Cooperative Information Dissemination with Graph-based Multi-Agent Reinforcement Learning

Raffaele Galliera[†,⋆], **Kristen Brent Venable**[†,⋆], **Matteo Bassani**[†], **Niranjan Suri**[†,⋆,‡]
[†]Institute for Human & Machine Cognition
[⋆]Department of Intelligent Systems & Robotics, The University of West Florida
Pensacola, FL, USA
[‡]US Army Research Laboratory
Adelphi, MD, USA
{rgalliera, bvenable, mbassani, nsuri}@ihmc.org

Abstract

Efficient information dissemination is crucial for supporting critical operations across domains like disaster response, autonomous vehicles, and sensor networks. This paper introduces a Multi-Agent Reinforcement Learning (MARL) approach as a significant step forward in achieving more decentralized, efficient, and collaborative information dissemination. We propose a Partially Observable Stochastic Game (POSG) formulation for information dissemination empowering each agent to decide on message forwarding independently, based on the observation of their one-hop neighborhood. This constitutes a significant paradigm shift from heuristics currently employed in real-world broadcast protocols. Our novel approach harnesses Graph Convolutional Reinforcement Learning and Graph Attention Networks (GATs) with dynamic attention to capture essential network features. We propose two approaches to accomplish cooperative information dissemination, L-DyAN and HL-DyAN, differing in terms of the information exchanged among agents. Our experimental results show that our trained policies outperform existing methods, including the state-of-the-art heuristic, in terms of network coverage and communication overhead on dynamic networks of varying density and behavior.

1 Introduction

Group communication, implemented in a broadcast or multicast fashion, finds a natural application in different networking scenarios, such as Vehicular Ad-hoc Networks (VANETs) (Tonguz et al., 2007; Ibrahim et al., 2020), with the necessity to disseminate information about the nodes participating, e.g. identity, status, or crucial events happening in the network. These systems can be characterized by congestion-prone networks and/or different resource constraints, such that message dissemination becomes considerably expensive if not adequately managed. For this matter, message forwarding calls for scalable and distributed solutions able to minimize the total number of forwards, while achieving the expected coverage. Moreover, modern broadcast communication protocols often require careful adjustments of their parameters before achieving adequate forwarding policies, which would otherwise result in sub-optimal performance in terms of delivery ratio and latency (Suri et al., 2022).

Recently, researchers have considered learning communication protocols (Foerster et al., 2016) with Multi-Agent Reinforcement Learning (MARL) (Buşoniu et al., 2010). At its core, MARL seeks to design systems where multiple agents learn to optimize their objective by interacting with the environment and the other entities involved. Such tasks can be competitive, cooperative, or a combination of both, depending on the scenario. As agents interact within a shared environment, they often find the need to exchange information to optimize their collective performance. This has led to the development of communication mechanisms that are learned rather than pre-defined, allowing agents to cooperate better utilizing their learned signaling system.

Nevertheless, learning to communicate with MARL comes with several challenges. In multi-agent systems, actions taken by one agent can significantly impact the rewards and state transitions of other agents, rendering the environment more complex and dynamic, and ensuring that agents develop a shared and consistent communication protocol, is an area of active research. Methods such as CommNet (Sukhbaatar et al., 2016) and BiCNet (Peng et al., 2017), focus on the communication of local encodings of agents' observations. These approaches allow agents to share a distilled version of their perspectives, enabling more informed collective decision-making. ATOC (Jiang and Lu, 2018) and TarMAC (Das et al., 2019) have ventured into the realm of attention mechanisms. By leveraging attention, these methods dynamically determine which agents to communicate with and what information to share, leading to more efficient and context-aware exchanges. Yet another approach, as exemplified by Graph Convolutional Reinforcement Learning (DGN) (Jiang et al., 2020), harnesses the power of Graph Neural Networks (GNNs) and attention mechanisms to model the interactions, relations, and communications between agents.

However, to the best of our knowledge, no MARL-based method involving proactive communication and GNNs has been proposed to address the unique challenges of optimizing the process of information dissemination within a broadcast dynamic network. In such a scenario, nodes need to cooperate to spread the information by forwarding it to their immediate neighbors, which might change over time, while relying on their limited observation of the entire graph. Furthermore, their collaboration and ability to accomplish dissemination are bound by the limitations of the underlying communication channels. This means that both the quantity of forwarding actions and the amount of information exchange needed for effective cooperation are constrained and should be minimized.

In this work, we introduce **a novel Partially Observable Stochastic Game (POSG) for optimized information dissemination** in dynamic broadcast networks, forming the basis for our MARL framework.[1] To this end, we design a MARL algorithm to encourage **cooperation within dynamic neighborhoods** where node connections are frequently changing. Furthermore, we design and test **two distinct architectures**, namely Local Dynamic Attention Network (L-DyAN) and Hyperlocal Dynamic Attention Network (HL-DyAN), which require different levels of communication leveraging Graph Attention Network (GAT) with dynamic attention (Brody et al., 2022) and Dueling Q-Networks (Wang et al., 2016).

Our **experimental study** demonstrates our methods' efficacy in achieving superior network coverage across dynamic graphs in different scenarios, outperforming DGN and the established Multipoint Relay (MPR) (Dearlove and Clausen, 2014) heuristic. Moreover, our approach operates on one-hop observations and empowers nodes to take independent forwarding decisions, unlike MPR. By exploring the potential of learning-based approaches for addressing information dissemination in dynamic networks, our work underscores the versatility of MARL in present and future, real-world applications such as information dissemination in social networks (Guille et al., 2013), space networks (Ye and Zhou, 2021), and vehicle communication services (Ma et al., 2012).

2 Background

Optimized Flooding in Broadcast Networks A dynamic broadcast network can be represented as a dynamic graph $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, where each node represents a (possibly) mobile node and an edge between two nodes at time t represents the two corresponding nodes being within each other's broadcasting range at that time. Hence, for every node $v \in \mathcal{V}$, the set of its neighbors at time t is defined as $\mathcal{N}_v(t) = \{u \in \mathcal{V} | (v, u) \in \mathcal{E}(t)\}$.

A main objective of broadcast communications over connected networks is called Optimized Flooding (Qayyum et al., 2002) and it is achieved when the information emitted from a given node $v \in \mathcal{V}$ reaches every other node $u \neq v$, thanks to forwarding actions of a set of nodes $\mathcal{D} \subseteq \mathcal{V}$. While maximizing coverage it is also desirable to minimize redundant transmissions, which might impact resource utilization, such as bandwidth, power consumption, and latency. From a graph-theoretic point of view this can be achieved by identifying a specific subset of nodes, called a Minimum Connected Dominating Set (MCDS), that will be tasked with forwarding the information. This task requires the introduction of a centralized entity with complete knowledge of the network state and has been shown to be NP-complete (Garey and Johnson, 1979). A much more efficient and realistic approach is to approximate the MCDS in a distributed manner, relying only on local observations of

[1] Repository link obscured.

the network made from each node's perspective. Indeed, this is the approach taken by the MPR heuristic and our MARL approach.

3 Related Work

The MPR selection algorithm is a technique developed to efficiently disseminate information in Mobile Ad-hoc Networks (MANETs) and wireless mesh networks. It achieves this by having each node designate certain one-hop neighbors to forward messages arriving from them, thereby reducing the overall transmission load and preventing excessive network broadcasting. This process involves nodes exchanging "HELLO messages" to identify and select their MPR sets, ensuring effective network coverage with minimal redundancy.

In real-world networking protocols, MPR plays an essential role. For instance, in the Optimized Link State Routing (OLSR) protocol (Dearlove and Clausen, 2014), MPR is fundamental in managing the distribution of Topology Control (TC) messages. Similarly, in Simplified Multicast Forwarding (SMF) (Macker, 2012), MPR is employed primarily for the efficient forwarding of multicast packets.

In this work, we compare our approach with the MPR selection algorithm, as outlined in the standard OLSR implementation (Dearlove and Clausen, 2014), leveraging this algorithm as a baseline for distributed message dissemination in dynamic graph structures.[2] However, we define a completely different approach that leverages MARL and, unlike MPR, only requires an anonymized knowledge of the one-hop neighborhood and empowers each agent to independently decide their message forwarding policy.

4 Method

In this section, we describe our novel MARL approach for optimizing information flooding in dynamic broadcast networks. We start by presenting a POSG formulation and then introduce our learning method and our two architectures, L-DyAN and HL-DyAN, designed to achieve efficient dissemination while requiring different degrees of communication.

4.1 MARL Formulation

We envision the dissemination process discretized into timesteps and episodes starting with a source node transmitting the information (a message) to its immediate (one-hop) neighbors. Each node in the graph corresponds to an agent observing its one-hop neighborhood and their features. At every timestep, nodes that have received the message will sense their neighborhood and decide whether to forward it to its current one-hop neighbors or stay silent. However, agents do not have any control or information on who will be part of their neighborhood at the next time-step. Finally, the agents' objective is to disseminate the message emitted from the source node, i.e. maximize the network coverage, while minimizing the amount of forwarding actions, i.e. the messages, required.

An agent becomes a meaningful actor, receiving appropriate reinforcement signals only once it receives the message and for a limited number of steps. We capture this by implementing two different elements of our POSG. On the one hand, we distinguish "Graph Episodes" from "Agent Episodes", allowing the agents to dynamically enter and leave the game independently. For this purpose, upon message reception, we limit the Agent's Episode to a fixed number of steps (*local horizon*) during which it decides whether to forward the message to its immediate neighbors or not. Graph Episodes, model the overall dissemination process, and terminate once every agent that has received the message has exhausted its local horizon. Given the agents' asynchronous presence, reward signals are issued individually to each agent, but capture the necessary degree of cooperation within their neighborhoods.

In our formulation agents are anonymous (i.e. not identified by any ID) and sense only their immediate neighborhood, accessing the degree of connectivity of such neighbors and observing their forwarding behavior. This is far more parsimonious than what is required by MPR that requires agents to obtain a complete, identified, two-hop knowledge. More specifically, given the broadcast network represented by graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E}_0)$ at

[2]The complete MPR selection algorithm used by OLSR is shown in Algorithm 1-Appendix A.1

time t_0, and node $n_s \in \mathcal{V}$, we define the POSG associated to the optimized flooding of $\mathcal{G}_0$ with source n_s and network update function $\mathcal{U}$, with the tuple $\langle \mathcal{I}, \mathcal{S}, \mathcal{A}^i_{i\in I}, \mathcal{U}, \mathcal{P}, \mathcal{R}^i_{i\in I}, \mathcal{O}^i_{i\in I}, \gamma \rangle$, where:

Agents set $\mathcal{I}$. Set $\mathcal{I}$ contains one agent for each node in $\mathcal{V}$. $\mathcal{I}$ is divided into three disjoint sets which are updated at every timestep t: the active set $\mathcal{I}_a(t)$, the done set $\mathcal{I}_d(t)$, and the idle set $\mathcal{I}_i(t)$. Agents in $\mathcal{I}_i(t)$ are inactive because they have not received the message yet. At the beginning of the process, $\mathcal{I}_i(t)$ will contain all agents except the one associated with n_s. Agents in $\mathcal{I}_d(t)$ are also inactive, after participating in the game and terminating their Agent Episode once the local horizon is reached. $\mathcal{I}_a(t)$, instead, includes the set of agents actively participating in the game at timestep t. Agents in $\mathcal{I}_i(t)$ are moved to $\mathcal{I}_a$ at time step $t+1$, if they have been forwarded the information, hence starting their Agent Episode.

Actions $\mathcal{A}^i_{i\in I}$. For any time step t, if agent i is in $\mathcal{I}_a(t)$, then, $\mathcal{A}^i$ contains two possible actions: forward the information to their neighbors or stay silent. The action set for agents in $\mathcal{I}_i(t)$ and $\mathcal{I}_d(t)$ is, instead, empty.

Environment Dynamics $\mathcal{P}$ and Network Update $\mathcal{U}$. The environment dynamics are defined by the transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^{|I|} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ represents the set of probability distributions over the state space $\mathcal{S}$. In our POSG model, we incorporate a general stochastic network update function, $\mathcal{U}$, controlling how the edges of the network change over time. This element allows us to capture various dynamics such as agent mobility or other factors that may affect a network's connectivity. More formally, at every timestep t, the graph structure is updated such that $\mathcal{G}_{t+1} = \mathcal{G}(\mathcal{V}, \mathcal{U}(\mathcal{E}_t))$.

The message-forwarding mechanism is purposefully modeled as deterministic and, at each timestep t, if an active agent i forwards the message, all nodes in $\mathcal{N}_i(t)$ will receive it.

Observations $\mathcal{O}^i_{i\in I}$ and State set $\mathcal{S}$. Each node in the graph has a set of three features observable by other neighboring agents at each time step t: neighborhood size, the number of messages transmitted, and its last action. The agents' observations are represented as the graph describing their one-hop neighborhood and the features associated with each node in this local structure. In our setting, the state $\mathcal{S}_t$ corresponds to the current graph structure $\mathcal{G}_t$ and the following information for each node: its features, the set to which the agent belongs $\mathcal{I}_a$, $\mathcal{I}_d$, or $\mathcal{I}_i$, and the remaining steps of the local horizon for those in $\mathcal{I}_a$.

Rewards $\mathcal{R}^i{}_{i\in I}$. At the end of each step every agent in $\mathcal{I}_a$ is issued with a reward signal with positive and negative components. The positive term rewards the agent based on its two-hop coverage, i.e. how many one- and two-hop neighbors have received the information. One of two penalties might be issued, based on the agent's behavior. If the agent has forwarded during its last action, it will participate in a shared transmission cost, punishing the agent for the number of messages sent by its neighborhood. Otherwise, it will receive penalties based on the unexploited coverage potential of neighbors who have not yet received the information. Formally, the reward signal for agent i at time t be defined as follows:

$$r_{i,t} = \frac{\upsilon(\mathcal{M}_i(t), t)}{|\mathcal{M}_i(t)|} - p(i,t), \mathcal{M}_i(t) = \bigcup_{u \in \mathcal{N}_i(t) \cup \{i\}} \mathcal{N}_u(t) \setminus \{i\} \tag{1}$$

$$p(i,t) = \begin{cases} m(\mathcal{N}_i(t), t), & \text{if } i \in \mathcal{T}(t) \\ \mu(\mathcal{N}_i(t), t), & \text{if } i \in \mathcal{I}_a(t) \setminus \mathcal{T}(t) \end{cases} \tag{2}$$

In Equation 1, $\mathcal{M}_i(t)$ represents the set of two-hop neighbors of agent i at t. $\upsilon(\mathcal{M}_i, t)$ denotes the number of them that by timestep t have already received the message, while $p(i,t)$, defines the penalties assigned to agent i. The latter is further described in Equation 2, where $\mathcal{T}(t)$ is the set of active agents that have forwarded the message at least once. Here $m(\mathcal{N}_i(t), t)$ denotes the sum of the number of messages transmitted by the current neighborhood of agent i by timestep t. The term $\mu(\mathcal{N}_i(t), t)$ instead defines the Maximum Normalized Coverage Potential of node i:

$$\mu(\mathcal{N}_i(t), t) = \frac{\max(\mathcal{C}_i(t))}{\sum \mathcal{C}_i(t)} \qquad \mathcal{C}_i(t) = \{|\mathcal{N}_j(t)| : j \in \mathcal{N}_i(t) \cap \mathcal{I}_i(t)\} \tag{3}$$

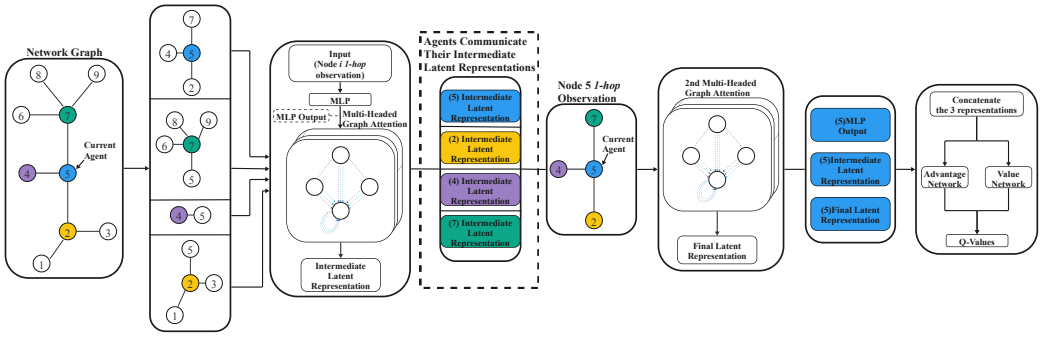

Figure 1: Information flow from a single agent observation (5) to the produced Q-Values in L-DyAN.

On the one hand, we note that by assessing the ability of an agent's neighborhood to reach nodes beyond its immediate neighbors, Equation 1, encourages agents to collectively cover more nodes through coordination within their vicinity. On the other hand, the neighborhood-shared transmission steers the agents away from redundancy, promoting efficient dissemination. Finally, the Maximum Normalized Coverage Potential counterbalances the shared transmission costs, by hastening transmission to nodes with highly populated neighborhoods that have not yet been reached.

4.2 Learning Approach

The idea behind L-DyAN and HL-DyAN is to encourage cooperation within dynamic neighborhoods, where links between nodes can form and/or disappear over time. We therefore propose a loss function comprising neighborhood experiences, the usage of GAT layer(s) with dynamic attention (Brody et al., 2022), and the presence of a dueling network to separately estimate the state-value and the advantages for each action (Wang et al., 2016). The choice of a GAT layer with dynamic attention is driven by its capability of capturing expressive attention mechanisms within a graph, a feature shown to be weaker in dot-product attention, as used in DGN (Brody et al., 2022).

4.2.1 Cooperative Dynamic Neighborhoods

During training, at each timestep t, the tuple $(\mathcal{O}_{\mathcal{I}_a(t)}, \mathcal{A}_{\mathcal{I}_a(t)}, \mathcal{R}_{\mathcal{I}_a(t)}, \mathcal{O}'_{\mathcal{I}_a(t)})$ is stored in a circular replay buffer with a fixed length. $\mathcal{O}_{\mathcal{I}_a(t)}$ indicates the set of observations of all agents in $\mathcal{I}_a(t)$, $\mathcal{A}_{\mathcal{I}_a(t)}$ the set of actions taken by these agents, $\mathcal{R}_{\mathcal{I}_a(t)}$ is the set of rewards, and $\mathcal{O}'_{\mathcal{I}_a(t)}$ the set of observations of agents in $\mathcal{I}_a(t)$ at the next timestep.

At each training step, we sample a random batch $\mathcal{B}$ from the replay buffer, with every sample containing the experience of some agent i and the ones of its current and active neighbors $\mathcal{N}_{i,\mathcal{I}_a(t)} = \mathcal{N}_i(t) \cap \mathcal{I}_a(t)$. The loss for each sample is computed not only based on the agent's own experience but also considering the experiences of its active neighbors. We denote $\mathcal{N}^{+i}_{i,\mathcal{I}_a(t)} = \mathcal{N}_{i,\mathcal{I}_a(t)} \cup \{i\}$ and define the loss function:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{\mathcal{B}} \frac{1}{|\mathcal{N}^{+i}_{i,\mathcal{I}_a(t)}|} \sum_{j \in \mathcal{N}^{+i}_{i,\mathcal{I}_a(t)}} \left(y_t^j - Q(o^j, a^j; \theta) \right)^2, \tag{4}$$

where, for each agent j, y_t^j is the target return and $Q(o^j, a^j; \theta)$ the predicted Q value, parameterized with θ, given the observation o^j and action a^j. From this point onward, we will drop the superscript j when referring to o, a, r, and y as they will refer to a single experience.

Additionally, we take advantage of the agents' short-lived experiences and perform n-step returns, with n equal to the local horizon (k). We note that the replay buffer is temporally sorted and organized such that every individual episode, ongoing or terminated with a length up to k, can be uniquely identified. If the buffer contains the remaining steps until the termination of the agent's episode, the n-step computation serves an unbiased value of the return: $y_t = \sum_{i=0}^{k-t} \gamma^i r_{t+i}$.

If the trajectory stored in the buffer contains only the next j steps before termination, y_t will be estimated as:

$$y_t = \sum_{i=0}^{j-1} \gamma^i r_{t+i} + \gamma^j Q(o_{t+i}, \text{argmax}_{a' \in \mathcal{A}} Q(o_{t+i}, a'; \theta); \bar{\theta}), \tag{5}$$

where θ is the current network and $\bar{\theta}$ is the target network.

4.2.2 Local-DyAN

The first architecture we propose is depicted in Figure 1 and consists of an encoder module comprised of three different stages: one Multi Layer Perceptron (MLP) followed by two multi-headed GATs (Veličković et al., 2018) with dynamic attention (Brody et al., 2022). The final latent representation will comprise the concatenation of each stage output, which is then fed to a dueling network decoding the final representation into the predicted Q values. After each encoding stage, a ReLU activation function is applied.

We now describe the flow from the agent's observation to the Q values prediction and we show how it can be integrated into broadcast communication protocols. Agent i's observation at time t is first fed to the MLP encoding stage. This results in a learned representation of the features belonging to agent i and its neighbors, denoted respectively $\mathbf{x}_i$ and $\mathbf{x}_j, \forall j \in \mathcal{N}_i(t)$. Following such encoding stage, the output of each of the M attention heads of the first GAT is:

$$\mathbf{x}_i^m = \alpha_{i,i}^m \mathbf{W}\mathbf{x}_i + \sum_{j \in \mathcal{N}_i(t)} \alpha_{i,j}^m \mathbf{W}\mathbf{x}_j \; \forall m \in \{0, ..., M-1\}, \tag{6}$$

where the dynamic attention α^m for the tuple (i, j), denoted as $\alpha_{i,j}^m$, is computed by:

$$\alpha_{i,j}^m = \frac{\exp\left(\mathbf{a}^\top \mathrm{LeakyReLU}\left(\mathbf{W}[\mathbf{x}_i \,\|\, \mathbf{x}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i(t) \cup \{i\}} \exp\left(\mathbf{a}^\top \mathrm{LeakyReLU}\left(\mathbf{W}[\mathbf{x}_i \,\|\, \mathbf{x}_k]\right)\right)}, \tag{7}$$

where $\mathbf{a}$ and $\mathbf{W}$ are learned. We denote $\hat{\mathbf{X}}_i = \mathbf{x}_i^0 || \mathbf{x}_i^1 || ... || \mathbf{x}_i^{M-1}$, where $||$ is the concatenation operator, as the concatenation of every attention output. Through message passing, each agent i receives $\hat{\mathbf{X}}_j, \forall j \in \mathcal{N}_i(t)$. These new representations are fed to the second GAT layer, where the computation follows the same logic seen in Equation 6 and 7, producing the embedding $\hat{\mathbf{Z}}_i$.

Finally, the output of each encoding stage is concatenated in a final latent representation $\mathbf{H}_i$:

$$\mathbf{H}_i = \mathbf{x}_i || \hat{\mathbf{X}}_i || \hat{\mathbf{Z}}_i. \tag{8}$$

At this point, $\mathbf{H}_i$ is fed to the two separate streams of the dueling network, namely the value network V and the advantage network A, parameterized by two separate MLPs with parameters α and β, respectively. Let us denote the parameterization previous to the dueling network, which produced the final latent representation $\mathbf{H}_i$ given o, as δ. The predicted Q values are then obtained as:

$$Q(o, a; \delta, \alpha, \beta) = V(o; \delta, \alpha) + \left(A(o, a; \delta, \beta) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} A(o, a'; \delta, \beta) \right). \tag{9}$$

We note that the encoding process described above harmoniously integrates with the communication mechanisms present in protocols deployed in the real world, such as OLSR. We envision every node (agent) in the network, feeding its neighborhood structure through the encoding process described above. Subsequently, every agent shares their intermediate representation $\hat{\mathbf{X}}_i$ with their one-hop neighbors, in a similar way to how nodes communicate their MPR sets in OLSR. Once the representations are collected, agents feed them to the second GAT layer, obtaining $\mathbf{H}_i$.

Table 1: Evaluation of L-DyAN, HL-DyAN, MPR, and DGN in terms of Coverage and Messages forwarded.

Initial Node Degree	Nodes Speed	Metric	L-DyAN	HL-DyAN	MPR	DGN
5 Neighbors	$1\frac{\text{unit}}{\text{step}}$	Messages	28.22	34.81	25.64	3.88
		Coverage	**93.78%**	90.02%	86.24%	24.34%
5 Neighbors	$6\frac{\text{units}}{\text{step}}$	Messages	24.05	33.40	8.21	6.69
		Coverage	79.34%	**83.64%**	34.02%	35.77%
10 Neighbors	$1\frac{\text{unit}}{\text{step}}$	Messages	23.26	33.06	22.95	15.91
		Coverage	88.51%	**89.59%**	85.73%	24.35%
10 Neighbors	$6\frac{\text{units}}{\text{step}}$	Messages	24.31	37.01	8.10	7.03
		Coverage	86.33%	**91.69%**	37.98%	42.74%
25 Neighbors	$1\frac{\text{unit}}{\text{step}}$	Messages	23.74	34.15	24.93	3.28
		Coverage	90.44%	**93.84%**	92.86%	26.33%
25 Neighbors	$6\frac{\text{units}}{\text{step}}$	Messages	24.29	36.35	10.03	6.96
		Coverage	88.19%	**92.23%**	46.80%	44.02%

However, embedding the communication process generates a communication overhead of size proportional to $\hat{\mathbf{X}}_i$, an aspect which might need to be further minimized in bandwidth-constrained networks (Suri et al., 2023; Galliera et al., 2023). This observation leads us to our second approach.

4.2.3 Hyperlocal-DyAN

Intending to generate less communication overhead, we design a second model, named HL-DyAN, which resembles L-DyAN in its form. We replace the three encoding stages with a single GAT layer with dynamic attention. Within agent i's observation, we apply the GAT encoding process to every node, followed by a ReLU activation function. Finally, a global max-pooling layer is applied to summarize the most salient neighborhood characteristics. The rationale for this approach is that agents can make informed decisions by processing their one-hop neighborhood dynamics from each neighbor's perspective, eliminating the need to share their latent representations, hence reducing the communication overhead, as seen in L-DyAN.

In detail, agent i's observation at time t is fed to the GAT layer and, as opposed to L-DyAN, such an operation is repeated for every node within the local observation of agent i, producing a set of latent representations comprising $\hat{\mathbf{Y}}_i$ and $\hat{\mathbf{Y}}_j, \forall j \in \mathcal{N}_i(t)$. We then perform global max pooling, obtained through a feature-wise max operation:

$$\mathbf{H}_i = \max_{j \in \mathcal{N}_i(t) \cup \{i\}} \hat{\mathbf{Y}}_j. \tag{10}$$

Finally, $\mathbf{H}_i$ is fed to the dueling network following the same process described in Equation 9.

5 Experiments

Experimental Setup We generated 50,000 connected graph topologies for training, each consisting of 50 nodes with a broadcasting range of 20 units and no constraints on the number of neighbors. For every learning algorithm, training was conducted five times adopting different random seeds (4, 9, 17, 42, 43) for 1 million agent steps. Furthermore, we utilized 4 distinct sets of connected starting graph topologies, not seen during training, for testing purposes. These sets comprised 50 nodes per graph with various constraints on the maximum intial node degree allowed (5, 10, and 25). Our evaluation process involved testing each graph 50 times and selecting a different node as the source n_s in each iteration. Additional evaluations were conducted on the impact of the nodes' velocity setting their speed to 1, 6, and 10 $\frac{\text{units}}{\text{step}}$. Further experiments with different constraints are presented in Section A.2 of the Appendix.

Our analysis compared L-DyAN and HL-DyAN with the MPR heuristic and DGN. The DGN methodology excluded Temporal Relation Regularization, as it was unnecessary in our setting where agent interaction is temporally bounded by a short local horizon. To ensure a fair evaluation, we maintained consistent hyperparameters across all models, the details of which are presented in Table 3-Appendix A.4.1.

Results Table 1 shows the results of our experiments in terms of coverage and messages required to achieve it, presenting the means and standard deviations. Our proposed methods, L-DyAN and HL-DyAN, consistently demonstrate higher coverage across different scenarios compared to MPR and DGN. In particular, HL-DyAN achieves the highest coverage with a mean of 90.37%. However, this comes at the cost of a higher number of messages, with HL-DyAN generating an average of 34.85 messages per episode. L-DyAN, while slightly less successful in coverage (87.70%), requires significantly fewer messages (24.32), indicating its suitability in scenarios where message efficiency is prioritized over coverage.

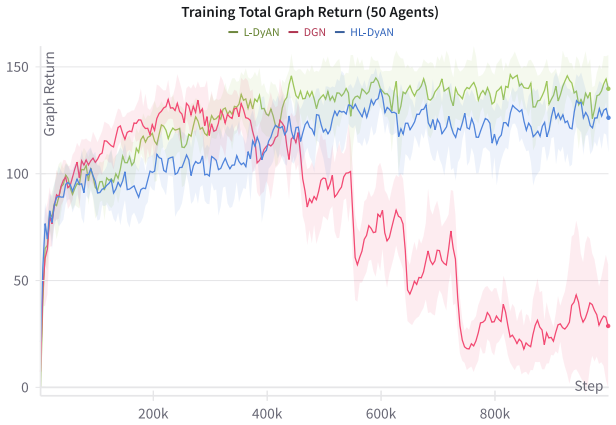

Figure 2: Training performance in terms of average return achieved by the agents.

As the nodes' speed increases, the performance gap between our proposed methods and MPR widens, with L-DyAN and HL-DyAN maintaining superior coverage across all tests with increased speed. This is further supported by additional tests we conducted with the nodes' speed set to 10, obtaining a coverage of 82.35%, resp. 88.83% with 23.88 resp. 36.61 messages, by L-DyAN and HL-DyAN. MPR, instead, struggled to reach 27% of coverage. Additionally, in the more dynamic scenarios with speed set to be greater than 1, the maximum node degree negatively influences the performance of all methods, but L-DyAN and HL-DyAN consistently outperform the other algorithms, which fail to reach 50% coverage. This indicates the robustness of L-DyAN and HL-DyAN to dynamic scenarios with less dense neighborhoods. In more static scenarios where the node speed is set to 1, L-DyAN and MPR reveal to be the more efficient reaching, respectively, 14.84 and 14.54 percent of coverage per message.

Figure 2 illustrates the training progress of L-DyAN, HL-DyAN, and DGN over multiple cycles using five distinct random seeds. Our proposed methodologies, L-DyAN and HL-DyAN, demonstrated an average of the total graph returns (sum of all the agents returns) of 141.47 ± 15.63 and 127.50 ± 13.57, respectively. In contrast, the training trajectory of DGN indicates a more brittle learning progress, unable to learn an effective multi-agent strategy for this task.

The results underscore the efficiency of L-DyAN and HL-DyAN in learning effective multi-agent strategies balancing message efficiency with coverage consistently across various scenarios. MPR falls short in both slightly and very dynamic and/or sparsely connected environments, with its performance worsening as nodes are faster and their starting neighborhood more sparse. The results also highlight the adaptability of L-DyAN and HL-DyAN in varying network densities and node velocities, making them suitable for a wide range of dynamic network environments.

6 Conclusion and Future Work

In this work, we captured the problem of information dissemination in dynamic broadcast networks in a novel POSG formulation and proposed two MARL methods to solve the task, namely L-DyAN and HL-DyAN. Our experiments showed how these methods outperform in terms of coverage and message efficiency both DGN and a popular heuristic employed in real-world networking scenarios.

Our future research agenda includes investigating more structured group communication tasks, where, for example, coverage is desired only for a subset of nodes or nodes with higher priority. We will also study methods to enable more controlled trade-offs between coverage and forwarded messages, as well as their application in deployed protocols for physical computer networks. Orthogonally, we will investigate the application of our approach to the dissemination of information in other domains, such as social networks and computational social choice.

References

Ozan Tonguz, Nawapom Wisitpongphan, Fan Bai, Priyantha Mudalige, and Varsha Sadekar. Broadcasting in vanet. In *2007 Mobile Networking for Vehicular Environments*, pages 7–12, 2007. doi: 10.1109/MOVE.2007.4300825.

Banar Fareed Ibrahim, Mehmet Toycan, and Hiwa Abdulkarim Mawlood. A comprehensive survey on vanet broadcast protocols. In *2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, pages 298–302, 2020. doi: 10.1109/ICCAKM46823.2020.9051462.

Niranjan Suri, Maggie Bredy, Lorenzo Campioni, Jan Nilsson, Eelco Cramer, Roberto Fronteddu, and Mauro Tortonesi. Comparing performance of group communications protocols over scb versus routed manet networks. In *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, pages 1011–1017, 2022. doi: 10.1109/MILCOM55135.2022.10017772.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2145–2153, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Lucian Buşoniu, Robert Babuška, and Bart De Schutter. *Multi-agent Reinforcement Learning: An Overview*, pages 183–221. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-14435-6. doi: 10.1007/978-3-642-14435-6_7. URL https://doi.org/10.1007/978-3-642-14435-6_7.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2252–2260, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

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, 2017.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 7265–7275, Red Hook, NY, USA, 2018. Curran Associates Inc.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1538–1546. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/das19a.html.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkxdQkSYDB.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations (Poster)*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1995–2003. JMLR.org, 2016.

Christopher Dearlove and Thomas H. Clausen. Optimized Link State Routing Protocol Version 2 (OLSRv2) and MANET Neighborhood Discovery Protocol (NHDP) Extension TLVs. RFC 7188, April 2014. URL https://www.rfc-editor.org/info/rfc7188.

Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A. Zighed. Information diffusion in online social networks: A survey. *SIGMOD Rec.*, 42(2):17–28, jul 2013. ISSN 0163-5808. doi: 10.1145/2503792.2503797. URL https://doi.org/10.1145/2503792.2503797.

Zipeng Ye and Qingrui Zhou. Performance evaluation indicators of space dynamic networks under broadcast mechanism. *Space: Science & Technology*, 2021, 2021. doi: 10.34133/2021/9826517. URL https://spj.science.org/doi/abs/10.34133/2021/9826517.

Xiaomin Ma, Jinsong Zhang, Xiaoyan Yin, and Kishor S. Trivedi. Design and analysis of a robust broadcast scheme for vanet safety-related services. *IEEE Transactions on Vehicular Technology*, 61(1):46–61, 2012. doi: 10.1109/TVT.2011.2177675.

A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 3866–3875, 2002. doi: 10.1109/HICSS.2002.994521.

M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN 0-7167-1044-7.

J. Macker. Rfc 6621: Simplified multicast forwarding, 2012.

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.

Niranjan Suri, Lorenzo Campioni, Edoardo Di Caro, Maggie Breedy, Alessandro Morelli, Roberto Fronteddu, and Mauro Tortonesi. Adaptive information dissemination over tactical edge networks. In *2023 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–7, 2023. doi: 10.1109/ICMCIS59922.2023.10253585.

Raffaele Galliera, Mattia Zaccarini, Alessandro Morelli, Roberto Fronteddu, Filippo Poltronieri, Niranjan Suri, and Mauro Tortonesi. Learning to sail dynamic networks: The marlin reinforcement learning framework for congestion control in tactical environments. In *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, pages 424–429, 2023. doi: 10.1109/MILCOM58377.2023.10356270.

Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267):1–6, 2022. URL http://jmlr.org/papers/v23/21-1127.html.

J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.

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

Aric Hagberg, Pieter J. Swart, and Daniel A. Schult. Exploring network structure, dynamics, and function using networkx. 1 2008. URL https://www.osti.gov/biblio/960616.

A Technical Appendix

A.1 Pseudo-code of the MPR Selection algorithm

Algorithm 1 MPR Selection Heuristic

Require: The set N of one-hop neighbors
Ensure: The MPR set

```
Initialize MPR set with all members of N with willingness equal to WILL_ALWAYS
for each node y ∈ N do
    Calculate D(y)
end for
Select nodes in N which cover the poorly covered nodes in N_2. Remove these nodes from N_2.
while nodes exist in N_2 not covered by at least MPR_COVERAGE nodes in the MPR set do
    for each node in N do
        Calculate reachability: number of nodes in N_2 not yet covered by at least MPR_COVERAGE nodes
    in the MPR set and are reachable through this 1-hop neighbor.
    end for
    Select as MPR the node with the highest willingness among nodes in N with non-zero reachability.
    if multiple choices then
        Select node providing maximum reachability to nodes in N_2.
        if multiple nodes provide same reachability then
            Select node as MPR with greater D(y).
        end if
    end if
    Remove nodes from N_2 now covered by MPR_COVERAGE nodes in the MPR set.
end while
```

$D(y)$ is defined as the number of symmetric neighbors of node y, excluding all the members of N and excluding the node performing the computation. A poorly covered node is a node in N_2 which is covered by less than `MPR_COVERAGE` nodes in N. Note that in our implementation every node has willingness set to `WILL_ALWAYS` and `MPR_COVERAGE` is set to 1 to ensure that the MPR heuristic's overhead is kept to the minimum.

A.2 Extended Results

In Table 2 we present an extended version of Table 1, where we include additional networking scenarios with node velocities set to 3 and $10\frac{\text{units}}{\text{step}}$ and 49 initial maximum node degree. We note that all of the additional experiments support the claims made in the paper.

A.3 Ablation Study

We investigate different ablations of L-DyAN, whose architectures lay between L-DyAN and HL-DyAN, and DGN. Their performance is measured in terms of the summation of the returns achieved by each agent that has participated in the dissemination task, named "graph return" (Figure 3). Given that our environment is highly dynamic in terms of the entities contributing to the dissemination task at each timestep, such a metric allows us to understand if the local rewards assigned to each agent correlate with a desired overall collaboration across the entire graph, measured in terms of summations of the rewards achieved. We trained these policies on static scenarios with graphs of 20 nodes.

L-DyAN-Duel. The implementation of this method lies between L-DyAN and DGN. Starting from the latter, we added the dueling network instead of a single MLP stream as the action decoder. Figure 3 shows the positive impact of the dueling network in the final strategy, which significantly outperforms L-DyAN after 600K steps. From such a learning trajectory, we can also deduce the impact of another main component

Initial Node Degree	Nodes Speed	Metric	L-DyAN	HL-DyAN	MPR Heuristic	DGN
5 Neighbors	$1\frac{\text{unit}}{\text{step}}$	Messages	28.22	34.81	25.64	3.88
		Coverage	**93.78%**	90.02%	86.24%	24.34%
5 Neighbors	$3\frac{\text{units}}{\text{step}}$	Messages	26.21	34.55	14.33	5.21
		Coverage	87.78%	**88.65%**	54.53%	30.49%
5 Neighbors	$6\frac{\text{units}}{\text{step}}$	Messages	24.05	33.40	8.21	6.69
		Coverage	79.34%	**83.64%**	34.02%	35.77%
5 Neighbors	$10\frac{\text{units}}{\text{step}}$	Messages	22.10	32.50	4.49	9.34
		Coverage	71.96%	**79.78%**	21.54%	45.32%
10 Neighbors	$1\frac{\text{unit}}{\text{step}}$	Messages	23.26	33.06	22.95	15.91
		Coverage	88.51%	**89.59%**	85.73%	24.35%
10 Neighbors	$3\frac{\text{units}}{\text{step}}$	Messages	24.34	35.97	16.10	4.07
		Coverage	88.73%	**93.30%**	65.4%	29.36%
10 Neighbors	$6\frac{\text{units}}{\text{step}}$	Messages	24.31	37.01	8.10	7.03
		Coverage	86.33%	**91.69%**	37.98%	42.74%
10 Neighbors	$10\frac{\text{units}}{\text{step}}$	Messages	24.66	35.98	5.05	9.72
		Coverage	82.32%	**88.06%**	26.61%	52.15%
25 Neighbors	$1\frac{\text{unit}}{\text{step}}$	Messages	23.74	34.15	24.93	3.28
		Coverage	90.44%	**93.84%**	92.86%	26.33%
25 Neighbors	$3\frac{\text{units}}{\text{step}}$	Messages	24.33	36.05	17.73	4.25
		Coverage	90.81%	**93.49%**	72.92%	31.04%
25 Neighbors	$6\frac{\text{units}}{\text{step}}$	Messages	24.29	36.35	10.03	6.96
		Coverage	88.19%	**92.23%**	46.80%	44.02%
25 Neighbors	$10\frac{\text{units}}{\text{step}}$	Messages	24.46	35.78	5.75	9.50
		Coverage	84.79%	**89.87%**	30.32%	51.89%
49 Neighbors	$1\frac{\text{unit}}{\text{step}}$	Messages	22.99	34.34	23.92	3.47
		Coverage	89.73%	**91.96%**	88.93%	27.16%
49 Neighbors	$3\frac{\text{units}}{\text{step}}$	Messages	23.9	35.8	16.45	4.76
		Coverage	90.05%	**93.02%**	67.91%	33.38%
49 Neighbors	$6\frac{\text{units}}{\text{step}}$	Messages	24.81	36.61	9.39	6.81
		Coverage	88.86%	**92.42%**	43.49%	41.33%
49 Neighbors	$10\frac{\text{units}}{\text{step}}$	Messages	23.88	35.55	4.81	9.81
		Coverage	82.35%	**88.83%**	26.68%	52.69%

Table 2: Evaluation of L-DyAN, HL-DyAN, MPR, and DGN in terms of Coverage and Messages forwarded involving different scenarios.

of our L-DyAN, the n-step return estimation proportional to the local horizon (see Cooperative Dynamic Neighborhoods). With the addition of such n-step returns, we obtain our L-DyAN architecture, and we can notice how such a component helps the learned strategy to converge earlier and less abruptly.

L-DyAN-MP. This method removes the second GAT layer of L-DyAN and replaces it with the global max pool operator (later adopted by HL-DyAN). The concatenation of the output of every encoding stage is still present here. We can notice a slight drop in performance when compared to L-DyAN.

L-DyAN-MPNC. This method removes both the second GAT layer of L-DyAN, as well as the concatenation of the output of every encoding stage. We notice a decrease in performance when compared to L-DyAN. It can also be seen that HL-DyAN can be derived from L-DyAN-MPNC after the ablation of the MLP encoding stage and that HL-DyAN does not suffer from such performance reduction.

In summary, these ablation studies centered around L-DyAN allow us to both understand the strengths of this approach when compared to DGN, as well as motivate the design of the HL-DyAN architecture, which exhibits a simplified structure, less communication overhead, and only slightly underperforms in terms of graph return during training.

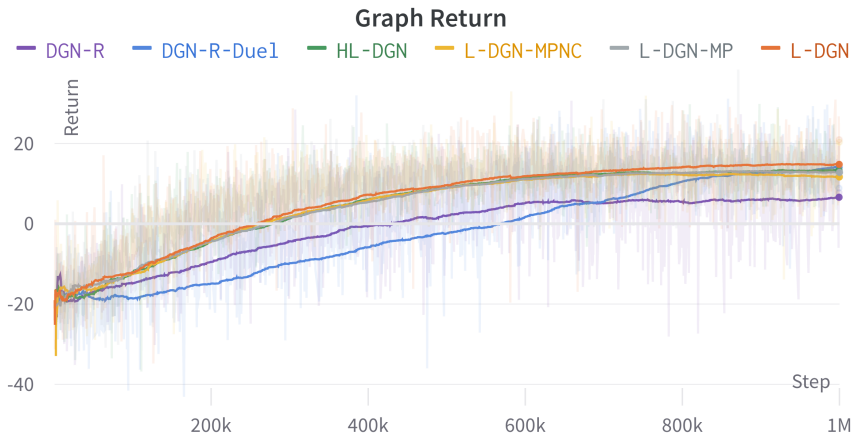

Figure 3: Graph return of the various methods used for the ablation study.

A.4 Additional Reproducibility Details and Instructions

A.4.1 Hyperparameters

	Hyperparameter	Value
Training		
	Training steps	1×10^6
	Learning rate	1×10^{-3}
	Buffer size	1×10^5
	Gamma	0.99
	Batch size	32
	Exploration Decay	Exponential
	Local Horizon	4
	N-Step Estimation	4
	Training Frequency	1 per 160 Agent steps
	Gradient Steps	1
	Parallel Training Envs	40
	Experience Replay	Uniform
	Seed	4, 9, 17, 42, 43
Network Parameters		
	MLP Hidden Size	32
	GAT Attention Heads	4
	GAT Hidden Size	32 (each head)
	A-Network Hidden Sizes	[128, 128]
	V-Network Hidden Sizes	[128, 128]

Table 3: Hyperparameters used across our experiments. “Uniform” indicates that no prioritized replay has been employed.

A.4.2 Implementation Details

Our framework, which is written in Python and based on PyTorch, implements a customized extension of Tianshou (Weng et al., 2022). The MARL environment is defined following the PettingZoo (Terry et al., 2021) API. The GAT, transformer-like dot product attention layer, and global max pooling follow the implementation provided by PyTorch Geometric (Fey and Lenssen, 2019). Training and testing graphs were generated with the aid of the NetworkX library (Hagberg et al., 2008).

A.4.3 Hardware Involved

Our policies were trained using 40 parallel environments on a workstation running Ubuntu 22.04 LTS, CUDA Toolkit v11.7, and equipped with an Intel i9-13900F CPU, 32GB DDR4 RAM, and an NVIDIA GeForce RTX 4090 GPU.