An Obstacle Avoidance-Specific Reinforcement Learning Method Based on Fuzzy Attention Mechanism and Heterogeneous Graph Neural Networks

Feng Zhang[a], Chengbin Xuan[a], Hak-Keung Lam[a,*]

[a]*Department of Engineering, King's College London, Strand, London, WC2R 2LS, London, United Kingdom*

Abstract

Deep reinforcement learning (RL) is an advancing learning tool to handle robotics control problems. However, it typically suffers from sample efficiency and effectiveness. The emergence of Graph Neural Networks (GNNs) enables the integration of the RL and graph representation learning techniques. It realises outstanding training performance and transfer capability by forming controlling scenarios into the corresponding graph domain. Nevertheless, the existing approaches strongly depend on the artificial graph formation processes with intensive bias and cannot propagate messages discriminatively on explicit physical dependence, which leads to restricted flexibility, size transfer capability and suboptimal performance. This paper proposes a fuzzy attention mechanism-based heterogeneous graph neural network (FAM-HGNN) framework for resolving the control problem under the RL context. FAM emphasises the significant connections and weakening of the trivial connections in a fully connected graph, which mitigates the potential negative influence caused by the artificial graph formation process. HGNN obtains a higher level of relational inductive bias by conducting graph propagations on a masked graph. Experimental results show that our FAM-HGNN outperforms the multi-layer perceptron-based and the existing GNN-based RL approaches regarding training performance and size transfer capability. We also conducted an ablation study and sensitivity analysis to validate the efficacy of the proposed method further.

Keywords: Graph Neural Networks, Fuzzy Logic System, Reinforcement Learning,

*Corresponding Author

Email addresses: `feng.1.zhang@kcl.ac.uk` (Feng Zhang), `chengbin.xuan@kcl.ac.uk` (Chengbin Xuan), `hak-keung.lam@kcl.ac.uk` (Hak-Keung Lam)

Mobile Robot, Obstacle Avoidance Navigation Control

1. Introduction

Effective and efficient collision-free navigation is critical to ensure the safety of the mobile robot and its surrounding environment, as collisions or other types of interference can damage equipment, injure individuals, or disrupt critical processes. Therefore, obstacle avoidance navigation control is a paramount functionality for mobile robots. However, the existing conventional obstacle avoidance techniques such as D* (Stentz, 1994), reactive obstacle avoidance (Lalish and Morgansen, 2012) and artificial potential field (Khatib, 1986) typically emanate inferior performance due to the dynamic or rapidly changing environments.

The advent of deep reinforcement learning signifies an advancing controller design strategy that frames the control problem as the optimisation of policies. RL can handle highly dynamic and complex environments through massive amounts of interactions with trial and error. The efficacy of this paradigm has been substantiated through its successful application to a diverse array of decision-making challenges (Vinyals et al., 2019; Mnih et al., 2013; Silver et al., 2016; Zhao et al., 2018), achieving performance levels that rival human intelligence. RL is also a promising alternative for developing automatic controllers for extensive robotics applications. To obtain the predictions for conducting interaction with the environments and learning, the conventional RL structures typically construct their corresponding policy networks and value networks with multi-layer perceptrons (MLPs). However, some imperceptible defects under this default setting hinder the RL involving real-world robotics applications. First, even though the underlying structural information is naturally plentiful within complicated RL environments, MLPs tend to neglect them, leading to suboptimal sample efficiency and effectiveness (Dulac-Arnold et al., 2021). Second, the intrinsic fixed MLP input size restricts the size transfer capability of the learned policies, impeding the application towards robotics applications that are supposed to work in a complex and varied environment.

One of the effective integrations with RL to mitigate the above issues is graph neural networks. GNNs are designed to capture the structural information from graph-structured data. Moreover, they can naturally generalise different input sizes over similar tasks without network modifications. Researchers typically introduce GNNs as part of the policy networks to conduct graph representation learning and extract problem-independent structural information to increase the efficiency of information utilisation.

Table 1: List of notations.

Notations For Reinforcement Learning	
$\mathcal{S}$	state space of the observations
$\mathcal{A}$	action space of the robot
$\mathcal{P}$	transition probability function
r	reward function
R	return; or discounted accumulated reward
π, π_θ, π^*	policy, policy function under parameter θ, optimal policy
s, s', s_t	state, next state, state at time t
γ	discount factor
Notations For Fuzzy Attention System	
R^i	the i_{th} rule of the TS-FIS
$f^i(\cdot)$	the i_{th} subsystem function
$\mathbf{x}, x_j$	input vector, the j_{th} scalar within the input vector
M_j^i	fuzzy set with respect to the i_{th} rule and the j_{th} input
$\mathbf{m}^i$	set of membership values with respect to the i_{th} rule
$w^i(\mathbf{x})$	truth value with respect to the i_{th} rule
y	the final crispy output
Notations For Graph Neural Network	
G	graph
V	set of nodes of the graph
E	set of edges of the graph
v	a node within the graph
(v, u)	a directed edge from node v to node u
p_u, P	the node type of the node u, the number of node type
$c_{(v,u)}, C$	edge type of the edge (v, u), number of the edge type
$\mathcal{N}(u)$	neighbourhood node sets of the node u
h_u^l	node features of the node u in layer l
$M_{c_{(v,u)}}^l$	message function with respect to edge type $c_{(v,u)}$ in layer l
$m_{(v,u)}^l$	the message from node v to node u in layer l
A^l	aggregation function in layer l
$\overline{m}_u^l$	the aggregated message of node u in layer l
$U_{p_u}^l$	update function with respect to node type p_u in layer l
READOUT	readout function
V_{out}	the set of specific nodes to be applied in the readout function
$H(t)$	readout output

Nervenet (Wang et al., 2018) formed the multi-joint agent as a graph based on its morphology and applied GNNs as part of the policy networks for conducting RL. Decentralised multi-robot collision-free path planning is another prominent application that benefited from GNNs thanks to the natural graph formed by robot-communication networks (Li et al., 2020; Tolstaya et al., 2020; Blumenkamp et al., 2022; Tzes et al., 2023). Even though superior training performance and salient generalisation capability are demonstrated in these approaches, two deficiencies potentially lead to suboptimal graph information extraction. First, connections are treated equally important even though different neighbourhoods could impose disparate influences on graph propagation. Taking the obstacle avoidance scenario as an example, the mobile robot should focus more on the closer obstacles than the spatial-distant obstacles. This philosophy is reflected in GNN as information from different node neighbourhoods should have different weights. Second, human bias and manual graph formation are immensely dependent. This priori assumption could result in redundancy connections and the disappearance of the essential edges.

A common methodology for addressing the above issues is the attention mechanism (Vaswani et al., 2017). Several attention-based methods have been proposed in the GNNs community to resolve classification tasks conditioned on node degree importance (Kipf and Welling, 2016), edge-wise self-attention (Veličković et al., 2017), motif-based attention (Peng et al., 2018), etc. The attention mechanism can generate edge-wise attention based on information from a pair of nodes. The attention works as a weighting mask that endows the edge-wise messages with different importance. Therefore, it can emphasise the important connections and understate negligible edges. MAGAT (Li et al., 2021) exploits an edge-wise self-attention paradigm to resolve the decentralised path planning problem. Despite outperforming the GNN-based methods that exclude an attention mechanism, MAGAT still fails to consider the communication between obstacles. Contrarily, G2ANet (Liu et al., 2020) proposes to introduce a two-stage learnable attention network into the multi-agent reinforcement learning problems that take all components into a fully connected graph. It first discards unrelated edges through hard attention in a fully connected map and then applies soft attention to discriminate the importance of different edges. However, both hard and soft attentions are conditioned on black-box learnable networks, lacking indispensable interpretability for real-world robotics applications. Yuying et al. proposed a gaze-modulated GCN-based RL (Chen et al., 2020) to resolve the navigation problem in crowds. The method determines the inter-human connections by attention learnt from the human gaze data. However, a shared weight for computing messages among edges fails to discriminate relationships such as robot-human and human-human, leading to suboptimal data efficiency. Zhe et al. proposed handling a similar problem with graph relational networks considering obstacle-robot, object-robot and obstacle-obstacle relationships (Liu et al., 2023). They also exploit trainable attention kernels for weighing different connects. However, their attention system lacks explicit physical meaning, leading to vulnerable credibility and limited interpretability.

A literature categorisation is provided in Table 2 to demonstrate the research gap intuitively. In summary, the existing GNN-based RL methods fail to handle the collision-free navigation control by satisfying the following features at the same time: 1) lightly or no reliance on artificial rules for graph formation, 2) the ability to propagate messages according to different relationship types, 3) able to distinguish the diverse contribution of the neighbourhood through an explainable attention mechanism with explicit physical meanings.

Table 2: A comparison of the surveyed papers.

Method	Adjacency	Attention	Relational	Interpretability
Nervenet (Wang et al., 2018)	structural bias	No	Yes	No
CNN-GNN (Li et al., 2020)	local field-of-view	No	Yes	No
MAGAT (Li et al., 2021)	inter-robot local field-of-view	self-attention	No	No
G2ANet (Liu et al., 2020)	fully connected graph	hard & soft attention	No	No
Gaze-Modulated GCN (Chen et al., 2020)	fully connected graph	learned from gaze data	No	Yes
Relational GNN (Liu et al., 2023)	local field-of-view	trainable attention kernel	Yes	No

Inspired by the works above and to address these research gaps, this study combines the fuzzy attention mechanism with heterogeneous graph neural network architecture as an RL integrating methodology. In this work, a fully connected graph that involves all components within the horizon is formed to introduce the FAM-HGNN algorithm. The antecedent knowledge is then fed into the T-S fuzzy system for generating physically explicit attention that weighs connections between different components. These fuzzy attention masks introduce the relational inductive bias concerning nodes' spatial similarity in a cybernetic way. The HGNN is finally exploited for excavating the structural information from the established graph. The overall structure of FAM-HGNN is illustrated in Fig. 1. The main contributions are summarised as follows:

- A fuzzy attention mechanism is combined with the heterogeneous graph neural network to improve the data efficiency under obstacle avoidance navigation control. We demonstrate the superiority regarding training return and success rate of our model on a customised obstacle avoidance control scenario.

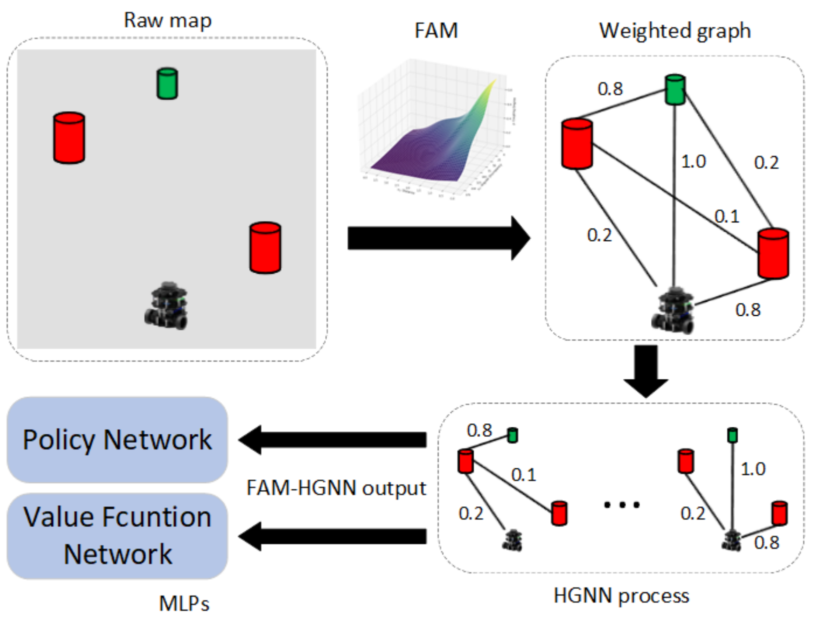

Figure 1: RL-based obstacle avoidance control based on Fuzzy Attention Mechanism and Heterogeneous Graph Neural Network.

- By testing the trained model on obstacle avoidance navigation control with different obstacle numbers with its training and comparing its performance with the baseline artificial potential filed method, the significant size transfer capability of our model is verified.
- We conduct an ablation study regarding FAM structure and demonstrate its efficacy.
- We perform sensitivity analysis for both MLP and GNN components regarding their network size, validating the robustness of our model structure regarding network size.

The rest of the paper is organised as follows: Section 2 provides the preliminaries of RL in continuous control, the T-S fuzzy system and the heterogeneous graph neural networks. Section 3 explains the proposed method with an obstacle avoidance instance. Experimental details are presented in Section 4. Results evaluations and analysis are discussed in Section 5.

2. Preliminaries

In this section, the notations and the necessary background knowledge are provided to support the development of the proposed method.

2.1. Markov Decision Process and Continuous Control

The continuous control problem can be formulated as a Markov decision process (MDP). MDP is defined as tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r)$, where $\mathcal{S}$ is the state space of the observations, $\mathcal{A}$ denotes the action space of the robot, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition probability function which takes as input state-action pair and output the next state. r represents the reward by transferring system from current state s into the next state s' through action a. π is the policy that generates instructive action based on the state s_t at time instant t. By denoting γ as the discount factor, the goal of RL is to obtain an optimal policy π^* that maximises expected discounted accumulative reward (Sutton and Barto, 2018):

$$R = \max_{\pi_\theta} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t)] \tag{1}$$

To investigate the effectiveness of graph representation learning, policy proximal optimisation (PPO) (Schulman et al., 2017) is chosen in this work as the RL algorithm for addressing the continuous control problem.

2.2. Takagi-Sugeno Fuzzy Inference System

The Fuzzy inference system (FIS) employs an evaluation methodology based on fuzzy rules that draw inspiration from the human decision-making process (Zadeh, 1965). In particular, this work focuses only on the Takagi-Sugeno (T-S) type fuzzy system (Takagi and Sugeno, 1985).

A T-S fuzzy system can typically be depicted as Fig. 2. The working principle of the TS-FIS can be interpreted as a rule-based piecewise aggregation of multi-subsystems. Each subsystem is determined by its own linguistic rule as the following:

$$R^i : \text{IF } x_1 \text{ is } M_1^i \text{ AND } \dots \text{ AND } x_I \text{ is } M_I^i \text{ THEN } y = f^i(\mathbf{x}) \tag{2}$$

where R^i denotes the i_{th} rule of the overall TS-FIS and $f^i(\cdot)$ is the i_{th} subsystem function. x_j is the j_{th} crispy input of the system and $M_j^i \in [0, 1]$ represents the fuzzy sets with respect to the i_{th} rule and the j_{th} input, typically for linguistically describing the input variables. In the above fuzzy rule, TS-FIS firstly take as inputs crisp numerical values $\mathbf{x} = [x_1, \dots, x_I] \in \mathbb{R}^I$ to their rule-based membership functions to generate a set of membership values $\mathbf{m}^i(\mathbf{x}) = [M_1^i(x_1), \dots, M_N^i(x_N)]$. The

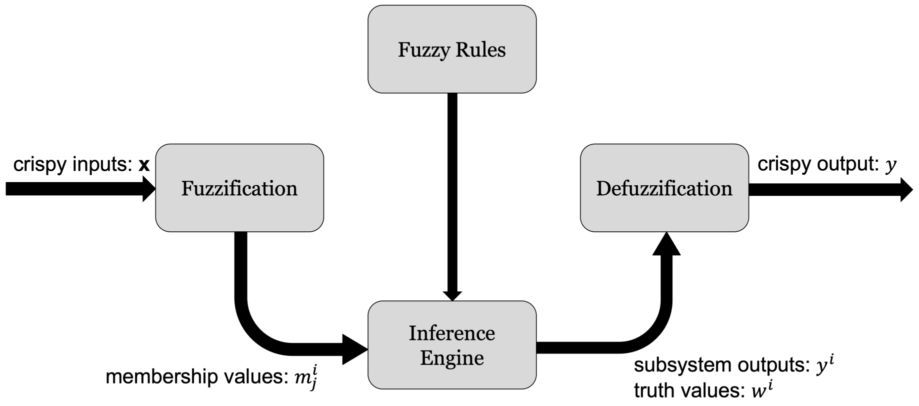

Figure 2: Block Diagram of the Takagi-Sugeno Fuzzy Inference System (Mehran, 2008).

membership values indicate the matching degrees between inputs and fuzzy sets. The above process is so-called fuzzification.

To obtain a rule-based truth value $w^i(\mathbf{x})$, TS-FIS exploits AND operations for aggregating the membership values, which is equivalent to the minimum operation over i_{th} rule-based membership values set $\mathbf{m}^i(\mathbf{x})$:

$$w^i(\mathbf{x}) = \min(\mathbf{m}^i(\mathbf{x})) \tag{3}$$

where $w^i(\mathbf{x}) \in [0, 1]$, indicates the rule firing strength and can be regarded as a relative weight of this rule. A defuzzification methodology is finally applied to extract a crisp final output $y(\mathbf{x})$ from the rule-based subsystems' outputs $f^i(\mathbf{x})$ and their corresponding truth values w^i, which is a softmax summation process as follows:

$$y(\mathbf{x}) = \frac{\sum_{i=1}^{n} w^i(\mathbf{x}) f^i(\mathbf{x})}{\sum_{i=1}^{n} w^i(\mathbf{x})} \tag{4}$$

Generally, a TS-FIS can be regarded as an epistemic non-linear function constructor. Unlike the obscureness of neural networks, the membership functions within FIS are endowed with explicit physical or logical meaning based on different systems' contexts, thereby enhancing interpretability.

2.3. Heterogeneous Graph Neural Networks

We focus on the heterogeneous graphs in our problem formulation. The graph structure is defined as $G = (V, E)$, where V represents the set of nodes and E is the set of edges. Without losing generality and following the format of a directed heterogeneous graph, a directed edge from node v to node u is denoted as (v, u), where $u, v \in V$. The neighbourhood set of node u is denoted as $\mathcal{N}(u)$, which comprise all node has an edge with node u. Unlike homogeneous graphs, a heterogeneous can have different node types and edge types, denoted as $p_u \in \{1, 2, \dots, P\}$ and $c_{(v,u)} \in \{1, 2, \dots, C\}$, respectively.

Although different types of nodes can have different feature dimensions in a heterogeneous graph, we uniformly define the dimension of the node features as D for the sake of simplicity. Under this setting, GNN learns a function $f(\cdot) : \mathbb{R}^{N \times D} \rightarrow \mathbb{R}^{O}$ that maps nodes' features into a compressed embedding, where N denotes the node number and O is the desired output dimension. The forward propagation of GNN typically consists of three processes, i.e., message computation, message aggregation and node embedding update. These processes can be represented in either spectrum form or vertex-expansion form, while the latter is adopted in this work.

The input of a GNN is a set of node features h_u^0 which are normally assigned from raw observation s_t from the environment. In a heterogeneous graph, messages from a source node v to a destination node u are calculated based on relation-based message functions M:

$$m^l_{(v,u)} = M^l_{c_{(v,u)}}(h^l_v) \tag{5}$$

where superscript l here denotes the current layer number. The total layer number is L and it determines the number of times the above processes are repeated. Once we obtain the inter-nodes messages, aggregation functions A are then applied for synthesising messages from all incoming edges:

$$\overline{m}^l_u = A^l(\{m^l_{(v,u)} | v \in \mathcal{N}(u)\}) \tag{6}$$

Node update is the final step in each propagation layer. By taking as inputs the incoming aggregated message of each node $\overline{m}^l_u$ and its current embedding h^l_u, an update function U that is dependent on the node type p_u conducts the nodes' embeddings update as follows:

$$h^{l+1}_u = U^l_{p_u}(h^l_u, \overline{m}^l_u) \tag{7}$$

A readout function is exploited after L times repeats of the above propagations. The readout function is defined as a function that aggregates information from nodes to output a compressed representation $H(t)$ of the entire graph. Based on this, a fixed-size composite output can be generated regardless of the graph's size and the nodes' feature size. A readout function is mathematically represented as:

$$H(t) = \underset{u \in V}{\text{READOUT}}(\{h_u^L | u \in V_{out}\}) \tag{8}$$

where V_{out} is a set consisting of specific nodes to be applied in the readout function. The design of the readout function is very specific to the objective. When integrating with the RL, the readout output $H(t)$ is fed into other MLPs for generating the final output that satisfies the dimensional requirement of the actions and the values.

3. Methodology

In this section, the technical details of FAM-HGNN are demonstrated by instantiating an obstacle avoidance navigation control scenario. The graph formation process is first interpreted. A TS-FIS-based FAM is then applied to weigh the different edges within the graph. A HGNN that takes as inputs the node embeddings and edge-wise attention is finally designed to generate the actions and values.

3.1. Graph Formation

In the obstacle avoidance navigation and control problem, we consider three primary types of components on our horizon: robots, targets and obstacles. Each element of these components is treated as a distinct node, carrying its respective physical information. For a better interpretation of the proposed method and without losing generality, an obstacle avoidance scenario with four obstacles, one mobile robot and one target is instantiated. A fully connected heterogeneous graph involving all available components is constructed to represent this system as a graph. The corresponding graph structure is visualised in Fig. 3. It can be expressed as $G = (V, E)$, where $V = \{n_{robot}, n_{target}, n^1_{obstacle}, n^2_{obstacle}, n^3_{obstacle}, n^4_{obstacle}\}$ and $E = \{(v, u) | u \in V \wedge v \in \complement_V u\}$. Specifically, the robot node n_{robot} is characterised by a concatenated feature vector $s_{n_{robot}} = (\overrightarrow{\mathbf{p}}, \overrightarrow{\mathbf{v}}, \overrightarrow{\mathbf{h}}) \in \mathbb{R}^6$ that consists of three elements: positional coordinates $\overrightarrow{\mathbf{p}} = (x_{position}, y_{position}) \in \mathbb{R}^2$, velocity $\overrightarrow{\mathbf{v}} = (x_{velocity}, y_{velocity}) \in \mathbb{R}^2$, and heading direction $\overrightarrow{\mathbf{h}} = (x_{heading}, y_{heading}) \in \mathbb{R}^2$. The target node and obstacle nodes are represented solely by their respective spatial coordinates. To guarantee consistent dimensional properties across all nodes, we employ zero padding for the target and obstacle nodes, i.e., $s_{n_{target}/n_{obstacle}} = (\overrightarrow{\mathbf{p}}, \overrightarrow{\mathbf{0}}, \overrightarrow{\mathbf{0}})$.

3.2. Fuzzy Attention Mechanism

A fully connected graph is uninformative in providing spatial independence between different components. Therefore, the challenge is obtaining an expressive graph representation with strong cognition capability based on relational inductive biases.

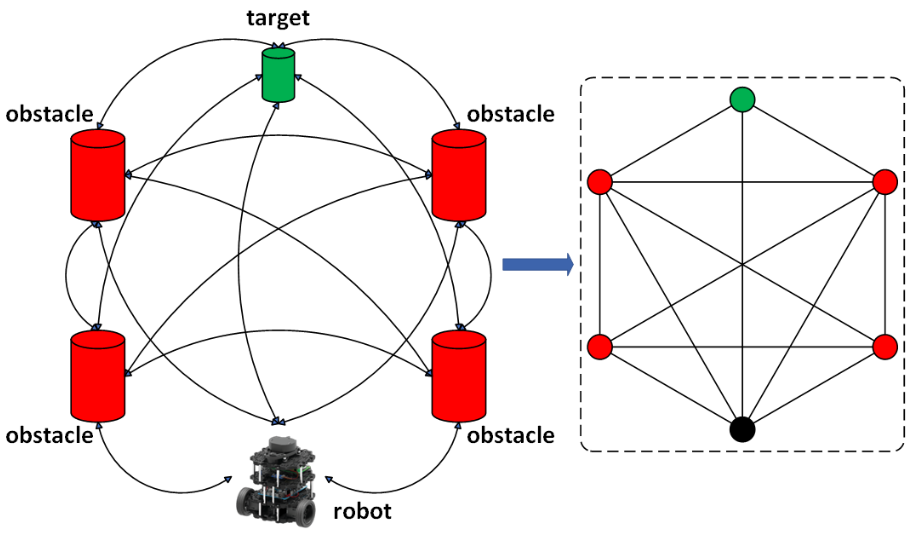

Figure 3: In this figure, we provide a graph instance with 1 robot, 1 target and 4 obstacles. We form a fully connected graph at this stage by taking all components. The arrows indicate the directed edges of the graph.

The attention mechanism is an effective method for masking connections with different weights without violating the original connectivity. However, it typically suffers from limited explicit physical meaning. As the previous introduction, the fuzzy logic system is a human-level reasoning framework that involves expert knowledge. This capability enables us to design an explicable expert system that evaluates the importance of connections among the given graph. Therefore, we introduce a fuzzy attention mechanism system for emphasising strong connections and neglecting trivial connections.

To evaluate the importance of connections between two nodes, the FAM generates edge-wise attention for every edge within the graph G:

$$y_{v,u}(\mathbf{x}) = \mathrm{FAM}(\mathbf{x}) \tag{9}$$

where $\mathbf{x} = [x_1, x_2]$ is the edge-wise vector of premise variables. x_1 is the Euclidean distance regarding the nodes' coordinates, and x_2 is the relative angular discrepancy between the destination node and source node, respectively. To account for the effect of both the velocity and the heading direction of the mobile robot, we define x_2 as follows:

$$x_2 = \arccos \frac{\alpha \cdot \beta}{\max(|\alpha| \cdot |\beta|, \epsilon)} \tag{10}$$

To mitigate the zero-division issue, a max operation along with a small value ϵ is applied in the denominator. The operator $\cdot$ employed in the numerator represents the dot product of the vectors, whereas it denotes the multiplication of two scalar numbers in the denominator. Specifically, α is the positional difference between the two nodes, and β is a composite vector combining information from velocities and heading angles. They are mathematically defined as:

$$\begin{aligned}
\alpha &= \overrightarrow{\mathbf{p}}^{destination} - \overrightarrow{\mathbf{p}}^{source} \\
\beta &= \beta_{velocity} + \beta_{heading} \\
\beta_{velocity} &= \overrightarrow{\mathbf{v}}^{source} - \overrightarrow{\mathbf{v}}^{destination} \\
\beta_{heading} &= \overrightarrow{\mathbf{h}}^{source} - \overrightarrow{\mathbf{h}}^{destination}
\end{aligned} \tag{11}$$

where the superscript *source* and *destination* are for specifying if the vector information is from the source node or destination node in a directed edge. To generally describe the vectors in Eq. (11) regardless of the source-destination relationship, we discard the superscript in the consequent discussion. Following the definition in Section3.1, $\overrightarrow{\mathbf{p}} = (x_{position}, y_{position})$ is the position vector of the node, $\overrightarrow{\mathbf{v}} = (x_{velocity}, y_{velocity})$ is the velocity vector of the node and $\overrightarrow{\mathbf{h}} = (x_{heading}, y_{heading})$ is the heading direction of the node, respectively. Figure 4 provides a graphical representation to interpret x_2. Specifically, since we apply zero padding to represent the velocity and heading direction of target nodes and obstacle nodes in Section3.1, their corresponding parts are equal to a zero vector, i.e., $\overrightarrow{\mathbf{v}} = \overrightarrow{\mathbf{h}} = \overrightarrow{\mathbf{0}}$.

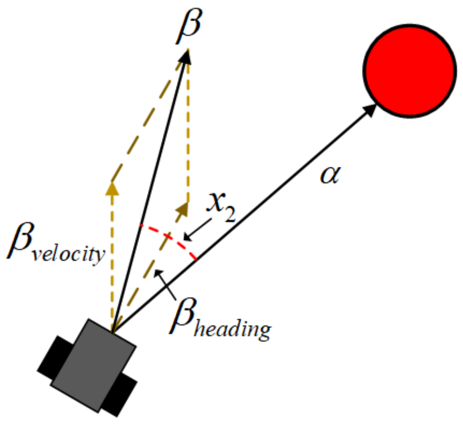

Figure 4: An illustration of the x_2 that takes a robot node as the source node and an obstacle node as the destination node.

By recalling the preliminaries in Eq. (3), this FAM system can be mathematically represented as:

$$\mathrm{FAM}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \min([M_1^i(x_1), M_2^i(x_2)]) f^i(\mathbf{x})}{\sum_{i=1}^{n} \min([M_1^i(x_1), M_2^i(x_2)])} \tag{12}$$

Each subsystem is described by its own fuzzy rule R^i and function $f^i(\mathbf{x})$. All fuzzy rules are listed R^i in Eq. (14). To explicitly indicate the relative scale of premise variables according to antecedent knowledge under the given environment, premise variables x_1 and x_2 are linguistically delineated by "Small", "Medium" or "Large". The corresponding unnormalised Gaussian membership functions with respect to both inputs are illustrated in Fig. 5. Their mathematical models are uniformly described as:

$$M_j^i = e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{13}$$

where μ is the mean and σ is the variance. Specifically, for "Small", "Medium" and "Large" of x_1, $\sigma = 0.75$, $\mu = 0, 2, 4$, respectively. For "Small", "Medium" and "Large" of x_2, $\sigma = 30$, $\mu = 0, 90, 180$, respectively.

Each subsystem within the FAM is a linear function $f^i(\mathbf{x}) = A^i\mathbf{x}^\intercal + b^i$, which is described by the corresponding subsystem parameters A^i, b^i. The exact values of all subsystem parameters are given in Eq. (15).

$$\begin{aligned}
&R^1 : \text{IF } x_1 \text{ is Small AND } x_2 \text{ is Small THEN } y = f^1(\mathbf{x}) = A^1\mathbf{x}^\intercal + b^1\\
&R^2 : \text{IF } x_1 \text{ is Small AND } x_2 \text{ is Medium THEN } y = f^2(\mathbf{x}) = A^2\mathbf{x}^\intercal + b^2\\
&R^3 : \text{IF } x_1 \text{ is Small AND } x_2 \text{ is Large THEN } y = f^3(\mathbf{x}) = A^3\mathbf{x}^\intercal + b^3\\
&R^4 : \text{IF } x_1 \text{ is Medium AND } x_2 \text{ is Small THEN } y = f^4(\mathbf{x}) = A^4\mathbf{x}^\intercal + b^4\\
&R^5 : \text{IF } x_1 \text{ is Medium AND } x_2 \text{ is Medium THEN } y = f^5(\mathbf{x}) = A^5\mathbf{x}^\intercal + b^5\\
&R^6 : \text{IF } x_1 \text{ is Medium AND } x_2 \text{ is Large THEN } y = f^6(\mathbf{x}) = A^6\mathbf{x}^\intercal + b^6\\
&R^7 : \text{IF } x_1 \text{ is Large AND } x_2 \text{ is Small THEN } y = f^7(\mathbf{x}) = A^7\mathbf{x}^\intercal + b^7\\
&R^8 : \text{IF } x_1 \text{ is Large AND } x_2 \text{ is Medium THEN } y = f^8(\mathbf{x}) = A^8\mathbf{x}^\intercal + b^8\\
&R^9 : \text{IF } x_1 \text{ is Large AND } x_2 \text{ is Large THEN } y = f^9(\mathbf{x}) = A^9\mathbf{x}^\intercal + b^9
\end{aligned} \tag{14}$$

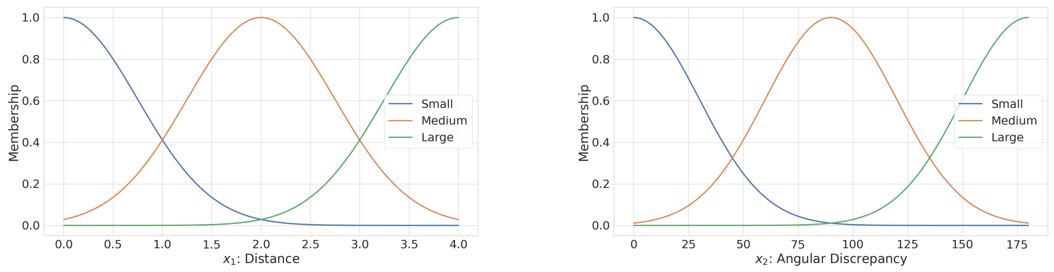

Figure 5: Unnormalised Gaussian antecedent membership functions f two premise inputs x_1 and x_2 .

$$\begin{aligned}
&A^1 = [-0.2, 0] && A^2 = [-0.1, -0.002] && A^3 = [-0.05, -0.001] \\
&A^4 = [-0.05, -0.001] && A^5 = [-0.05, -0.001] && A^6 = [-0.01, -0.0002] \\
&A^7 = [-0.055, 0.0008] && A^8 = [-0.01, -0.0002] && A^9 = [0, 0] \\
&b^1 = 1 && b^2 = 0.7 && b^3 = 0.275 \\
&b^4 = 0.4 && b^5 = 0.25 && b^6 = 0.07 \\
&b^7 = 0.2225 && b^8 = 0.07 && b^9 = 0
\end{aligned} \tag{15}$$

The T-S system outputs are designated as the coupling degrees or graph edges-wise attentions, signifying the physical and logical interdependence of the two nodes. Fig. 6 illustrates the ultimate input-output attention value surface. Specifically, the coupling degree exerted on the edges between the robot and the target node, i.e., $y_{n_{robot},\ n_{target}}$ and $y_{n_{target},\ n_{robot}}$, is always set to 1 for manifesting the global goal of reaching the target. Essentially, these attention weights demonstrate the spatial similarity of the nodes in the context of navigation and obstacle avoidance problems. FAM is a transparent rule-based system synthesising the effect of "IF/THEN" rules and subsystems. All of these are designed by conditioning on prior knowledge with artificial inference towards the environment. These characteristics contribute to its high interpretability compared with the concurrently learned style attention system (Veličković et al., 2017; Vaswani et al., 2017).

Under this framework, the graph connection attentions $\{y_{v,u}(\mathbf{x}_t)|(v, u) \in E\}$ are dynamically updated to indicate the variable relationships between different components over time. Despite the methodology conditioned on a naive fully connected graph with limited spatial structural bias, the introduction of FAM significantly magnifies its expressiveness by converting hard edges into soft connections. This transformation alleviates the difficulty of forming a graph artificially while simultaneously retaining the relational inductive bias through soft connections. It also avoids the possible sparsity adjacency under the hard connection setting, which leads to inefficient representation and learning (Abadal et al., 2021).

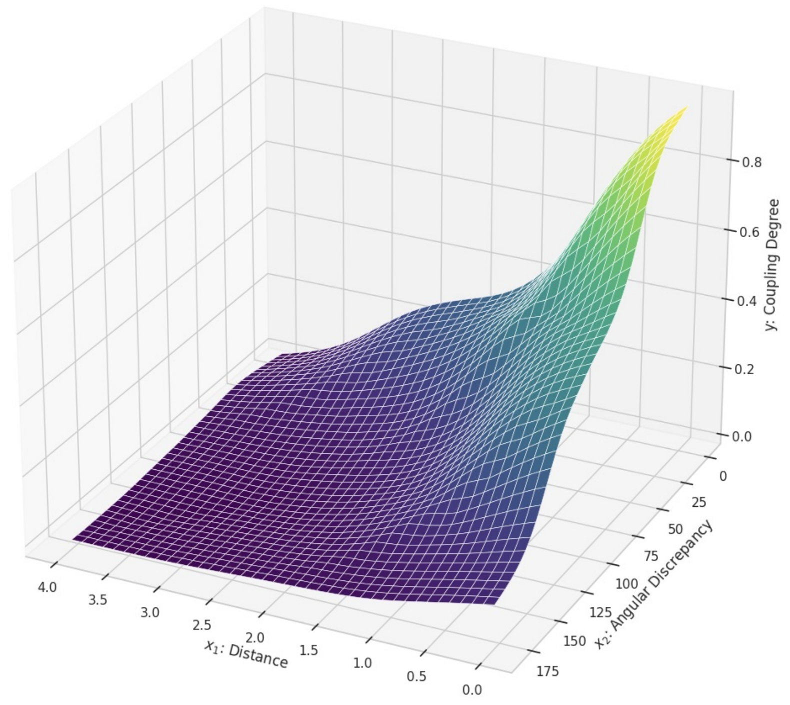

Figure 6: 3D surface plot with two independent inputs x_1 and x_2, and attention output y. A high coupling degree is designated only when both x_1 and x_2 are tiny, indicating a strong spatial similarity.

3.3. Heterogeneous Graph Neural Networks as Policy and Value Network

To exploit the structural information and thereby increase the data efficiency, A HGNN is introduced as a spatial information extractor. Conditioned on the graph G as Fig 3, HGNN extracts expressive node embeddings through graph propagation processes. The HGNN structure involves edge-relationship-based message propagation, attention-based message aggregation and node-type-based feature update. The overall FAM-HGNN is described as Algorithm 1.

At every timestep interacting with the environment, an amalgamated state vector $s_t = s_{robot} \parallel s_{target} \parallel s^1_{obstacle} \parallel \ldots \parallel s^{n_o}_{obstacle}$ is generated from the physical simulator as the environmental observation, where $\parallel$ denotes the concatenation. This state vector

Algorithm 1 Fuzzy Attention Mechanism Heterogeneous Graph Neural Network

```
1:  Build a fully connected graph $G$ based on the environment settings.
2:  for Each timestep encountering with the environment do
3:      Disassemble the observation vector $s_t$ into $N$ different parts, which corresponding to the $N$ nodes' features $h_u^0$.
4:      Take raw node features as input to the T-S fuzzy system and generate the edge-wise attentions:
            $y_{v,u}(\mathbf{x}) = \text{FAM}(\mathbf{x})$
5:      for $l = 0, 1, \dots, L-1$ do
6:          For $(v,u) \in E$, compute the message from node $v$ to node $u$:
                $m^l_{(v,u)} = M^l_{c_{(v,u)}}(h^l_v) = W^l_{c_{(v,u)}} h^l_v + B^l_{c_{(v,u)}}$    (16)
7:          Aggregate the messages for each node $u \in V$ according to their incoming edges $(v,u) \in E$:
                $\overline{m}^l_u = A(\{m^l_{(v,u)} | v \in \mathcal{N}(u)\}) = \sum_{v \in \mathcal{N}(u)} y_{(v,u)} m^l_{(v,u)}$    (17)
          state Update the node features for each node within the graph $u \in V$:
                $h^{l+1}_u = U^l_{p_u}(h^l_u, \overline{m}^l_u) = \sigma(\overline{m}^l_u + W^l_{p_u} h^l_u + B^l_{p_u})$    (18)
8:      end for
9:      Output the concatenation $H(t)$ through a readout function:
            $H(t) = \underset{u \in V}{\text{READOUT}}(\{h^L_u | u \in V_{out}\})$
            $= h^L_{robot} \parallel h^L_{target} \parallel \text{mean}(\{\mathbf{h}^L_o\}) \parallel \max(\{\mathbf{h}^L_o\}) \parallel \min(\{\mathbf{h}^L_o\})$    (19)
10: end for
```

indicates the physical variables associated with the robot, target, and obstacles. This consolidated tensor is then partitioned into N sub-tensors and allocated to the corresponding N nodes. The set of node features is denoted as $\{h_u^0 | u \in V\}$.

Following the convention in Section 2, each layer of the HGNN consists of three steps of operations, i.e., message calculation M (5), message aggregation A (6) and node embedding update U (7). In FAM-HGNN, the message function M is dependent on the edge type $c_{(v,u)}$. Specifically, each edge type is assigned with an individual instance of linear transformation message function (16) for computing the message from a source node v to a destination node u. To emphasise the strong connections and weaken the influence of trivial connections, the message aggregation is defined as a weighted summation that takes the precalculated attention $y_{v,u}$ as the weighting coefficients (17). The node embedding for each node $u \in V$ is finally updated based on the aggregated extraneous message $\overline{m}_u^l$ and a self-loop message $W_{p_u}^l h_u^l + B_{p_u}^l$ (18). σ denotes the nonlinear activation, and the self-loop linear transformation holds distinct instances dependent on node type p_u. The above processes constitute a one-layer operation within the HGNN, executed for L times, where L is the total layer number. Notably, the aforementioned functions (16) (17) (18) are all dependent on the layer number, i.e., function instances are distinct across different layers. The above is designed on account of the inherent characteristics of the obstacle avoidance problem. Specifically, each component within the problem should exhibit a diverse range of physical representations, resulting in the presence of distinct physical messaging mechanisms.

To attain the transfer capability and incorporate it with the consequent MLPs, a novel readout function is defined as Eq. (19). Where $\{\mathbf{h}_o^L\} = (\{h_u^L | p_u = p_{obstacle}\})$ is the set comprise all obstacle node embedding in the output layer. The output from the readout function is a mixture of structural inductive bias and high-dimensional embeddings obtained by propagating the original observation on a graph. The output concatenation is fed into conventional MLP and generates the readable output of action a_t and value v_t. For a better interpretation of the FAM-HGNN structure, based on the graph in Fig. 3, an instance that only refers to the propagation of the robot node is illustrated in Fig. 7.

The whole FAM-HGNN block can be regarded as a shared network of the policy and value networks that extract the graph features. When the RL agent generates the prediction of actions and values, the FAM-HGNN will work as an auxiliary graph representation learning layer for extracting spatial information. The parameters within the FAM-HGNN block will be concurrently learned when conducting the RL-based backpropagation and parameter update (1).

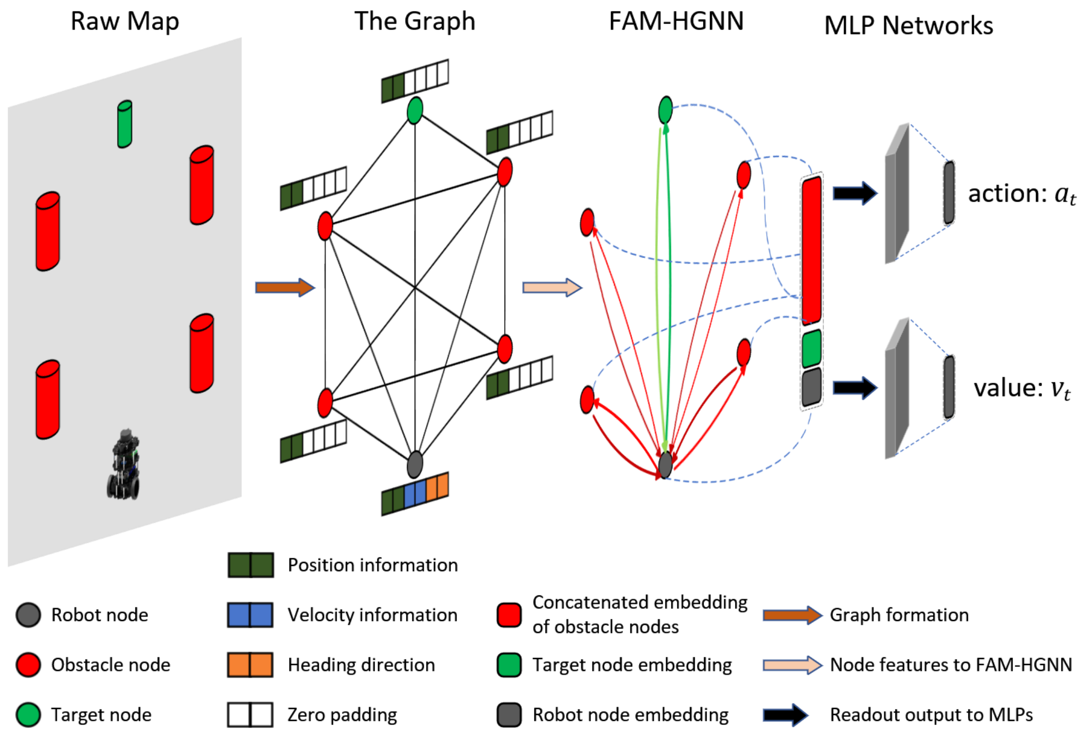

Figure 7: Schematic representation of the FAM-HGNN. Note that the graph structure here is the same as in Fig. 3, but we only illustrate the edges that are connected with the robot node for a simpler interpretation.

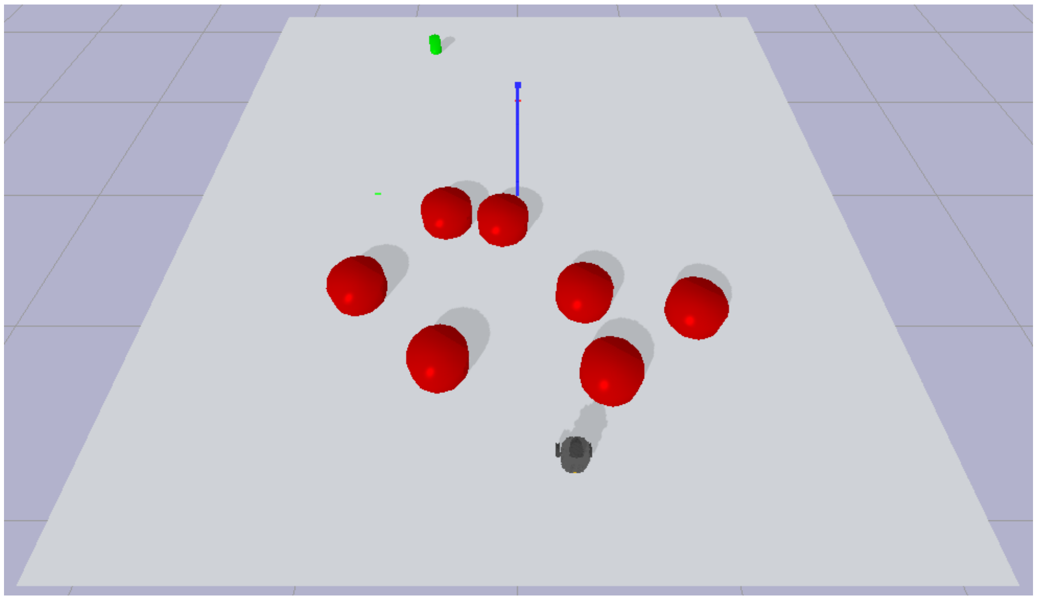

Figure 8: An illustration of the customised environment. The grey plane is the horizon of the map. The dark grey component is the Turtlebot3 Burger, the red cylinders are the obstacles and the green cylinder denotes the target point.

4. Implementation

In this section, a customised obstacle avoidance environment is exploited to evaluate the proposed FAM-HGNN method. The customised obstacle avoidance environment and simulation details are introduced in Section 4.1. Its reward setting is introduced in Section 4.2. The choice of baselines is given in Section 4.3. The hyperparameter settings for RL and GNN candidates are provided in Section 4.4.

4.1. Environment and Simulation

The simulations were conducted on an 8-core, 2.2Ghz i7-10870 CPU with 16 GB memory and an Nvidia GeForce RTX 3060 GPU with 6 GB memory. All models were implemented in PyTorch v1.13 and accelerated through CUDA 11.7. GNNs were constructed and accelerated using the same PyTorch backend with the assistance of Deep Graph Library (DGL) 1.0.2 Wang et al. (2019) APIs. To validate the proposed FAM-HGNN method, a customised OpenAI gym (Brockman et al., 2016) environment is designed based on Pybullet physical simulation engine (Coumans and Bai, 2016–2022), which is illustrated in Fig. 8.

This environment has three primary components, i.e., robot, target and obstacles. At each time instant t in this environment, the RL agent will receive an observation state vector $s_t = s_{robot} \| s_{target} \| s^1_{obstacle} \| \ldots \| s^{n_o}_{obstacle}$. The state vector is fed into a policy model to output a control signal a_t for controlling the behaviour of the robot. The robot follows all the dynamics and kinematics of the differential mobile robot. Specifically, for the differential mobile robot in this environment, $a_t = [v_{left\_wheel}, v_{right\_wheel}]$, denotes the velocity control command for left wheel and right wheel. The physical model of the differential mobile robot is chosen as Turtlebot3 Burger (Robotis, 2017). The actual implementation of the environment is a transplant version of its original model file without the lidar components. Additionally, The wheel speed is amplified proportionally to expedite the environmental interaction process. The target and the obstacles are defined as cylinder areas with a radius equal to $0.05m$ and $0.17m$, respectively. Notably, the obstacle areas are free of physical collision. All components are placed in a $4.5m \times 4.5m$ square plane with a lateral friction feature. The experiments were conducted by assuming that the agent can perceive the location information of all the components within the map.

Based on this environment, a fully connected graph that is analogous to the graph in Fig. 3 can be generated. The graph has three node types ($P = 3$) for indicating robots, targets and obstacles. It has seven edge types ($C = 7$), indicating the relationships of robot-target, target-robot, robot-obstacle, obstacle-robot, target-obstacle, obstacle-target and obstacle-obstacle.

4.2. Reward Settings

Reward function r plays a prominent role in reinforcement learning problems (1). In the customised environment, the reward function can be categorised into two parts: terminal rewards and non-terminal rewards. Specifically, the episode will terminate only in three situations: 1) reach the target 2) run out of the map 3) reach the predefined timestep limitation for each episode. For each of the aforementioned circumstances, the agent will receive a $+50$, -10 and 0 instant reward, respectively.

Unlike other obstacle avoidance environment settings, the collision in this environment is set as a non-terminal state. This setting expedites the global exploration by enabling the agent to explore the transitions after the collision. In summary, the reward function r_t in non-terminal timesteps is comprised of four parts as follows:

$$r_t = r_g + r_o + r_c + r_p \tag{20}$$

The first term r_g is an incentive-guiding reward that encourages the robot to minimise the distance between its current position and the target:

$$r_g = \begin{cases} c_d e_{d_t} & \text{if } e_{d_t} \geq 0 \\ \kappa c_d e_{d_t} & \text{if } e_{d_t} < 0 \end{cases} \tag{21}$$

where e_{d_t} denotes the distance variation between the robot and the target. c_d is a scaling factor mapping it to r_g. It should be noted that the reward function incorporates a discount factor κ when e_{d_t} is negative. This adjustment avoids large negative rewards for behaviours like avoiding obstacles or taking circuitous routes to reach the target. Reducing the penalty for such behaviours encourages effective obstacle avoidance navigation in complex and dynamic environments. The second term r_o is a guiding penalty concerning the obstacle avoidance functionality, mathematically defined as:

$$r_o = \sum_{i=1}^{n} \begin{cases} c_{d_o} e_{d^i_{ot}} & \text{if } d^i_{ot} \leq \rho \\ 0 & \text{if } d^i_{ot} > \rho \end{cases} \tag{22}$$

where $e_{d_{ot}}$ represents the distance variation between the robot and the obstacles. c_{d_o} is its corresponding coefficient. ρ denotes the influential area of the obstacle. Only if the distance between the robot and one of the obstacles is under this certain range will the term r_o work. To further prevent collision behaviours, a collision-trigger penalty r_c is introduced as follows:

$$r_c = \sum_{i=1}^{n} \begin{cases} p_c & \text{if collision} \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

The agent will receive a penalty p_c when the robot overlaps the region of obstacles. Otherwise, this term equals to 0. The last term $r_p = p$ is a constant time penalty for facilitating the robot to reach the target as soon as possible. The overall reward function parameters are given in Table 3.

Table 3: Reward function parameters

Parameter	Value
Target guiding coefficient c_d	20
Obstacle guiding coefficient c_{d_o}	-40
Collision penalty p_c	-0.2
Time penalty p	-0.01
Avoiding coefficient κ	0
Obstacle penalty range ρ	0.6

4.3. Baselines

We choose the Baselines3 PPO implementation (Raffin et al., 2021) as the basic RL architecture for breeding methods that rely on either MLPs or GNNs. Considering the attribute of obstacle avoidance, the relationships between components (robots, targets and obstacles) and their relative importance could make a significant impact. In this case, the following baselines are chosen:

GCN: Graph Convolutional Network (GCN) (Kipf and Welling, 2016) is a classical GNN model that perform graph aggreagation.

RGCN: Relational Graph Convolutional Network (RGCN) (Schlichtkrull et al., 2018) is similar to GCN but takes different relationships into account.

GAT: Graph Attention Network (GAT) (Veličković et al., 2017) integrate graph aggregations with the self-attention mechanism.

4.4. Hyperparameters

PPO hyperparameters are determined by grid search, which is provided in table 4. Specifically, for the MLP-based PPO, the input features first fed into a $[64, 64]$ size shared network. After which, the output from the shared network is fed into policy and value function networks to obtain the action and value prediction, respectively. The embedding size of both policy and function networks is $[64]$. The term *n steps* indicates the number of steps to run for each environment per update. For each environment, once 20480 transitions are collected, the simulation is paused and then moved to the training. The training uses the Adam optimiser with a fixed learning rate of 3×10^{-4}. The epoch number is 40 and the batch size is 2048. To mitigate the randomness led by a highly stochastic environment, for each model, we run 6 experiments with different seed settings. To further mitigate the environmental randomness and increase the sample efficiency, 4 parallel environments with different seeds are run for each experiment. For each experiment, the agent is learned with 300 iterations, namely $24576000 = 300 \times 20480 \times 4$ timesteps.

Hyperparameters of the GNN candidates are provided in Table 5. Similar to the structure of the MLP-based method, the network structures of all GNN-based blocks consist of two parts: 1) a shared network with two GNN layers, which take as inputs the node features and output a condensed graph output. 2) Two individual MLP networks that take as input condensed graph output to predict policies and values, respectively. Note that only RGCN and FAM-HGNN are relational GNNs. Node type is only applicable in FAM-HGNN. Besides, GAT is the only self-attention-based method that needs multi-head (Vaswani et al., 2017).

5. Simulation Results

To fully investigate the effectiveness of the proposed FAM-HGNN method, the following four experiments are conducted in this section:

Table 4: PPO Hyperparameters

Hyperparameter	Value
Shared network size	[64, 64]
Policy network size	[64]
Value function network size	[64]
Learning rate	3×10^{-4}
n steps	20480
Batch size	2048
Number of epochs	40
Discount factor γ	0.99
GAE parameter λ	0.97
Value function coefficient	0.5
PPO clip range	0.2
Gradient clip range	0.5
Target KL	0.005

Table 5: GNNs Hyperparameters

	GAT	GCN	RGCN	FAM-HGNN
Num. GNN layers	2	2	2	2
GNN embedding sizes	[6, 10, 8]	[6, 10, 8]	[6, 10, 8]	[6, 10, 8]
Num. relationships	N/A	N/A	7	7
Num. node types	N/A	N/A	N/A	3
Num. heads	3	N/A	N/A	N/A
MLP network sizes	[64]	[64]	[64]	[64]

- To evaluate the training performance concerning the episodic return and success rate, we conduct the training of FAM-HGNN and other GNN-based structures under the customised obstacle avoidance navigation control environment. (Section 5.1)

- To reconfirm the effectiveness of the FAM structure, an ablation study is conducted as follows: 1) incorporate the FAM structure with other GNN-based methods. 2) Abandon the FAM structure from FAM-HGNN and implement the HGNN component solely. (Section 5.2)

- To investigate the size transfer capability, the trained models are exploited in similar scenarios with diverse numbers of obstacle settings. We further implement an artificial potential field as a baseline for investigating the effectiveness of the transferred models. (Section 5.3)

- To investigate the sensitivity of the proposed model to different hyperparameters, a sensitivity analysis is conducted in Section 5.4.

5.1. Training Performance

Training is conducted on the customised obstacle avoidance environment with 7 obstacles to evaluate the performance of all structures. We design two measurement indices as follows:

- Average Episodic Return: The episodic return is computed using Eq. 1. The average episodic return is the mean value of the episodic returns in the latest 100 episodes.

- Success Rate: A success episode is when the mobile robot is driven from its initial position to the target without any collision and boundary violation under the episodic timestep limitation. When calculating the success rate, the latest 100 episode is considered. Success rate equals $n_{successful\_episode}/100$, where $n_{successful\_episode}$ is the number of successful episodes within the latest 100 episode.

The result plots shown in Fig. 9 are generated based on means and standard deviations from all experiments. The results show that in the training cases, the episodic return and success rate of the FAM-HGNN method can reach about 100 and 80%, respectively. The MLP-based method can only achieve about 40 episodic return and 15% success rate, while the other GNN methods demonstrate worse performance. Note that the random exploration process is likely to cause a collision when the

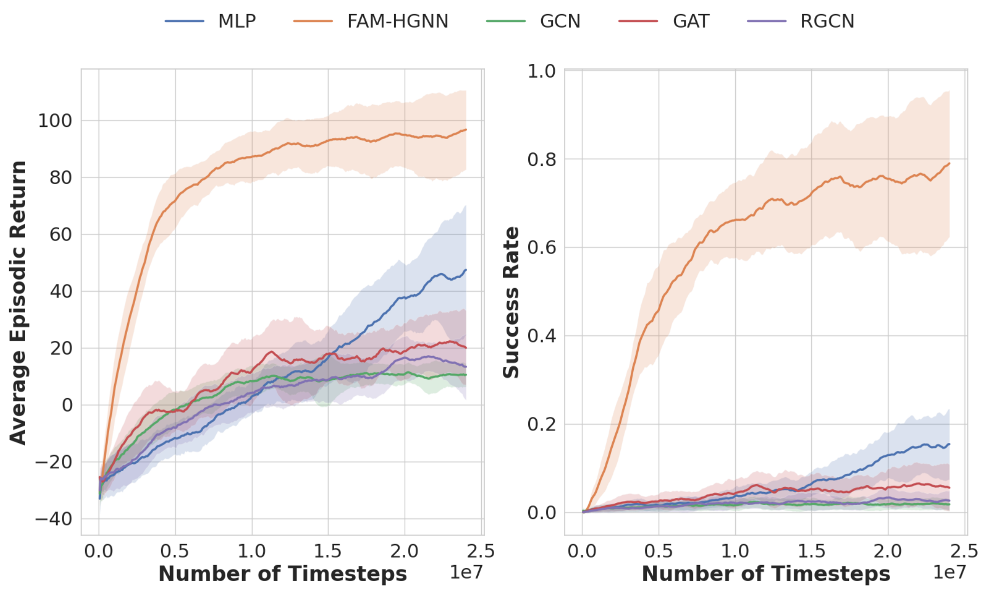

Figure 9: Training performance of all algorithms on customised obstacle avoidance environment with 7 obstacles setting.

robot is near an obstacle, significantly reducing its success rate in training cases. This assumption is validated by the results in the test stage (Section 5.3). Overall, the above result validates the superiority of the proposed FAM-HGNN structure.

5.2. *Ablation Study*

An ablation study is conducted in this section to investigate the influence of the FAM structure. Specifically, for FAM-HGNN, we compare its training results with the HGNN, which is the version without the FAM structure. For GCN and RGCN, we implemented them with an additional FAM component. GAT is already an attention method and is mutually exclusive with the FAM. Note that the hyperparameters of FAM-GCN, FAM-RGCN and HGNN are the same as those of GCN, RGCN and FAM-HGNN, respectively. By setting the MLP-based method as a baseline, Fig. 10 demonstrates the comparison results of all FAM-based methods and their corresponding non-FAM-based methods. The following conclusions are drawn according to the results:

1) The results demonstrate an obvious gap between all FAM-based GNN structures (FAM-HGNN, FAM-RGCN, FAM-GCN) and their corresponding GNN methods without FAM components (HGNN, RGCN, GCN). FAM can considerably promote the training performance of all GNN methods, which validates the effectiveness

of the FAM structure. This is because the FAM evaluate the relative importance of different edges, thereby providing a strong reference to determine which connections should be focused on in graph propagation processes.

2) The results also show that the outperformance of the FAM-HGNN method is mainly attributed to the FAM components. This is validated through the comparison between HGNN and other non-FAM GNN methods, as well as the comparison between FAM-based and non-FAM methods. Even though the FAM-HGNN method overperforms the other methods, the HGNN structure exhibits an average level of performance compared with other GNN candidates. Moreover, all FAM-based methods show a significant improvement compared with non-FAM structures. These phenomenons indicate the FAM structure's efficacy.

3) The MLP-based method outperforms all non-FAM methods while inferior to FAN-based methods. This phenomenon indicates two conclusions. First, the inductive bias is missing since all non-FAM methods work on a fully connected graph. It leads to an invalid graph propagation process and fails to learn the structural information. Second, the promising results of the FAM-based methods justify that FAM could transform a fully connected graph into a soft-connected graph with abundant relational inductive bias through an attention mechanism.

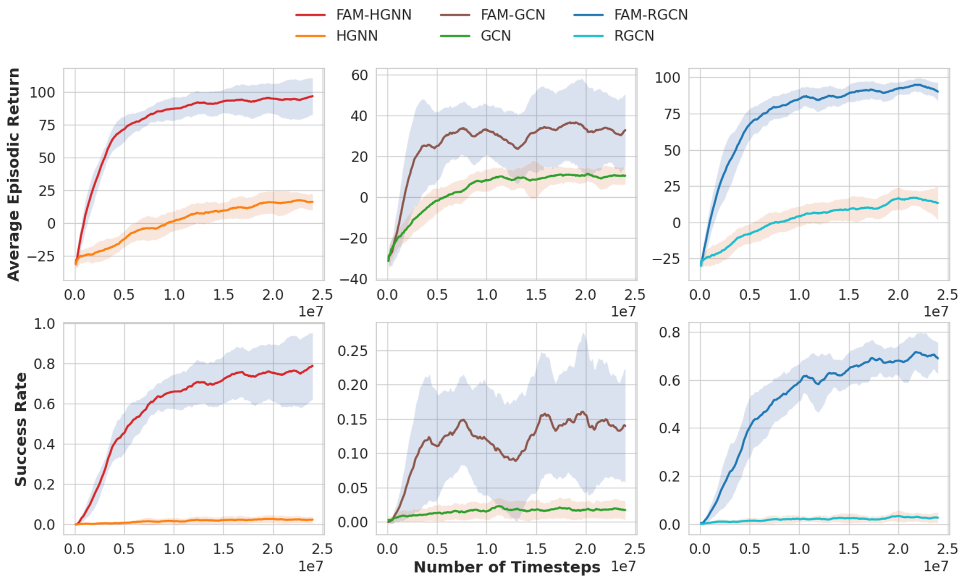

Figure 10: FAM-HGNN and APF test results with diverse obstacles setting.

5.3. Size Transfer Comparison

Size transfer promotes reusing the knowledge from the trained models to resolve tasks with variant input sizes. It is the crux of applying the RL control method to practical control problems. Most of the benchmark RL algorithms neglect this issue due to the limitation caused by default MLP network structures. GNN can generate a dimension-invariant output regardless of the number of nodes without rectifying the network structure. This saliency brings its potential for extending transfer capability into the conventional RL algorithm. The size transfer under the customised obstacle avoidance environment is defined as applying the aforementioned trained models in different numbers of obstacle scenarios without any additional training and tuning. We investigate the transfer capability of FAM-GNN by comparing it with other GNN methods. We further implement artificial potential field (APF) (Khatib, 1986) as a learning-free benchmark for indicating the environmental difficulty. Notably, as the training performance of the MLP-based model is already unsatisfactory and whose implementation with size transfer capability requires additional dimensionality reduction layers such as TreeNet (Wang et al., 2018), we exclude its size transfer from this work. We obtain the average episodic return and success rate results by exploiting the learned models to different scenarios with 100 episodes run, as shown in Table 6. Besides, more detailed FAM-HGNN and APF performance curves to different obstacle number settings are shown in Fig. 11.

Num. Obstacles	3	5	7	9	11	13
APF	**100% (108.43)**	**96% (106.54)**	87% (102.17)	82% (98.60)	75% (92.77)	**63**% (84.39)
GAT	22% (43.58)	8% (33.76)	5% (30.23)	6% (31.54)	2% (14.78)	3% (-1.47)
GCN	6% (27.82)	3% (20.12)	3% (15.07)	0% (6.29)	3% (-2.80)	% (-17.28)
FAM-GCN	0% (13.90)	11% (39.28)	34% (60.18)	23% (48.31)	11% (38.74)	4% (25.02)
RGCN	3% (24.25)	2% (21.00)	1% (9.41)	0% (-3.40)	1% (-14.62)	0% (-31.31)
FAM-RGCN	56% (94.36)	61% (100.06)	76% (98.90)	75% (95.37)	46% (74.76)	25% (55.13)
HGNN	4% (28.70)	4% (21.55)	1% (18.25)	1% (13.73)	4% (4.10)	0% (-10.10)
FAM-HGNN	96% (107.50)	93% (107.24)	**96% (109.29)**	**96% (109.20)**	**76% (97.68)**	62% **(87.77)**

Table 6: The success rate (outside the brackets) and the average episodic accumulated discounted rewards (inside the brackets) of the size transfer performance. The bold numbers denote the best performance of success rate or average episodic accumulated discounted rewards under specific obstacle numbers among the algorithms.

The action generation in the training stage follows a Gaussian distribution sample process while outputting a deterministic value in the test stage. In this case, a better performance is normally demonstrated in the test stage, even though the environment remains the same.

The results show that FAM-HGNN successfully realised size transfer in the customised environment with diverse obstacle number settings. For FAM class methods, the best performance is normally shown in the same scenarios as their training. Their performance tends to decline slightly in less obstacle scenarios while demonstrating

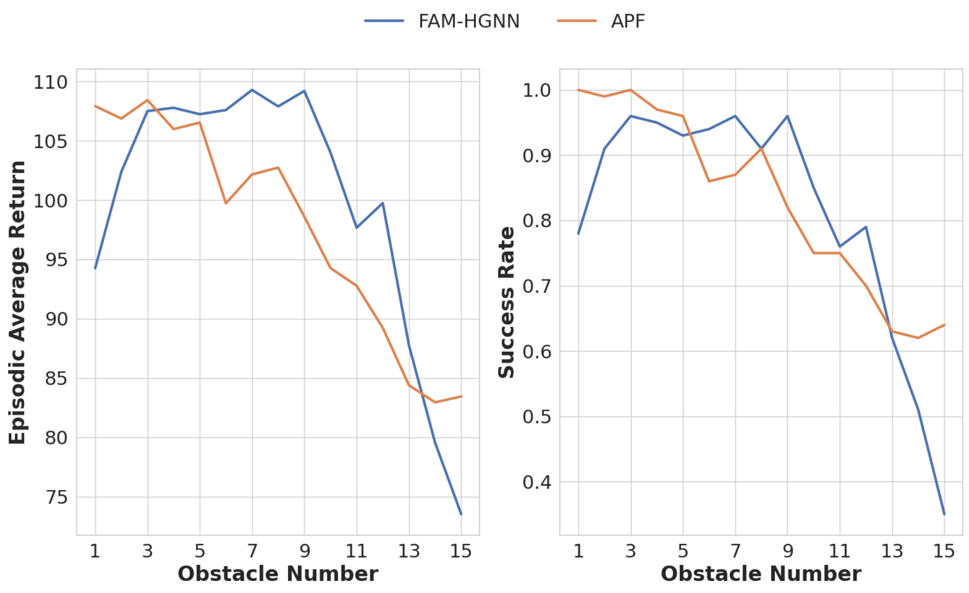

Figure 11: FAM-HGNN and APF test results with diverse obstacles setting.

significant inferiority in scenarios with more obstacles. We assume the limitation of the size transfer capability is the cause. Furthermore, the increasing task difficulty also results in inferior performance in large obstacle number settings. These are justified by comparing the performance of APF in different obstacle number settings (Fig. 11). The performance of APF is as expected: the more obstacles, the worse the performance.

5.4. *Sensitivity Analysis*

In this section, a sensitivity analysis is conducted to investigate the influence of 1) policy and the value function networks MLP embedding size and 2) the choice of readout function within the FAM-HGNN method.

The training results regarding different embedding sizes are given in Fig. 12. In Fig. 12, we can see that the FAM-HGNN performs well when the MLP embedding size is relatively small. Notably, the readout output size is set as 40. In this case, we assume that to order a considerable training performance, the MLP embedding size should be smaller or similar to the readout output size. The results also indicate that FAM-HGNN is robust to the MLP embedding size within a certain range.

Furthermore, to investigate the effect of the readout function, we conduct the training with the same settings in Section 5.1. We defined the following 4 different readout functions as candidates:

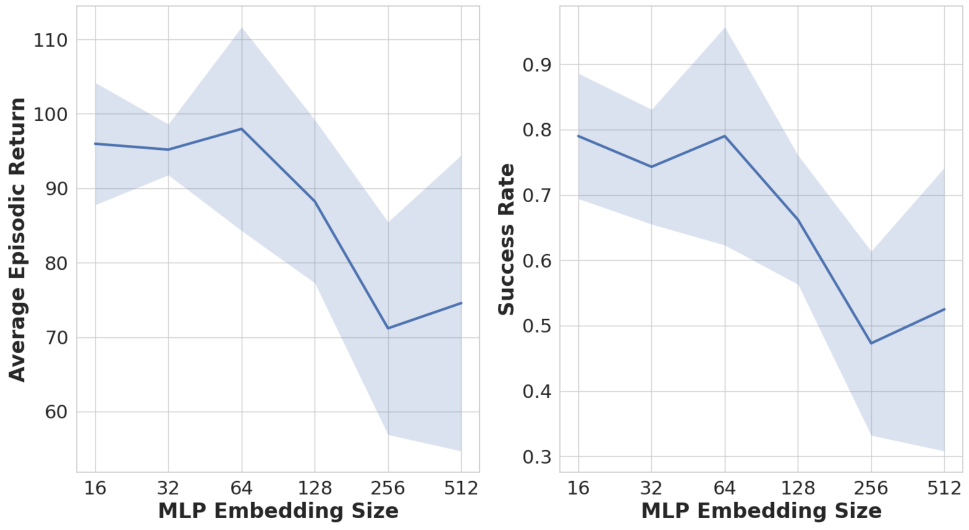

Figure 12: FAM-HGNN and APF test results with diverse obstacles setting.

- Tate robot node and target node: $H(t) = h^L_{robot} \parallel h^L_{target}$
- Mean: $H(t) = \text{mean}\{h^L_u | u \in V\}$
- Max: $H(t) = \max\{h^L_u | u \in V\}$
- Min: $H(t) = \min\{h^L_u | u \in V\}$

where the first readout function concatenates the node embedding of the robot node and the target node as the graph output. The remains are classic GNN readout methods, namely element-wise mean, max, min operations over all nodes, respectively.

The results in Fig. 13 show that all other readout functions have a degraded performance compared with the readout function in Eq. (19). Specifically, the element-wise mean, max, min readout functions fail to train the agent while the robot & target readout function demonstrate a relative ascendant performance. This is due to the characteristics of the GNN-RL-based control problems. Unlike typical GNN applications such as large-scale homogeneous graph classification or edge prediction problems, the discrepancy of the nodes and edges has a great impact on graph propagation in GNN-RL-based control problems. Therefore, the resulting nodes' embeddings in the output layer indicate distinct representations. The element-wise mean, max, min readout functions miss discriminatory treatment to the nodes and obtain an inexpressive graph representation. Meanwhile, even though the robot & target readout function considers a subset of the node embedding, its graph representation is tangible and results in passable performance.

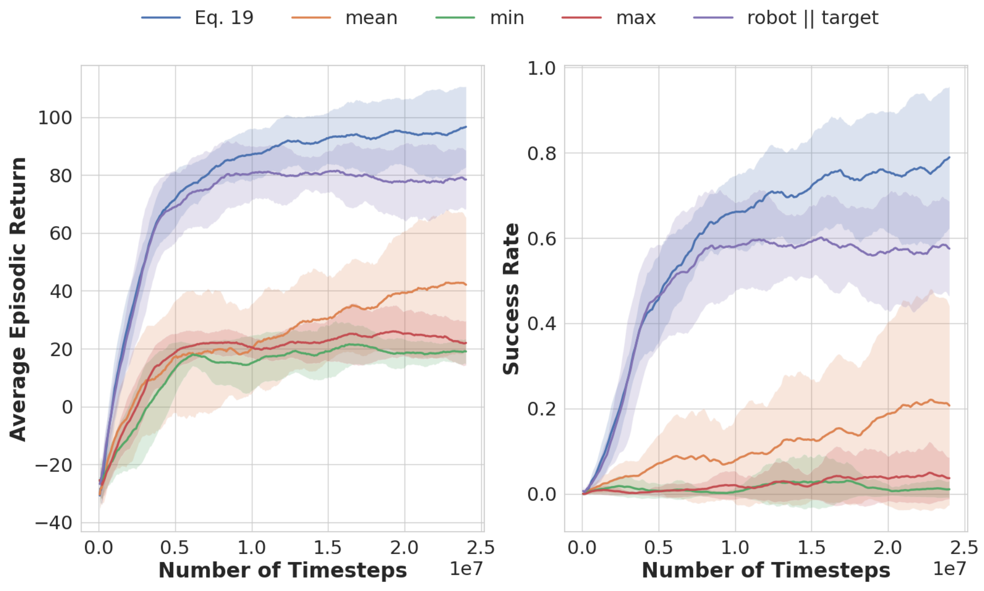

Figure 13: FAM-HGNN and APF test results with diverse obstacles setting.

6. Conclusion

In this paper, we propose a novel FAM-HGNN framework for resolving the obstacle avoidance navigation control problem. To the best of our knowledge, FAM-HGNN is the first framework that exploits a fuzzy system to achieve the attention mechanism of the GNN. It avoids artificial graph formation processes and considerably promotes the performance of GNN-based RL by propagating messages discriminatively on explicit physical dependence. Experimental results on a customised obstacle avoidance environment show that the FAM-HGNN outperforms the other GNN-based and MLP-based RL agents. Specifically, FAM-HGNN demonstrates its superiority in success rate and episodic return, which is up to 96% and 109.29 in the testing cases, respectively. In the size-transfer comparison, FAM-HGNN shows its strong size-transfer capability in obstacle avoidance scenarios with different obstacle number settings. Despite the inspiring results, FAM-HGNN demonstrates limited resistance to environmental randomness. Moreover, the numerous function instances conditioned on node types and edge types bring computational burden and increase the training time. In future work, we will extend FAM-HGNN to real-life multi-robot and multi-target scenarios and explore how to compress the function instances within the GNNs.

Acknowledgement

This work was supported by King's College London.

Disclosure statement

No potential conflict of interest is declared.

References

Abadal, S., Jain, A., Guirado, R., López-Alonso, J., Alarcón, E., 2021. Computing graph neural networks: A survey from algorithms to accelerators. ACM Computing Surveys (CSUR) 54, 1–38.

Blumenkamp, J., Morad, S., Gielis, J., Li, Q., Prorok, A., 2022. A framework for real-world multi-robot systems running decentralized gnn-based policies, in: 2022 International Conference on Robotics and Automation (ICRA), IEEE. pp. 8772–8778.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W., 2016. Openai gym. arXiv preprint arXiv:1606.01540 .

Chen, Y., Liu, C., Shi, B.E., Liu, M., 2020. Robot navigation in crowds by graph convolutional networks with attention learned from human gaze. IEEE Robotics and Automation Letters 5, 2754–2761.

Coumans, E., Bai, Y., 2016–2022. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`.

Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T., 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning 110, 2419–2468.

Khatib, O., 1986. Real-time obstacle avoidance for manipulators and mobile robots. The international journal of robotics research 5, 90–98.

Kipf, T.N., Welling, M., 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 .

Lalish, E., Morgansen, K.A., 2012. Distributed reactive collision avoidance. Autonomous Robots 32, 207–226.

Li, Q., Gama, F., Ribeiro, A., Prorok, A., 2020. Graph neural networks for decentralized multi-robot path planning, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 11785–11792.

Li, Q., Lin, W., Liu, Z., Prorok, A., 2021. Message-aware graph attention networks for large-scale multi-robot path planning. IEEE Robotics and Automation Letters 6, 5533–5540.

Liu, Y., Wang, W., Hu, Y., Hao, J., Chen, X., Gao, Y., 2020. Multi-agent game abstraction via graph attention neural network, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 7211–7218.

Liu, Z., Zhai, Y., Li, J., Wang, G., Miao, Y., Wang, H., 2023. Graph relational reinforcement learning for mobile robot navigation in large-scale crowded environments. IEEE Transactions on Intelligent Transportation Systems .

Mehran, K., 2008. Takagi-sugeno fuzzy modeling for process control. Industrial Automation, Robotics and Artificial Intelligence (EEE8005) 262, 1–31.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 .

Peng, H., Li, J., Gong, Q., Wang, S., Ning, Y., Yu, P.S., 2018. Graph convolutional neural networks via motif-based attention. arXiv preprint arXiv:1811.08270 .

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N., 2021. Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research .

Robotis, 2017. TurtleBot3 Burger. `http://emanual.robotis.com/docs/en/platform/turtlebot3/o` [Online; accessed 28-April-2023].

Schlichtkrull, M., Kipf, T.N., Bloem, P., Berg, R.v.d., Titov, I., Welling, M., 2018. Modeling relational data with graph convolutional networks, in: European semantic web conference, Springer. pp. 593–607.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 .

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., 2016. Mastering the game of go with deep neural networks and tree search. nature 529, 484–489.

Stentz, A., 1994. Optimal and efficient path planning for partially-known environments, in: Proceedings of the 1994 IEEE international conference on robotics and automation, IEEE. pp. 3310–3317.

Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Takagi, T., Sugeno, M., 1985. Fuzzy identification of systems and its applications to modeling and control. IEEE transactions on systems, man, and cybernetics , 116–132.

Tolstaya, E., Gama, F., Paulos, J., Pappas, G., Kumar, V., Ribeiro, A., 2020. Learning decentralized controllers for robot swarms with graph neural networks, in: Conference on robot learning, PMLR. pp. 671–682.

Tzes, M., Bousias, N., Chatzipantazis, E., Pappas, G.J., 2023. Graph neural networks for multi-robot active information acquisition, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 3497–3503.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems 30.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 .

Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al., 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature 575, 350–354.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., et al., 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv preprint arXiv:1909.01315 .

Wang, T., Liao, R., Ba, J., Fidler, S., 2018. Nervenet: Learning structured policy with graph neural networks, in: International conference on learning representations.

Zadeh, L., 1965. Fuzzy sets. Inform Control 8, 338–353.

Zhao, X., Xia, L., Zhang, L., Ding, Z., Yin, D., Tang, J., 2018. Deep reinforcement learning for page-wise recommendations, in: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 95–103.

Appendix A. APF Implementation Details

APF is a robot path planning method that perfectly suits our environment formulation. In our obstacle avoidance environment, APF formulates targets as attractive fields U_a and obstacles as repulsive fields U_r. The mobile robot is driven by the gradient direction of a compound field U_{total}, which synthesises the influence of both components. Considering the obstacle avoidance setting with one target and multiple obstacles, the universal potential field is written as:

$$U_{total} = U_a + U_r \tag{A.1}$$

$$U_a = \begin{cases} \frac{1}{2}\zeta_1 \|q, q_{goal}\|_2^2 & \text{if } \|q, q_{goal}\|_2^2 \leq \rho_t \\ \frac{1}{2}\zeta_2 \|q, q_{goal}\|_2 & \text{if } \|q, q_{goal}\|_2 > \rho_t \end{cases} \tag{A.2}$$

$$U_r = \sum_{i=1}^{n} U_r^i = \sum_{i=1}^{n} \begin{cases} \frac{1}{2}\eta(\frac{1}{\|q, q_o^i\|_2} - \frac{1}{\rho_o})^2 & \text{if } \|q, q_o^i\|_2 \leq \rho_o \\ 0 & \text{if } \|q, q_o^i\|_2 > \rho_o \end{cases} \tag{A.3}$$

based on which, their corresponding gradient expression is given as follows:

$$F_{total} = F_a + F_r = -\nabla U_a - \nabla U_r \tag{A.4}$$

$$\nabla U_a = \begin{cases} \zeta_1 \|q, q_{goal}\|_2 & \text{if } \|q, q_{goal}\|_2 \leq \rho_t \\ \zeta_2 \|q, q_{goal}\|_2 & \text{if } \|q, q_{goal}\|_2 > \rho_t \end{cases} \tag{A.5}$$

$$\nabla U_r = \sum_{i=1}^{n} \nabla U_r^i = \sum_{i=1}^{n} \begin{cases} \eta(\frac{1}{\rho_o} - \frac{1}{\|q, q_o^i\|_2})\frac{1}{\|q, q_o^i\|_2^2} \nabla \|q, q_o^i\|_2 & \text{if } \|q, q_o^i\|_2 \leq \rho_o \\ 0 & \text{if } \|q, q_o^i\|_2 > \rho_o \end{cases} \tag{A.6}$$

As the investigation of APF itself is beyond the scope of this work, and there is no uniform regulation for designing potential functions, we provide effective potential functions conditioned on our environment settings as in previous sections. Note that the original APF is a path planning methodology rather than a motion control policy, therefore, we implement the kinematic control by setting constant linear velocity v while the angular velocity w is oriented by the potential force F_{total}. Furthermore, since the direct control input of the differential mobile robot is the velocity of both the left wheel and right wheel, we finally transfer it into the corresponding control signal by its kinematic model:

$$
\begin{aligned}
v &= 0.75 \\
w &= c_w \theta_{\theta_h, F_{total}} \\
w_l &= \frac{2v + wD}{2r} \\
w_r &= \frac{2v - wD}{2r}
\end{aligned}
$$

Considering the environment's complexity and to comply with the theoretical maximum linear velocity of 1.0725 associated with the learning-based method, we have opted to adopt a constant linear velocity of 0.75 in this study. The sole control strategy employed involves proportionally mapping the angular difference, denoted as θ_{ϕ_h, ϕ_F}, between the heading direction ϕ_h and the synthetic force direction ϕ_F to an angular velocity command w. Given that the direct control input for the differential mobile robot is based on the velocities of its two wheels, it is necessary to further translate the linear and angular velocities into corresponding wheel velocities, accounting for the robot's kinematic characteristics. The values of r and D correspond to the wheel radius and the distance between the two wheels, respectively, and are consistent with the model of the Turtlebot3 Burger, where r is equal to 0.033 and D is equal to 0.22. An overview of the hyperparameters employed in the Artificial Potential Field (APF) approach is presented in Table A.7.

Table A.7: APF parameters

Parameter	Value
ζ_1	5
ζ_2	7.5
ρ_t	1.5
η	8
ρ_o	0.3
c_w	2