Expressive Attentional Communication Learning using Graph Neural Networks

Yu Quan Chong

CMU-RI-TR-24-51

July 31, 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Prof. Katia Sycara, *chair*
Prof. Jiaoyang Li
Benjamin Freed

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my family and friends.

Abstract

Multi-agent reinforcement learning presents unique hurdles such as the non-stationary problem beyond single-agent reinforcement learning that makes learning effective decentralized cooperative policies using an agent's local state extremely challenging. Effective communication to share information and coordinate is vital for agents to work together and solve cooperative tasks, as the ubiquitous evidence of communication in nature would highlight. Hence, communication within a framework between agents can potentially alleviate the problems of non-stationarity and partial observability while being highly scalable. This work examines graph neural networks (GNNs), whose message-passing mechanisms synergize well with differentiable communication learning (CL) methods. We investigate the inherent limitations of attention-based GNNs regarding their expressive power and propose a new GNN, Graph Attention Isomorphism Network (GAIN). We evaluated GAIN on the Open Graph Benchmark and showed that it outperforms state-of-the-art GNNs on various graph, link, and node property tasks across a GNN design space using GraphGym. We incorporate GAIN into a simple architecture called Graph Communication Network (GCNet) and evaluate it on tasks in the StarCraft Multi-Agent Challenge. We show that it outperforms GCNet using state-of-the-art GNNs and other baseline CL methods.

Acknowledgments

I am deeply grateful to my advisor, Prof. Sycara, for her invaluable guidance and support throughout my academic endeavors. I sincerely thank Prof. Li for her mentorship in my research pursuits. I want to express my heartfelt thanks to Prof. Sartoretti for his guidance throughout my research journey. Additionally, I am thankful to Dana Hughes, Joseph Campbell, Simon Benjamin Stepputtis, Huao Li, Ini Oguntola, Silong Yong, Xinran Zhao, and Yutong Wang for their valuable contributions and kind assistance. Lastly, I extend my immense gratitude to my family and friends for their unwavering support and encouragement.

Funding

This work was partially supported by DARPA award HR001120C0036 and AFOSR award FA9550-18-1-0097.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Reinforcement learning (RL) has seen immense success in areas such as robotics [47] and games [57, 79]. However, many real-world problems such as autonomous driving [77], transportation systems [2], and warehouse logistics [96] present themselves as multi-agent settings to which multi-agent reinforcement learning (MARL) offers solutions. However, there are challenges in MARL beyond single agent RL such as the non-stationary problem [66] arising from learning in a dynamic environment with changing policies from other learning agents that makes learning effective decentralized cooperative policies using an agent's local state extremely challenging. Effective communication to share information and coordinate is vital for agents to work together and solve cooperative tasks, as the ubiquitous evidence of communication in nature would highlight. For example, bees perform a waggle dance to communicate the location of food sources to other members of the hive [9] and dolphins use echolocation to communicate and navigate underwater, helping them find food and avoid obstacles [16].

This work examines graph neural networks (GNNs), which are a powerful class of neural networks designed to perform inference on graph-structured data. GNNs are designed to take advantage of the relationships and interactions between nodes, representing entities, and edges, representing connections or relationships, to learn meaningful representations. This capability makes them particularly suitable for a wide range of applications, including social network recommendation systems [17] and molecular chemistry [90]. In addition, its message-passing mechanism is particularly suitable for communication learning (CL) because it allows nodes, representing agents, to iteratively exchange and aggregate information from their neighbors, hence synergizing well with differentiable CL methods. However, one of the known limitations of popular GNNs lies in their limited expressive power [59, 98] that is bounded by the 1-dimensional Weisfeiler-Leman algorithm (1-WL) [92] in distinguishing non-isomorphic graphs. Given that most GNN-based CL methods fall under the paradigm of graph decision networks (GDNs) [63], it can be shown that a GDN's capability to learn a sufficiently expressive communication framework between agents to learn policies to solve a task is tied to its GNN's expressive power based on the WL hierarchy. Hence, we investigate the inherent limitations of attention-based GNNs in terms of their expressive power and propose a new GNN, Graph Attention Isomorphism Network (GAIN), that is provably as expressive as 1-WL. We evaluated GAIN on the Open Graph Benchmark (OGB) [35] and showed that it outperforms state-of-the-art GNNs on various graph, link, and node property tasks across a GNN design space using GraphGym [100]. We incorporate GAIN into a simple GDN-based architecture called the Graph Communication Network (GCNet) and evaluate it on tasks in the StarCraft Multi-Agent Challenge (SMAC) [72]. We show that it outperforms GCNet using state-of-the-art GNNs and other baseline CL methods.

Chapter 2

Background

This chapter provides the necessary preliminaries and related work.

2.1 Preliminaries

The following sections provide the necessary technical background.

2.1.1 Notations

Let $[n] = \{1, \ldots, n\} \in \mathbb{N}$ for $n \geq 1$. Let $\{\}$ and $\{\!\!\{\}\!\!\}$ denote a set and a multiset, respectively. Formally, a multiset is a tuple $X = \langle S, \mu \rangle$, where S is the underlying set that represents the distinct elements in X and $\mu : S \rightarrow \mathbb{N}^+$ gives the multiplicity of the elements. A graph (undirected) G is a tuple $\langle V, E \rangle$ with a finite set of nodes $V(G)$ and a set of edges $E(G) \subseteq \{\{u, v\} \subseteq V(G) \mid u \neq v\}$, where the edge $\{u, v\}$ can be denoted by (u, v) or (v, u). A labeled graph is a tuple $\langle V, E, l \rangle$ with the label function $l : V(G) \cup E(G) \rightarrow \sum$, where $\sum$ is a subset of natural numbers and $l(w)$ is the label of $w \in V(G) \cup E(G)$. An attributed graph is a tuple $\langle V, E, a \rangle$ with an attribute function $a : V(G) \cup E(G) \rightarrow \mathbb{R}^d$, where $d > 0$. Hence, $a(w)$ is an attribute or continuous label of a node or edge $w \in V(G) \cup E(G)$. Let $\mathcal{G}$ denote the set of all labeled or attributed graphs. The neighborhood of $v \in V(G)$ is $N(v) = \{u \in V(G) \mid (v, u) \in E(G)\}$ and $\tilde{N}(v) = N(v) \cup \{v\}$. For a set $S \subseteq V(G)$, S induces a subgraph (S, E_S) where $E_S = \{(u, v) \in E(G) \mid (u, v) \in S \times S\}$. Two graphs G and H are isomorphic, denoted by $G \simeq H$, if there exists an edge-preserving bijection (graph isomorphism) $\varphi : V(G) \to V(H)$ where $\forall (u,v) \in E(G), (u,v) \in E(G) \Leftrightarrow (\varphi(u), \varphi(v)) \in E(H)$. In addition, for labeled graphs, a graph isomorphism requires that $\forall v \in V(G)$, $l(v) = l(\varphi(v))$ and likewise for edge labels. The graph isomorphism problem is concerned with deciding whether two graphs are isomorphic or not.

2.1.2 Multi-Agent Reinforcement Learning

MARL involves multiple agents learning to make decisions through interactions within a shared environment to achieve individual or collective goals.

Partially Observable Stochastic Games

The multi-agent problem can be defined in general as a Partially Observable Stochastic Game (POSG) [30, 99, 111]. The tuple $\langle \mathcal{I}, \mathcal{S}, \rho^0, \{\mathcal{A}_i\}_{i\in\mathcal{I}}, P, \{\mathcal{O}_i\}_{i\in\mathcal{I}}, O, \{R_i\}_{i\in\mathcal{I}}, \gamma \rangle$ describes a POSG, where $\mathcal{I}$ is a finite set of agents indexed by $[n]$, $\mathcal{S}$ is the set of states shared by all agents, $\rho^0 \in \Delta(\mathcal{S})$ is the initial state distribution, $\mathcal{A}_i$ is the set of actions available to agent i, $\mathcal{O}_i$ is the set of observations of agent i, and $\gamma \in [0,1]$ is the discount rate. Let $\boldsymbol{\mathcal{A}} = \times_{i\in\mathcal{I}}\mathcal{A}_i$ and $\boldsymbol{\mathcal{O}} = \times_{i\in\mathcal{I}}\mathcal{O}_i$ denote the joint action space and the joint observation space, respectively. $P : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \Delta(\mathcal{S})$ is the transition probability from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ in the next time step given the joint action $\boldsymbol{a} = \langle a_i \rangle_{i\in\mathcal{I}}$, where $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$. $O : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \Delta(\boldsymbol{\mathcal{O}})$ is the probability of observing joint observation $\boldsymbol{o} = \langle o_i \rangle_{i\in\mathcal{I}}$, where $\boldsymbol{o} \in \boldsymbol{\mathcal{O}}$, when transitioning to the state $s' \in \mathcal{S}$ given the joint action $\boldsymbol{a}$. $R_i : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to \mathbb{R}$ is the reward function for agent i with $\boldsymbol{r} = \langle r_i \rangle_{i\in\mathcal{I}}$ as the joint reward. In cooperative settings where this work focuses, all agents share a common goal, i.e. $r_1 = r_2 = \cdots = r_n$, which reduces POSG to a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [65].

Policy Gradients

Unlike value-based methods that focus on estimating the optimal Q-value function from which the policy is derived implicitly, policy gradient (PG) methods optimize the policy directly over the policy space. Specifically, each agent learns its own policy $\pi_{\theta_i} : \mathcal{S} \to \Delta(\mathcal{A}_i)$ by updating the policy parameters θ_i, where $\boldsymbol{\theta} = \langle \theta_i \rangle_{i\in\mathcal{I}}$ is the joint policy parameters, $\boldsymbol{\pi_\theta} = \langle \pi_{\theta_i} \rangle_{i\in\mathcal{I}}$ is the joint policy, and $\rho^{\boldsymbol{\pi_\theta}}$ is the state distribution under $\boldsymbol{\pi_\theta}$. With the expected discounted return as the agent's objective function, $J_i(\boldsymbol{\theta}) = \mathbb{E}_{s_0 \sim \rho^0, s \sim P, \boldsymbol{a} \sim \boldsymbol{\pi_\theta}} \left[\sum_{t=0}^{\infty} \gamma^t R_i\left(s_t, \boldsymbol{a}, s_{t+1}\right) | s_0\right]$, the policy gradients, extended to the multi-agent context, are expressed by the policy gradient theorem [83] in Equation 2.1 as follows:

$$\nabla_{\theta_i} J_i(\boldsymbol{\theta}) = \mathbb{E}_{s \sim \rho^{\boldsymbol{\pi_\theta}}(\cdot), \boldsymbol{a} \sim \boldsymbol{\pi_\theta}(\cdot|s)} \left[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|s) Q_i^{\boldsymbol{\pi_\theta}}(s|\boldsymbol{a})\right] \tag{2.1}$$

In single agent RL, examples of PG methods include REINFORCE algorithms [93] that use Monte Carlo methods to estimate the value function and actor-critic methods [48, 70], where the actor and critic represent the model used to approximate the policy and the value function, respectively. Important variants of actor-critic methods include PG with optimal baselines [91, 108], deep deterministic policy gradients (DDPG) methods [49], soft actor-critic methods [28] and trust-region methods [74, 76]. Extensions of actor-critic methods to MARL include multi-agent deep deterministic policy gradients (MADDPG) [51] and multi-agent proximal policy optimization (MAPPO) [102], where MAPPO is the underlying MARL algorithm being used in this work. Equations 2.2 and 2.3 show the losses of the actor $L(\boldsymbol{\theta})$ and the critic $L(\boldsymbol{\phi})$, parameterized by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, for MAPPO respectively, where B and n are the batch size and the number of agents, respectively:

$$\begin{aligned} L(\boldsymbol{\theta}) = \frac{1}{Bn} \sum_{k=1}^{B} \sum_{i=1}^{n} \Bigg(& \min\left(r_{\theta_i}^{(k)} A_i^{(k)}, \operatorname{clip}\left(r_{\theta_i}^{(k)}, 1-\epsilon, 1+\epsilon\right) A_i^{(k)}\right) + \\ & \sigma S\left(\pi_{\theta_i}\left(s_i^{(k)}\right)\right)\Bigg) \end{aligned} \tag{2.2}$$

$$\begin{aligned} L(\boldsymbol{\phi}) = \frac{1}{Bn} \sum_{k=1}^{B} \sum_{i=1}^{n} \max \Bigg(& \left(V_{\phi_i}\left(s_i^{(k)}\right) - \hat{R}_i^{(k)}\right)^2, \\ & \left(\operatorname{clip}\left(V_{\phi_i}\left(s_i^{(k)}\right), V_{\phi_i^{old}}\left(s_i^{(k)}\right) - \varepsilon, V_{\phi_i^{old}}\left(s_i^{(k)}\right) + \varepsilon\right) - \hat{R}_i^{(k)}\right)^2 \Bigg) \end{aligned} \tag{2.3}$$

For $L(\boldsymbol{\theta})$, $r_{\theta_i}^{(k)} = \frac{\pi_{\theta_i}\left(a_i^{(k)}|s_i^{(k)}\right)}{\pi_{\theta_i^{old}}\left(a_i^{(k)}|s_i^{(k)}\right)}$ and ϵ is a hyperparameter to control the extent to which the updated policy can deviate from the old policy to prevent excessively large policy updates that can make learning inefficient and unstable. $A_i^{(k)}$ is computed using generalized advantange estimation (GAE) [75], S is the policy entropy function and

σ is the coefficient hyperparameter for the policy entropy. Similarly for $L(\boldsymbol{\phi})$, ε is a hyperparameter that controls the extent of the temporal difference (TD) error between the value function $V_{\phi_i}\left(s_i^{(k)}\right)$ and the discounted reward-to-go $\hat{R}_i^{(k)}$. In addition, if the actor and critic networks involve recurrent neural networks (RNN), the loss functions sum over time and are trained via backpropagation through time (BPTT).

2.1.3 Weisfeiler–Leman Algorithm

The 1-WL algorithm [92] proposed by Weisfeiler and Leman addresses the graph isomorphism problem through a simple but powerful iterative node labeling heuristic that can successfully test the isomorphism for a broad class of graphs [6].

1-dimensional Weisfeiler-Leman Algorithm

Given a labeled graph $G = \langle V, E, l \rangle$, 1-WL computes a node coloring $C_i^1 : V(G) \to \mathbb{N}$ that is dependent on the coloring of the neighbors for each iteration $i > 0$ as shown in Equation 2.4:

$$C_i^1(v) = \text{RELABEL}\left(C_{i-1}^1(v), \{\!\{C_{i-1}^1(u) \mid u \in N(v)\}\!\}\right) \tag{2.4}$$

RELABEL is an injective hash function that maps to a unique natural number that has not been used in previous iterations and when $i = 0$, $C_i^1 = l$ or a constant label if none is provided. To test if two graphs are isomorphic using 1-WL, we execute the above algorithm shown in Equation 2.4 in parallel for both graphs. For a given iteration i, should the two graphs have a different number of nodes colored as $c \in \mathbb{N}$, 1-WL concludes that the two graphs are not isomorphic. Otherwise, the algorithm terminates when $\forall u, v, C_i^1(u) = C_i^1(v) \Leftrightarrow C_{i+1}^1(u) = C_{i+1}^1(v)$ for a given iteration i, and we can define the stable coloring $\forall v \in V(G), C_i^1(v) = C_\infty^1(v)$. The stable coloring is reached after at most $\max(|V(G)|, |V(H)|)$ iterations [26].

k-dimensional Weisfeiler-Leman Algorithm

However, 1-WL cannot distinguish all non-isomorphic graphs [11], leading to the development of a more powerful generalization of 1-WL known as the k-dimensional Weisfeiler-Leman algorithm [5, 37]. In the literature, two versions of the algorithm are known, which have minor differences in their aggregation methods [25, 27, 54, 59, 61]: 1) the folklore k-dimensional Weisfeiler-Leman algorithm (k-FWL) and 2) the oblivious k-dimensional Weisfeiler-Leman algorithm (k-OWL). By labeling subgraphs instead of individual nodes similar to 1-WL given some definition of neighborhood between subgraphs, both variants exceed 1-WL in terms of expressive power. For $k \geq 1$, there are non-isomorphic graphs distinguished by $(k+1)$-WL that k-WL cannot [11]. Formally, with $\mathbf{v}$ as a tuple in $V(G)^k$, $G[\mathbf{v}]$ is the subgraph induced by $\mathbf{v}$, where G is a graph, $k \geq 2$ and the nodes in $\mathbf{v}$ are labeled with integers from $[k]$ corresponding to their indices. Like 1-WL, k-FWL computes a coloring $C_i^k : V(G)^k \to \mathbb{N}$ in each iteration $i \geq 0$, where two tuples $\mathbf{v}, \mathbf{w} \in V(G)^k$ get the same color if the map $v_i \mapsto w_i$ induces an isomorphism between $G[\mathbf{v}]$ and $G[\mathbf{w}]$. For $i > 0$, C_{i+1}^k is defined as follows in Equation 2.5:

$$C_{i+1}^k(\mathbf{v}) = \text{RELABEL}\left(\left(C_i^k(\mathbf{v}), M_i(\mathbf{v})\right)\right) \tag{2.5}$$

$$M_i(\mathbf{v}) = \{\!\!\{\left(C_i^k\left(\phi_1(\mathbf{v}, w)\right), \ldots, C_i^k\left(\phi_k(\mathbf{v}, w)\right)\right) \mid w \in V(G)\}\!\!\} \tag{2.6}$$

$$\phi_j(\mathbf{v}, w) = (v_1, \ldots, v_{j-1}, w, v_{j+1}, \ldots, v_k) \tag{2.7}$$

$\phi_j(\mathbf{v}, w)$ in Equation 2.7 replaces v_j in $\mathbf{v}$ with w, where two tuples are adjacent or j-neighbors if they differ by v_j and the multiset $M_i(\mathbf{v})$ in Equation 2.6 groups the colors of the k-tuples across the replaced nodes. In prose, two tuples $\mathbf{v}$ and $\mathbf{w}$ with the same color at iteration $i-1$ get a different color in iteration i if there exists a $j \in [k]$ such that the number of j-neighbors for $\mathbf{v}$ and $\mathbf{w}$ are different. Like the 1-WL, the algorithm terminates when $\forall \mathbf{v}, \mathbf{w}, C_i^k(\mathbf{v}) = C_i^k(\mathbf{w}) \Leftrightarrow C_{i+1}^k(\mathbf{v}) = C_{i+1}^k(\mathbf{w})$ for a given iteration i, and we can define the stable partition $\forall \mathbf{v} \in V(G), C_i^k(\mathbf{v}) = C_\infty^k(\mathbf{v})$. k-OWL differs from k-FWL in Equation 2.6, where $M_i(\mathbf{v})$ is replaced by $M_i^*(\mathbf{v})$ as shown in Equation 2.8:

$$M_i^*(\mathbf{v}) = \left(\{\!\!\{C_i^k\left(\phi_1(\mathbf{v}, w)\right) \mid w \in V(G)\}\!\!\}, \ldots, \{\!\!\{C_i^k\left(\phi_k(\mathbf{v}, w)\right) \mid w \in V(G)\}\!\!\}\right) \tag{2.8}$$

Due to the stated difference in color aggregation between k-FWL and k-OWL, k-OWL has a lower expressive power compared to k-FWL. To be precise, it is found that 1-OWL and 2-OWL have the same expressive power and $(k+1)$-OWL has the same expressive power as k-FWL [27].

2.1.4 Graph Neural Networks

In general, GNNs update the vectorial representation of each node in a graph by aggregating information from its neighboring nodes to be merged with its current representation. Formally, the problem is initialized as a labeled graph $G = \langle V, E, l \rangle$ with initial node feature vectors of $f^{(0)} : V(G) \to \mathbb{R}^d$ that are consistent with l, i.e. $\forall u, v,\ l(u) = l(v) \to f^{(0)}(u) = f^{(0)}(v)$. An example of $f^{(0)}$ would be the one-hot encoding function of l. The general architecture of a GNN is shown in Equation 2.9:

$$f^{(t)}(v) = f^{\boldsymbol{W}_1}_{merge}\left(f^{(t-1)}(v), f^{\boldsymbol{W}_2}_{aggr}\left(\{\!\!\{f^{(t-1)}(w) \mid w \in N(v)\}\!\!\}\right)\right) \tag{2.9}$$

$f^{\boldsymbol{W}_2}_{aggr}$ aggregates the multiset of neighborhood feature vectors, $f^{\boldsymbol{W}_1}_{merge}$ merges the node's feature vector currently with the neighborhood features aggregated from $f^{\boldsymbol{W}_2}_{aggr}$. Both $f^{\boldsymbol{W}_1}_{merge}$ and $f^{\boldsymbol{W}_2}_{aggr}$ are arbitrary differentiable functions (e.g., summation, vector concatenation) parameterized by $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, respectively, which can be learned end-to-end for classification or regression tasks. For local/node-level tasks such as node classification, the node feature vector at the final layer T, $f^{(T)}(v)$, can be used for prediction. For global/graph-level tasks such as graph classification, a readout function $f^{\boldsymbol{W}_3}_{readout}$ parameterized by $\boldsymbol{W}_3$ is required to aggregate node feature vectors from all nodes at the final layer T to obtain the entire graph's feature vector f_G for prediction as shown in Equation 2.10 as follows:

$$f_G = f^{\boldsymbol{W}_3}_{readout}\left(\{f^{(T)}(v) \mid v \in V(G)\}\right) \tag{2.10}$$

Both Xu et al. [98] and Morris et al. [59] showed that the expressive powers of GNNs are upper-bounded by 1-WL in terms of distinguishing non-isomorphic graphs, i.e. any $f^{\boldsymbol{W}_1}_{merge}$ and $f^{\boldsymbol{W}_2}_{aggr}$ is unable to learn node features to distinguish non-isomorphic graphs if 1-WL cannot distinguish them. Formally, letting $\boldsymbol{W}^{(t)}$ denote the set of weights up to layer t, the above is expressed in Theorem 1 as follows:

Theorem 1 (Morris et al., 2019; Xu et al., 2019)**.** *For all $t \geq 0$, weights $\boldsymbol{W}^{(t)}$, choices of $f^{(0)}$, and nodes $u, v \in V(G)$, where $G = \langle V, E, l \rangle$ is a labeled graph:*

$$C^1_t(u) = C^1_t(v) \Rightarrow f^{(t)}(u) = f^{(t)}(v)$$

Morris et al. also showed that there exists a sequence of weights $\boldsymbol{W}^{(t)}$ where GNNs have the same expressive power as 1-WL in distinguishing non-isomorphic graphs through injective $f_{merge}^{\boldsymbol{W}_1}$ and $f_{aggr}^{\boldsymbol{W}_2}$ functions, formally stated in Theorem 2 as follows:

Theorem 2 (Morris et al., 2019)**.** *For all $t \geq 0$ and nodes $u, v \in V(G)$, where $G = \langle V, E, l \rangle$ is a labeled graph, there exists a sequence of weights $\boldsymbol{W}^{(t)}$ and a GNN architecture such that:*

$$C_t^1(u) = C_t^1(v) \Leftrightarrow f^{(t)}(u) = f^{(t)}(v)$$

Xu et al. corroborated with Morris et al. by developing the Graph Isomorphism Network (GIN) and showing that it has the same expressive power as 1-WL in distinguishing non-isomorphic graphs.

2.1.5 Graph Decision Networks

Most GNN-based CL methods fall into the paradigm of GDNs described as follows. At each time step, let an attributed graph $G = \langle V, E, a \rangle$ represent the communication graph, where the nodes in $V(G)$ represent the agents $\mathcal{I}$, and the edges in $E(G)$ represent the (undirected) communication link between two agent nodes. Each agent node has the agent's observations as its attribute, that is, $a(v_i) = o_i$, where $v_i \in V(G)$. The communication graph is then passed through layers of GNNs that represent multiple rounds of communication. The final layer of the GNN is an actor network shared between all agents that produces the policy for each agent given the learned feature vector of the node representing the agent, i.e. $f_{merge}^{\boldsymbol{W}_1}$ in Equation 2.9 ignores $f_{aggr}^{\boldsymbol{W}_2}$. Given this framework, Morris et al. [63] states that any GDN reduces to a GNN node labeling problem, where the correct label for a given node is the optimal action from the shared actor network that maximizes the agent's return. As a result, GDNs are limited to 1-WL in their expressive power, as GNNs.

2.2 Related Work

The following sections introduce relevant work.

2.2.1 Communication Learning

Early seminal work in CL includes RIAL and DIAL by Foerster et al. [20] and CommNet by Sukhbaatar et al. [82]. RIAL treats communication as a discrete action sent to all agents and is end-to-end trainable within each agent. On the other hand, DIAL has continuous communication directly connected to all other agents that is only discretized during execution time, allowing backpropagation through the communication channels across agents, laying the groundwork for differentiable CL methods. Similarly, CommNet utilizes a differentiable global communication channel that communicates the average hidden states of all agents to all. Diff Discrete [23] uses an encoder-channel-decoder architecture to backpropagate gradients through a discrete communication channel with unknown noise for two agents. BiCNet [69] and FCMNet [89] use recurrent neural networks (RNNs) to connect all agents in various sequences with communication from various directions. Variable-length coding [22] builds upon Diff Discrete by encouraging shorter message length using a regularization-inspired message-length penalty term. TarMAC [14] uses multiple rounds of communication between all agents and the soft-attention mechanism to weigh messages. Similarly, IMAC [88] learns a scheduler to reweight all agent messages while explicitly modeling bandwidth limitation as an optimization constraint to encourage agents to convey low entropy but useful messages. InfoLewis [41] uses variational compositional communication to adequately embed coordination information and provide a contrastive objective to ground communication in intent-specific features.

The methods discussed above utilize a communication policy that assumes a complete communication graph where messages are broadcasted to all agents, giving rise to an inherent drawback with regard to communication bandwidth. Hence, some methods consider a predefined partial graph to reduce overall communication. MAIC [103] allows agents to learn estimated models of their teammates to generate customized messages that can be dynamically pruned based on their weights given a specified threshold for effective and sparse communications. DGN [40] restricts communication within a specified number (3) of closest neighbors, using a GNN based on attention for communication between neighbors with temporal regularization of the attention weights for consistent cooperation. Similarly, Agent-Entity Graph [3] restricts communication to a specified distance, using a shared agent-entity graph with both agents and environmental entities with a GNN based on attention for communication. MAGNet [52] also uses a generated agent-entity relevance graph and a GNN for communication.

Other methods allow agents to individually determine whether to communicate with other agents. IC3Net [80] uses a binary gating mechanism to allow agents to decide if they are to communicate their internal state to all agents deterministically. Enforcer [43] builds on IC3Net for discrete and sparse communication focusing on learning human interpretable communication. ATOC [39] uses an attention model to determine the probability that an agent initiates communication with a dynamically formed local communication group. Gated-ACML [53] learns a probabilistic gating mechanism to block communications between each agent and a central message coordinator. ETCNet [34] uses an event-triggered module with a gating policy to determine whether communication occurs, where the gating policy is optimized with limited bandwidth as a constraint to minimize unnecessary communication. IMGS-MAC [42] uses information maximization autoencoder and individualized communication regularization to learn a gating function given over-constrained communication budgets. LSC [78] allows each group of agents, defined using a specified proximity, to learn the weights to elect a leader, from which a GNN-based mechanism is used for the communication of messages: 1) from agents to their leaders, 2) between leaders, and 3) from leaders to their group of agents.

Lastly, some methods learn a global communication policy that decides the communication links between all agents. SchedNet [45] deploys a global scheduler that restricts the number of agents capable of broadcasting their message according to their importance. GA-Comm [50] uses a two-stage attention network, where hard attention is used to remove unrelated edges and soft attention is used to learn the importance weight of the edges, to derive a communication graph from an initial complete graph representing all agents, from which a GNN is used for communication. Similarly, MAGIC [64] learns a global scheduler to encode when to communicate and to whom to communicate in a directed communication graph, where a GNN based on attention, Graph Attention Network (GAT) [85], is used for communication.

2.2.2 Expressive Graph Neural Networks

GNN architectures that are widely used in popular GNN libraries such as PyTorch Geometric [19] and DGL [87] include the Graph Convolutional Network (GCN) [46], GraphSAGE [29], GAT [85], GATv2 [10] and GIN [98]. Xu et al. [98] showed that GCN and GraphSAGE are not as expressive as 1-WL due to their aggregation function $f_{aggr}^{\mathbf{W_2}}$. The mean aggregator in GCN captures the multiset's distribution of elements but not the exact multiset, hence being able to perform well in tasks where the statistical information of the graph is vital. Similarly, the max-pooling aggregator in GraphSAGE captures neither the exact structure nor its distribution, but is able to identify representive elements that allows it to perform well for specific tasks. Zhang and Xie [106] showed that attention-based GNNs under a specific attention aggregation framework are also not as expressive as 1-WL, losing cardinality information similar to the mean aggregator.

Numerous works have investigated improving the expressive power of GNNs. One of the approaches would be to design higher-order GNNs based on the k-dimensional Weisfeiler-Leman algorithm, giving rise to k-GNN [58] and Invariant Graph Networks (k-IGNs) [55, 56] from k-OWL and k-FGNN [54] from k-FWL. These architectures have been shown to be as expressive as their corresponding higher-order tests [4, 24] mentioned in Section 2.1.3, where the expressiveness strictly increases as k increases [12] and when k approaches infinity, they can universally approximate any continuous graph function [44]. However, by requiring one to consider k tuples of nodes, these architectures can become highly intractable computationally and memory wise and hence are generally not practical for real-world applications, including for CL in this work.

To develop more practical GNN architectures, a further line of work examines local GNNs, which takes advantage of the local/sparse nature of graphs that can serve as a powerful inductive bias not used by higher-order GNNs given that their aggregation mechanisms are fundamentally global and graph adjacency information is only encoded in initial node attributes. The local k-GNN [60], shown to be strictly more expressive than k-GNN, integrates the adjacency of the graph directly into the layers of the network and aggregates information from neighbors rather than globally. Building on local k-GNN, (k, s)-SpeqNet [62] improves computational efficiency by considering a subset of k-tuples whose vertices can be grouped into no more than s connected components. Similarly, k-SetGNN considers k-sets instead of k-tuples. Other architectures can be interpreted as local variants of k-IGN [21] and k-FGNN [18, 104, 110].

Another line of research to enhance the expressive power of GNNs considers subgraph GNNs that represent graphs as multisets of subgraphs and feeding them to GNNs. The usage of subgraphs breaks the symmetries induced by the local aggregation function in GNNs, allowing it to extract more structural information from a given graph. Cotta et al. [13] and Papp et al. [68] proposed using node-deleted subgraphs while Papp and Wattenhofer [67] recommended node marking as opposed to node deletion for better expressive power. The Nested GNN [105] represents a graph using rooted subgraphs by extracting a local K-hop subgraph around each node. Similarly, Identity-Aware GNN [101] extracts a K-hop ego network of each node and assigns a unique coloring to the central node of the ego network.

Rather than feeding each subgraph independently into a GNN, subgraph GNNs have been extended to allow interactions between subgraphs through cross-subgraph aggregation layers [7, 107], which are shown to strictly improve the expressivity [104]. Frasca et al. [21] proposes a novel symmetry analysis that links previous work in invariant and equivariant models for graphs [55] to show that subgraphs GNNs are bounded by a variant of 2-IGN (local 2-IGN), which is then bounded by 2-FWL/3-OWL. Zhang et al. [104] later proved that Local 2-IGN is as expressive as Local 2-GNN and strictly less expressive than 2-FGNN. Qian et al. [71] introduced the higher order subgraph GNNs, subgraph k-GNN, by marking k nodes per subgraph for a graph with n vertices, giving rise to n^k unique subgraphs. The authors showed that subgraph k-GNN is strictly bounded by $(k+1)$-FWL and is incomparable to k-FWL for $k > 1$. Zhou et al. [109] further generalizes subgraph k-GNN to k, l-GNN, where k-GNNs run independently on all l-labeled graphs, where they showed that k, l-GNN is bounded by $(k+1)$-GNN for $l > 2$ and l is the subgraph size.

Nevertheless, the gained expressive power from the above methods still comes with significant computational costs that are especially expensive and hence undesirable in the context of CL under the GDN framework. Morris et al. [63] proposes two existing GNN augmentations to GDNs for universal expressivity for equivariant graph functions and symmetry breaking for coordination problems with minimal computational costs. One of the GNN enhancements is random node initialization (RNI), where Sato et al. [73] showed that concatenating random features (sampled from a standard uniform distribution) to initial node features can improve GIN performance in common combinatorial optimization problems. Abboud et al. [1] showed that RNI results in, with high probability, universal expressivity for invariant graph functions, which Morris et al. extend to equivariant graph functions such as GDNs. In the context of GDNs, RNI can be viewed as concatenating noise to the observations of the agents. Another GNN enhancement are unique node identifiers, where Dasoulas et al. [15] defined the colored local iterative procedure (CLIP) to differentiate identical node attributes using colors. Dasoulas et al. showed that assigning unique IDs to nodes (the observations of agents) is equivalent to 1-CLIP, which can approximate any invariant graph function. Like RNI, Morris et al. extend the stated universality result to equivariant graph functions. However, the stated GNN enhancements do not provide much improved performance as shown from the empirical results shown in [63], possibly due to the large training steps required for convergence given the augmentations in practice.

Chapter 3

Method

Given the limitations of the various variants of higher-order GNNs and universal GNN augmentations elaborated in Section 2.2.2, this work focuses on deriving maximally expressive (1-WL) GNNs for CL under the GDN framework, where communication costs remain reasonable and justified given the success of various GNN-based CL methods mentioned in Section 2.2.1. Although it is possible to build an architecture based on GIN [98], which is proven to be as expressive as 1-WL, this work investigates attention-based GNNs that are proven to be not as expressive as 1-WL [106], but are widely used in a significant number of CL methods such as GA-Comm [50] and MAGIC [64] with good empirical results. In our work, we expand on the Cardinality Preserved Attention (CPA) model proposed by Zhang and Xie [106] and propose a new GNN, the Graph Attention Isomorphism Network (GAIN), which can be proven to be as expressive as 1-WL. We incorporate GAIN into a simple GDN architecture called Graph Communication Network (GCNet).

3.1 Attention-Based Graph Neural Networks

The following sections describe the attention mechanism used in attention-based GNNs in general, highlight their inherent limitations in their expressive power, and propose solutions such that they can be as expressive as 1-WL.

3.1.1 Attention Score

In the t-th layer of the attention-based GNN, the attention score between node i and its neighbor j (including itself) can be calculated as follows in Equation 3.1 using a scoring function $\mathrm{Att} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, where $f^{(t-1)}(i), f^{(t-1)}(j) \in \mathbb{R}^d$ are feature vectors of the nodes in the $(t-1)$-th layer. The attention scores are then normalized using softmax as shown in Equation 3.2:

$$e_{ij}^{(t-1)} = \mathrm{Att}\left(f^{(t-1)}(i), f^{(t-1)}(j)\right) \tag{3.1}$$

$$\alpha_{ij}^{(t-1)} = \mathrm{softmax}\left(e_{ij}^{(t-1)}\right) = \frac{\exp\left(e_{ij}^{(t-1)}\right)}{\sum_{k \in \tilde{N}(i)} \exp\left(e_{ik}^{(t-1)}\right)} \tag{3.2}$$

Given a query vector, which is the feature vector of node i, the attention mechanism, based on the scoring function Att, computes a distribution on a set of key vectors, which are the feature vectors of neighbors of node i. Brody et al. [10] provide the definitions for static and dynamic scoring as shown in Definition 1 and 2 respectively as follows:

Definition 1 (Static scoring)**.** *A (possibly infinite) family of scoring functions* $\mathcal{F} \subseteq (\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R})$ *computes static scoring for a given set of query vectors* $\boldsymbol{\mathcal{Q}} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\} \subset \mathbb{R}^d$ *and key vectors* $\boldsymbol{\mathcal{K}} = \{\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n\} \subset \mathbb{R}^d$*, if for every* $f \in \mathcal{F}$*, there exists a key* $\boldsymbol{k}_{j_f}$*, where* $j_f \in [n]$*, such that for every query* $\boldsymbol{q}_i$ *and key* $\boldsymbol{k}_j$*, where* $i \in [m]$ *and* $j \in [n]$*, the following holds:*

$$f\left(\boldsymbol{q}_i, \boldsymbol{k}_{j_f}\right) \geq f\left(\boldsymbol{q}_i, \boldsymbol{k}_j\right)$$

Definition 2 (Dynamic scoring)**.** *A (possibly infinite) family of scoring functions* $\mathcal{F} \subseteq (\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R})$ *computes dynamic scoring for a given set of query vectors* $\boldsymbol{\mathcal{Q}} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\} \subset \mathbb{R}^d$ *and key vectors* $\boldsymbol{\mathcal{K}} = \{\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n\} \subset \mathbb{R}^d$*, if for any mapping* $\varphi : [m] \to [n]$ *there exists* $f \in \mathcal{F}$ *such that for any query* $\boldsymbol{q}_i$ *and any key* $\boldsymbol{k}_j$*, where* $i \in [m]$ *and* $j \in [n] \setminus \varphi(i)$*, the following holds:*

$$f\left(\boldsymbol{q}_i, \boldsymbol{k}_{\varphi(i)}\right) > f\left(\boldsymbol{q}_i, \boldsymbol{k}_j\right)$$

A family of attention functions computes static or dynamic attention for $\boldsymbol{\mathcal{K}}$ and

$\boldsymbol{\mathcal{Q}}$, if its scoring function computes static or dynamic scoring, respectively, possibly followed by monotonic normalization such as softmax. It can be seen that static attention is highly limited, given that every function $f \in \mathcal{F}$ has a key $\boldsymbol{k}_{j_f}$ that makes f maximal independent of query $\boldsymbol{q}_i$. On the other hand, dynamic attention is expressive as it can select every key $\boldsymbol{k}_{\varphi(i)}$ for a given query $\boldsymbol{q}_i$ by making $f\left(\boldsymbol{q}_i, \boldsymbol{k}_{\varphi(i)}\right)$ maximal within $\{f\left(\boldsymbol{q}_i, \boldsymbol{k}_j\right) \mid j \in [n]\}$. Equation 3.3 shows the scoring function Att used by GAT [85], where $\boldsymbol{a} \in \mathbb{R}^{2d'}$ and $\boldsymbol{W} \in \mathbb{R}^{d' \times d}$ are learned.

$$\text{GAT :} \qquad e_{ij}^{(t-1)} = \text{LeakyReLu}\left(\boldsymbol{a}^{\top} \cdot \left[\boldsymbol{W} f^{(t-1)}(i) \,||\, \boldsymbol{W} f^{(t-1)}(j)\right]\right) \tag{3.3}$$

However, Brody et al. shows (in Theorem 1) that GAT computes only static attention for any set of node feature vectors $\boldsymbol{\mathcal{K}} = \boldsymbol{\mathcal{Q}} = \{f^{(t)}(i)\}_{i \in [|V(G)|]}$ and developed GATv2 as shown in Equation 3.4 to address the limitations of GAT, where $\boldsymbol{a} \in \mathbb{R}^{d'}$ and $\boldsymbol{W} \in \mathbb{R}^{d' \times 2d}$ are learned.

$$\text{GATv2 :} \qquad e_{ij}^{(t-1)} = \boldsymbol{a}^{\top} \text{LeakyReLu}\left(\boldsymbol{W} \cdot \left[f^{(t-1)}(i) \,||\, f^{(t-1)}(j)\right]\right) \tag{3.4}$$

Brody et al. shows (in Theorem 2) that GATv2 computes dynamic attention for any set of node feature vectors $\boldsymbol{\mathcal{K}} = \boldsymbol{\mathcal{Q}} = \{f^{(t)}(i)\}_{i \in [|V(G)|]}$.

3.1.2 Attention Aggregator

Having calculated the attention score elaborated in Section 3.1.1, both GAT and GATv2 compute the weighted average using the normalized attention scores of the transformed feature vectors of the neighbor nodes of node i to obtain its new feature vector as shown in Equation 3.5, where $\boldsymbol{W}$ are the learned weights from calculating the unormalized attention scores and σ is a nonlinear function.

$$\text{GAT / GATv2 :} \qquad f^{(t)}(i) = \sigma\left(\sum_{j \in \tilde{N}(i)} \alpha_{ij}^{(t-1)} \boldsymbol{W} f^{(t-1)}(j)\right) \tag{3.5}$$

Theorem 3 (proof in Appendix A.1) generalizes Theorem 1 in [106] to show the cases where the attention mechanism is not injective. It accounts for the case where a matrix is used to transform the original node feature vectors in the attention aggregation mechanism (e.g., GAT, GATv2) instead of the framework used in [106] used by Theorem 1 that only considers the original node feature vectors as shown Equation 3.6:

$$\text{Zhang and Xie, 2020}: \qquad f^{(t)}(i) = \sigma\left(\sum_{j \in \tilde{N}(i)} \alpha_{ij}^{(t-1)} f^{(t-1)}(j)\right) \tag{3.6}$$

Theorem 3. *Assume that the input feature space $\mathcal{X}$ is countable. For a multiset $X \subset \mathcal{X}$ of bounded size and node feature c, let $h(c, X) = \sum_{x \in X} \alpha_{cx} \boldsymbol{W}_p f(x)$, where $f : \mathcal{X} \to \mathbb{R}^d$, α_{cx} is the attention score between $f(c)$ and $f(x)$ calculated using the attention function, Att, and softmax in Equations 3.1 and 3.2, respectively, and $\boldsymbol{W}_p \in \mathbb{R}^{d' \times d}$ is a full rank matrix, i.e. $rank\left(\boldsymbol{W}_p\right) = min(d', d)$. Given the stated, the following holds:*

$$\forall f, Att: \ h(c_1, X_1) = h(c_2, X_2) \Leftrightarrow c_1 = c_2, X_1 = \langle S, \mu \rangle, X_2 = \langle S, k \cdot \mu \rangle, k \in \mathbb{N}^+$$

In summary, Theorem 3 demonstrates that h will produce the same embedding for different multisets if and only if they share the same central node feature and identical distributions of node features, where the attention aggregation mechanism is shown in Equation 3.7 as follows:

$$f^{(t)}(i) = \sigma\left(\sum_{j \in \tilde{N}(i)} \alpha_{ij}^{(t-1)} \boldsymbol{W}_p f^{(t-1)}(j)\right) \tag{3.7}$$

Note that Theorem 3 and Equation 3.7 assumes that $\boldsymbol{W}_p$ has full rank. In other words, when: 1) $d' > d : rank\left(\boldsymbol{W}_p\right) = d$, $\boldsymbol{W}_p$ has full column rank and is injective, 2) $d' < d : rank\left(\boldsymbol{W}_p\right) = d'$, $\boldsymbol{W}_p$ has full row rank and is surjective, and 3) $d' = d : rank\left(\boldsymbol{W}_p\right) = d$, $\boldsymbol{W}_p$ has full rank and is invertible. As the constraints stated for $\boldsymbol{W}_p$ are not enforced for the learned weights $\boldsymbol{W}$ from the calculation of the unormalized attention scores for GAT/GATv2, their attention aggregation mechanism in Equation 3.5 does not fall within the framework of Theorem 3 in general. Hence, their limitations cannot be specifically expressed to be improved upon. Furthermore, letting $\boldsymbol{W}_p = \boldsymbol{I}_d$, where $\boldsymbol{I}_d \in R^{d \times d}$ is the identity matrix, we recover Theorem 1 and

Equation 3.6 from [106] from Theorem 3 and Equation 3.7 respectively.

3.1.3 Cardinality Preserved Attention Model

Theorem 4 states the conditions for maximal expressive power for the attention-based GNNs described in Sections 3.1.1 and 3.1.2 in differentiating different elements, namely local and global structures. Each local structure belonging to a node $v \in V(G)$ is the t-height subtree rooted at the node represented by feature vector $f^{(t)}(v)$ after t layers of GNN. The global structure is the aggregated representation of the local structures of all nodes in the graph. Theorem 4 can be proven identically to Lemma 1 in [106], which is based on Lemma 2 and Theorem 3 in [98].

Theorem 4. *Assume that the input feature space $\mathcal{X}$ is countable. Let $\mathcal{A}_{GNN} : \mathcal{G} \to \mathbb{R}^g$ be a GNN whose architecture is based on the attention aggregation mechanism in Equation 3.7. For global-level tasks, a readout function from Equation 2.10 is used. With sufficient GNN layers, $\mathcal{A}_{GNN}$ is maximally expressive, i.e. can differentiate all distinct local structures or be as powerful as 1-WL in differentiating global structures if the following conditions hold true:*

- ***Local level**: The weighted summation and σ in Equation 3.7 is injective.*
- ***Global level**: The readout function of Equation 2.10 is injective, along with conditions from the local level.*

From Theorem 4, it can be inferred that $\mathcal{A}_{GNN}$ is not maximally expressive, since Theorem 3 shows that the weighted summation in Equation 3.7 is not injective. It is simple to show that there exist subtrees or graphs that 1-WL determines to be non-isomorphic that $\mathcal{A}_{GNN}$ cannot differentiate due to its inability to extract cardinality information from multisets with identical distribution (Corollary 1 and 2 from [106]). Building on the Additive and Scaled models in the CPA model in [106], we propose similar augmentations to Equation 3.7 so that it can be injective by retaining cardinality information as shown in Equations 3.8 and 3.9, respectively, as follows:

$$\text{Additive}: \quad f^{(t)}(i) = \sigma\left(\sum_{j\in\tilde{N}(i)} \alpha_{ij}^{(t-1)} \boldsymbol{W}_p f^{(t-1)}(j) + \sum_{j\in\tilde{N}(i)} \boldsymbol{W}_q f^{(t-1)}(j)\right) \tag{3.8}$$

$$\text{Scaled}: \quad f^{(t)}(i) = \sigma\left(\psi^{(t-1)}\left(\left|\tilde{N}(i)\right|\right) \odot \sum_{j\in\tilde{N}(i)} \alpha_{ij}^{(t-1)} \boldsymbol{W}_p f^{(t-1)}(j)\right) \tag{3.9}$$

where $\odot$ is the Hadamard product. For the Additive model, $\boldsymbol{W}_p, \boldsymbol{W}_q \in \mathbb{R}^{d' \times d}$ are full rank matrices where $d' \geq d$, i.e. $\boldsymbol{W}_p$ and $\boldsymbol{W}_q$ are injective or invertible. For the Scaled model, $\boldsymbol{W}_p \in \mathbb{R}^{d' \times d}$ has full rank and $\psi : \mathbb{N} \to \mathbb{R}^{d'}$ is an injective function. Note that the models stated are applicable for any given Att in Equation 3.1. Corollary 1 (proof in Appendix A.2) shows that the Additive and Scaled models can be injective with the correct weights or functions, respectively.

Corollary 1. *Assume that the input feature space $\mathcal{X}$ is countable. There exist $f : \mathcal{X} \to \mathbb{R}^d$, $\boldsymbol{W}_q$ for Equation 3.8 (Additive model) and ψ for Equation 3.9 (Scaled model) such that the Additive and Scaled models can differentiate the multisets that Equation 3.7 cannot, as stated in Theorem 3.*

3.1.4 Graph Attention Isomorphism Network

To reduce the complexity of the Scaled model, we can fix ψ to the mapping used in the proof of Corollary 1 shown in Appendix A.2, where ψ maps $|\tilde{N}(i)|$ to a d'-dimensional vector $\boldsymbol{x}_i$ where each element is $|\tilde{N}(i)|$. As such, we can simplify the Scaled model to the following variant as shown in Equation 3.10 similar to the f-Scaled variant in the CPA model in [106]:

$$f^{(t)}(i) = \sigma\left(\left|\tilde{N}(i)\right| \cdot \sum_{j\in\tilde{N}(i)} \alpha_{ij}^{(t-1)} \boldsymbol{W}_p f^{(t-1)}(j)\right) \tag{3.10}$$

With the attention aggregation mechanisms stated in Equations 3.8 and 3.10, there is now a myraid of possible variants of GAIN architectures that is provably as expressive as 1-WL, depending on the choices of $\boldsymbol{W}_p$, $\boldsymbol{W}_q$ and Att. For the Scaled model, given that $\boldsymbol{W}_p \in \mathbb{R}^{d' \times d}$ can be any full rank matrix with no constraints on d', which can be trivially generated by filling the main diagonal of zero matrix with any non-zero constant. Hence, it can be used to reduce the dimensonality of node feature vectors if necessary, which is not possible for the Additive model as $\boldsymbol{W}_p, \boldsymbol{W}_q \in \mathbb{R}^{d' \times d}$ have to be full rank matrices where $d' \geq d$. However, by letting $\boldsymbol{W}_p = \boldsymbol{I}_d$ and $\boldsymbol{W}_q = \text{diag}(\epsilon)$, where $\text{diag}(\epsilon) \in \mathbb{R}^{d \times d}$ is a square diagonal matrix with diagonal values of ϵ, we obtain a variant as shown in Equation 3.11 similar to the f-Additive variant in the CPA model in [106]:

$$f^{(t)}(i) = \sigma \left(\sum_{j \in \tilde{N}(i)} \left(\alpha_{ij}^{(t-1)} + \epsilon \right) f^{(t-1)}(j) \right) \tag{3.11}$$

Equation 3.11 is extremely simple to implement and ϵ can be a learnable parameter or a fixed constant as long as its non-zero. This GAIN variant can be seen as the attention variant of GIN that is 1-WL expresssive. Lastly, GAIN uses the Att function of GATv2 given that it computes dynamic attention and like GIN, GAIN uses multi-layer perceptrons (MLPs) to learn σ as an injective non-linearity given the universal approximation theorem [32, 33].

3.2 Graph Communication Network

GCNet basically has the same architecture as GDN described in Section 2.1.5, except for a small difference in terms of input to the shared actor network. In GDN, the shared actor network takes in the node feature vector of the final layer to produce a policy from which an action is extracted. On the other hand, GCNet takes the concatenated node feature vector from all layers, $f_{cat}(i) = ||_{t=0}^{T} f^{(t)}(i)$, as input to the shared actor network. This architecture takes inspiration from the graph readout function used in GIN based on Jumping Knowledge Networks [97], where our motivation is to capture the local structure information from all layers instead of just the final layer, as the features in the early layers may generalize better / provide a better representation for the shared actor network.

Chapter 4

Results

The following sections detail the results of this work in which we evaluated GAIN on various graph, link, and node property tasks in OGB [35]. Details of the tasks and datasets in OGB used in this work can be found in Appendix B.1. In addition, we integrate GAIN into GCNet (GCNet-GAIN) and evaluate it on tasks in SMAC [72]. For simplicity, we use the GAIN variant based on Equation 3.8 where $\boldsymbol{W}_p = \boldsymbol{W}_q = \boldsymbol{I}_d$ for all experiments and refer to it as GAIN.

4.1 Graph Tasks

The current literature often focuses on proposing and evaluating GNN models of specific architectural designs, which are specific instances of a GNN design space consisting of a cross product of various design dimensions, such as the number of layers or the type of the aggregation function. Hence, such a pipeline offers a narrow window for evaluating the performance of GNNs in general. Therefore, this work uses GraphGym [100] to evaluate GNNs in a general design space across a diverse set of graph, link, and node property tasks in OGB to provide a more balanced and fair evaluation of GNN architectures in general. In particular, GraphGym considers variations in intra-layer design, inter-layer design, and training configurations. Intra-layer design involves design dimensions within a GNN layer, such as the use of batch normalization [38], dropout [81], type of activation function, and aggregation function $\left(f_{merge}^{\boldsymbol{W}_1}\right)$, where the options for aggregation functions are summation (add), element-wise mean (mean), or element-wise max (max). Inter-layer design involves how GNN layers are organized within a neural network model. GraphGym considers directly stacking multiple GNN layers, residual connections (skip-sum) [31] and dense connections that concatenate feature vectors from all previous layers (skip-cat) [36]. In addition, it also considers adding MLPs before and after GNN layers. Training configurations involve standard design dimensions such as batch size, learning rate, optimizer type, and training epochs. In this work, we perform a grid search over several key design dimensions highlighted in Appendix B.2 and report the best evaluated test performance for each GNN architecture found during the grid search across 3 seeds. The best performing model configurations are shown in Appendix B.3. We consider GCN [46], GAT [85], GATv2 [10] and GIN [98] as baselines for GAIN.

4.1.1 Graph Property Prediction

Table 4.1 shows the results for the graph tasks, where ROC-AUC is the area under the receiver operating characteristic curve. It can be seen that GAIN performance is comparable to the best model for **ogbg_molhiv** and is the best model by a significant margin for **ogbg_ppa**.

Layer Type	**ogbg_molhiv**		**ogbg_ppa**	
	ROC-AUC	**p-value**	**Accuracy**	**p-value**
GCN	73.38 (1.05)	0.172	**18.90 (0.61)**	0.002
GAT	73.31 (2.38)	0.433	**10.44 (1.66)**	0.003
GATv2	73.01 (1.99)	0.475	**10.44 (1.66)**	0.003
GIN	72.36 (1.25)	0.740	**18.32 (0.29)**	0.001
GAIN	72.04 (0.21)	-	34.94 (1.23)	-

Table 4.1: Results for graph tasks. The standard deviations are in parentheses. The p-values are generated by the 2-tail paired t-test. The results that are statistically significant from GAIN at the significance level $p = 0.1$ are in bold.

4.1.2 Link Property Prediction

Table 4.2 shows the results for the link tasks. The Hits@K metric is calculated by randomly sampling a specific number of negative edges (e.g. $100,000$ for **ogbl_ddi**) and counting the ratio of positive edges that are ranked in the K-th place or higher. It can be seen that GAIN performance is comparable to the best models for **ogbl_collab** and **ogbl_ddi**. However, we note that the models' performance for **ogbl_ddi** have especially large variances, possibly arising due to the small $K = 20$ used and model overfitting.

Layer Type	**ogbl_collab**		**ogbl_ddi**	
	Hits@50	**p-value**	**Hits@20**	**p-value**
GCN	**16.13 (0.38)**	0.001	3.19 (5.28)	0.293
GAT	**15.11 (1.14)**	0.000	31.94 (55.33)	0.982
GATv2	**14.50 (0.32)**	0.001	30.71 (53.18)	0.999
GIN	49.40 (1.54)	0.359	11.09 (18.99)	0.244
GAIN	47.66 (1.66)	-	30.66 (38.77)	-

Table 4.2: Results for link tasks. The standard deviations are in parentheses. The p-values are generated by the 2-tail paired t-test. The results that are statistically significant from GAIN at the significance level $p = 0.1$ are in bold.

4.1.3 Node Property Prediction

Table 4.3 shows the results for the node tasks. It can be seen that GAIN performance is comparable to the best models for **ogbn_mag** and **ogbn_products**, while having relatively good performance in terms of absolute difference for **ogbn_arxiv**.

Layer Type	**ogbn_arxiv**		**ogbn_mag**		**ogbn_products**	
	Accuracy	**p-value**	**Accuracy**	**p-value**	**Accuracy**	**p-value**
GCN	62.21 (0.12)	0.017	27.17 (0.09)	0.253	56.33 (0.22)	0.111
GAT	62.34 (0.07)	0.976	27.09 (0.08)	0.828	55.96 (0.04)	0.212
GATv2	62.22 (0.16)	0.493	27.02 (0.06)	0.814	55.67 (0.43)	0.799
GIN	**62.87 (0.09)**	0.026	27.16 (0.18)	0.445	55.90 (0.36)	0.695
GAIN	62.35 (0.11)	-	27.06 (0.14)	-	55.77 (0.17)	-

Table 4.3: Results for node tasks. The standard deviations are in parentheses. The p-values are generated by the 2-tail paired t-test. The results that are statistically significant from GAIN at the significance level $p = 0.1$ are in bold.

4.2 StarCraft Multi-Agent Challenge

SMAC Maps	CommNet	GA-Comm	MAGIC	GCNet-GAIN
10m_vs_11m	20.00 (1.71)	40.63 (31.79)	69.38 (27.98)	**100.00 (0.00)**
1c3s5z	98.13 (4.19)	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
25m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
27m_vs_30m	**100.00 (0.00)**	**100.00 (0.00)**	95.63 (8.15)	91.25 (6.77)
2c_vs_64zg	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2m_vs_1z	79.38 (44.39)	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2s3z	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2s_vs_1sc	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3m	98.75 (1.71)	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3s5z	**100.00 (0.00)**	**100.00 (0.00)**	97.50 (3.42)	93.13 (5.59)
3s5z_vs_3s6z	17.50 (25.83)	**21.88 (14.99)**	12.50 (12.31)	11.25 (12.81)
3s_vs_3z	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3s_vs_4z	**100.00 (0.00)**	62.50 (51.35)	**100.00 (0.00)**	80.63 (43.32)
3s_vs_5z	**100.00 (0.00)**	20.00 (44.72)	96.88 (4.42)	92.50 (13.37)
5m_vs_6m	**92.50 (4.74)**	73.75 (12.02)	86.88 (15.84)	**92.50 (3.56)**
6h_vs_8z	4.38 (1.71)	3.75 (1.40)	1.88 (2.80)	**64.38 (40.30)**
8m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
8m_vs_9m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
bane_vs_bane	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
corridor	36.88 (50.60)	**80.00 (44.72)**	58.13 (53.11)	75.00 (42.04)
MMM	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
MMM2	21.88 (30.46)	35.63 (49.09)	**95.63 (2.80)**	58.75 (21.01)
so_many_baneling	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**

Table 4.4: Results for SMAC comparing GCNet-GAIN with CL baselines. The standard deviations are in parentheses. The best results are in bold.

We evaluated GCNet-GAIN on a range of SMAC maps listed in [102] with MAPPO as the underlying MARL algorithm. Similarly to the graph tasks, we consider GCN, GAT, GATv2 and GIN as baselines to be integrated into GCNet. In addition, we also consider CommNet[82], GA-Comm [50] and MAGIC [64] as CL baselines, given that their CL architectures are extremely similar to GNNs. In particular, GA-Comm and MAGIC build upon GAT at its essence. The hyperparameters used for the experiments, which are generally adapted from [102] with minor changes, are detailed

SMAC Maps	**GCNet**				
	GCN	**GAT**	**GATv2**	**GIN**	**GAIN**
10m_vs_11m	98.13 (4.19)	**100.00 (0.00)**	99.38 (1.40)	98.75 (2.80)	**100.00 (0.00)**
1c3s5z	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
25m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
27m_vs_30m	88.13 (11.35)	88.13 (4.08)	88.75 (6.48)	88.75 (8.15)	**91.25 (6.77)**
2c_vs_64zg	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2m_vs_1z	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2s3z	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2s_vs_1sc	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3s5z	92.50 (6.48)	**96.88 (6.99)**	89.38 (10.03)	93.13 (5.59)	93.13 (5.59)
3s5z_vs_3s6z	4.38 (6.85)	3.13 (3.13)	6.25 (7.65)	3.75 (5.59)	**11.25 (12.81)**
3s_vs_3z	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3s_vs_4z	80.00 (44.72)	80.00 (44.72)	80.00 (44.72)	80.00 (44.72)	**80.63 (43.32)**
3s_vs_5z	21.88 (43.69)	21.88 (40.32)	40.63 (48.91)	40.63 (54.22)	**92.50 (13.37)**
5m_vs_6m	92.50 (4.19)	91.88 (4.19)	92.50 (3.56)	**94.39 (2.60)**	92.50 (3.56)
6h_vs_8z	**66.88 (36.88)**	64.38 (40.30)	64.38 (41.73)	60.63 (45.75)	64.38 (40.30)
8m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
8m_vs_9m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
bane_vs_bane	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
corridor	**75.00 (42.04)**	73.75 (41.49)	74.38 (41.72)	74.38 (41.72)	**75.00 (42.04)**
MMM	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
MMM2	**62.50 (25.00)**	48.13 (30.51)	59.38 (21.20)	51.25 (20.80)	58.75 (21.01)
so_many_baneling	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**

Table 4.5: Results for SMAC comparing GCNet-GAIN with GCNet with other GNN baselines. The standard deviations are in parentheses. The best results are in bold.

in Appendix B.4. For each random seed, we compute the win rate for 32 evaluation games after each training iteration and take the best evaluation win rates as the performance for each seed. Tables 4.4 and 4.5 compare GCNet-GAIN with CL and other GCNet GNN baselines, respectively, where the results show the best evaluated win rate from each seed averaged across 5 seeds. It can be observed that GCNet-GAIN has comparable, if not significantly better performance against CL and other GCNet GNN baselines in general, validating the value of expressive attention-based GNN architectures in CL. For example, GCNet-GAIN has strong performance on maps such as 10m_vs_11m, 3s_vs_5z, 5m_vs_6m and 6h_vs_8z that are ranked as Hard or Super Hard tasks in [102] where the agents in those maps are outnumbers by their opponents. The complete results table with all the baselines can be found in Appendix B.5.

Chapter 5

Conclusions

In conclusion, this work explores attention-based GNNs for CL. In particular, we highlight their inherent limitations in terms of expressive power in a generalized framework and propose a class of maximally expressive (as 1-WL) attention-based GNNs. We implement a simple variant with that class of GNN architecture, which we term GAIN, and showcase its strong empirical performance against other state-of-the-art GNNs on various graph, link, and node property prediction tasks in OGB. We integrate GAIN into a simple CL architecture called GCNet and demonstrate that it outperforms GCNet using state-of-the-art GNNs and other baseline CL methods. We hope that the proposed architecture can serve as the basis for future work in expressive CL. We note that the contributions of many existing CL works can be built on top of GCNet-GAIN, such as the learning of a scheduler to determine who to communicate with via methods like hard attention weights in GA-Comm and MAGIC. This is so because the scheduler merely affects the graph structure in terms of edge connectivity. Hence, the expressivity guarantees of GCNet-GAIN remain. Future work would involve delving into further expressive GNN models for CL beyond 1-WL through methods such as subgraph GNNs.

Appendix A

Proofs

The following sections details the proofs used in this work.

A.1 Proof for Theorem 3

Proof. Assume that the input feature space $\mathcal{X}$ is countable. Given a multiset $X \subset \mathcal{X}$ of bounded size and node feature c, let $h(c, X) = \sum_{x \in X} \alpha_{cx} \boldsymbol{W}_p f(x)$, where $f : \mathcal{X} \rightarrow \mathbb{R}^d$, α_{cx} is the attention score between $f(c)$ and $f(x)$ calculated using the attention function, Att and softmax in Equations 3.1 and 3.2, respectively, and $\boldsymbol{W}_p \in \mathbb{R}^{d' \times d}$ is a full rank matrix, i.e. $rank\,(\boldsymbol{W}_p) = min(d', d)$. We proof both directions of the implication to prove Theorem 3.

(1) $\forall f$, Att, let $c_1 = c_2 = c$, $X_1 = \langle S, \mu \rangle$, $X_2 = \langle S, k \cdot \mu \rangle$, where $k \in \mathbb{N}^+$. Consider the following which expands from Equations 3.1 and 3.2:

$$h(c_1, X_1) = \sum_{s \in S} \mu(s) \alpha_{cs1} \boldsymbol{W}_p f(s) = \frac{\sum_{s \in S} \mu(s) \exp\left(e_{cs}\right) \boldsymbol{W}_p f(s)}{\sum_{x \in X_1} \exp\left(e_{cx}\right)}$$

$$h(c_2, X_2) = \sum_{s \in S} k \cdot \mu(s) \alpha_{cs2} \boldsymbol{W}_p f(s) = \frac{k \cdot \sum_{s \in S} \mu(s) \exp\left(e_{cs}\right) \boldsymbol{W}_p f(s)}{\sum_{x \in X_2} \exp\left(e_{cx}\right)}$$

where α_{csi} is the attention score between $f(c)$ and $f(s)$ from X_i, e_{cs} and e_{cx} are the unnormalized attention scores between $f(c)$ and $f(s)/f(x)$ from X_i, respectively, and $i \in \{1, 2\}$. Note that the following is true given that X_2 has k copies of the elements in X_1:

$$k \cdot \sum_{x \in X_1} \exp\left(e_{cx}\right) = \sum_{x \in X_2} \exp\left(e_{cx}\right)$$

As such, we can show the following:

$$\begin{aligned} h(c_1, X_1) &= \frac{\sum_{s \in S} \mu(s) \exp\left(e_{cs}\right) \boldsymbol{W}_p f(s)}{\sum_{x \in X_1} \exp\left(e_{cx}\right)} \\ &= \frac{k \cdot \sum_{s \in S} \mu(s) \exp\left(e_{cs}\right) \boldsymbol{W}_p f(s)}{k \cdot \sum_{x \in X_1} \exp\left(e_{cx}\right)} \\ &= \frac{k \cdot \sum_{s \in S} \mu(s) \exp\left(e_{cs}\right) \boldsymbol{W}_p f(s)}{\sum_{x \in X_2} \exp\left(e_{cx}\right)} \\ &= h(c_2, X_2) \end{aligned}$$

(2) $\forall f$, Att, let $h(c_1, X_1) = h(c_2, X_2)$, which can be stated as follows:

$$\begin{aligned} h(c_1, X_1) &= \sum_{x \in X_1} \alpha_{cx1} \boldsymbol{W}_p f(x) = \sum_{s \in S_1} \mu_1(s) \alpha_{cs1} \boldsymbol{W}_p f(s) \\ h(c_2, X_2) &= \sum_{x \in X_2} \alpha_{cx2} \boldsymbol{W}_p f(x) = \sum_{s \in S_2} \mu_2(s) \alpha_{cs2} \boldsymbol{W}_p f(s) \end{aligned}$$

where α_{cxi} and α_{csi} are the attention score between $f(c_i)$ and $f(x)/f(s)$ from X_i, respectively, $X_i = \langle S_i, \mu_i \rangle$ and $i \in \{1, 2\}$. The above can be rewritten as follows, where $\mathbf{0}_{d'} \in \mathbb{R}^{d'}$ is the zero vector:

$$\begin{aligned} &\sum_{s \in S_1 \cap S_2} \left(\mu_1(s) \alpha_{cs1} - \mu_2(s) \alpha_{cs2}\right) \boldsymbol{W}_p f(s) + \\ &\sum_{s \in S_1 \setminus S_2} \mu_1(s) \alpha_{cs1} \boldsymbol{W}_p f(s) - \sum_{s \in S_2 \setminus S_1} \mu_2(s) \alpha_{cs2} \boldsymbol{W}_p f(s) = \mathbf{0}_{d'} \end{aligned} \tag{A.1}$$

We first want to prove that $S_1 = S_2 = S$ by contradiction. Assume that $S_1 \neq S_2$. Given that the above is true $\forall f$, it must also be true for the following defined functions f_1 and f_2, where $\boldsymbol{x} \in \mathbb{R}^d \setminus \{\mathbf{0}_d\}$ is any arbitrary vector other than the zero vector,

$\mathbf{0}_d \in \mathbb{R}^d$.

$$
\begin{aligned}
f_1(s) &= f_2(s), && \forall s \in S_1 \cap S_2 \\
f_1(s) &= f_2(s) - \boldsymbol{x}, && \forall s \in S_1 \setminus S_2 \\
f_1(s) &= f_2(s) + \boldsymbol{x}, && \forall s \in S_2 \setminus S_1
\end{aligned}
$$

Substituting f with f_1:

$$
\begin{aligned}
&\sum_{s \in S_1 \cap S_2} \left(\mu_1(s)\alpha_{cs1} - \mu_2(s)\alpha_{cs2}\right) \boldsymbol{W}_p f_1(s) + \\
&\sum_{s \in S_1 \setminus S_2} \mu_1(s)\alpha_{cs1} \boldsymbol{W}_p f_1(s) - \sum_{s \in S_2 \setminus S_1} \mu_2(s)\alpha_{cs2} \boldsymbol{W}_p f_1(s) = \mathbf{0}_{d'}
\end{aligned}
$$

Substituting f_1 with f_2:

$$
\begin{aligned}
&\sum_{s \in S_1 \cap S_2} \left(\mu_1(s)\alpha_{cs1} - \mu_2(s)\alpha_{cs2}\right) \boldsymbol{W}_p f_2(s) + \\
&\sum_{s \in S_1 \setminus S_2} \mu_1(s)\alpha_{cs1} \boldsymbol{W}_p \left(f_2(s) - \boldsymbol{x}\right) - \\
&\sum_{s \in S_2 \setminus S_1} \mu_2(s)\alpha_{cs2} \boldsymbol{W}_p \left(f_2(s) + \boldsymbol{x}\right) = \mathbf{0}_{d'}
\end{aligned}
$$

Rewriting:

$$
\begin{aligned}
&\sum_{s \in S_1 \cap S_2} \left(\mu_1(s)\alpha_{cs1} - \mu_2(s)\alpha_{cs2}\right) \boldsymbol{W}_p f_2(s) + \\
&\sum_{s \in S_1 \setminus S_2} \mu_1(s)\alpha_{cs1} \boldsymbol{W}_p f_2(s) - \sum_{s \in S_2 \setminus S_1} \mu_2(s)\alpha_{cs2} \boldsymbol{W}_p f_2(s) \\
&= \sum_{s \in S_1 \setminus S_2} \mu_1(s)\alpha_{cs1} \boldsymbol{W}_p \boldsymbol{x} + \sum_{s \in S_2 \setminus S_1} \mu_2(s)\alpha_{cs2} \boldsymbol{W}_p \boldsymbol{x}
\end{aligned}
$$

LHS of the equation above is equal to $\mathbf{0}_{d'}$ from Equation A.1. Rewriting and rearranging terms:

$$\sum_{s \in S_1 \setminus S_2} \mu_1(s) \alpha_{cs1} \boldsymbol{W}_p \boldsymbol{x} + \sum_{s \in S_2 \setminus S_1} \mu_2(s) \alpha_{cs2} \boldsymbol{W}_p \boldsymbol{x} = \mathbf{0}_{d'}$$
$$\left(\sum_{s \in S_1 \setminus S_2} \mu_1(s) \alpha_{cs1} + \sum_{s \in S_2 \setminus S_1} \mu_2(s) \alpha_{cs2} \right) \boldsymbol{W}_p \boldsymbol{x} = \mathbf{0}_{d'}$$

Given that $\forall s \in S_i$ where $i \in \{1, 2\}$, the multiplicity function $\mu_i(s) \geq 1$ by definition, $\alpha_{csi} > 0$ by the properties of softmax, $\left(\sum_{s \in S_1 \setminus S_2} \mu_1(s) \alpha_{cs1} + \sum_{s \in S_2 \setminus S_1} \mu_2(s) \alpha_{cs2} \right) > 0$. As $\boldsymbol{x}$ is not the zero vector, $\mathbf{0}_d$, the null space of $\boldsymbol{W}_p$ is non-trivial, implying that $\boldsymbol{W}_p$ is not full rank. This contradicts the initial given fact that $\boldsymbol{W}_p$ is a full rank matrix. Hence, the assumption that $S_1 \neq S_2$ must be false. Therefore, $S_1 = S_2 = S$ is true. As a result, Equation A.1 can be rewritten as follows:

$$\sum_{s \in S} \left(\mu_1(s) \alpha_{cs1} - \mu_2(s) \alpha_{cs2} \right) \boldsymbol{W}_p f(s) = \mathbf{0}_{d'}$$

For the above equation to hold true $\forall f$, Att, the following would have to be true $\forall s \in S$:

$$\mu_1(s) \alpha_{cs1} - \mu_2(s) \alpha_{cs2} = 0$$
$$\frac{\mu_1(s)}{\mu_2(s)} = \frac{\exp(e_{cs2}) \sum_{x \in X_1} \exp(e_{cx1})}{\exp(e_{cs1}) \sum_{x \in X_2} \exp(e_{cx2})} \tag{A.2}$$

where e_{csi} and e_{cxi} are the unnormalized attention scores between $f(c_i)$ and $f(s)/f(x)$ from X_i, respectively, and $i \in \{1, 2\}$. We want to prove that $c_1 = c_2 = c$ using contradiction. If $|S| = 1$, $c_1 = c_2$ is trivially true. For $|S| > 1$, assume that $c_1 \neq c_2$. As Equation A.2 is true $\forall$ Att, consider an Att given that $c_1 \neq c_2$ which results in unnormalized attention scores as follows:

$$e_{cs1} = 1, \qquad \forall s \in S$$
$$e_{cs2} = \begin{cases} 1, & \text{for } s = s_0 \\ 2, & \forall s \neq s_0 \in S \end{cases}$$

Hence, when $s = s_0$, the RHS of Equation A.2 can be expressed as follows:

$$\frac{e \cdot |X_1| e}{e \cdot ((|X_2| - \mu_2(s_0))\, e^2 + \mu_2(s_0) e)} = \frac{|X_1|}{(|X_2| - \mu_2(s_0))\, e + \mu_2(s_0)}$$

It can be observed that the RHS of Equation A.2 is irrational while the LHS is rational given the definition of the multiplicity function, implying that Equation A.2 is not true, leading to a contradiction. Hence, the assumption that $c_1 \neq c_2$ is false. Therefore, $c_1 = c_2 = c$ is true, allowing Equation A.2 to be further simplified as follows $\forall s' \in S$:

$$\frac{\mu_1(s')}{\mu_2(s')} = \frac{\sum_{x \in X_1} \exp(e_{cx})}{\sum_{x \in X_2} \exp(e_{cx})} = \frac{\sum_{s \in S} \mu_1(s) \exp(e_{cs})}{\sum_{s \in S} \mu_2(s) \exp(e_{cs})} = k$$

where $e_{cs1} = e_{cs2} = e_{cs}$, $e_{cx1} = e_{cx2} = e_{cx}$ and $k \in \mathbb{N}^+$. Hence, by denoting $\mu_1 = \mu$, we show that $c_1 = c_2 = c$, $X_1 = \langle S, \mu \rangle$, $X_2 = \langle S, k \cdot \mu \rangle$, where $k \in \mathbb{N}^+$. □

A.2 Proof for Corollary 1

Proof. Assume that the input feature space $\mathcal{X}$ is countable. Consider $X_1 = \langle S, \mu \rangle$, $X_2 = \langle S, k \cdot \mu \rangle \subset \mathcal{X}$, where $k \in \mathbb{N}^+$ and $c \in S$. Given the stated, from Theorem 3, $H = \sum_{x \in X_1} \alpha_{cx1} \boldsymbol{W}_p f(x) = \sum_{x \in X_2} \alpha_{cx2} \boldsymbol{W}_p f(x)$, where α_{cxi} is the attention score between $f(c_i)$ and $f(x)$ from X_i, W_p is a full rank matrix and $i \in \{1, 2\}$. Using the Additive and Scaled models, the attention aggregation mechanism can be rewritten as follows:

$$\begin{aligned} \text{Additive}: \quad & h_{add}(c, X_i) = H + \sum_{x \in X_i} \boldsymbol{W}_q f(x) \\ \text{Scaled}: \quad & h_{scaled}(c, X_i) = \psi(|X_i|) \odot H \end{aligned}$$

(Additive) From Lemma 5 in [98], given that $\mathcal{X}$ is countable, there exists a mapping $Z : \mathcal{X} \to \mathbb{N}$ from $x \in \mathcal{X}$ to natural numbers. As $X \subset \mathcal{X}$ has a bounded size, there exists a number $N \in \mathbb{N}$ such that $\forall X, |X| < N$. Then an example of f is d-dimensional vector where each element is $N^{-Z(x)}$ and $\sum_{x \in X_i} f(x)$ is an injective function of multisets. Given that $\boldsymbol{W}_p, \boldsymbol{W}_q \in \mathbb{R}^{d' \times d}$ are full rank matrices where

$d' \geq d$, i.e. $\boldsymbol{W}_p$ and $\boldsymbol{W}_q$ are injective or invertible, $\sum_{x \in X_i} \boldsymbol{W}_q f(x)$ is injective as well. Hence, when $X_1 \neq X_2$, $\sum_{x \in X_1} \boldsymbol{W}_q f(x) \neq \sum_{x \in X_2} \boldsymbol{W}_q f(x)$. Therefore, $h_{add}(c, X_1) \neq h_{add}(c, X_2)$.

(Scaled) Let ψ map $|X_i|$ to a d'-dimensional vector $\boldsymbol{x}_i$ where each element is $|X_i|$. Given that $f(x)$ is not the zero vector $\mathbf{0}_d$ by definition, $\boldsymbol{W}_p$ is a full rank matrix with a trivial nullspace and attention score $\alpha_{cxi} > 0$ by definition, we can infer that H is not a zero vector $\mathbf{0}_{d'}$. Hence, when $X_1 \neq X_2$, $h_{scaled}(c, X_1) - h_{scaled}(c, X_2) = (\boldsymbol{x}_1 - \boldsymbol{x}_2) \odot H \neq \mathbf{0}_{d'}$. $\square$

Appendix B

Experiment Details

The following sections highlight the details of the experiments conducted in this work.

B.1 Datasets

The following sections summarize the OGB datasets used in this work. For further information, refer to the paper by Hu et al. [35] and https://ogb.stanford.edu/.

B.1.1 Graph Property Prediction

ogbg_molhiv: The **ogbg_molhiv** dataset consists of 41,127 molecular graphs adopted from MoleculeNet [95], with 25.5 nodes per graph and 27.5 edges per graph. Each graph (molecule) consists of atoms (nodes) and chemical bonds (edges), where each node has features such as atom type and chirality, and each edge (bond) has features such as bond type and aromaticity. The task is a binary classification problem to predict whether a molecule is HIV inhibitory or not with ROC-AUC as the evalation metric.

ogbg_ppa: The **ogbg_ppa** dataset consists of 158,100 undirected protein association graphs extracted from protein-protein association networks of 1,581 different species [84] that cover 37 broad taxonomic groups (e.g., mammals, bacterial families, archaeans), with 243.4 nodes per graph and 2,266.1 edges per graph. In each graph, nodes represent proteins and edges represent biologically significant protein-protein associations such as gene co-occurrence, gene fusion events, and co-expression. The task is a multi-class classification problem to predict what taxonomic group the graph originates from with accuracy as the evalation metric.

B.1.2 Link Property Prediction

ogbl_collab: The **ogbl_collab** dataset comprises of an undirected graph with $235,868$ nodes and $1,285,465$ edges representing a subset of the collaboration network between authors indexed by Microsoft Academic Graph (MAG) [86]. In the graph, nodes represent authors, and edges indicate collaborations between authors, such as co-authorship of academic papers. The task is a link prediction problem, where the goal is to determine whether a link (collaboration) will exist between two given nodes (authors) in the future based on the existing graph structure and properties. The evaluation metric used is Hits@K, where $K = 50$.

ogbl_ddi: The **ogbl_ddi** dataset comprises of a homogeneous, unweighted, undirected graph with $4,267$ nodes and $1,334,889$ edges representing the drug-drug interaction (DDI) network [94]. In the graph, nodes correspond to drugs, and edges represent known drug-drug interactions. The task is a link prediction problem, in which the goal is to determine whether a link (DDI) exists between two given nodes (drugs). The evaluation metric used is Hits@K, where $K = 20$.

B.1.3 Node Property Prediction

ogbn_arxiv: The **ogbn_arxiv** dataset consist of a directed graph with $169,343$ nodes and $1,166,243$ edges representing the citation network of all Computer Science (CS) arXiv papers indexed by MAG [86]. Each node on the graph corresponds to a paper and each directed edge indicates a citation from one paper to another. The nodes are equipped with various input features that include word embeddings of the title and abstracts of the papers. The task is a multi-class classification problem to predict the subject areas of the papers (nodes) into one of the given categories based on the graph structure and input features. The evaluation metric used is accuracy.

ogbn_mag: The **ogbn_mag** dataset consist of a heterogeneous graph with $1,939,743$ nodes and $21,111,007$ edges representing a large-scale academic graph that is a subset of MAG [86]. It consists of four types of entities: papers, authors, institutions, and fields of study. These entities are connected by multiple types of directed edges, representing various relationships like authorship, citation, authors' affiliations, and the relationships between papers and fields of study. The task is a multi-class classification problem to predict the venue of a given academic paper. The evaluation metric used is accuracy.

ogbn_products: The **ogbn_products** dataset consist of an undirected and unweighted graph with $2,449,029$ nodes and $61,859,140$ edges representing an Amazon product co-purchasing network [8]. Each node represents a product in Amazon with node features derived from product descriptions, and each edge between two nodes indicates that the two products are frequently co-purchased. The task is a multi-class classification problem to predict the category of a product out of 47 categories. The evaluation metric used is accuracy.

B.2 Design Space Grid Search

The following sections describe the range of values that the design dimensions take for the grid search for all datasets. For all datasets, the design dimensions shown in Table B.1 are fixed:

Design Dimension	Value
Batch Normalization	[True]
Activation Function	[PReLU]
Dropout	[0]
Aggregation Function	[add]
Optimizer	[Adam]
Learning Rate	[0.01]
Max Epoch	[200]

Table B.1: Fixed design dimensions for all datasets

Design Dimension	**Value**
Batch Size	[32, 128]
Graph Pooling	[add, mean, max]
MLP Layers Pre-Message-Passing	[2]
Message-Passing Layers	[2, 4]
MLP Layers Post-Message-Passing	[2]
Hidden Dimensions	[32, 128]
Layer Type	[GCN, GAT, GATv2, GIN, GAIN]
Stage Type	[stack, skip-sum, skip-concat]

Table B.2: Design space for **ogbg_molhiv** and **ogbg_ppa**. Total of 360 configurations each.

Design Dimension	**Value**
Batch Size	[512]
MLP Layers Pre-Message-Passing	[2]
Message-Passing Layers	[2]
MLP Layers Post-Message-Passing	[2]
Hidden Dimensions	[32, 128]
Layer Type	[GCN, GAT, GATv2, GIN, GAIN]
Stage Type	[stack, skip-sum, skip-concat]

Table B.3: Design space for **ogbl_collab**, **ogbn_mag** and **ogbn_products**. Total of 30 configurations each.

Design Dimension	**Value**
Batch Size	[512]
MLP Layers Pre-Message-Passing	[2]
Message-Passing Layers	[2, 4]
MLP Layers Post-Message-Passing	[2]
Hidden Dimensions	[32, 128]
Layer Type	[GCN, GAT, GATv2, GIN, GAIN]
Stage Type	[stack, skip-sum, skip-concat]

Table B.4: Design space for grid search for **ogbl_ddi**. Total of 60 configurations.

Graph pooling of add, mean and max represents global add, mean and max pooling, respectively, for a graph readout function for graph tasks (Equation 2.10). For GAT,

Design Dimension	Value
Batch Size	[128]
MLP Layers Pre-Message-Passing	[2]
Message-Passing Layers	[2, 4]
MLP Layers Post-Message-Passing	[2]
Hidden Dimensions	[32, 128]
Layer Type	[GCN, GAT, GATv2, GIN, GAIN]
Stage Type	[stack, skip-sum, skip-concat]

Table B.5: Design space for grid search for **ogbn_arxiv**. Total of 60 configurations.

GATv2 and GAIN, the number of attention heads used is 1. For GIN and GAIN, the MLP used for injective nonlinearity (σ) is a 2 layer MLP with ReLU activation function. In particular, we use GIN-0 for GIN where the learnable parameter ϵ in GIN is fixed to 0.

B.3 Best Model Design

The following sections state the model designs that gave the best test performance.

B.3.1 Graph Property Prediction

Tables B.6 and B.7 state the model designs that give the best test performance for **ogbg_molhiv** and **ogbg_ppa** respectively.

Layer Type	Batch Size	Graph Pooling	Message-Passing Layers	Hidden Dimensions	Stage Type
GCN	128	max	2	128	stack
GAT	128	add	2	128	skip-sum
GATv2	128	add	4	32	stack
GIN	32	add	2	32	skip-sum
GAIN	32	add	2	32	stack

Table B.6: Design for models with the best test results for **ogbg_molhiv**

Layer Type	**Batch Size**	**Graph Pooling**	**Message-Passing Layers**	**Hidden Dimensions**	**Stage Type**
GCN	128	add	4	128	stack
GAT	128	max	4	32	skip-concat
GATv2	128	add	4	32	skip-concat
GIN	128	mean	4	128	skip-sum
GAIN	128	add	4	128	skip-sum

Table B.7: Design for models with the best test results for **ogbg_ppa**

B.3.2 Link Property Prediction

Tables B.8 and B.9 state the model designs that give the best test performance for **ogbl_collab** and **ogbl_ddi** respectively.

Layer Type	**Batch Size**	**Message-Passing Layers**	**Hidden Dimensions**	**Stage Type**
GCN	512	2	128	skip-sum
GAT	512	2	128	skip-sum
GATv2	512	2	32	skip-sum
GIN	512	2	32	skip-sum
GAIN	512	2	128	skip-sum

Table B.8: Design for models with the best test results for **ogbl_collab**

Layer Type	**Batch Size**	**Message-Passing Layers**	**Hidden Dimensions**	**Stage Type**
GCN	512	4	128	skip-concat
GAT	512	4	128	stack
GATv2	512	4	128	skip-sum
GIN	512	4	128	stack
GAIN	512	4	128	skip-concat

Table B.9: Design for models with the best test results for **ogbl_ddi**

B.3.3 Node Property Prediction

Tables B.10, B.11 and B.12 state the model designs that give the best test performance for **ogbn_arxiv**, **ogbn_mag**, and **ogbn_products** respectively.

Layer Type	**Batch Size**	**Message-Passing Layers**	**Hidden Dimensions**	**Stage Type**
GCN	128	4	32	stack
GAT	128	4	32	stack
GATv2	128	4	32	skip-sum
GIN	128	4	32	stack
GAIN	128	4	32	stack

Table B.10: Design for models with the best test results for **ogbn_arxiv**

Layer Type	**Batch Size**	**Message-Passing Layers**	**Hidden Dimensions**	**Stage Type**
GCN	512	2	32	skip-concat
GAT	512	2	32	skip-concat
GATv2	512	2	32	skip-sum
GIN	512	2	128	stack
GAIN	512	2	128	stack

Table B.11: Design for models with the best test results for **ogbn_mag**

Layer Type	**Batch Size**	**Message-Passing Layers**	**Hidden Dimensions**	**Stage Type**
GCN	512	2	128	skip-concat
GAT	512	2	128	stack
GATv2	512	2	128	stack
GIN	512	2	128	skip-concat
GAIN	512	2	128	stack

Table B.12: Design for models with the best test results for **ogbn_products**

B.4 StarCraft Multi-Agent Challenge Hyperparameters

In general, we adopt the hyperparameters for MAPPO and SMAC from [102] with minimal changes. In particular, for all GCNet methods, we use the following MAPPO hyperparameters: buffer length of 200, mini-batch of 2, and MAPPO epoch of 5. Across all baselines, we alter the MAPPO clip parameter for some SMAC maps as stated in Table B.13 as follows:

SMAC Maps	**CommNet**	**GA-Comm**	**MAGIC**	**GCNet**				
				GCN	**GAT**	**GATv2**	**GIN**	**GAIN**
10m_vs_11m	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
1c3s5z	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
25m	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
27m_vs_30m	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
2c_vs_64zg	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
2m_vs_1z	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
2s3z	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
2s_vs_1sc	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
3m	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
3s5z	0.2	0.05	0.05	0.2	0.2	0.2	0.05	0.05
3s5z_vs_3s6z	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
3s_vs_3z	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
3s_vs_4z	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
3s_vs_5z	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
5m_vs_6m	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
6h_vs_8z	0.2	0.05	0.05	0.05	0.05	0.05	0.05	0.05
8m	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
8m_vs_9m	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
bane_vs_bane	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
corridor	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
MMM	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
MMM2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
so_many_baneling	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2

Table B.13: MAPPO clip parameter used for all methods and SMAC maps. The majority of the values remain the same from [102].

For GNN hyperparameters, we use the following: hidden dimensions of 128 and 4 layers of GNNs. For GAT, GATv2, and GAIN, the number of attention heads used is 8. For GIN and GAIN, the MLP used for injective nonlinearity (σ) is a 2 layer

MLP with ReLU activation function. In particular, we use GIN-0 for GIN where the learnable parameter ϵ in GIN is fixed to 0.

B.5 Additional Results

SMAC Maps	**CommNet**	**GA-Comm**	**MAGIC**	**GCNet**				
				GCN	**GAT**	**GATv2**	**GIN**	**GAIN**
10m_vs_11m	20.00 (1.71)	40.63 (31.79)	69.38 (27.98)	98.13 (4.19)	**100.00 (0.00)**	99.38 (1.40)	98.75 (2.80)	**100.00 (0.00)**
1c3s5z	98.13 (4.19)	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
25m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
27m_vs_30m	**100.00 (0.00)**	**100.00 (0.00)**	95.63 (8.15)	88.13 (11.35)	88.13 (4.08)	88.75 (6.48)	88.75 (8.15)	91.25 (6.77)
2c_vs_64zg	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2m_vs_1z	79.38 (44.39)	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2s3z	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
2s_vs_1sc	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3m	98.75 (1.71)	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3s5z	**100.00 (0.00)**	**100.00 (0.00)**	97.50 (3.42)	92.50 (6.48)	96.88 (6.99)	89.38 (10.03)	93.13 (5.59)	93.13 (5.59)
3s5z_vs_3s6z	17.50 (25.83)	**21.88 (14.99)**	12.50 (12.31)	4.38 (6.85)	3.13 (3.13)	6.25 (7.65)	3.75 (5.59)	11.25 (12.81)
3s_vs_3z	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
3s_vs_4z	**100.00 (0.00)**	62.50 (51.35)	**100.00 (0.00)**	80.00 (44.72)	80.00 (44.72)	80.00 (44.72)	80.00 (44.72)	80.63 (43.32)
3s_vs_5z	**100.00 (0.00)**	20.00 (44.72)	96.88 (4.42)	21.88 (43.69)	21.88 (40.32)	40.63 (48.91)	40.63 (54.22)	92.50 (13.37)
5m_vs_6m	92.50 (4.74)	73.75 (12.02)	86.88 (15.84)	92.50 (4.19)	91.88 (4.19)	92.50 (3.56)	**94.39 (2.60)**	92.50 (3.56)
6h_vs_8z	4.38 (1.71)	3.75 (1.40)	1.88 (2.80)	**66.88 (36.88)**	64.38 (40.30)	64.38 (41.73)	60.63 (45.75)	64.38 (40.30)
8m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
8m_vs_9m	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
bane_vs_bane	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
corridor	36.88 (50.60)	**80.00 (44.72)**	58.13 (53.11)	75.00 (42.04)	73.75 (41.49)	74.38 (41.72)	74.38 (41.72)	75.00 (42.04)
MMM	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**
MMM2	21.88 (30.46)	35.63 (49.09)	**95.63 (2.80)**	62.50 (25.00)	48.13 (30.51)	59.38 (21.20)	51.25 (20.80)	58.75 (21.01)
so_many_baneling	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**

Table B.14: Full results for SMAC. The standard deviations are in parentheses. The best results are in bold.

Bibliography

[1] Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *Proceedings of the Thirtieth International Joint Conference on Artifical Intelligence (IJCAI)*, 2021. 2.2.2

[2] Jeffrey L Adler and Victor J Blue. A cooperative multi-agent transportation management and route guidance system. *Transportation Research Part C: Emerging Technologies*, 10(5-6):433–454, 2002. 1

[3] Akshat Agarwal, Sumit Kumar, Katia Sycara, and Michael Lewis. Learning transferable cooperative behavior in multi-agent teams. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 1741–1743, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184. 2.2.1

[4] Waiss Azizian and Marc Lelarge. Expressive power of invariant and equivariant graph neural networks. *arXiv preprint arXiv:2006.15646*, 2020. 2.2.2

[5] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697, 2016. 2.1.3

[6] László Babai and Ludik Kucera. Canonical labelling of graphs in linear average time. In *20th annual symposium on foundations of computer science (sfcs 1979)*, pages 39–46. IEEE, 1979. 2.1.3

[7] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022. 2.2.2

[8] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016. URL `http://manikvarma.org/downloads/XC/XMLRepository.html`. B.1.3

[9] Jacobus C Biesmeijer and Thomas D Seeley. The use of waggle dance information by honey bees throughout their foraging careers. *Behavioral Ecology and*

Sociobiology, 59:133–142, 2005. 1

[10] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=F72ximsx7C1`. 2.2.2, 3.1.1, 3.1.1, 3.1.1, 4.1

[11] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. 2.1.3

[12] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019. 2.2.2

[13] Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. *Advances in Neural Information Processing Systems*, 34:1713–1726, 2021. 2.2.2

[14] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on machine learning*, pages 1538–1546. PMLR, 2019. 2.2.1

[15] George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. *arXiv preprint arXiv:1912.06058*, 2019. 2.2.2

[16] Stephen M Dawson. Clicks and communication: the behavioural and social contexts of hector's dolphin vocalizations. *Ethology*, 88(4):265–276, 1991. 1

[17] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019. 1

[18] Jiarui Feng, Lecheng Kong, Hao Liu, Dacheng Tao, Fuhai Li, Muhan Zhang, and Yixin Chen. Extending the design space of graph neural networks by rethinking folklore weisfeiler-lehman. *Advances in Neural Information Processing Systems*, 36, 2024. 2.2.2

[19] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 2.2.2

[20] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016. 2.2.1

[21] Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron.

Understanding and extending subgraph gnns by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35:31376–31390, 2022. 2.2.2

[22] Benjamin Freed, Rohan James, Guillaume Sartoretti, and Howie Choset. Sparse discrete communication learning for multi-agent cooperation through backpropagation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7993–7998. IEEE, 2020. 2.2.1

[23] Benjamin Freed, Guillaume Sartoretti, Jiaheng Hu, and Howie Choset. Communication learning via backpropagation in discrete channels with unknown noise. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7160–7168, 2020. 2.2.1

[24] Floris Geerts and Juan L Reutter. Expressiveness and approximation properties of graph neural networks. *arXiv preprint arXiv:2204.04661*, 2022. 2.2.2

[25] Martin Grohe. Isomorphism testing for embeddable graphs through definability. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 63–72, 2000. 2.1.3

[26] Martin Grohe. *Descriptive complexity, canonisation, and definable graph structure theory*, volume 47. Cambridge University Press, 2017. 2.1.3

[27] Martin Grohe. The logic of graph neural networks. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–17. IEEE, 2021. 2.1.3, 2.1.3

[28] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 2.1.2

[29] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 2.2.2

[30] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004. 2.1.2

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4.1

[32] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. 3.1.4

[33] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward

networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 3.1.4

[34] Guangzheng Hu, Yuanheng Zhu, Dongbin Zhao, Mengchen Zhao, and Jianye Hao. Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):3966–3978, 2021. 2.2.1

[35] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020. 1, 4, B.1

[36] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 4.1

[37] Neil Immerman and Eric Lander. *Describing graphs: A first-order approach to graph canonization*. Springer, 1990. 2.1.3

[38] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 4.1

[39] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31, 2018. 2.2.1

[40] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *ICLR*, 2020. 2.2.1

[41] Seth Karten and Katia Sycara. Intent-grounded compositional communication through mutual information in multi-agent teams. *Workshop on Decision Making in Multi-Agent Systems at International Conference on Intelligent Robots and Systems (IROS)*, 2022. 2.2.1

[42] Seth Karten, Mycal Tucker, Siva Kailas, and Katia Sycara. Towards true lossless sparse communication in multi-agent systems. In *2023 ieee international conference on robotics and automation (icra)*, pages 7191–7197. IEEE, 2023. 2.2.1

[43] Seth Karten, Mycal Tucker, Huao Li, Siva Kailas, Michael Lewis, and Katia Sycara. Interpretable learned emergent communication for human–agent teams. *IEEE Transactions on Cognitive and Developmental Systems*, 15(4):1801–1811, 2023. 2.2.1

[44] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

2.2.2

[45] Woojun Kim, Jongeui Park, and Youngchul Sung. Communication in multi-agent reinforcement learning: Intention sharing. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=qpsl2dR9twy`. 2.2.1

[46] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`. 2.2.2, 4.1

[47] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274, 2013. 1

[48] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999. 2.1.2

[49] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 2.1.2

[50] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7211–7218, 2020. 2.2.1, 3, 4.2

[51] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017. 2.1.2

[52] Aleksandra Malysheva, Tegg Taekyong Sung, Chae-Bong Sohn, Daniel Kudenko, and Aleksei Shpilman. Deep multi-agent reinforcement learning with relevance graphs. *arXiv preprint arXiv:1811.12557*, 2018. 2.2.1

[53] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni. Learning agent communication under limited bandwidth by message pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5142–5149, 2020. 2.2.1

[54] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019. 2.1.3, 2.2.2

[55] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Rep-*

resentations, 2019. URL https://openreview.net/forum?id=Syx72jC9tm. 2.2.2

[56] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pages 4363–4371. PMLR, 2019. 2.2.2

[57] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 1

[58] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019. 2.2.2

[59] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019. 1, 2.1.3, 2.1.4, 1, 2.1.4, 2, 2.1.4

[60] Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. *Advances in Neural Information Processing Systems*, 33:21824–21840, 2020. 2.2.2

[61] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far, 2021. 2.1.3

[62] Christopher Morris, Gaurav Rattan, Sandra Kiefer, and Siamak Ravanbakhsh. Speqnets: Sparsity-aware permutation-equivariant graph networks. In *International Conference on Machine Learning*, pages 16017–16042. PMLR, 2022. 2.2.2

[63] Matthew Morris, Thomas D Barrett, and Arnu Pretorius. Universally expressive communication in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:33508–33522, 2022. 1, 2.1.5, 2.2.2

[64] Yaru Niu, Rohan R Paleja, and Matthew C Gombolay. Multi-agent graph-attention communication and teaming. In *AAMAS*, volume 21, page 20th, 2021. 2.2.1, 3, 4.2

[65] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016. 2.1.2

[66] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V

Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019. 1

[67] Pál András Papp and Roger Wattenhofer. A theoretical comparison of graph neural network extensions. In *International Conference on Machine Learning*, pages 17323–17345. PMLR, 2022. 2.2.2

[68] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021. 2.2.2

[69] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017. 2.2.1

[70] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9): 1180–1190, 2008. 2.1.2

[71] Chendi Qian, Gaurav Rattan, Floris Geerts, Mathias Niepert, and Christopher Morris. Ordered subgraph aggregation networks. *Advances in Neural Information Processing Systems*, 35:21030–21045, 2022. 2.2.2

[72] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019. 1, 4

[73] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pages 333–341. SIAM, 2021. 2.2.2

[74] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015. 2.1.2

[75] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 2.1.2

[76] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2.1.2

[77] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016. 1

[78] Junjie Sheng, Xiangfeng Wang, Bo Jin, Junchi Yan, Wenhao Li, Tsung-Hui Chang, Jun Wang, and Hongyuan Zha. Learning structured communication for multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(2):50, 2022. 2.2.1

[79] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550 (7676):354–359, 2017. 1

[80] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018. 2.2.1

[81] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 4.1

[82] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016. 2.2.1, 4.2

[83] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2.1.2

[84] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 47(D1):D607–D613, 2019. B.1.1

[85] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`. 2.2.1, 2.2.2, 3.1.1, 4.1

[86] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020. B.1.2, B.1.3, B.1.3

[87] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019. 2.2.2

[88] Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich.

Learning efficient multi-agent communication: An information bottleneck approach. In *International Conference on Machine Learning*, pages 9908–9918. PMLR, 2020. 2.2.1

[89] Yutong Wang and Guillaume Sartoretti. Fcmnet: Full communication memory net for team-level cooperation in multi-agent systems. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, page 1355–1363, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136. 2.2.1

[90] Yuyang Wang, Zijie Li, and Amir Barati Farimani. Graph neural networks for molecules. In *Machine Learning in Molecular Sciences*, pages 21–66. Springer, 2023. 1

[91] Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. *arXiv preprint arXiv:1301.2315*, 2013. 2.1.2

[92] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968. 1, 2.1.3

[93] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992. 2.1.2

[94] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018. B.1.2

[95] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. B.1.1

[96] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008. 1

[97] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018. 3.2

[98] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`. 1, 2.1.4, 1, 2.1.4, 2.2.2, 3, 3.1.3, 4.1, A.2

[99] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement

learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020. 2.1.2

[100] Jiaxuan You, Rex Ying, and Jure Leskovec. Design space for graph neural networks. In *NeurIPS*, 2020. 1, 4.1

[101] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10737–10745, 2021. 2.2.2

[102] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022. (document), 2.1.2, 4.2, B.4, B.13

[103] Lei Yuan, Jianhao Wang, Fuxiang Zhang, Chenghe Wang, Zongzhang Zhang, Yang Yu, and Chongjie Zhang. Multi-agent incentive communication via decentralized teammate modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9466–9474, 2022. 2.2.1

[104] Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests. In *International Conference on Machine Learning*, pages 41019–41077. PMLR, 2023. 2.2.2

[105] Muhan Zhang and Pan Li. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34:15734–15747, 2021. 2.2.2

[106] Shuo Zhang and Lei Xie. Improving attention mechanism in graph neural networks via cardinality preservation. In *IJCAI: proceedings of the conference*, volume 2020, page 1395. NIH Public Access, 2020. 2.2.2, 3, 3.1.2, 3.1.2, 3.1.3, 3.1.3, 3.1.4, 3.1.4

[107] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=Mspk_WYKoEH`. 2.2.2

[108] Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. Analysis and improvement of policy gradient estimation. *Advances in Neural Information Processing Systems*, 24, 2011. 2.1.2

[109] Cai Zhou, Xiyuan Wang, and Muhan Zhang. From relational pooling to subgraph gnns: A universal framework for more expressive graph neural networks. In *International Conference on Machine Learning*, pages 42742–42768. PMLR, 2023. 2.2.2

[110] Junru Zhou, Jiarui Feng, Xiyuan Wang, and Muhan Zhang. Distance-restricted

folklore weisfeiler-leman gnns with provable cycle counting power. *Advances in Neural Information Processing Systems*, 36, 2024. 2.2.2

[111] Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(1), 2024. 2.1.2