HKIE THE HONG KONG INSTITUTION OF ENGINEERS 香港工程師學會

Multi-agent pathfinding for deadlock handling in rotational movements

Frodo Kin Sun Chan, Yan Nei Law and Edmond Shiao Bun Lai
Hong Kong Industrial Artificial Intelligence and Robotics Centre, Hong Kong, People's Republic of China

ABSTRACT

Multi-agent pathfinding (MAPF) is a challenging problem especially in environments with a large number of agents and small map size. Although some recent studies revealed that the reinforcement learning approach is feasible to train the agents for resolving the deadlock problem at large scales in terms of the number of agents, the robots, which commonly appear in warehouses, are ignored in their design. As the robots in warehouses using zero-radius right-angle turns (e.g., KIVA robots) usually stay in the same location for a long time with the same rotation movements, the deadlock problem becomes more severe especially in dense areas. In this paper, a reinforcement and imitation learning algorithm with a deadlock avoidance scheme with reinforcement and imitation learning, a deadlock breaking scheme and a deadlock resolving scheme are proposed for resolving the deadlock problem by considering the configurations for a rotation-based warehouse environment including moving forward, rotating clockwise, and rotating anticlockwise only. In addition, a light model of the proposed algorithm is also presented to speed up the inference process. In the experimental results, the proposed algorithm and its light version are shown to be effective and useful for handling different scenarios of the deadlock problem.

KEYWORDS Multi-agent systems; robotics and automation; reinforcement learning; multi-agent pathfinding; zero-radius turning; right angle turning; mobile robots

CONTACT Frodo Kin Sun Chan frodochan@hkflair.org
Received 28 March 2022

1. Introduction

MAPF planning has been widely applied in different scenarios including but not limited to search and rescue (Baxter et al., 2007), airports (Morris et al., 2016), offices (Veloso et al., 2015), warehouses (Wurman et al., 2008), and video games (Silver, 2005), which have different configurations (e.g., rotation movement) for agents. The aim of MAPF is to allow agents to move from their starting location to their target location without colliding with others. When the number of agents and the map size increase, the computation problem for path finding becomes resource demanding and intractable.

Normally, holonomic robots have an action space including staying idle and moving in four cardinal directions. However, the action space of robots with zero-radius turning (ZRT) by right-angle is different. In this paper, we consider the agents with an action space defined as staying idle, moving forward, and rotating 90° clockwise or anticlockwise, which is like a warehouse's setting. This is the same setting as the real automated guided vehicle (AGV) system in the grid-type floor layout.

Based on the low cost, easy control, and simple design (Li et al., 2022), this kind of robot is commonly used in warehouses. Also, the configuration of moving forward, and rotating clockwise and anticlockwise is used in recent research (Hönig et al., 2019). In order to simulate the real settings of warehouses, the same configuration is adopted for this paper.

As many recent studies have been focused on the use of the action space of holonomic robots (Ferner et al., 2013; Sartoretti et al., 2019), the focus on the action space of other kinds of robots is relatively less. In the rotational settings of ZRT robots, the agents move left and right in two steps, i.e., rotation followed by moving forward. Also, if the agents move backwards, three steps (i.e., two rotations followed by moving forward) are applied. This will introduce an extra dimension to the searching space for the rotational setting. In addition, the inference speed is also a problem when a larger number of agents are used. Even though parallel programming is a way to handle this issue, the light version of the model is better for use in embedded devices. In this paper, the algorithm including a deadlock avoidance scheme with reinforcement and imitation learning, a deadlock breaking scheme, and a deadlock resolving scheme and the light version of the proposed model are proposed for tackling the deadlock in the rotational settings of ZRT robots and speed up the inference speed of the proposed model.

In our previous work (Chan et al., 2022), only deadlock avoidance and a breaking scheme are proposed for the deadlock issue. However, deadlock may still occur in a dense environment. Thus, an integrated system including deadlock avoidance, breaking, and resolving schemes and the speed-up version of deadlock avoidance are proposed to further improve the completeness of the targets by considering the real scenarios. For the deadlock resolving scheme, a novel rollback mechanism based on two mapping tables is proposed by considering the properties of rotational movement in this paper. Also, a

novel speed-up version is proposed to effectively reduce the inference time by introducing a new design of the structure of the filtering layers and recurrent layers in this paper. The experimental results in Section 4 show that the proposed algorithm (MAPF-rot*+) not only achieves comparable performance as MAPF-rot (Chan et al., 2022) in general, but also performs better than MAPF-rot in some cases. In addition, the speed-up version also shows a significant improvement (38.23% reduction) in the inference time.

In the following sections, a brief review of related works is given in Section 2, and the details of the proposed algorithm and its light version are explained in Section 3. Then, Section 4 shows the effectiveness and superiority of the proposed approaches based on the experimental results with different settings. Finally, conclusions are drawn in Section 5.

2. Related works

The MAPF solution is generally divided into coupled, decoupled, and dynamically decoupled methods. For the coupled method, the paths of all agents are included to search for the optimal solution. When there is a large number of agents, its complexity is very high and finding the solution becomes intractable. A typical example is the A* method, which involves all agents searching for the optimal solution by estimating the total costs based on the movements and the heuristic costs. The second method is the decoupled method which reduces the complexity by planning the paths individually for each agent, but the path adjustment is performed on all agents with proper cooperation. Thus, the optimality of the solution is not always guaranteed and can be suboptimal (Sanchez and Latombe, 2002). Some examples of existing planners are the velocity planner (Cui et al., 2011; Van et al., 2011) and priority planner (Cáp et al., 2013; Ma et al., 2016). The third method is the dynamically decoupled method which also plans the paths individually for each agent, but only partial agents are involved for path adjustment through cooperation, for example, the conflict-based planner (Sharon et al., 2012).

Apart from these branches, reinforcement learning is also suggested for MAPF (Foerster et al., 2016; Foerster et al., 2017; Gupta et al., 2017; Lowe et al., 2017). For these papers, the partially observable Markov decision process (POMDP), which allows agents to have a local map for observation without knowing the information of other agents outside the map, is often assumed. By using the reinforcement learning approach, a policy is learned from training for different agents. As there are many agents involved, different policies can be learned for different agents (Foerster et al., 2016; Foerster et al., 2017; Gupta et al., 2017; Lowe et al., 2017). In addition, Gupta et al. (2017) showed that learning one policy, which enables parameter sharing for homogeneous agents, is better than many policies, which is used in a centralised and concurrent approach. In their paper, Lowe et al. (2017) suggested applying the actor-critic approach for learning different policies for different agents. Their approach allows multiple critics, which receive information from all agents, and multiple actors, which have the same number of agents. When the number of agents increases, the complexity of their method becomes very high. Foerster et al. (2017) also used the actor-critic approach, but only one critic was used for estimating the value function. They also had the problem of generalisation to the case of a large number of agents because only two actors were involved in their design. Foerster et al. (2016) further considered information sharing for the learning of different policies, but such sharing still has problems of handling a large number of agents.

To cope with the large number of agents in a complicated environment, PRIMAL (Sartoretti et al., 2019) is applied for learning the policy through the actor-critic approach. In their approach, only one actor and critic are trained for multiple agents, which are homogeneous. By using this approach, the model training does not depend on the number of agents and is able to learn for a large number of agents. However, the training of reinforcement learning is still very challenging for a large model. Although the reinforcement learning method enhances the exploration of the model, it has a risk of falling in a local minimum. For example, Sartoretti et al. (2019) mentioned that the agents learned to achieve goals as fast as possible, but they did not cooperate with other agents for higher rewards. To overcome this problem of reinforcement learning, Sartoretti et al. (2019) also applied three methods including adding a penalty for enhancing the movement of other agents, applying imitation learning for training, and generating random environments in the training process for agents to learn different scenarios, especially the difficult cluttered one. On the other hand, the PRIMAL model only supports five actions including staying idle, moving up, moving down, moving left, and moving right, which may not be sufficient for agents to move in different settings. Furthermore, some similar work in other applications can be found in Wang et al. (2022, 2023).

3. Proposed algorithm

3.1. Configurations for a rotation-based environment

In this paper, simulations are performed in a rotation-based environment. The simulation environment is in a grid form, which contains some obstacles and agents, and each agent has its starting position and its goal.

As the rotation-based environment requires the agents to move in four cardinal directions, these directions can be represented by $o \times 90°$, where the orientation index $o \in \{0,1,2,3\}$. Note that the initial direction of the agents is

set as 0°. The actions taken by an agent in a non-rotation-based environment and a rotation-based environment are different and are illustrated in Figure 1. For the non-rotation-based environment, an agent has four neighbours of the current state (x, y), which is also the current coordinate of the agent on the 2D grid map (Figure 1(a)). However, the rotation-based environment has only three neighbours ($\boldsymbol{N}_{left}$, $\boldsymbol{N}_{right}$, and $\boldsymbol{N}_{forward}$) of the current state (x, y, o), where o is the orientation index. The neighbours $\boldsymbol{N}_{left}$, $\boldsymbol{N}_{right}$, and $\boldsymbol{N}_{forward}$ are (x, y, (o-1) *mod* 4), (x, y, (o+1) *mod* 4), and ($x'(o)$, $y'(o)$, o), respectively, where ($x'(o)$, $y'(o)$, o) is defined as follows:

$$(x'(o), y'(o), o) = \begin{cases} (x, y+1, 0) & if\ o = 0 \\ (x+1, y, 1) & if\ o = 1 \\ (x, y-1, 2) & if\ o = 2 \\ (x-1, y, 3) & if\ o = 3 \end{cases} \quad (1)$$

Equation (1) is formed to represent the state of $\boldsymbol{N}_{forward}$ when an agent faces in different directions (in terms of o). For example, when an agent (x, y, o) faces east (o=1), $\boldsymbol{N}_{forward}$ becomes (x+1, y, 1). Thus, $\boldsymbol{N}_{forward}$ is set as ($x'(o)$, $y'(o)$, o) because its first two state values depend on o, where o represents the facing direction of the agent and its neighbour ($\boldsymbol{N}_{forward}$).

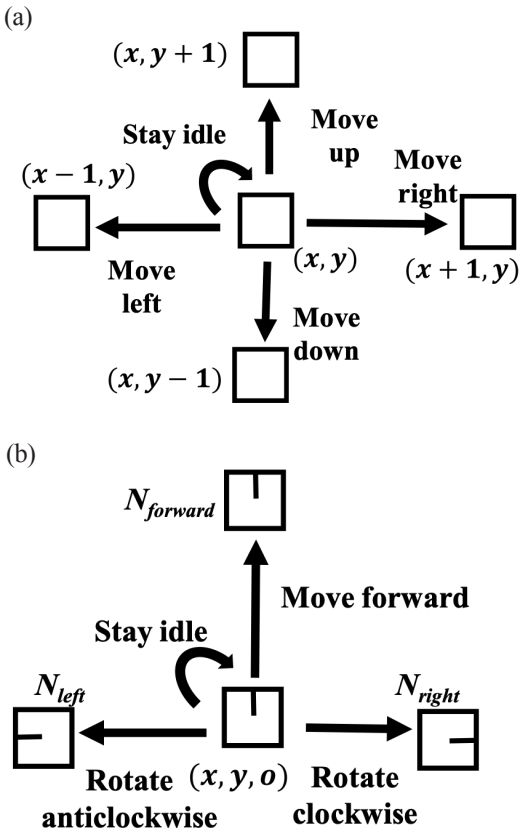


Figure 1. The neighbours of an agent in the (a) non-rotation-based environment and (b) rotation-based environment under different actions.

As the CBS (Sharon et al., 2012) and ODrM* (Ferner et al., 2013) methods are based on graph construction and normally perform searching in four cardinal directions only, their node neighbours and conflicts have to be modified to enable rotation-based movement. Specifically, their node neighbours are required to change to (x, y, o), and the agent movement is restricted by orientation index o. In addition, the vertex conflict and the swap conflict only consider the position (x, y) without including the orientation index o in the graph computation.

3.2. MAPF-rot

The main objective of the MAPF-rot (Chan et al., 2022) algorithm is to include the rotation-related factor (e.g., facing direction of the agents) in the model training and inference processes for homogeneous policy learning. To achieve this objective, the observation space is set to have six maps including obstacle locations, agents' positions, neighbours' goals, agents' goals, and vertical and horizontal maps. For the vertical map, an agent location on a vertical map (d_{NS}) is marked as 1/-1 when an agent faces north/south. Similarly, the agent location on a horizontal map (d_{EW}) contains 1/-1 to represent that the agent faces east/west. Then, these six maps are inputted into the model training and inference processes in 10x10 windows, which is a limited field of view (FOV) to filter out the information far beyond the agent locations and reduce the number of parameters in the model design. The overall structure of the training model is shown in Figure 2. In Figure 2, the input tensors are the concatenation of six maps and (g_x, g_y, m) is a goal vector, where g = (g_x, g_y, m) = (x_a - x_{goal}, y_a - y_{goal}), m = |g|, (x_a, y_a) is the current position of an agent and (x_{goal}, y_{goal}) is the goal of the agent. In the learning, the input vectors and the goal vector are generated based on the observations of the rotation-based environment. Then, a model is learned and its network includes filtering layers, which consist of several 2D convolution layers and maxpooling layers, and recurrent layers, which is a LSTM module in the implementation. The training is based on the A3C scheme, which makes the model robust to different situations by updating the gradients based on several local gradients in different environments. The outputs of the model are the value and the policy vector, which are used for estimating the discounted return and indicating the probability of different actions, for calculating the loss including the policy loss, the value loss, and the validity loss defined by PRIMAL (Sartoretti et al., 2019) in the training. And then, an action is selected with the maximum value in the policy vector. By inputting the agent action, the environment computes the rewards, which are set in Training rewards, and has the feedback to the learning model for moving to the next state. The initial learning rate and the decay factor of the Nadam optimiser are 2×10^{-5} and $\sqrt{1+5\times10^{-5}s}$, where s is the episode count. Scheme 1 shows the training procedure of MAPF-rot via reinforcement and imitation learning, and the details of the procedure can be found in the paper by Chan et al. (2022).

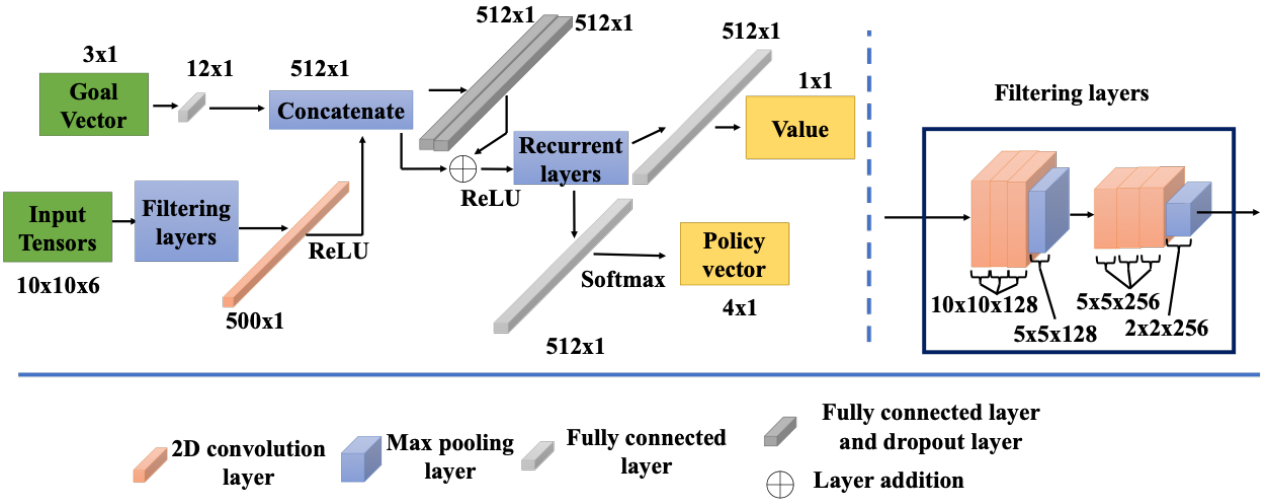


Figure 2. The network structure of the training model in MAPF-rot.

To set the training rewards for actions and collisions, the rewards of PRIMAL (Sartoretti et al., 2019) are used for reference. Based on their experience, the penalisation of agents slightly more for staying still than for moving can prevent poor convergence. Furthermore, the punishment on rotation is also added to stop an agent which keeps on rotating at the same location.

Training rewards

Reward Type	Reward
Staying idle	-0.5
Moving forward	-0.3
Rotating 90° clockwise/anticlockwise	-0.5
Collision	-2.0
Reaching a goal	0.0
Finish Episodes	+35

Scheme 1. Training procedure

```
function Training
create the global network of MAPF-rot for parameter updates
create the local network of MAPF-rot for each thread/agent
for each episode do
    generate a world with a random size and obstacle density
    update the parameters of the local network from the global network
    generate a random number x
    if x <0.5:
        # imitation learning
        obtain the ground truth by ODrM*
        calculate the gradient based on the imitation loss from the local network
    else:
        # reinforcement learning
        extract the actions obtained by the local network with current maps
        calculate the gradient based on extracted actions
        update the global network by using the gradient from the local network
```

3.3. Deadlock breaking scheme (MAPF)*

In a rotation-based environment, the deadlock problem becomes serious because the rotations cause the agents to stay longer in the same location and only the action of moving forward causes a location change. To further reduce the occurrences of deadlocks, the deadlock breaking scheme is also proposed in Scheme 2. The main idea is to generate a rarely triggered action to break the deadlock if the agents stay too long in the same position. To determine how long the agents stay, c_{ij} is the counter for each agent i and each action j. If an action j rarely occurs, its c_{ij} will be low, and there is a high probability p for generating this action. The setting of N is important for this scheme. If it is too small, the agent movement mainly relies on the action randomly generated by this scheme. On the other hand, a large value of N causes a high occurrence of deadlock. Thus, N is set as 5 empirically.

Scheme 2. Deadlock breaking scheme

1: function DeadlockBreaking
2: for each episode do
3: initialise row vector $\boldsymbol{c_i}$ = (0 0 0 0) for action counting of each agent i
4: for each agent i do
5: obtain a policy vector P_i and action A_i from the proposed algorithm
6: if agent i is staying in the same location:
7: $c_{ij} += 1$ # j is the index of action A_i
8: set $M_i = \sum_{j=0}^{3} c_{ij}$
9: if $M_i = N$: # N is a preset threshold for action generation
10: set $m_j = e^{\frac{p_{ij}}{1+c_{ij}/M_i}}$
11: the probability of each action $p = \frac{m_j}{\sum_{j=0}^{3} m_j}$
12: generate a random action A_i' based on the distribution p
13: reset c_i = (0 0 0 0)
14: return A_i'
15: else:
16: return A_i

3.4. Deadlock resolving scheme (MAPF+)

Although the proposed algorithms are effective to resolve the deadlocks, the deadlock may still have a chance to happen due to the dense environment. In a real situation, the workers may manually reset the agents to resolve the deadlocks, but the manual solution is very time consuming. In this paper, an automatic deadlock resolving scheme (Scheme 3), which includes a rotation-based rollback mechanism and conventional multiple agent pathfinding method, is also proposed to tackle the deadlock problem in the industrial environment.

Scheme 3. Deadlock resolving scheme

1: function DeadlockResolving
2: for each episode do
3: if timeout:
4: extract a rollback sequence by running the rollback mechanism
5: rollback with the number of timestep $\boldsymbol{T}$
6: run the rollback sequence from ODrM* in the current environment

In Scheme 3, the deadlock is assumed to exist among the agents when the timeout, which is preset before the system runs, occurs. Once deadlock happens, the rollback mechanism is used to reverse the actions of the active agents that have not reached the goal within $\boldsymbol{T}$ steps, where $\boldsymbol{T}$ = [(**timeout** + **1**)/**2**] and [•] is a flooring operator. As there are not as many active agents as there are agents at the beginning, a less dense environment is better to obtain a path finding solution for multiple agents. Thus, ODrM* is applied for searching for the path of the multiple agents in the simplified environment.

The rollback mechanism is different for different configurations (rotation-based movement, movement with a 4-connected path and 8-connected path). The rollback sequence for movement with a 4-connected path and 8-connected path can be formed by replacing the reverse action (e.g., Move Left is replaced with Move Right). However, the rollback mechanism for the rotation-based movement requires extra actions for reversing the original status. For example, the reverse motion of moving forward is {turning 90° clockwise, turning 90° clockwise, moving forward, turning 90° clockwise, turning 90° clockwise}. Scheme 4(a) shows the mapping table for the rollback action sequence of each action, and every action requires five actions in the rollback action. On the other hand, the rollback action sequence can be reduced to three steps for each action (six for two actions) when two successive actions are combined, and the combined sequence is shown in Scheme 4(b). When there is one action, the mapping in Scheme 4(a) is applied. When there are two actions, the mapping in Scheme 4(b) is used. When there is a rollback action following the two actions, the rollback action sequence of the action can also be reduced to have three elements by combining the idle (Action ⧖) and the rotation (Action ↺/↻). Note that there are only three possible cases of the last two elements in the tail based on our setting: {{⧖,⧖}, {↺,↺}, {↺, ⧖}}. If the rollback action is Action ⧖, ↺, ↻, the last two idle actions shown in Scheme 4(a) can be removed before merging with the tail. If the rollback action is Action →, three cases are treated as follows. For the case of {⧖,⧖}, the last two elements can be replaced with {↺,↺}, and the first two elements in Action → are removed. For the case {↺,↺}, the last two elements can be replaced with {⧖,⧖}, and the first two elements in Action → are removed. For the case of {↺, ⧖}, the only possible combination of the last three elements is {↺,↺, ⧖} (cf. Scheme 4(b)). For merging, the tail can be replaced with {⧖,⧖,⧖}, and the first two elements in Action F are removed.

The combined version of MAPF* and MAPF+ is MAPF*+. As the deadlock breaking scheme and the deadlock resolving scheme are the schemes for different situations, they can be applied together with MAPF-rot, i.e., MAPF-rot executes the deadlock breaking scheme before the timeout happens and the deadlock resolving scheme after the timeout happens.

3.5. Light version of MAPF-rot

In real scenarios, the inference speed is always the concern of industry. Although MAPF-rot is not slow in the inference step, the speed is still a problem when there are many agents for inference, for example, 1,000 agents. Parallel programming is a way to tackle the problem with a large number of agents. In another way, a light model of MAPF-rot is proposed for speeding up the process in this paper. To build a light model, two modifications

of the filtering layers and recurrent layers in Figure 3, are proposed. In MAPF-rot, the filtering layers are built of convolution layers, which have a high computation cost. Thus, two fire modules of SqueezeNet (Iandola et al., 2016), whose CNN architecture that has 50× fewer parameters than AlexNet and maintains the AlexNet-level accuracy on ImageNet, is proposed to replace the existing layers. The new filtering layers and fire module are shown in Figure 3 (a-b).

In addition, the recurrent module of MAPF-rot is changed from the LSTM to GRU module because the GRU module has fewer parameters than that of the LSTM module. Although it is yet to be verified that these modules have similar performance (Chung et al., 2014), the fewer parameters of GRU makes the computation more efficient. Furthermore, the size of the light version (209.18MB) has a 27.10% reduction from that of the original model (286.94MB).

There is a trade-off between the MAPF-rot algorithm and its light version. The advantage of MAPF-rot is that it can handle complicated cases more effectively. Since MAPF-rot has a more complicated model for path finding, it is more efficient in handling different environments. Besides, MAPF-rot generally performs better than its light version when the density of an obstacle increases. The disadvantage of MAPF-rot is that it is slower than its light version. As MAPF-rot has several convolution filters for learning, it requires much time to execute. Furthermore, MAPF-rot performs worse than its light version when the number of agents increases because its light version is sensitive to the dynamic objects by using a 1x1 convolution filter.

Scheme 4. The mapping scheme for (a) 1-step action and (b) 2-step action

(a)

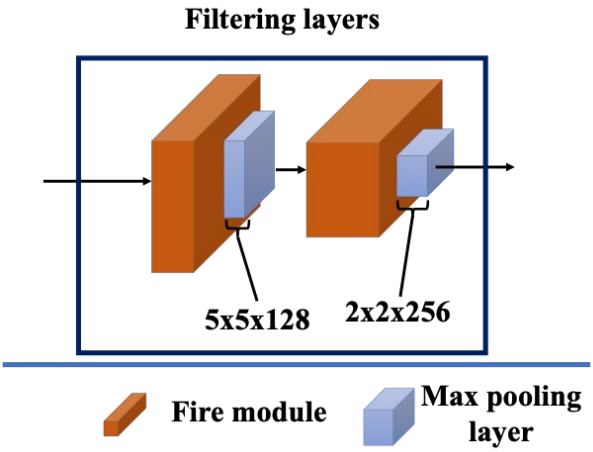
Action (1 step)	Rollback action sequence
⧖	{⧖,⧖,⧖,⧖,⧖}
→	{↺,↺,→,↺,↺}
↻	{↺,⧖,⧖,⧖,⧖}
↺	{↻,⧖,⧖,⧖,⧖}

(b)

Action (2 steps) (i.e. (*n*-1)th and *n*th step)	Rollback action sequence
⧖ ⧖	{⧖,⧖,⧖,⧖,⧖,⧖}
⧖ →	{↻,↻,→,↻,↻,⧖}
⧖ ↻	{↺,⧖,⧖,⧖,⧖,⧖}
⧖ ↺	{↻,⧖,⧖,⧖,⧖,⧖}
→ ⧖	{⧖,↻,↻,→,↻,↻}
→ →	{↻,↻,→,→,↻,↻}
→ ↻	{↻,⧖,⧖,→,↻,↻}
→ ↺	{↺,⧖,⧖,→,↻,↻}
↻ ⧖	{⧖,⧖,⧖,↺,⧖,⧖}
↻ →	{↻,↻,→,↻,⧖,⧖}
↻ ↻	{↺,⧖,⧖,↺,⧖,⧖}
↻ ↺	{⧖,⧖,⧖,⧖,⧖,⧖}
↺ ⧖	{⧖,⧖,⧖,↻,⧖,⧖}
↺ →	{↻,↻,→,↺,⧖,⧖}
↺ ↻	{⧖,⧖,⧖,⧖,⧖,⧖}
↺ ↺	{↻,⧖,⧖,↻,⧖,⧖}

⧖: Idle, →: Move forward, ↻: Rotate 90° clockwise, ↺: Rotate 90° anti-clockwise

(a)



(b)

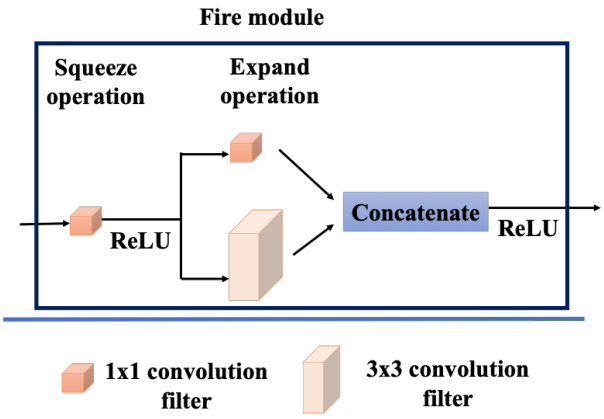


Figure 3. (a) The filtering layers of the fast version and (b) its fire module.

4. Experimental results

4.1. Testing dataset of rotational movement

In order to test the performance of the proposed algorithm, a dataset is created with different combinations

of obstacle locations, agent locations, and agent goals based on different map sizes {10, 40, 80, 160}, different numbers of agents {4, 32, 128, 256, 1024}, and the densities of obstacles {0, 0.1, 0.2, 0.3}. Note that the densities of obstacles are defined as the probability of the occurrence of obstacles at certain locations. For each setting, five sets of trials are formed, and each set contains different map sizes, number of agents, and densities of obstacles. The 10-sized map only supports 4 and 32 agents and the 40-sized map only supports up to 256 agents. As these settings allow rotational movement, the agent has four actions including staying idle, moving forward, rotating 90° clockwise, and rotating 90° anti-clockwise. For each step in the episode, the agent decides the action to prevent the collisions and deadlock with other agents by using the state-of-the-art methods or the proposed algorithm. When the agent reaches its goal, it is set to disappear in one-shot settings.

Each setting has a timeout {256,256,384,512} based on the map size (Sartoretti et al., 2019). Then, a trial is considered as incomplete if not all the agents can reach their goals before the timeout. By convention, MAPF-rot+ is used to represent MAPF-rot with the deadlock breaking scheme. Also, MAPF-rot*+ is used to represent MAPF-rot with the deadlock breaking and resolving scheme in the following experimental results. The experiments are performed on a computer with 2T RAM and 8xA100 GPUs.

The evaluations of the feasibility of the proposed schemes and the proposed schemes under the warehouse settings are shown in Supp. D and E of the online supplementary information (https://github.com/F0048/MAPF/blob/main/supp.pdf).

4.2. Comparisons with different methods

To evaluate the performance of the proposed algorithms, the state-of-the-art methods including CBS (Sharon et al., 2012) (source: https://github.com/whoenig/libMultiRobotPlanning) and ODrM* (Ferner et al., 2013) (source: https://github.com/gsartoretti/PRIMAL), which are modified for rotational movement as stated in Section 2.1, are used for comparison. The case of density of 0.1 with 40-sized map is shown in Figure 4. The success percentages are calculated based on the average percentage of agents reaching the goals among all the sets of trials. If all the agents reach the goals for one trial before the timeout, the trial is considered a complete trial. According to Figure 4(b), the average timestamps are reported only when at least half of the trials are complete. The measurement of timestamps for different methods follows the experimental settings in Sartoretti et al. (2019).

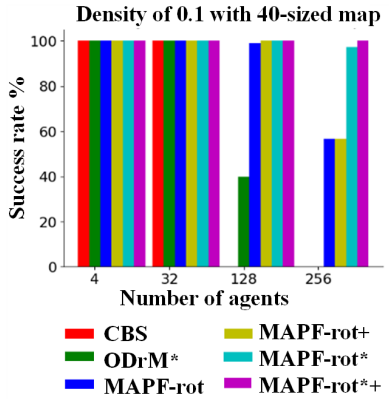
In Figure 4(a), MAPF-rot performs better than CBS and ODrM*, but it is still far from perfect when the density of obstacles increases. Then, MAPF-rot+ and MAPF-rot*+ further enhance their performance. In Figure 4(b), although the average timestamps of CBS and ODrM* are fewer than that of the others for 4 and 32 agents, they are not able to have half of the trials completed when the number of agents increases. On the other hand, MAPF-rot can still provide the solution for the settings with a larger number of agents. Again, further improvement can be achieved by MAPF-rot+ and MAPF-rot*+. Observe that incomplete trials usually are caused by deadlock occurrence and the design of MAPF-rot*+ can resolve the deadlock after it happens. For completeness, the average timestamp of MAPF-rot*+ exceeding the timeout is shown in the results.

To further study the difference between MAPF-rot+ and MAPF-rot*+, the average percentage of the completed agents for all trials in each setting by the proposed algorithms are reported in Supp. B of the online supplementary information (https://github.com/F0048/MAPF/blob/main/supp.pdf).

4.3. Comparisons of the inference time for MAPF-rot and its light version

In practice, speed is also a key factor for real situations besides performance. In Section 3.5, a light version of MAPF-rot is introduced. An experiment is designed to compare the inference time of MAPF-rot to that of its light version by using the testing dataset in Section 4.1. In the training process of the light version, the model proposed in Section 3.5 is implemented with tensorflow, and the GRU module is implemented with the operations of tensorflow.while and tensorflow.nn.rnn_cell.GRUCell. For each setting, five trials are also performed. In each trial, the total inference time for all agents in the setting is recorded for each step. Then, the result for each trial is calculated by averaging the inference time in all steps, and then each setting has an average inference time by taking the average of the results of five trials. In Figure 5(a), MAPF-rot has a longer inference time than its light version for different numbers of agents. When the number of agents is 1,024, the reduction rate is 38.23% (1.02 s and 0.63 s for 1,024 agents). The case of density of 0.1 with 40-sized map is shown in Figure 5(b). Since the light version has used a 1x1 convolution filter for speeding up the process and size reduction, this filter is more sensitive to the local change compared to that of MAPF-rot. In this case, the light version tends to move actively to avoid congestion with many agents because of the sensitivity of the local change. So, the light version is better than MAPF-rot when many agents appear in a small area.

(a)



(b)

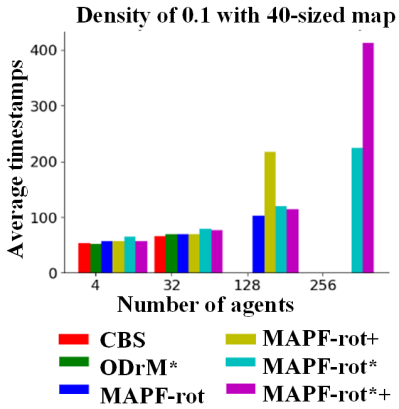
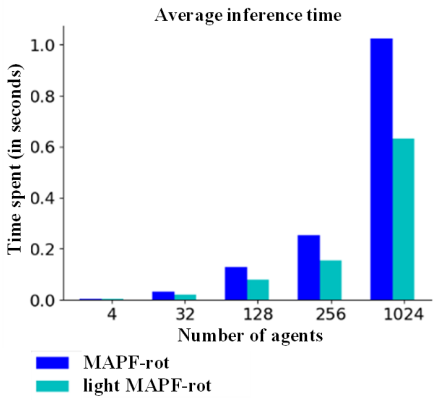


Figure 4. Success rate % and average timestamps on density of 0.1 with a 40-sized map. The results are shown as empty when more than half of the trials are not completed. MAPF-rot+ means MAPF-rot with a deadlock resolving scheme, MAPF-rot* means MAPF-rot with a deadlock breaking scheme, MAPF-rot*+ means MAPF-rot with a deadlock breaking and resolving scheme.

(a)



(b)

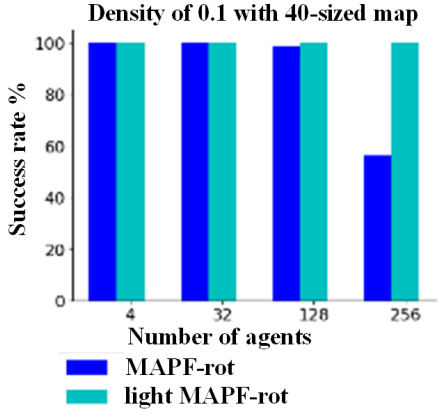


Figure 5. (a) The average inference time with different numbers of agents and (b) the success rate of a density of 0.1 with a 40-sized map.

5. Conclusion

Although there are some recent research works on MAPF, different configurations of robots with zero-radius turning by right-angle are ignored. In this work, a solution, which includes a reinforcement and imitation learning algorithm with a deadlock avoidance scheme, a deadlock breaking scheme for deadlock avoidance, and a deadlock resolving scheme for tackling the deadlock, is proposed to effectively handle the deadlock problem by considering the configurations for a rotation-based warehouse environment including moving forward, rotating clockwise, and rotating anticlockwise only. Furthermore, in order to speed up the model inference, especially with a large number of agents, a light model of MAPF-rot is also proposed. In the experimental results, the proposed solution is demonstrated to be robust and efficient for handling the deadlock problem, and it outperforms the state-of-the-art models including CBS (Sharon et al., 2012) and OrDM* (Ferner et al., 2013). Also, the proposed schemes not only can be applied in rotation-based movements, but they are also useful in other configurations of movements (4-connected

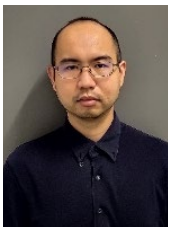
path and 8-connected path) according to the experimental results. In short, the proposed solution and schemes are advantageous to improve the success rate of MAPF and shorten the searching time and model inference time.

The limitation of this work is that it was applied in a grid-based environment only. The future works of this study aim to include kinematic constraints and continuous movement on the AGV for the realistic cases.

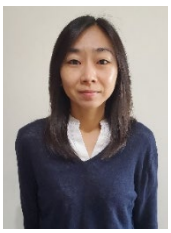
Acknowledgement

This work was supported by the InnoHK funding launched by the Innovation and Technology Commission, Hong Kong SAR.

Notes on contributors



Dr Frodo Kin Sun Chan received his bachelor's degree from the University of Hong Kong and his Ph.D. from Nanyang Technological University, NTU Singapore. He has broad exposure to R&D development by working in different companies including but not limited to Cherrypicks and LSCM in Hong Kong. Currently, he is a senior algorithm and software engineer in FLAIR. His research interests include computer vision, pattern recognition, and image processing for industrial uses. He has more than 5 years of research experience in the fields of human activity tracking, human skeleton tracking, object recognition and other fields. He has participated in a number of R&D projects of the Hong Kong Innovation and Technology Commission and has participated in the research and development projects of local universities and companies in Hong Kong. He actively participates in academic activities in related fields, and has published four papers in international academic journals and five papers in international conferences.



Dr Ivy Law received her B.Sc. and M.Phil. degrees in Mathematics from the University of Hong Kong and her Ph.D. degree in Computer Science from the University of California, Los Angeles. Currently she is Head of AI in the Hong Kong Industrial Artificial Intelligence & Robotics Centre (FLAIR). Her work involves artificial intelligence and computer vision for robotics and industrial automation. She was a Principal Architect/Assistant Director of the Robotics & Control team in LSCM from 2017 to 2022. She held a Senior Engineer position with ASTRI from 2014 to 2017. Ivy was an Assistant Principal Investigator of the Imaging Informatics Division in the Bioinformatics Institute, Singapore, from 2006 to 2014. She was also an Adjunct Assistant Professor in the Department of Computer Science in the National University of Singapore (NUS), where she has been teaching since the fall of 2013. Ivy's research interests focus on developing image processing and data mining techniques for various applications. Her works cover a broad range of topics in computer vision, artificial intelligence and robotics.

Mr Edmond Lai obtained his Bachelor of Engineering (Computer Engineering) and Master of Philosophy (Computer Science) degrees from the University of Hong Kong. Mr. Edmond Lai Shiao-bun joined HKPC in 2018 to lead the digital transformation, smart manufacturing and Mainland businesses of HKPC. Mr. Lai is an expert in Industry 4.0 (i4.0) and Enterprise 4.0 (e4.0) business transformation, as well as digital product development, with experience in local and overseas market development. He assumed the position of Member of the Board and Chief Executive Officer, Hong Kong Industrial Artificial Intelligence & Robotics Centre (FLAIR) in 2019. Mr. Lai worked in General Electric (GE) for more than 20 years. He successfully completed the management trainee programme in the information technology department of the multinational company, and then also served in their internal audit and finance functions. He also spearheaded expertise across other business units including aviation, capital, healthcare, plastics, power, and transportation among various regions including Greater China, Australia, Japan, Singapore, Switzerland, and the United States.

References

[1] Baxter JL, Burke E, Garibaldi JM and Norman M (2007). Multi-robot search and rescue: A potential field based approach. *Autonomous Robots and Agents*, 76, pp. 9-16.

[2] Cáp M, Novák P, Selecký M, Faigl J and Vokffnek J (2013). *Asynchronous decentralized prioritized planning for coordination in multirobot system*. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. Tokyo: IEEE, pp. 3822–3829.

[3] Chan FKS, Law YN, Lu B, Chick T, Lai ESB and Ge M (2022). *Multi-Agent Pathfinding for Deadlock Avoidance on Rotational Movements*. In: 2022 17th International Conference on Control, Automation, Robotics and Vision. Singapore: IEEE, pp. 765-770.

[4] Chung J, Gulcehre C, Cho K and Bengio Y (2014). Empirical evaluation of gated recurrent neural networks on sequence modelling. *arXiv preprint*, abs/1412.3555.

[5] Cui R, Gao B and Guo J (2011). Pareto-optimal coordination of multiple robots with safety guarantees. *Autonomous Robots*, 32(3), pp. 189-205.

[6] Ferner C, Wagner G and Choset H (2013). *ODrM*: optimal multirobot path planning in low dimensional search spaces*. In: 2013 IEEE International Conference on Robotics and Automation. Karlsruhe: IEEE, pp. 3854-3859.

[7] Foerster J, Assael IA, de Freitas N and Whiteson S (2016). *Learning to communicate with deep multi-agent reinforcement learning*. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. Barcelona:Curran Associates Inc., pp. 2145–2153.

[8] Foerster J, Farquhar G, Afouras T, Nardelli N and Whiteson S (2017). Counterfactual multi-agent policy gradients. *arXiv preprint*, abs/1705.08926.

[9] Gupta JK, Egorov M and Kochenderfer M (2017). *Cooperative multiagent control using deep reinforcement learning*. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. São Paulo: International Foundation for Autonomous Agents and Multiagent Systems, pp. 66-83.

[10] Hönig W, Kiesel S, Tinka A, Durham JW and Ayanian N (2019). Persistent and Robust Execution of MAPF Schedules in Warehouses. *IEEE Robotics and Automation Letters*, 4(2), pp. 1125-1131.

[11] Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint*, abs/1602.07360.

[12] Li P, Yang H, Li H and Liang S (2022). Nonlinear ESO-based tracking control for warehouse mobile robots with detachable loads. *Robotics and Autonomous Systems*, 149(C), pp. 103945.

[13] Lowe R, Wu Y, Tamar A, Harb J, Abbeel OP and Mordatch, I. (2017). *Multi-agent actor-critic for mixed cooperative-competitive environments*. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach: Curran Associates Inc., pp. 6382–6393.

[14] Ma H, Tovey CA, Sharon G, Kumar TS and Koenig S (2016). *Multi-Agent Path Finding with Payload Transfers and the Package-Exchange Robot-Routing Problem*. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. Phoenix: AAAI press, pp. 3166–3173.

[15] Morris R, Pasareanu C, Luckow K, Malik W, Ma H, Kumar S and Koenig S (2016). *Planning, scheduling and monitoring for airport surface operations*. In: AAAI-16 Workshop on Planning for Hybrid Systems.

[16] Sanchez G and Latombe J (2002). *Using a PRM planner to compare centralized and decoupled planning for multi-robot systems*. In: Proceedings 2002 IEEE International Conference on Robotics and Automation. Washington: IEEE, pp. 2112-2119.

[17] Sartoretti G, Kerr J, Shi Y, Wagner G, Satish Kumar TK, Koenig S and Choset H (2019). PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters*, 4(3), pp. 2378-2385.

[18] Sharon G, Stern R, Felner A and Sturtevant N (2012). *Conflict-based search for optimal multi-agent path finding*. In: Proceedings of the AAAI Conference on Artificial Intelligence 26(1). Toronto: AAAI Press, pp. 563-569.

[19] Silver D (2005). *Cooperative pathfinding*. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Marina Del Rey: AAAI Press, pp. 117-122.

[20] Van Den Berg J, Guy SJ, Lin M and Manocha D (2011). *Reciprocal n-body collision avoidance*. In: 14th International Symposium of Robotic Research. Lucerne: Springer Tracts in Advanced Robotics, pp. 3-19.

[21] Veloso M, Biswas J, Coltin B and Rosenthal S (2015). *CoBots: Robust symbiotic autonomous mobile service robots*. In: Proceedings of the 24th International Conference on Artificial Intelligence. Buenos Aires: AAAI Press, pp. 4423-4429.

[22] Wang C, Deng D, Xu L and Wang W (2022). Resource Scheduling Based on Deep Reinforcement Learning in UAV Assisted Emergency Communication Networks. *IEEE Transactions on Communications*, 70(6), pp. 3834-3848.

[23] Wang G, Wu C, Du Z, Tsutomu Y, Rui Y and Lei Z (2023). DRL-Assisted Network Selection for Federated IoV. *IEEE Internet of Things Magazine*, 6(3), pp. 86-90.

[24] Wurman PR, D'Andrea R and Mountz M (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1), pp. 9-20.

Appendix

Nomenclature

o	Orientation index to represent four different facing directions
(x, y)	XY Coordinate in cartesian space
(x, y, o)	XY Coordinate in cartesian space with different facing direction
$\boldsymbol{N}_{left}$	Left neighbour in cartesian space
$\boldsymbol{N}_{right}$	Right neighbour in cartesian space
$\boldsymbol{N}_{forward}$	Neighbour in the agent facing direction
d_{NS}	The value (1/-1) of an agent location on a vertical map when an agent faces to north/south
d_{EW}	The value (1/-1) of an agent location on horizontal map when an agent faces to east/west
$g = (g_x, g_y, m)$	Goal vector on the 2D map
(x_a, y_a)	Current position of an agent