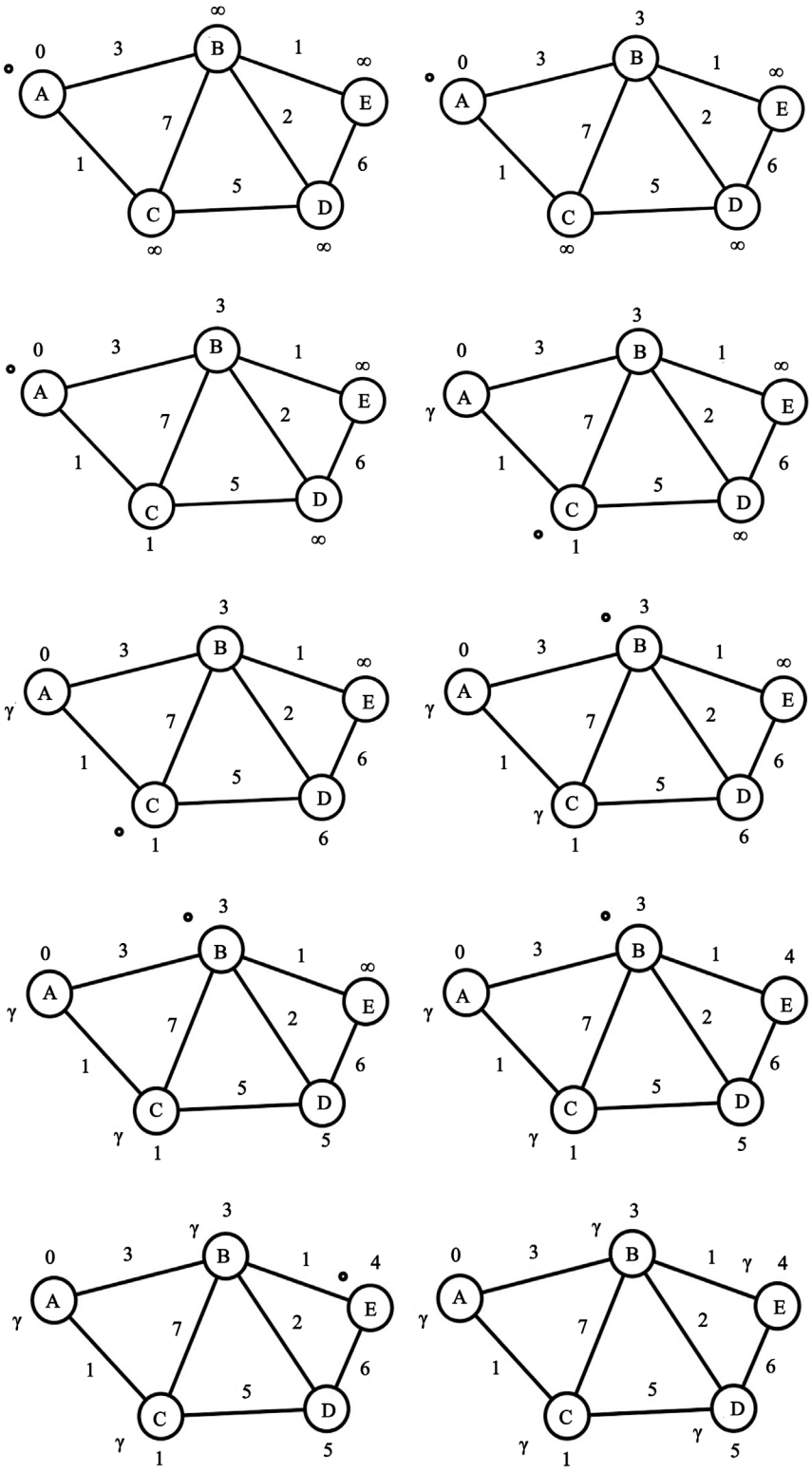

Figure 2. Example of Dijkstra algorithm. *Note: In **Figure 2**, the "small dot" is used to determine the "current node", which is being checked to get the shortest distance with the other nodes. The "check mark" is used to determine the node that has been checked the distance with its neighbors. The number on the line connecting 2 nodes is the distance between them. The number of its node is the shortest distance from node A to it.

set current node is node A. In Figure 2(a), the current node is marked with a small dot nearby.

Now, we check the neighbors of our current node (B and C) in no specific order. Let's begin with B. We add the minimum distance of the current node (in this case is 0) with the weight of the edge that connects our current node with B (in this case is 3) and we obtain 0 + 3 = 3. We compare that value with the minimum distance of B (infinity), the lower value is the one that remains as the minimum distance of B (in this case 3 is less than infinity) (Figure 2(b)).

Then, we move to check neighbor C. We add the minimum distance of the current node (in this case is 0) with the weight of the edge that connects our current node with C (in this case is 1) and we obtain 0 + 1 = 1. We compare that value with the minimum distance of C (infinity), the lower value is the one that remains as the minimum distance of C (in this case 1 is less than infinity) (Figure 2(c)).

We have to check all the neighbors of A. Therefore, we mark it as visited. The visited nodes are represented with a check mark and we pick a new current node. That node must be an unvisited node with the smallest minimum distance (node C). Mark it with a small dot nearby (Figure 2(d)).

Now we repeat the algorithm. We check the neighbors of our current node, ignoring the visited nodes. This means we just need to check node B and node D. For B, we add 1 (the minimum distance of C, our current node) with 7 (the weight of the edge connecting B and C) to obtain 8. We compare that value with the minimum distance of B (3) and leave the smaller value: 3.

Continue with node D, we add 1 (the minimum distance of C, our current node) with 5 (the weight of the edge connecting D and C) to obtain 6. We compare that value with the minimum distance of D (infinity) and leave the smaller value: 6 (Figure 2(e)).

Afterwards, we mark C as visited and pick a new current node (B), which is non-visited node with the smallest current distance. Repeat the algorithm, this time we check D and E (Figure 2(f)).

For D, we add 3 (the minimum distance of B, our current node) with 2 (the weight of the edge connecting B and D) to obtain 5. We compare that value with the minimum distance of D (6) and leave the smaller value: 5 (Figure 2(g)).

For E, we add 3 (the minimum distance of B, our current node) with 1 (the weight of the edge connecting B and E) to obtain 4. We compare that value with the minimum distance of E (infinity) and leave the smaller value: 4 (Figure 2(h)).

We mark B as visited and set node E to be current node. Repeat the algorithm; we just need to check node D. For D, we add 4 (the minimum distance of E, our current node) with 6 (the weight of the edge connecting B and D) to obtain 10. We compare that value with the minimum distance of D (5) and leave the smaller value: 5 (Figure 2(i)).

Then we mark E as visited and set D as the current node. Because node E

doesn't have any non-visited so we don't need to check anything. We mark it as visited and stop the algorithm (Figure 2(j)).

2.2. Shuttles Calling Algorithm

About shuttles calling algorithm, which decides the case of which shuttle should get to start-node, will be almost based on the Dijkstra algorithm. Firstly, we use a Dijkstra algorithm to calculate the distances from each shuttle to start-node to find out the shortest distance to decide which shuttle should be called. If that shuttle is in "busy" state, there would be some situations we need to consider reducing the operating time. These situations will be presented as below:

Situation 1: There is a "busy" shuttle on start-node. In this situation, the best way to solve this problem is to wait for the shuttle to complete its task and then it will become the chosen to transport goods to end-node.

Situation 2: There is another shuttle which is farther in the distance but we can save more time. In this situation, because the velocity of every shuttle in this system is constant and we have the distance of each node, we can compute the time that a shuttle will reach the start-node. If there are no obstacles on the way that one shuttle gets to start-node, we will compare the total time (time to finish his old task and time to get to start-node) with respect to the condition of which shuttles have the smaller sum time will become the one to be chosen. For more detail, there is an example of this situation will be presented in Figure 3 below.

In Figure 3, we suppose that node 0 is the start-node and there are no shuttles on these nodes 0, 1 and 2. Therefore, shuttle 1 and shuttle 2 are two shuttles which can reach node 0 without obstacles on their ways. As we can see, the shuttle 2 has the shortest path to node 0. However, this shuttle is in "busy" state and it needs ten minutes to finish its old task. Shuttle 1 is in "busy" state too, and it requires three minutes to finish its old task. Moreover, the velocity of all shuttles is equal and the distance between the node and the node is supposed to be the same. Therefore, we can assume the time that one shuttle need to move from one node to the next node is one minute. Consequently, in this example the total time of shuttle 2 to reach to start-node (node 0) is 10 + 1 = 11 minutes and that of shuttle 1 is 3 + 3 = 6 minutes. According to the two total time values, shuttle 1 will become the chosen one and will be called to start-node.

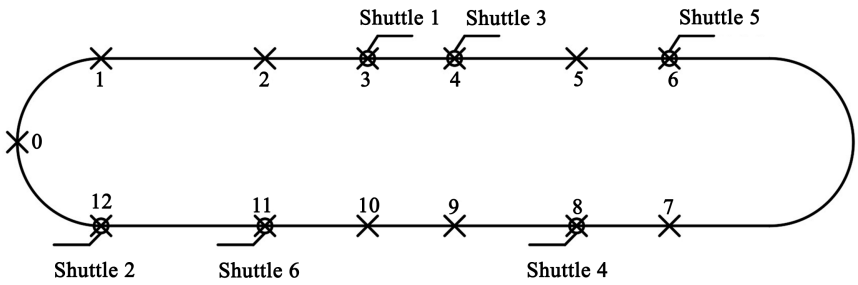

Figure 3. Illustration for situation 2 in shuttles calling algorithm.

2.3. Shuttles Communicating Algorithm

About shuttles communicating algorithm, we design an interactive algorithm which helps one shuttle to impact another one and become a loop of impacts. Each step of the algorithm is described as below steps:

Step 1: Get the positions of the shuttles, start the algorithm which the chosen shuttle in shuttle calling algorithm and create a "count" variable to break the loop of the algorithm.

Step 2: Apply the Dijkstra algorithm for the chosen shuttle and send signals to the shuttle (called shuttle A) which lies on the path and has the smallest distance to the start position of the chosen shuttle.

Step 3: Repeat Step 2 with shuttle A instead of the chosen shuttle.

Step 4: Until no shuttle is found on the path of the previous shuttle or "count" variable is satisfied the condition which prevent the last shuttle takes an effect on the first shuttle that could make the shuttles communicating algorithm becomes an infinity loop, the end-nodes of all shuttles are updated

Step 5: After all the shuttles arrive at their end-nodes, the server allows shuttles to move to their new positions and stops the algorithm.

To clarify this algorithm, we will use the example in Figure 3 which describes the shuttle calling algorithm in detail. In this example, the start-node is node 0 and the end-node is node 6, and the shuttle which is called to start-node is shuttle 1 like we have performed. By applying Djikstra algorithm, the path to go from node 0 to node 6 is: node 0 - node 1 - node 2 - node 3 - node 4 - node 5 - node 6.

On the way to node 6, shuttle 1 meets shuttle 3 first so it asks shuttle 3 to move to the node after the end-node (node 7) by the Web Server in Figure 4. After that, Dijkstra algorithm is applied for shuttle 3 which the path is: node 4 - node 5 - node 6 - node 7.

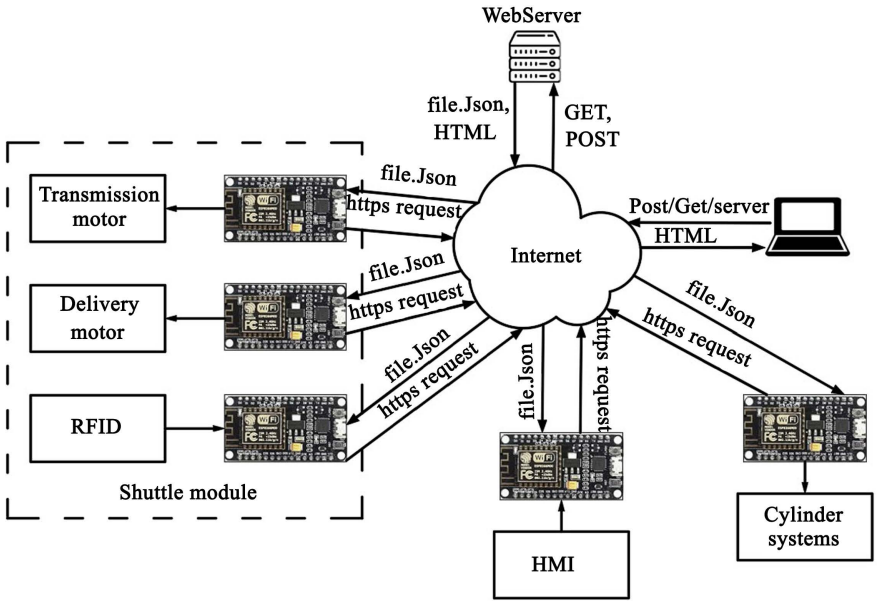

Figure 4. The conversation structure among the shuttles.

On the way to node 7, shuttle 3 meets shuttle 5 at first and it asks shuttle 5 to move to node 8. On the way to node 8 of shuttle 5, there is shuttle 4, which has been already on node 8. Therefore, shuttle 4 is asked to get to node 9 which the path is: node 8 - node 9.

On the way to node 9 of shuttle 4, there is no shuttle on its path. At this time, the end-node of each shuttle is updated, the shuttles communication algorithm is stopped and all the shuttles which have been impacted by the algorithm start to move to their end-nodes and the rest of them which has not been impacted by the algorithm hold their positions. Figure 5 presents each step of the example.

In Figure 4, we can see that the data transmission between ESP with the server is based on request and response of HTTP protocol. In HTTP model, the Web

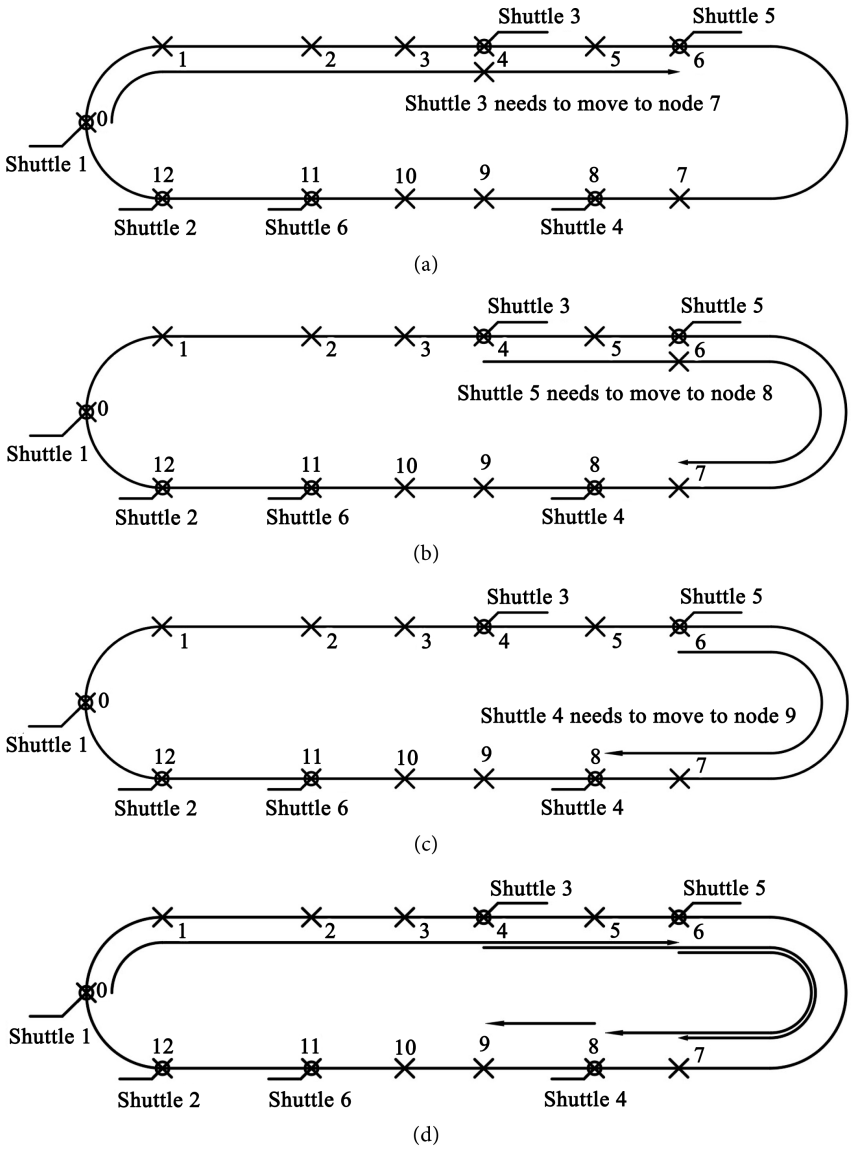

Figure 5. Illustration for a shuttle communicating algorithm: (a) Step 1: Shuttle 1 puts an impact on shuttle 3; (b) Step 2: Shuttle 3 puts an impact on shuttle 5; (c) Step 3: Shuttle 5 puts an impact on shuttle 4; (d) Step 4: Stop the loop and start the moves.

server is also the TCP Server, opening the default port for HTTP services which is TCP 80, ready to wait for connection requests from clients. The clients will initiate a TCP connection through this port, after the Webserver accepts the connection; the client will send an HTTP message called HTTP request to the Server on the newly established TCP connection. The server will respond with another HTTP response. This message will contain the requested Web page content (written in HTML language). With the characteristics as mentioned briefly above, the work to use the Web server for data collection and control can be easily realized by embedding sensor data fields in HTTP requests and responses.

GET and POST are the two methods of the HTTP protocol, both sending data to the server for processing after the user enters information in the form and submits it. Before sending information, it is encoded using a scheme called URL encoding. This scheme is name/value pairs combined with "=" symbol and different symbol separated by "&". Ex: a = value 1 & b = value 2 & c = value 3.

2.4. Control Algorithm of the System

The control algorithm of the whole system is introduced in **Figure 6**.

Figure 6 introduces control algorithm of the whole system. The 4 sub-programs as "choose car to get to start-node", "receive good", "check the route" and "change floor". The "choose car to get to start-node" presented the shuttles calling algorithm, which determines which shuttle will move to start-node. The "receive good" takes information from the user to make an impact on cylinder system if a user wants to deliver goods at floor 1 or floor 2. The "check the route" will update the destination of each shuttle if they lie on the route of the others. The "change floor" makes an impact on the cylinder systems if the user wants to receive goods at another stage that they deliver.

3. The Controller Design of Cylinder Module

There are 2 types of cylinder, which are popularly used nowadays: pneumatic cylinder and hydraulic cylinder. However, pneumatic cylinder is by far better than the other one because it is quieter, cleaner and does not require large amounts of space for fluid storage. The reason why we need a cylinder system right here is it helps to transport goods from the station with the shuttle as well as exchange parcels between two stages. There are other choices such as using the motor combined with the winch system or pulley and toothed belt system. However, they have some drawbacks like occupying lots of space to set up and pulley and toothed belt system is very weak in vertical mounting.

About the controller design of cylinder module: When the server sends signals to the controller by a Wi-Fi module, the controller sends back a signal to make sure that the communication is set up and starts to impact on the cylinders to receive or deliver packages.

In this situation, there are four situations that the cylinder system needs to handle: 1) Receiving packages on floor 1, 2) receiving packages on floor 2, 3)

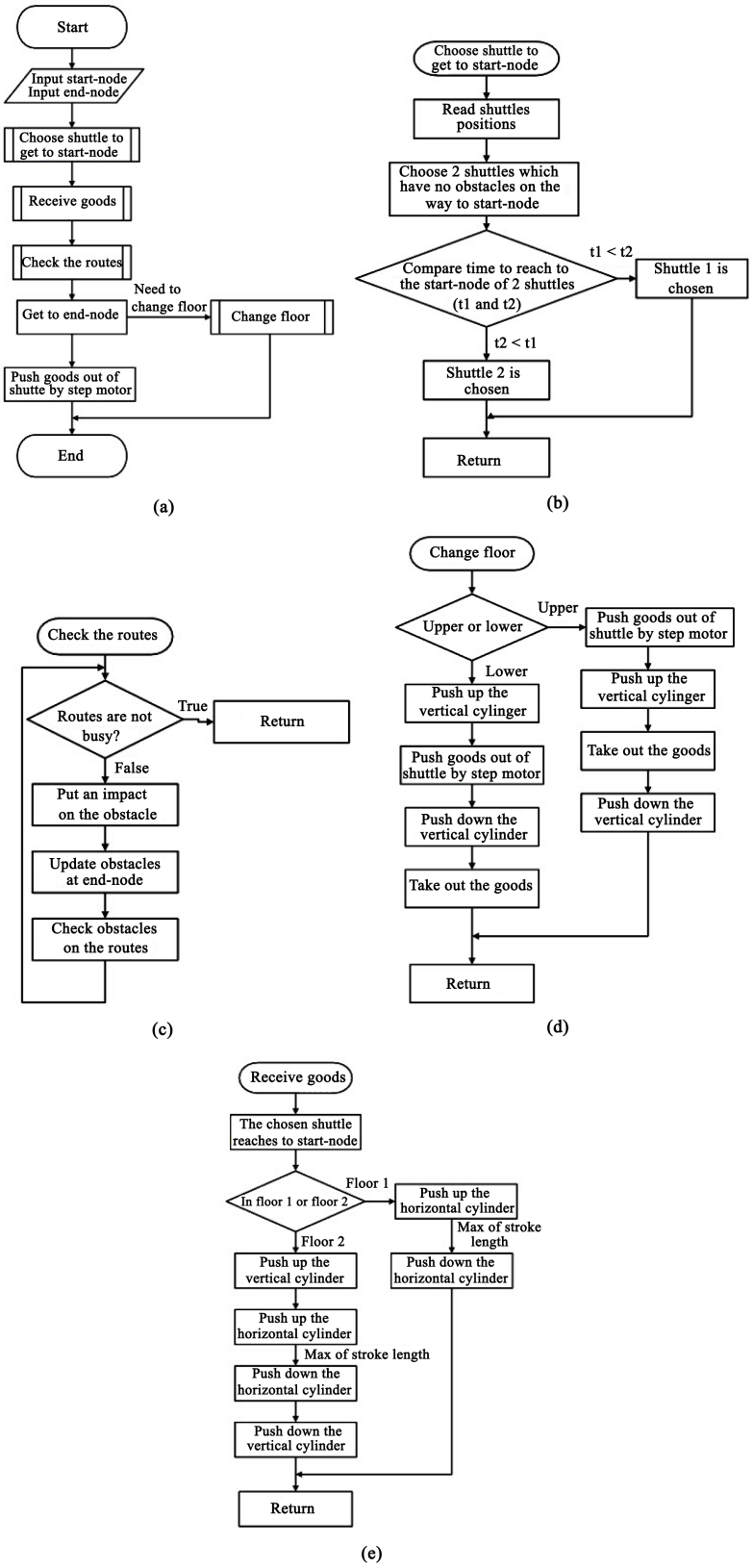

Figure 6. Control algorithm of the system: (a) Main algorithm; (b) "Choose shuttle to get to start-node" algorithm; (c) "Check the route" algorithm; (d) "Change floor" algorithm; (e) "Receive goods" algorithm.

delivering packages on floor 1, 4) delivering packages on floor 2. For each situation, the server sends a variable k which has different values for each case of the controller of the cylinder system. Corresponding to each value of variable k, the microprocessor has a different effect on the pneumatic valve system. The handling details for each case of the pneumatic valves are shown in Figure 7.

In Figure 7(a), server sends variable $k = 0$ to require cylinder system to deliver packages on floor 1. Therefore, the only horizontal cylinder is impacted. At this time, the controller will send a digital signal (1S1) to change the state of valve 1 and horizontal cylinder starts to push the packages into the shuttle, when the horizontal cylinder meets its max length, a signal (1S2) is sent back to valve 1 to change its state, the horizontal cylinder comes back to its old shape and the "delivering packages on floor 1" process ends.

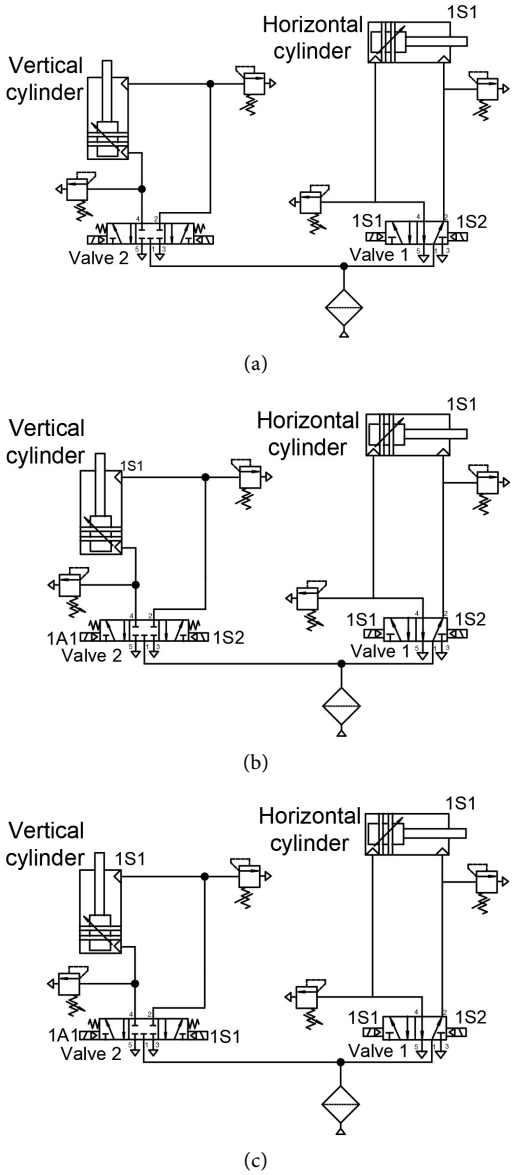

Figure 7. Control schematics of cylinder system: (a) $k = 0$; (b) $k = 1$; (c) $k = 2$.

In Figure 7(b), the server sends variable $k = 1$ to require cylinder system to deliver packages on floor 2. Therefore, the controller sends a digital signal (1A1) to change the state of valve 2 and vertical cylinder starts to lift the packages. Until it reaches its max stroke length, a signal 1S1 is sent to valve 1 to change its state and the horizontal cylinder begins to push the packages. When the horizontal cylinder reaches its max stroke length, a signal (1S2) is sent back to both valves and both cylinders recover their old shape. Finally, the "delivering packages on floor 2" process ends.

In Figure 7(c), the server sends variable $k = 2$ to require cylinder system to receive packages on floor 2. Therefore, the only vertical cylinder is impacted. At this time, microchip sends a digital signal (1A1) to change the state of valve 2 and vertical cylinder starts to lift the packages, when the vertical cylinder meets its max length, a signal (1S1) is sent back to valve 2 to change its state, the vertical cylinder comes back to its old shape and the "receiving packages on floor 2" process ends. With "receiving packages on floor 1" case, the server sends variable $k = 3$ to inform the cylinder system not to do anything. Figure 8 shows the control system of cylinder module.

4. The Controller Design of Shuttles Module

About the controller design of shuttles module, we use a microcontroller to control the direction of the shuttle's movement. When the server sends signals to the microcontroller by a Wi-Fi module, the microcontroller sends back an acknowledge signal to make sure that the communication is set up and starts to operate the electric motors. After the shuttles communication algorithm ends, the end-node of each shuttle is sent to its microcontroller by a server and the shuttles start to move to their new positions.

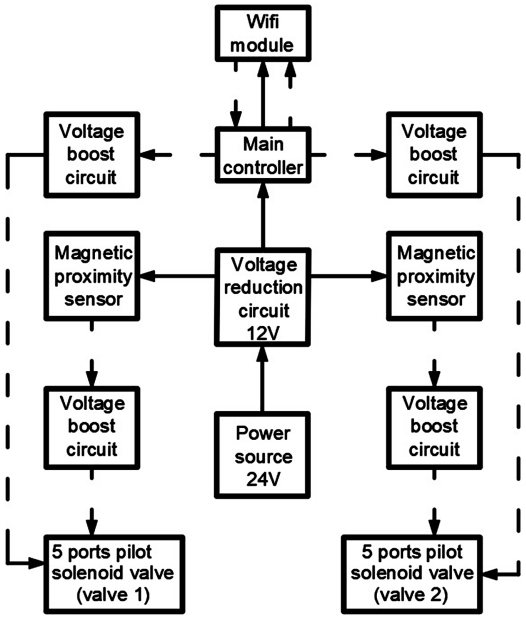

Figure 8. Electric block diagram of cylinder module.

The control unit is the heart of the system, where shuttles will be functioning on commands given by this unit. The block diagram (Figure 9) shows the control system which consists of the microcontroller and the Wi-Fi module which acts as a communication between the shuttle unit and the user. The data are transmitted to shuttle through this device which is connected to control system. The brain of the system—the microcontroller is responsible for decision making. For our smaller size, multi-station transportation system built in our BK-Recme BioMech Lab at Ho Chi Minh City University of Technology (VNU-HCM), the microcontroller used in this shuttle is ATmega2560. The motor drive TB6560, which is the most commonly used type today for step motor, the circuit can control a 2-phase stepper motor (10 - 35 VDC) with a maximum capacity of 3A. The TB6560 stepper motor control circuit is used to control stepper motors, CNC machine or precision mechanical system.

The RFID reader unit is responsible for the vehicle to move along the path, detects the marker and source point. It extracts data from its working environment and gives it as input to control unit. Like barcode technology, RFID scanner recognizes locations and identification of tagged item—but instead of reading laser light reflections from printed barcode labels, it leverages low-power radio frequencies to collect and store data. In a warehouse or distribution center, this technology is used to automate data collection. The transceiver reads radio frequencies and transmits them to an RFID tag. The identification information is then transmitted from a tiny computer chip embedded in the tag and broadcasted to the RFID reader.

The communication plays an important role in the efficient working of the vehicle. This is done initially when the user making a call to the mobile in the control system and the destinations are sent through Wi-Fi module from the server to each shuttle. There are two potential ways to set up the communication between the user and the vehicles:

- The input is given by an operator who is at the source or start point as per the requirement and the bot moves as per the signal to the corresponding stations.

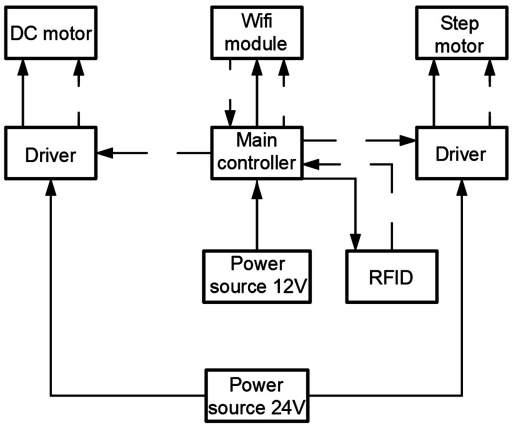

Figure 9. Electric block diagram of shuttle module.

- If there are multiple operators, the user at each station works as an operator by giving input to control unit from their respective stations as per requirement.

In this research, the first way is the chosen way to communicate between users and vehicles to save costs. By the way, the number of vehicles as well as the number of stations is not too large, that is enough for a modern computer today can handle. The Wi-Fi module we use is ESP8266, which is often used for IOTs applications. This module has preloaded firmware to help users communicate with Wi-Fi very easily via AT script through UART communication. ESP8266 is in consideration with its salient features including 802.11 b/g/n support, 2.4 GHz Wi-Fi, support WPA/WPA2, standard UART serial communication baud rate up to 115,200, support security standards such as: OPEN, WEP, WPA_PSK, WPA2_PSK, WPA_WPA2_PSK, support both TCP and UDP communication.

5. Simulation Results

In this simulation, there are some parameters that are needed to be set up as: start-node, end-node, path matrix and start position of each shuttle. The simulations are carried out with two different sets of input numbers. These two sets of input numbers are chosen randomly and there is no rule in choosing them, these parameters are described in Table 1 and Table 2. The reason that we apply two random input numbers is to check the correctness, feasibility and stability of our proposed algorithm. The simulation is conducted in Dev-C++ software. The row of the matrix is the route map of one node to the other nodes. For example, the second row of the table is represented in the route map of node 0. If the value is 0, there is no way to go straight from node 0 to that node. If the value is 1, there is a way to go straight from node 0 to that node and the distance between them is 1 (we can change the distance that we want), which means there is a way to go straight from node 0 to node 1 and from node 0 to node 12 in this example. However, it is just a one-way route, there is a way to go straight from node 0 to node 1 doesn't mean there is a way to go straight from node 1 to node 0. Similarly, we have the route map of the others.

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Table 1. The input parameters used in first simulation.

Parameters	Value
Start-node	1
End-node	8
Start position of shuttle 1	3 (1) (3)
Start position of shuttle 2	11 (1) (1)
Start position of shuttle 3	4 (1) (4)
Start position of shuttle 4	5 (0)
Start position of shuttle 5	6 (1) (5)
Start position of shuttle 6	7 (1) (6)

*The values behind the start position number of each shuttle is its state ("ready" = 0; "busy" = 1) and its time needed to finish the old task. The other values are the number of the nodes, in this simulation, the user wants the system to transport goods from node 1 to node 8 and the position of six shuttles from 1 to 6 are 3; 11; 4; 5; 6; 7 respectively.

Table 2. The input parameters used in second simulation.

Parameters	Value
Start-node	4
End-node	11
Start position of shuttle 1	1 (1) (3)
Start position of shuttle 2	5 (0)
Start position of shuttle 3	8 (1) (4)
Start position of shuttle 4	12 (0)
Start position of shuttle 5	9 (1) (5)
Start position of shuttle 6	7 (1) (6)

*The values behind the start position number of each shuttle is its state ("ready" = 0; "busy" = 1) and its time needed to finish the old task. The other values are the number of the nodes, in this simulation, the user wants the system to transport goods from node 1 to node 8 and the position of six shuttles from 1 to 6 are 1; 5; 8; 12; 9; 7 respectively.

The result of the first simulation is presented in **Figure 10**. In this simulation, shuttle 1 and shuttle 2 are two shuttles that can reach the start-node without any obstacles so they become two candidates to be chosen to get to start-node. Because the distance from the start position of shuttle 1 to start-node is 2 and the distance from the start position of shuttle 2 to start-node is 3, assuming that every shuttle takes 1 minute to travel at 1 unit of distance. To that end, shuttle 1 takes 2 minutes to get to start-node and shuttle 2 takes 3 minutes to get to start-node. However, the time that is needed to finish the old task of shuttle 1 and shuttle 2 is 3 minutes and 1 minute respectively. In consequence, the total time of shuttle 1 to reach to start-node is 5 minutes and the total time of shuttle 2 to reach start-node is 4 minutes. As a result, shuttle 2 wastes less time to reach start-node than shuttle 1 and become the chosen to move to start-node. We don't care about the time needed to finish the old task of the other shuttles

```
Shuttle 2 moves to start-node
Route shuttle 1:
Hold position
Route shuttle 2:
Route = 8 → 9 → 10 → 11 → 12 → 0 → 1
Route shuttle 3:
Hold position
Route shuttle 4:
Hold position
Route shuttle 5:
Hold position
Route shuttle 6:
Hold position
```

Figure 10. The simulation routing result 1.

because we suppose that each car only requires 10 minutes to finish its delivery or receiving process which means the maximum "busy" time of each shuttle is 10 minutes, so when shuttle 2 gets to start-node and finish its receiving process, that requires about 14 minutes, therefore the other shuttles are already in its "ready" state. In practical situations, the value of this 10 minutes will be changes based on the structure and the layout of the system in factories. On the way to move to end-node of shuttle 2, there aren't any shuttles on the path, so we have a result in **Figure 10**. Respectively, we have the results of second and third simulation in **Figure 11**.

With the parameters in **Table 1**, we have the travel route of each shuttle. At first, shuttle 2 gets to start-node (node 1) to receive goods and get to end-node (node 8). By applying the Dijkstra algorithm, we have the travel route of shuttle 2. Because there aren't any shuttles on shuttle 2's path so they remain their positions.

The result of the first simulation is introduced in **Figure 11**. In this simulation, shuttle 1 and shuttle 2 are two shuttles which can reach the start-node without any obstacles so they become two candidates to be the chosen to get to start-node. Because the distance from the start position of shuttle 1 to start-node is 3 and the distance from the start position of shuttle 2 to start-node is 1, assuming that every shuttle takes 1 minute to travel 1 unit of distance. To that end, shuttle 1 takes 3 minutes to get to start-node and shuttle 2 takes 1 minute to get to start-node. However, the time that is needed to finish the old task of shuttle 1 and shuttle 2 is 3 minutes and 0 minutes respectively. In consequence, the total time of shuttle 1 to reach start-node is 4 minutes and the total time of shuttle 2 to reach start-node is 1 minute. As a result, shuttle 2 takes less time to reach start-node than shuttle 1 and becomes the chosen to move to start-node. We don't care about the time needed to finish the old task of the other shuttles because we suppose that each car only requires 10 minutes to finish its delivery or receiving process which means the maximum "busy" time of each shuttle is 10 minutes, so when shuttle 2 gets to start-node and finish its receiving process, that requires about 11 minutes, therefore the other shuttles are already in its "ready" state. We have a result of travel routes of every shuttle in **Figure 11**.

```
Shuttle 2 moves to start-node
Route shuttle 1:
Route = 1 → 0 → 12 → 11 → 10
Route shuttle 2:
Route = 4 → 3 → 2 → 1 → 0 → 12 → 11
Route shuttle 3:
Route = 8 → 7
Route shuttle 4:
Route = 12 → 11 → 10 → 9
Route shuttle 5:
Route = 9 → 8
Route shuttle 6:
Route = 7 → 6
```

Figure 11. The simulation routing result 2.

With the parameters in Table 2, we have the travel route of each shuttle. At first, shuttle 2 gets to start-node (node 1) to receive goods and get to end-node (node 8). By applying the Dijkstra algorithm, we have the travel route of shuttle 2 is 4 → 3 → 2 → 1 → 0 → 12 → 11. Because shuttle 1 is on shuttle 2's path and nearest the start point of shuttle 2 so it needs to move out of shuttle 2's route (route of shuttle 1: 1 → 0 → 12 → 11 → 10). On the route of shuttle 1, there is shuttle 4, which has the nearest distance to start point of shuttle 1, we have a route of shuttle 4: 1 → 12 → 11 → 10 → 9. On the route of shuttle 4, there is shuttle 5, which has the nearest distance to start point of shuttle 4, we have a route of shuttle 5: 9 → 8. On the route of shuttle 5, there is shuttle 3, which has the nearest distance to start point of shuttle 5, we have a route of shuttle 3: 8 → 7. On the route of shuttle 3, there is shuttle 6, which has the nearest distance to start point of shuttle 3, we have a route of shuttle 5: 7 → 6.

6. Conclusion

By carrying out some simulation results with two series of different numbers in the previous section, the results obtained are very positive which shows the correctness, the effectiveness of the proposed control algorithm for our multi-station transportation system. No undesirable errors occurred during the program execution and the results are exactly as intended. However, the availability of the algorithm at this time also has one minor disadvantage as it cannot handle calling a shuttle at many stations at the same time. For our expectation in the next research, we will develop the algorithm that can handle calling multiple vehicles at the same time, build a user-friendly widget allowing users to enter data as well as observe the entire process of vehicle operation and proceed to install the actual product. More and more simulations and experiments will be carried out in the next steps to develop our proposed control algorithm and also check the agreement between the simulation results and the experimental results.

Acknowledgements

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number B2021-20-04.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Tiwari, P. and Tiwari, M.K. (2014) Knowledge Driven Sensor Placement in Multi-Station Manufacturing Process. *International Journal of Intelligent Engineering Informatics*, **2**, 118-138. https://doi.org/10.1504/IJIEI.2014.066202

[2] Hitchcock, F.L. (1941) The Distribution of a Product from Several Sources to Numerous Locations. *Journal of Mathematics and Physics*, **20**, 224-230. https://doi.org/10.1002/sapm1941201224

[3] Ashayeri, J. and Gelders, L.F. (1985) Warehouse Design Optimization. *European Journal of Operational Research*, **21**, 285-294. https://doi.org/10.1016/0377-2217(85)90149-3

[4] Visschers, J., Adan, I. and Weiss, G. (2011) A Product Form Solution to a System with Multi-Type Jobs and Multi-Type Servers. *Queueing Systems*, **70**, 269-298.

[5] Mather, W.H., Hasty, J., Tsimring, L.S. and Williams, R.J. (2011) Factorized Time-Dependent Distributions for Certain Multi-Class Queueing Networks and an Application to Enzymatic Processing Networks. *Queueing Systems*, **69**, 313-328. https://doi.org/10.1007/s11134-011-9216-3

[6] Kim, W.-S. and Morrison, J.R. (2010) On Equilibrium Probabilities in a Class of Two Station Closed Queueing Network. *ICCAS* 2010: *Proceeding of the International Conference and Control, Automation and System*, Gyeonggi-do, 27-30 October 2010, 237-242. https://doi.org/10.1109/ICCAS.2010.5670326

[7] Jung, S. and Morrison, J.R. (2010) Closed Form Solutions for the Equilibrium Probability Distribution in the Close Lu-Kumar Network under Two Buffer Priority Policies. *Proceeding of the* 8*th IEEE International Conference on Control and Automation*, Xiamen, 9-11 June 2010, 1488-1495. https://doi.org/10.1109/ICCA.2010.5524336

[8] Harrison, J.M. (1988) Brownian Models of Queueing Networks with Heterogeneous Costumer Populations. In: Fleming, W. and Lions, P.-L., Eds., *Stochastic Differential Systems, Stochastic Control Theory and Applications*, Volume 10, Springer, Berlin, 147-186. https://doi.org/10.1007/978-1-4613-8762-6_11

[9] Adan, I. and Weiss, G. (2005) A Two Node Jackson Network with Infinite Supply of Work. *Probability in the Engineering and Informational Sciences*, **19**, 191-212. https://doi.org/10.1017/S0269964805050102

[10] Adan, I. and Weiss, G. (2006) Analysis of a Simple Markovian Re-Entrant Line with Infinite Supply of Work under the LBFS Policy. *Queueing Systems*, **54**, 169-183. https://doi.org/10.1007/s11134-006-0065-4

[11] Nazarathy, Y. and Weiss, G. (2008) Near Optimal Control of Queueing Networks over a Finite Time Horizon. *Annals of Operations Research*, **170**, 223-249. https://doi.org/10.1007/s10479-008-0443-x

[12] Weiss, G. (2004) Stability of a Simple Re-Entrant Line with Infinite Supply of Work—The Case of Exponential Processing Times. *Journal Operations Research Society of Japan*, **47**, 304-313. https://doi.org/10.15807/jorsj.47.304

[13] Weiss, G. (2005) Jackson Networks with Unlimited Supply of Work. *Journal of Ap-*

plied Probability, **42**, 879-882. https://doi.org/10.1239/jap/1127322036

[14] Nazarathy, Y. (2008) On Control of Queueing Networks and the Asymptotic Variance Rate of Outputs. PhD Thesis, University of Haifa, Haifa.