Abstract—Swarm robot systems are widely applied in the warehouse logistics, which effectively improve logistics efficiency. Task allocation strategy is the core problem for swarm robot systems. Therefore, testifying the approach has a practical significance. The current study confirms the strategy using general robotics tools (GAZEBO). However, for the warehousing logistics, a few special factors need to be considered: man-robot coexistence in working environment, the energy consumption of robots and collision avoidance among robots. We propose a novel swarm robot simulation platform, called MultiBots, based on multi-agent pathfinding (MAPF) method and collision avoidance strategy, which can correctly verify the effectiveness of task allocation. Besides, we design the charging process to supplement the energy consumption of robots. The experimental results show that the MultiBots satisfies the requirements of the warehouse logistics scenarios. The MultiBots can be applied in testifying the efficiency of task allocation strategy for logistics systems.

I. INTRODUCTION

The development of Internet business economy has put forward new demands to the logistics. Therefore, the multi-agent system applied in logistics, such as a Kiva warehouse logistics system [1]. Also, the application of task allocation strategies in logistics can reduce the logistics cost [2] [3]. However, the lack of a suitable emulator makes the strategy unverified. Previous research in multi-agent simulator included TeamBots®, SWARM®, and NetLogo®, but these simulators are not suitable for the warehouse logistics scenarios. In this paper, we design a simulation platform named MultiBots, which is a Python-based logistics system simulator.

The MultiBots mainly considers three factors: the multi-agent pathfinding (MAPF) method, the collision avoidance strategy, and the charging process. Because the application scenarios are in the warehouse, robots share their work spaces with humans. Therefore, the resulting predictability of their motion is necessary for the safety of the humans [4]. We adopt a scheme for MAPF, called Highways [5] [6], based on the ideas behind experience graphs [7] [8]. Because the MultiBots is a multi-agent system, the collision avoidance is a signification issue [9]. Highways solves the head-to-head collisions among robots. And the collision waiting strategy deals with other situations. Considering the charging process, because the system energy consumption is an important factor in evaluating the task allocation strategy.

The remainder of this paper is organized as follows. Section II introduces the related work about the multi-agent emulator. Section III describes our MultiBots including application scenarios, the MAPF method, the collision avoidance strategy, and the charging process. Section IV shows the implementation and analyses experimental results. The last section concludes this paper and provides our future work.

II. RELATED WORK

Multi-agent is a hot issue in artificial intelligence and computational science. A simulation system is a valuable tool for studying multi-agent. Some simulation and experimental systems have already been developed. The Cellular Robotics System (CEBOT) can reconfigure itself to optimal structure depending on the purpose and environment [10]. ACTor-based Robots and Equipments Synthetic System (ACTRESS) is an autonomous and distributed robot system composed of multi robotic elements which are provided with functions to decide with understanding the target of tasks, recognizing surrounding communicate with any other components [11]. A swarm-bot is comprised of autonomous mobile robots called s-bot which can either cat independently or self-assemble into a swarm-bot by using their grippers [12]. MASON is discrete-event multi-agent simulation toolkit in Java [13]. And serve as the basis for a broad range of multi-agent simulation tasks ranging from swarm robotics to machine learning to social complexity environments.

Related work above are used as multi-agent simulation toolkits which do not target specific areas. For unmanned autonomous vehicles, [14] presents a hardware-in-the-loop development simulation framework for multi-vehicle autonomous systems. For mobile robot localization, [15] proposes a multi-agent system using the SPADE MAS platform to improve the location of mobile robots in dynamic scenarios. For traffic, [16] presents the concept of an integrated multi-agent simulation platform to support the development and validation

*This work was supported by the National Natural Science Foundation of China No. 61603376, awarded to Dr. Lujia Wang; and the Research Grant Council of Hong Kong SAR Government, China, under project No. 21202816 and No. 16212815, HKUST IGN16EG12, awarded to Prof. Ming Liu.

1https://www.cs.cmu.edu/~trb/TeamBots/
2http://www.swarm.org/wiki/Main_Page
3https://ccl.northwestern.edu/netlogo/
of autonomic cooperative car-to-car systems. For formation
control, [17] provides two algorithms to tackle the multi-
team formation problem. And two algorithms are evaluated
in RMASBench (a rescue multi-agent benchmarking platform
used in the RoboCup Rescue Simulation League).

For the warehouse logistics scenarios, several studies have
designed the simulation model from a different perspective:
[18] presents an agent-based simulation as a tool for decision-
making about automatic warehouses management. [19] pro-
toses towards a multi-agent logistics and commercial trans-
port model. [20] provides advanced approaches for multi-
robot coordination to resolve a fundamental problem which
is for robots to make individual decisions so to optimize
a system-wide objective function. However, the simulation
process based on ROS and GAZEBO is complex and does
not consider the energy consumption of the system. The
above emulators do not verify the logistics task allocation
strategy. The quality of the strategy decides the efficiency of
logistics which has the economic value [21]. To fill such a
gap, we design the MultiBots as Multi-Agent logistics system
simulation platform.

III. THE MULTIBOTS SYSTEM

A. The Warehouse Logistics scenarios

To achieve the logistics scenarios, we design a Multi-agent
system simulation platform divided into charging area and
working area. Simulator system uses a pixel as a unit length.
As shown in Fig.1(a), an 800*550 white rectangle stands
for the warehouse, which is the working area of the robot
loading and unloading goods. There are six available departure
zones, i.e., Fig.1(b). There are correspondingly six available
unloading zones, i.e., Fig.1(c). There are eighty 50*50 red
squares in the warehouse, divided into ten groups of 8 each,
which denotes the storage area of the goods. And they are
numbered from 0 to 79, as shown in Fig.2. When a red square
is filled with black, it denotes there are no goods. On the
contrary, the red means goods. Between the two groups, there
is a 50 width passageway, which allows circle robots with a
radius of 12.5 successfully passed. The straight line in the
circle represents the head of the robot, and there is a life bar
at the end of the robot. The three different filling colors in the
circle represent the three different status of the robots. Blue
means no load and the robot is on the way to pick up; Red
means carrying goods and the robot is on the way to unload.
Green means idle and the robot completes the task or goes
back to charge. Above the warehouse, there are six available
charging zones, i.e., Fig.1(d). The small red square located
left below the warehouse denotes the goods taken. MutilBots
parameters and their value are in the TABLE I.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warehouse size</td>
<td>800*550</td>
</tr>
<tr>
<td>Goods size</td>
<td>50*50</td>
</tr>
<tr>
<td>The number of goods</td>
<td>80</td>
</tr>
<tr>
<td>Radius of the robot</td>
<td>12.5</td>
</tr>
<tr>
<td>Width of passageways</td>
<td>50</td>
</tr>
</tbody>
</table>

B. Multi-agent Pathfinding Highways

Taking into account the application scenarios, in the ware-
house, we use the Highways method to guarantee human safety
and avoid head-to-head collisions among robots. Fig.2 shows
a directed graph from ‘A’ to ‘R’, which robots must move
along the narrow directed passageways between the storage
locations based on Highways.

Algorithm 1 presents the whole framework of the Highways
path planning procedure. First, we initialize the starting point
and end point of the robots, the direction among letters.
Second, we set the letter coordinate and their next letter of
Algorithm 1 Highways algorithm

1: Initialize the letter list, the StartLetter and GoalLetter
2: Initialize the path direction
3: Initialize the list holder to store temporary paths
4: for the LetterMark in the letter list do
5:   Set the LetterMark coordinates
6:   Set NextLetter of LetterMark
7: end for
8: Generate a list graph to store the NextLetter
9: Set the temporary path: temppath = [StartLetter] and holder = [temppath]
10: while holder ≠ empty do
11:   if temppath[LastLetter] = GoalLetter then
12:     return temppath
13:   end if
14:   for the NextLetter of the LastLetter do
15:     if the NextLetter is not in the temppath then
16:       Generate a newpath: newpath = temppath + NextLetter
17:       Add the newpath to the list holder
18:     end if
19:   end for
20: end while

arrival. Third, we take out the first path in the queue of the road and check that its last letter is the goal. Last, if the result is not the goal, we will generate a new path by adding next letter. Loop the above process until the checked result is the aim.

For example, if the robot starts from ‘A’, loading the goods at the storage location filled with red in Fig.2, to ‘M’ unloading. It can not choose the nearest method, which faces the direction of the arrow. The right way is ‘A→B→C→D→J→P→O’, loading at the storage location, and move along the ‘I→J→P→O→N→M’, unloading at the end ‘M’, as shown by the yellow solid line in Fig.2.

C. A collision avoidance strategy

Highways can avoid the head-to-head collisions among robots. However, other situations are leading to crashes, such as rear-ending during loading and unloading, side collision at the crossroads. We propose collision waiting method to solve the above problems. When an accident occurs, the active robot stops working, and the passive robot normally works. We determine whether the collision happens through the following two conditions: distance satisfies the equation \( \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq r_i + r_j \). \((x, y), r \) are respectively the coordinate and radius of the robots; angle satisfies the equation \( |\alpha - \beta| \leq \theta \). In Fig.3, we set Robot i as the active collision robot and Robot j as the passive collision robot. \( \alpha \) is the angle of Robot i. \( \beta \) is the angle between robot i and robot j. We call \( \gamma \) the collision angle, \( \gamma = |\alpha - \beta| \). \( \theta \) is set the threshold. In the case of satisfying the first condition, if \( \gamma \) is less than the \( \theta \), the collision happens. The set threshold smaller is, the probability of a collision is smaller. If the \( \theta \rightarrow 0^\circ \), only in the case of Fig.4(a) will be a collision. If the \( \theta \rightarrow 90^\circ \), in the case of Fig.4(b) will also be a collision.

D. The charging process

Fig.5 shows the logistics charging process including loading and unloading, accompanied by the energy consumption of robots. When the remaining energy is less than 35%, robots without goods directly return to the charging area to charge, and others firstly unload the goods. There are other charging
parameters in TABLE II. These parameters are set to ensure that the robot can return to the charging area when it needs to charge.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full power</td>
<td>10</td>
</tr>
<tr>
<td>Threshold power</td>
<td>3.5</td>
</tr>
<tr>
<td>Consuming rate</td>
<td>0.005</td>
</tr>
<tr>
<td>Charging rate</td>
<td>0.05</td>
</tr>
</tbody>
</table>

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, we describe the implementation of MultiBots and analyze experimental results. We provide three experimental indicators to access the performance of MultiBots:

- Task time $T$, it denotes the total time, which begins from the first robot into the warehouse to the last robot to leave off.
- The number of collisions $S$, it denotes the number of collisions happen among robots during the working.
- The number of charging $C$, it denotes the number of charging is during the working status of robots.

<table>
<thead>
<tr>
<th>Group number</th>
<th>The location of goods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group1</td>
<td>[11, 8, 39, 28, 53, 1, 69, 42, 79, 6]</td>
</tr>
<tr>
<td>Group2</td>
<td>[74, 50, 62, 15, 49, 34, 14, 72, 31, 37]</td>
</tr>
<tr>
<td>Group3</td>
<td>[68, 79, 56, 47, 48, 75, 23, 50, 76, 16]</td>
</tr>
<tr>
<td>Group4</td>
<td>[59, 6, 74, 2, 67, 49, 53, 28, 25, 79]</td>
</tr>
<tr>
<td>Group5</td>
<td>[14, 65, 41, 75, 22, 36, 54, 74, 9, 48]</td>
</tr>
</tbody>
</table>

1) Experiment Setup: To ensure the accuracy of the experiment, the research is conducted with six different number of robots, $N = 1, 2, 3, 4, 5, 6$, and each number has five diverse groups of goods, which the number of goods is ten. TABLE III shows the location of goods. The robots fetch in turn according to the order of the goods. In order to complete the experiment, other parameters are needed: the starting positions are [A, B, C, D, E, F] and the same unloading position is [M]; loading time and unloading time are 10 and 20 time steps respectively; the robot speed is 4 pixels per time step; the collision angle is $30^\circ$.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting positions</td>
<td>[A, B, C, D, E, F]</td>
</tr>
<tr>
<td>Unloading position</td>
<td>[M]</td>
</tr>
<tr>
<td>Loading time</td>
<td>10 time steps</td>
</tr>
<tr>
<td>Unloading time</td>
<td>20 time steps</td>
</tr>
<tr>
<td>Robot speed</td>
<td>4 pixels per time step</td>
</tr>
<tr>
<td>Collision angle</td>
<td>$30^\circ$</td>
</tr>
</tbody>
</table>

Fig. 6: The task time of six different number of robots

2) Experimental Results and Analysis: Fig.6, Fig.7, and Fig.8 respectively show three experimental indicators of six different number of robots. Part(a) is the specific distribution of indicators according to TABLE III. Part(b) is the mean of indicators of the five groups of goods.

Fig.6 shows the task time of six different number of robots. In general, for the different number of robots, the task time is completely different. As shown in Fig.6, the more the number of robots is, the less the task time spend. The same means less the amount of charging is, in Fig.7(b). The above description shows that the number of robots in the task has a fundamental impact on the task time and the number of charging. Fig.6(a) shows that the location of goods in the warehouse effects the task time. And this effect is based on the number of robots. For the number of charging is basically in line with this trend, in Fig.7(a). However, for the particular group, the impact of the location of goods is greater than the number of robots, e.g., group 2 in Fig.7(a).

As shown in Fig.8(b), the more the number of robots is, the higher the probability of collisions is. Fig.8(a) shows the distribution of crashes is complex, particularly when the number of robots is two and three. This is also confirmed from Fig.8(b). We conclude that when the difference in the number of robots is not great, the location of goods is the determinant of the number of collisions.

There is a link among the three experimental indicators. When the number of working robots is equal, the more the number of charging is, the longer the task time is. What is more, the impact of collisions on the task time is not visible. Because the setting of the parameters, loading time and unloading time, do not cause too much time to wait. Shortening the task time and reducing the number of collisions...
are the significant issues, which need to resolve in the future work.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we designed a multi-agent logistics simulation platform named MultiBots which can simulate the real warehouse logistics system and evaluate the task allocation strategies. In addition, we adopted the Highways as the multi-agent path planning method and the collision waiting strategy to guarantee the proper operation of the MultiBots. The experimental results demonstrate that the Highways and the collision waiting strategy are useful. Moreover, the task allocation strategies can be testified in the MultiBots.

In the future, we will work on efficient multi-agents pathfinding method for the MultiBots. At the same time, taking into account the distance, the task time, the number of collisions, and other factors, we will propose an efficient task allocation strategy to optimize the objective function $Cost_{total} = Cost(distance) + Cost(time) + Cost(collision)$, which will be testified in the MultiBots.

REFERENCES


