Analyzing RL Components for Wagner's framework via Brouwer's Conjecture

Flora Angileri[1], Giulia Lombardi[2], Andrea Fois, Renato Faraone[3], Carlo Metta[4], Michele Salvi[1], Luigi Amedeo Bianchi[2], Marco Fantozzi[3], Silvia Giulia Galfrè[5], Daniele Pavesi[3], Maurizio Parton[6*], Francesco Morandin[3]

[1]Tor Vergata University of Rome, Italy, Rome.
[2]University of Trento, Italy, Trento.
[3]University of Parma, Italy, Parma.
[4]ISTI-CNR, Italy, Pisa.
[5]University of Pisa, Italy, Pisa.
[6]University of Chieti-Pescara, Italy, Pescara.

*Corresponding author(s). E-mail(s): curiosailab@gmail.com;

Abstract

In this work, we continue the study and systematization started in our previous work ([2]) of Wagner's Reinforcement Learning framework to investigate graph conjectures. After identifying three main directions that impact the framework's performance (the environment dynamics, the RL algorithm and the neural network used as a function approximator), we conduct an ablation study to evaluate the effectiveness of each component, analyzing several variations of them. The experiments compare three environment dynamics, implemented as Gym spaces (*Linear*, *Local* and *Global*), two algorithms (PPO and the Cross-Entropy method [3]), different neural network structures (Multi-Layer Perceptron and Graph Neural Networks) and reward systems. This study was intended not only to test the framework's capabilities, but also to identify a configuration of environment, algorithm, and neural network that can be effective when exploring graph spaces, even with a complex target. For this reason, all the experiments

C. Metta is funded by EU Horizon 2020: G.A. 871042 SoBig-Data++, NextGenEU - PNRR-PEAI (M4C2, investment 1.3) FAIR and "SoBigData.it". G. Lombardi is funded by FSE REACT-EU, PON Ricerca e Innovazione 2014-2020. M. Salvi is supported by the PRIN project GRAFIA, the MURExcellence Department Project MatMod@TOV, awarded to the Department of Mathematics, University of Rome Tor Vergata, CUP E83C18000100006. G. Lombardi, M. Salvi, L. A. Bianchi, M. Parton, and F. Morandin are partially funded by INdAM groups GNAMPA and GNSAGA. Computational resources provided by the CLAI laboratory [1], University of Chieti-Pescara, Italy.

were executed on Brouwer's Conjecture. We also present the data collected with the various trained models, as these interesting configurations can be used in the inference process on the problem. Our analysis shows that a proper calibration of the individual components of the framework can significantly improve its performance, suggesting effective settings for addressing complex problems and contributing to the study of Brouwer's Conjecture. All the codes and data are open source and available at https://github.com/CuriosAI/graph_conjectures.

1 Introduction

The innovative idea presented in [3] by Adam Zsolt Wagner falls within an interesting line of research concerning the use of Reinforcement Learning (RL) techniques to investigate conjectures in combinatorics, with a primary focus on graph theory problems. The main vision of this work is that RL can be used as a tool to perform an 'intelligent' exploration of huge spaces of configurations, in search of counterexamples or interesting instances.

Wagner's approach can be applied to conjectures expressed in the normalized form

$$f(S) < 0 \text{ (or } f(S) \leq 0) \quad \forall\, S \in \mathcal{S},$$

where $\mathcal{S}$ is some configurations' space, and f is a real-valued function on $\mathcal{S}$. The core of this framework consists of formulating a Reinforcement Learning (RL) problem in which the interaction between the agent and the environment is aimed at constructing a particular configuration $S^* \in \mathcal{S}$ such that $f(S^*) > 0$. To achieve this, the reward system defined should be strictly related to the function f under exam, so that the more a configuration is near to disproving the conjecture, the higher the total reward obtained-by the agent-will be. This idea is brilliant not only when counterexamples actually emerge: in fact, as shown in Wagner's work, the data collected during the search process can still be used by mathematicians to gain useful insights, potentially guiding them toward a rigorous proof or a confutation.

In our previous work [2], we proposed a systematization of the framework in [3], starting to analyze which elements are the most relevant for the agent's learning, and how to choose and set them in order to improve the search. Among all the factors involved, we identified three pivotal directions to explore further: the environment's dynamics, the RL algorithm implemented, and the neural network used to approximate policies and value functions. We showed how just changing the algorithm already improved the performances, by exhibiting a new, smaller counterexample to Conjecture 2.1 in [3] that was not found by Wagner's original algorithm. To enhance the research in this direction, we provided a set of tools, available at https://github.com/CuriosAI/graph_conjectures, that can be customized and used to experiment on a wide range of graph conjectures. This set consists of four Gym environments (*Linear*, *Local*, *Global*, *Flip*) that allow for testing different ways of moving through graph spaces, termination conditions, types of actions, and reward systems. Moreover, we remarked on the importance of a proper neural network architecture

when used as a policy approximator. In fact, since most graph statements involve only isomorphism-invariant graph functions, it is evident that the agent's behavior should depend only on the topology of the graph and not be influenced by the labeling of the nodes. This crucial observation moved us towards considering Graph Neural Networks (GNNs) and suggesting them as optimal structures. Moreover, we suggested that pretraining the networks on a supervised learning task related to the problem might be useful to identify the neural network architecture that best extracts relevant features for the task. Once identified, this neural network—whether pretrained or trained from scratch—can then be incorporated into the RL algorithm. We anticipate that this approach will yield a more accurate search process, as the network can more reliably estimate the total return of a given graph. To this end, we provided a dataset of 1198 graphs with 11 nodes, labeled with their Laplacian spectra, that can be used on problems regarding Laplacian eigenvalues, like Brouwer's Conjecture. Also, the dataset can be found at https://github.com/CuriosAI/graph_conjectures.

In this work, we continue the analysis started in [2] by testing several reinforcement learning frameworks on Brouwer's Conjecture, a challenging open problem in graph theory.

Novel Contributions

First, we contribute to the study of Brouwer's Conjecture by providing the most relevant data obtained during the experiments. These data are graphs with a high value of the functions *complete brouwer* and *central brouwer*, as defined in Section 4.1. Observing these configurations can help infer which factors are most relevant, in the topology of a graph, for Brouwer's Conjecture, and so it might guide a confutation or a rigorous proof of the statement. To this end, we varied a great portion of the factors involved in the RL process, and aimed to maintain an appropriate level of exploration during the agent's learning to prevent it from getting stuck in local maxima. This contribution aligns with the perspective of introducing a complementary strategy to the well-established lines of research on Brouwer's Conjecture: leveraging reinforcement learning as a tool to support mathematical discovery. Specifically, we explore the potential of RL algorithms to identify rare instances that may either challenge the conjecture or provide structurally interesting examples to inspire deeper mathematical insights.

We also provide two new versions of *Local* and *Global* environments, which are characterized by the monotonicity of the game dynamics.

Another contribution of this work is a wide analysis of several components of this general RL framework. This comparative study allows both to see the impact of each element on the learning process and to offer a range of possible options, evaluated on a difficult problem such as Brouwer's Conjecture. The main aim of this study is to help identify a framework capable of dealing with complex open problems, as this will constitute a valuable tool for mathematical research. All the comparisons and collected data are presented in Section 5.

In order to assist the agent in understanding the relevant factors, we considered the use of a Graph Neural Network (GNN). After pondering various architectures and pre-selecting the more promising, the final choice landed on the one that outperformed

the task of Laplacian spectrum prediction, on the dataset provided in [2]. As emerged from the experiments, this method attests its effectiveness by regularizing the learning process, as we will see in Section 5.

In Section 2 we survey the related works. In Section 3, we provide an overview of the fundamental reinforcement learning concepts that represent the theoretical foundation of the paper. Section 4 presents the experimental setup in detail, including updated versions of the *Local* and *Global* environments and a detailed description of the Graph Neural Network used. In Section 5, we compare different environments, reward functions, and neural network architectures—offering a comprehensive overview of the main training metrics per epoch, such as the average and maximal reward. Furthermore, the best graphs obtained with the experiments are analyzed in Section 5.3, with visual representations also available in Appendix A.

2 Related Work

2.1 Reinforcement Learning and Graph Conjectures

Wagner's original paper [3] has significantly impacted the field, inspiring numerous follow-up studies that leverage AI techniques for mathematical exploration. This line of work fits into the broader context of machine assistance in mathematics, recently surveyed by Terence Tao in his Presidential Address [4] and related article [5]. In his overview of historical and recent developments, Tao speculates on the future roles of machines and cites Wagner's contribution among examples where computational methods address challenging mathematical problems. Wagner's framework, therefore, serves as an instance within the growing field of machine-assisted proof and discovery.

The most recent paper using Wagner's idea is [6]. Here, Davies et al. construct a novel family of geometric objects relevant to the Hirsch Conjecture on polytope diameters. They introduce a specific class of 5-dimensional prismatoids, referred to as 'drums', and demonstrate that their 'width'—a measure related to the diameter—grows *Linear*ly with the number of vertices. This result provides a new infinite family of counterexamples to the Hirsch Conjecture, exhibiting *Linear* growth in excess width and answering an open question previously posed by Matschke, Santos, and Weibel regarding such behavior in low dimensions.

Another paper leveraging Wagner's framework is [7], where the authors leverage multi-agent RL to generate novel and complex high-dimensional polytopes exhibiting specific extremal properties. Similar to Wagner's original setup, their system is framed as a game where agents collaboratively refine polytope examples. They apply it to challenging geometric problems like the k-neighbourly problem and the search for longest monotone paths, and generate millions of new counterexamples to the 50-year-old Hirsch Conjecture, surpassing previously known human-generated constructions both in quantity and diversity. This highlights the potential of AI as a tool for mathematical discovery in geometry, consistent with Tao's vision.

In [8], Charton, Ellenberg, Wagner, and Williamson introduce PatternBoost, a hybrid methodology combining classical search algorithms with transformer neural networks. The approach conceptually parallels Wagner's original framework, but substitutes reinforcement learning with transformer-based models to generate promising

candidate constructions (the "play" phase), while the policy improvement step of Wagner's cross-entropy method is replaced by classical exhaustive search (the "learning" phase). The method operates iteratively, alternating between a '*Local*' search phase that generates many candidate constructions and a '*Global*' phase where a transformer learns patterns from the best examples found, subsequently seeding the next *Local* search. Successfully applied to various problems in extremal combinatorics, PatternBoost has generated solutions matching or surpassing the best previously known results, including finding a counterexample to a conjecture that stood for 30 years.

In [9] Stevanović, Damnjanović, and Stevanović investigate a conjecture posed by Akbari, Alazemi, and Anđelić concerning the relationship between the *energy* of a graph $E(G)$ and its matching number $\mu(G)$. The conjecture proposed that the inequality $E(G) \leq 2\mu(G)$ holds for connected graphs with maximum vertex degree Δ satisfying $2 \leq \Delta \leq 5$, extending a previously proven result for $\Delta \geq 6$. To identify potential counterexamples, the authors apply Wagner's approach, discovering that likely counterexamples occur at $\Delta = 3$. Subsequently, they perform an exhaustive search among connected graphs with $\Delta \leq 3$ and at most 19 vertices, explicitly enumerating these counterexamples and thus disproving the conjecture. Here, Wagner's approach guides and informs mathematical intuition, in the spirit of [10].

Wagner's method is not always better than classical graph search algorithms. In this direction, Parczyk, Pokutta, Spiegel, and Szabó [11] compared Wagner's method with exhaustive search (when possible) and Tabu Search for finding extremal graphs related to Ramsey multiplicity. Based on their comparison, they reported inferior performance for the RL approach in terms of solution quality and computation time for medium and large-sized problems, highlighting the effectiveness of other heuristics in their study.

Adopting Wagner's approach of framing conjecture refutation as a graph construction game, in [12–14] Roucairol, Cazenave, Vito, and Stefanus explored alternative search algorithms for this task, particularly focusing on Monte Carlo Search (MCS) techniques such as Nested Monte Carlo Search, Nested Rollout Policy Adaptation, and Adaptive Monte Carlo Search. This MCS-driven 'game playing' proved effective and highly efficient, rapidly finding counterexamples for several existing spectral and chemical graph theory conjectures, suggesting that more sophisticated RL methods like AlphaZero, which relies on Monte Carlo search, may improve the original Wagner approach in these domains.

In [15, 16], Ghebleh, Al-Yakoob, Kanso, and Stevanović first reimplemented Wagner's approach, and then successfully applied this implementation to generate counterexamples for conjectures concerning the Laplacian spectral radius of graphs and to determine new or improved lower bounds for several small Ramsey numbers.

In [17], Mehrabian, Wagner et al. searched for graphs of a given size that maximize edge count while avoiding 3-cycles and 4-cycles. They compared Wagner's framework with AlphaZero and Tabu Search; using a curriculum learning strategy (that is, using good graphs found at smaller sizes to initialize the search for larger graphs), both approaches successfully improved several state-of-the-art lower bounds on the maximum number of edges for this problem across various sizes.

In [18], Vott and Lehavi introduce RamseyRL, a framework using reinforcement learning to guide a best-first search for counterexample graphs for classical Ramsey numbers, $R(s,t)$. Their approach uses a deep neural network as a learned heuristic to estimate a graph's potential to become a counterexample, with the work focusing on presenting and evaluating this RL-guided search methodology rather than finding new record-breaking counterexamples.

In [19], Gladkov, Pak, and Zimin provide a refutation of the long-standing Bunkbed Conjecture. Their successful disproof relies on constructing an explicit, structural counterexample graph based on prior work by Hollom. Interestingly, the authors also report on their separate, unsuccessful attempts to find a counterexample via computational search; for these experiments, they started from Wagner's reinforcement learning approach and used their own reimplementation in Julia, highlighting the practical difficulties in finding such counterexamples computationally despite leveraging AI techniques.

2.2 Brouwer's Conjecture

Brouwer's Conjecture was originally formulated in [20], as a variation of the Grone-Merris theorem (proven in [21]), and has since been widely studied, leading to proofs of conjecture 1 for various graph families: trees [22], unicyclic and bicyclic graphs [23], regular graphs, split graphs, and cographs [24], to name a few. Although a general proof has not yet been given, the literature is rich in results on very specific cases, like [25–28]. These results already reduce the search range for counterexamples.

A worth-mentioning paper is [29] by I. Rocha, which gives an asymptotic result about Brouwer's Conjecture. In this work, Rocha proves that the probability for a graph G with n vertices to violate the statement decays exponentially in n. It follows that, from a measure-theoretic standpoint, the set of graphs violating the conjecture forms a negligible subset of the entire graph space and that potential counterexamples, if they exist, become rarer as the size increases. As a result, [29] supports a shift in research focus: instead of continuing to identify new families of graphs for which the conjecture holds, future efforts should aim to detect and analyze the exceptional structures that might lead to its violation. Moreover, the decay rate—illustrated in Appendix C—shows that even small increases in the number of nodes lead to a significant reduction in the number of potential counterexamples. For this reason, our analysis focuses on small graphs, with particular attention to the case of 11 nodes.

3 Notation

Here we recall reinforcement learning's basic concepts and notations. An RL environment is described by a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p)$ consisting of state space, action space, rewards, and transition model $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{R})$, where Δ denotes the space of probability distributions. A transition $p(s', r|s, a)$ is the probability of reaching the next state s' with a reward $r \in \mathcal{R} \subset \mathbb{R}$ when the action a is executed in the state s. An agent interacts with the environment by sampling actions from a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. This agent-environment interaction gives rise to a sequence $\mathcal{T} = (S_0, A_0, R_1, S_1, A_1, R_2, \dots)$, called *trajectory*. Here $S_t, A_t, R_{t+1}, S_{t+1}$

denote the state at time t, the action executed at time t sampled from $\pi(\cdot|S_t)$, the reward received at time $t+1$, and the state reached at time $t+1$, respectively.

An environment dynamics can be classified in two ways: if there are absorbing states reachable from every state under a uniform policy, it is called *episodic* and interactions will eventually terminate; otherwise, it is called *continuing* and the trajectory will never end, unless an additional termination condition is given.

We refer to the situation in which the only non-zero reward occurs at the end of the episode as a *sparse* reward: $R_t = R(S_t, t) = f(S_T)$ if $t = T$, and 0 otherwise.

4 Methods

The first set of experiments observed the effects of changing the environment by testing the behavior of *Linear*-as presented in [2]-and a new implementation of *Local* and *Global*, with the cross-entropy method and an MLP network. We worked with new versions of our *Linear*, *Local*, and *Global* Gym environments. Minor modifications were applied to enable their further use in combination with the GIN architecture and to prevent the agent from visiting the same configuration multiple times. In this section, we briefly recall the logic of each environment and explain these variations. We refer the reader to [2] for details. The second batch of experiments is on the neural network's impact. As highlighted in [2], it is reasonable that the agent's behavior depends only on the graph's topology. MLPs are unsuitable for this purpose, since this network's outputs depend on nodes' labeling.

Starting from this observation, we replaced the MLP with a Graph Isomorphism Network (GIN) [30], leaving the algorithm unchanged. GIN networks are an obvious alternative to test for graph problems, as they are capable of deriving useful graph representations—more informative than the simple adjacency matrix—and encoding the graph structure more effectively. In particular, we employed GINEConv, a variant of GIN introduced in [31] that enables the addition of edge features. The GIN architecture used is described in detail in Section 4.1.1.

Given the remarkable results obtained in [2] with PPO on Conjecture 2.1 in [3], the third batch of experiments is intended to analyze its performance on this complex problem as well, and across all three environments. PPO has been used combined with an MLP, while the integration of the GIN network as a feature extractor poses additional complexity. We are currently working on preliminary experiments.

Before delving into the experimental setup (Section 4.1), we briefly recall the different logics of the environments, present their variations, and report all the proven facts on Brouwer's Conjecture that we considered in designing the experiments.

Linear

In *Linear*, the agent considers the elements of the set $E = \{(i,j)\}_{i,j=1,\dots,n}$ of all possible in a fixed ordering[1]. At each time t the agent can choose between leaving the edge number t as it is (i.e. passing it), or flipping it. The Edge-flipping operation (as defined in [17]) changes the state of an edge like a boolean *not* operator, as follows:

[1] Self-loop edges $\{(i,i)\}_{i=1,\dots,n}$ can be included/excluded in this list by setting an opportune flag to True/False at game's initialization.

let $e \in \{0, 1\}$ be the single bit representing the edge, then

$$\text{flip}(e) = \begin{cases} 1 & \text{if } e = 0 \\ 0 & \text{otherwise} \end{cases}$$

The state is given by the graph and the current time t, and the action space is $\{0, 1\}$, where 0 means that the current edge is left unchanged, and 1 that the edge is flipped. The game naturally ends when the agent has taken its decision for every edge in E, i.e. every episode goes on for $T = \frac{n(n-1)}{2}$ rounds, if self-loops are not allowed, and for $T = \frac{n(n+1)}{2}$ otherwise.

Local

In *Local*, the agent explores the graph space by moving from one node to another. When moving from node i to node j, the agent has the option to flip the edge (i, j) or do nothing. The state is given by the current graph, the current node i where the agent is located, and the current time. In its original version, an action was given by a target node j to move to and a binary value $\{0, 1\}$, with meaning as *Linear*'s actions.

Note that in *Local* the agent can cross a specific edge more times: if this happens, and flipping edges is allowed, as it was in the original version, the edge could be added and removed multiple times (see Figure 1). This unnecessarily complicates the agent's exploration, making it very easy to encounter previously seen configurations.

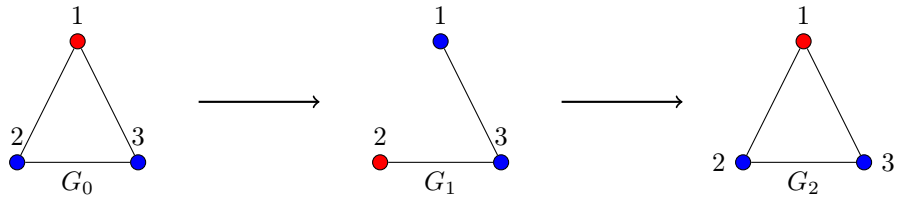

Fig. 1: During round $t = 0$ the agent moves from node 1 to node 2, and in round $t = 1$ it goes back to node 1. The edge $(1, 2)$ is flipped in both cases, so that $G_0 = G_2$.

To avoid this behavior, the new version of *Local* maintains the logic of moving between the nodes, but allows only one type of action: if the game starts from the empty graph, the only available action will be adding edges; if the starting point is the complete graph, the action will be removing edges; in case the starting graph is a particular configuration given, also the action type (adding or removing) must be specified by the user, otherwise the game cannot start. With this logic, some rounds might be "passed". Consider the case of an agent attempting to cross an edge that has already been removed: the agent will try to remove it again, but since the action type is fixed, it cannot be added back. In this case, the agent's action will have no effect on the graph. We refer to this characteristic as a *monotonicity* of the agent's trajectories, since for every t we will have $G_t \subseteq G_{t+1}$, if edges can be added, or $G_{t+1} \subseteq G_t$, if edges can be removed.

Unlike the *Linear* environment, *Local* does not have a natural stopping time, allowing for a continuous version to be implemented. However, we always tested an episodic version of it by setting a fixed episode duration at the game's initialization.

Global

In *Global*, the agent explores the graph space by acting on any edge across the entire graph. The agent can choose any edge, and as in *Local*, it can decide to flip it or do nothing. As highlighted in the *Local* case, this behavior is not optimal; thus, we created a new version of the *Global* environment, with the same action logic described in the *Local* case. *Global* environment has also been used with an episodic setting.

Conjecture 1 (Brouwer)**.** *Let G be a simple*[2] *undirected graph with n vertices and $L(G) = D(G) - A(G)$ its Laplacian matrix, where* D(G) *is the diagonal degree-matrix: $D_{ii} = d(i) \quad \forall i = 1, \dots, n$, and* A(G) *is the adjacency matrix. Let $(\lambda_i(G))_{i=1,\dots,n}$ be the eigenvalues of L with descending ordering. Then*

$$\sum_{i=1}^{t} \lambda_i(G) \leq m(G) + \binom{t+1}{2} \quad \forall\, t = 1, \dots, n \tag{1}$$

The inequality 1 is known to be true for any undirected, simple graph G for any t in $\{1, 2, n-1, n\}$. The cases $t \in \{1, n-1, n\}$ are consequences of basic facts on the Laplacian spectrum [3], while the proof of case $t = 2$ is given in [22]. Moreover, in [20], Brouwer states that the conjecture also holds for graphs with at most 10 vertices.

4.1 Experimental Setup

Considering the state-of-art on Brouwer's Conjecture (Section 2.2), and in particular the asymptotic result [29], all the experiments have been executed on graphs with a low number of nodes, letting n vary in $\{11, \dots, 15\}$. We recall that all the game modes used are designed to explore some subspace (depending on initial settings) of $\mathcal{G}_n = \{G \mid |V(G)| = n,\ G \text{ is undirected}\}$, thus a single experiment was needed for every fixed value of n.

The experiments were executed by fixing an arbitrary stopping time T for both *Local* and *Global* environments, while *Linear* dynamics already consists of a finite number of rounds. Therefore, every experiment gave rise to a finite trajectory $S_0, A_0, R_0, \dots, S_T, A_T, R_T$ where the graph G_t constructed by the agent at time t is included in the state information S_t. All the episodes have been started on the complete graph K_n.

[2] Without self-loops and multiple edges.
[3] Properties of the Laplacian spectrum can be found in [32, 33].

Motivation for the two reward functions.

We designed two distinct reward strategies to incorporate structural insights about the problem: the *complete reward*

$$\max_{t=3,\dots,n-2} \quad \sum_{i=1}^{t} \lambda_i(G) - m(G) - \binom{t+1}{2} \tag{2}$$

and the *central reward*

$$\max_{t=\lfloor \frac{n}{4} \rfloor,\dots,\lfloor \frac{3}{4} n \rfloor} \quad \sum_{i=1}^{t} \lambda_i(G) - m(G) - \binom{t+1}{2}. \tag{3}$$

The *complete reward* is the most natural function to design for Conjecture 1, since it evaluates the agent's performance over the entire range of spectral indices $t \in \{3, \dots, n-2\}$. While this choice of reward provides a broader signal, the learning's effectiveness is weakened by the presence of graphs that are easy to obtain-starting from K_n-and for which the *complete reward* is too high. These elements represent potential local maxima that prevent the agent from exploring more optimal solutions. In our experimental setting, one of this "tricky" elements is the graph $G = K_n \setminus \{e\}$, where e is an arbitrary edge: in fact

$$\sum_{i=1}^{n-2} \lambda_i(G) = m(G) + \binom{n-1}{2}.$$

and so the *complete reward* of G is 0. The experiments confirmed that the agent struggles more to explore when using the *complete reward*, due to this configuration.

To mitigate this issue and guide the agent's learning on the more informative part of the spectrum, we introduced the *central reward*, which restricts the reward computation to the region of the spectral range with indexes $\lfloor \frac{n}{4} \rfloor \leq t \leq \lfloor \frac{3n}{4} \rfloor$. Even if this score does not cover all the cases to be considered for Conjecture 1, it might encourage the agent to explore, in search of higher rewards.

Both reward functions were used as *sparse* rewards, as recalled in Section 3.

4.1.1 GIN network

To select a GNN for Conjecture 1, we tested GCN and GIN on the supervised task of predicting Laplacian eigenvalues $\lambda_1(G), \dots, \lambda_n(G)$. Training was performed on the $n = 11$ Laplacian dataset from [2]. Both networks use convolutional layers to extract graph topology features, and were implemented with PyTorch Geometric [34].

The GCN uses three `GCNConv` layers [35] with channels $(1, 32), (32, 32), (32, 32)$, followed by `Jumping Knowledge(max)` [36] and a two-layer ReLU MLP.

The GIN uses three `GINEConv` layers [31], each with a different 4-layer MLP of 32 units. Edge features are used; aggregation via `Jumping Knowledge(max)`.

Both networks have a single input channel with every value set to the constant 1 on the first layer (because we do not have features on the nodes). The graph structure

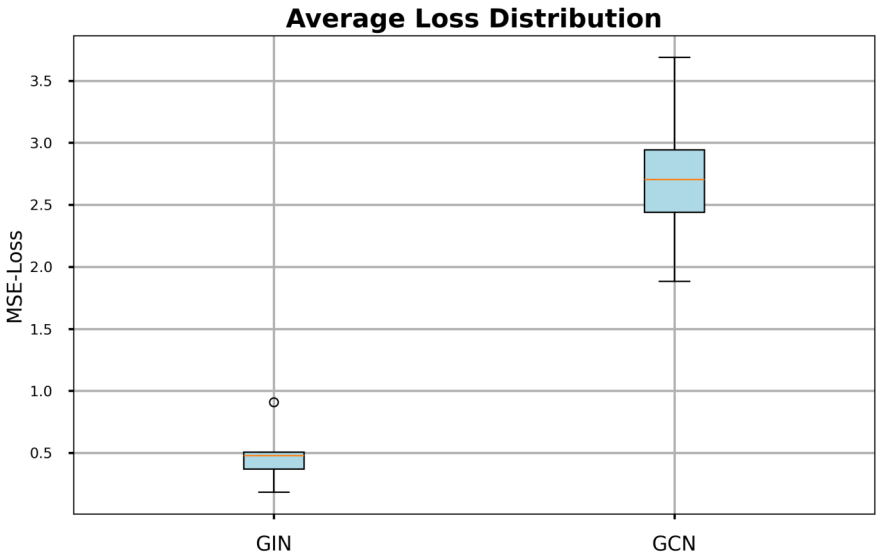

Fig. 2: Boxplots of the average validation loss in 5 training epochs of the GIN and GCN (from left to right) on the Laplacian dataset.

used is the complete graph. The rationale for the choice of the complete graph as the underlying structure is due to the fact that if the current observation is used and the graph is not connected, then the information of the different connected components can not cross the barrier comprised of the missing edges. The actual information about the current state of the graph is passed as edge information to the convolutional layers. This is the necessity which made us choose the GINEConv layer instead of the GINConv layer for the GIN net, and decide to put the information on the structure of the graph into the edge information parameter instead.

The training has been for 5 epochs, using Mean-Square loss, Adam optimizer, and 0.001 learning rate. The dataset has been split into 70% training, 15% validation, and 15% test.

Figure ?? shows the comparison between the average loss obtained on the validation set in every epoch, while the final mean test loss is given in Table 1. The results clearly show how the GINEConv layers are more accurate in capturing information about the graph's structure. This leads to the use of the GIN architecture in our experiments.

	GCN	**GIN**
Mean test loss	0.0561782897%	0.0098509728%

Table 1: Final average test loss of the two networks on approximately 1.797 elements.

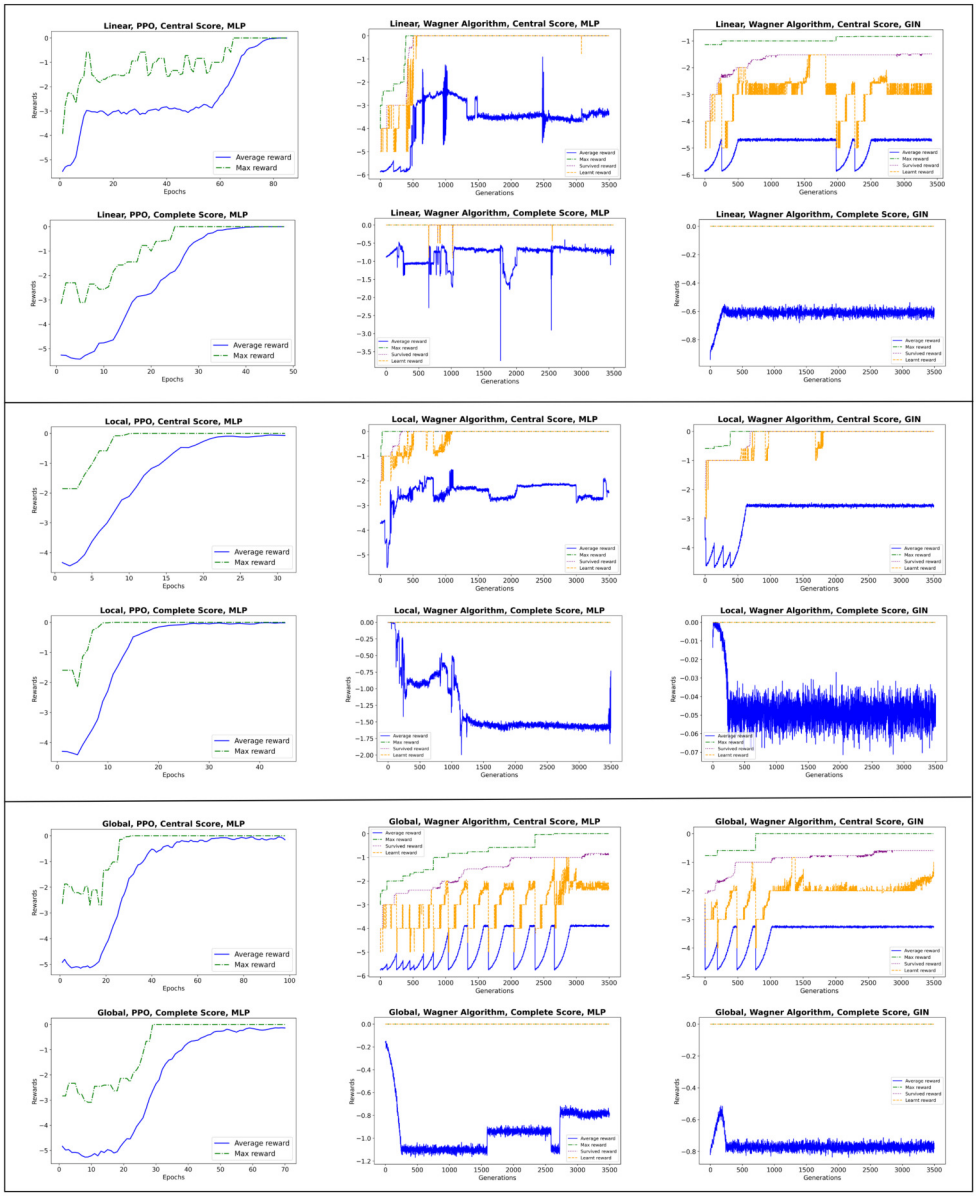

Fig. 3: Comparison of training performance across architectures, reward functions, and environments for graphs with 11 nodes. Each pair of rows represents a different environment (*Linear*, *Local*, *Global*), comparing central (top) and complete (bottom) rewards. Columns show results for PPO with MLP (left), and Wagner with MLP (center) or GIN (right). PPO reports average and max rewards per epoch; Wagner includes additional metrics such as survived and learnt rewards.

Remark 1 Since our work focuses on Brouwer's Conjecture, for which the asymptotic result in [29] holds, we do not investigate scalability, as it is not expected to yield particularly insightful observations regarding Brouwer's Conjecture. Moreover, due to the pre-training on graphs of size $n = 11$, we focus on this dimension. A systematic analysis of the method's scalability in the face of the combinatorial explosion of the graph space will be the subject of future work.

4.1.2 PPO Setup

This subsection describes the experimental setup for training Proximal Policy Optimization (PPO) in our reinforcement learning framework. We tested three custom environments—*Linear*, *Local*, and *Global*—for graph sizes $n = 11$–15, using two sparse reward functions (*complete score* and *central score*), each provided only at episode end, to analyze the effect of reward design on PPO learning.

Network and Hyperparameters. The policy network is a two-layer MLP (128 units each). Fixed hyperparameters for all runs:

- Learning rate: 0.0003
- Discount factor (γ): 0.99
- GAE parameter (λ): 0.95
- Entropy coefficient: 0.01
- Batch size: 2048
- Max steps: 200,000

The discount factor $\gamma = 0.99$ prioritizes long-term rewards, which is important since the final graph structure depends on a sequence of actions. GAE with $\lambda = 0.95$ helps stabilize learning by balancing bias and variance, resulting in smoother updates—especially valuable when rewards are delayed and sparse. The remaining hyperparameters use standard PPO defaults for stable training. All values were kept constant to ensure fair and consistent comparisons across experiments.

Early Stopping. Training stops when the variance of average episode rewards over five rollouts drops below $5 \cdot 10^{-4}$ and the last reward deviates from the rolling mean by less than 10^{-3}.

Remark 2 A key goal for future work is to use PPO with a GIN architecture. However, this integration is not a straightforward network replacement. A robust PPO+GIN agent requires substantial re-engineering of the PPO's data handling pipeline to properly interface with the GIN encoder. This will be the subject of a future paper.

4.1.3 Cross-Entropy setup

The code used for implementing the cross-entropy method is presented from Ghebleh et al. in [15] and available at https://github.com/dragance106/cema-for-graphs. Each experiment has produced 3500 generations of 1000 matches.
Hyperparameters: The cross-entropy method counts several degrees of freedom. The most characteristic are

- `percent learn`: the top percents of graphs used for training the agent's neural network
- `percent survive`: the top percents of graphs that are kept in the next generation
- `act rndness`, `act rndness wait`, `act rndness mult`, `act rndness max`: these parameters regulate the exploration. In each round, `act rndness` actions of the 1000 total are chosen randomly. This value grows by an `act rndness mult` factor if after `act rndness wait` generations the maximum reward did not increase.`act rndness max` is the maximum exploration factor allowed.

`percent learn`, `percent survive` has been left to default, while the randomness parameters have been increased to encourage exploration. However, `act rndness max` was set to 0.05.

This algorithm has been tested with the GIN pre-trained architecture and an MLP, to compare the different structures. The learning rate was set to 0.003.

5 Results

In this section, we present and analyze the results obtained from our framework. The analysis is divided into three parts: the first examines the reward behavior across different settings during training (Section 5.1), the second provides an overview of the performance of the various frameworks (Section 5.2), and the third investigates the structural properties of the best graphs identified during testing (Section 5.3).

5.1 Training Phase

Figure ?? provides a comprehensive comparison of training dynamics for all models applied to graphs of size 11. In these plots, as in the rest of Section 5, the cross-entropy method is referred to as the "Wagner algorithm" to highlight its connection with the framework proposed in [3]. Note that the curves *Learnt Reward* and *Survived Reward* represent the minimum reward observed in the training batch and in the survived batch, respectively. Several key observations can be drawn from this comparison, shedding light on how different model architectures and training schemes influence learning behavior.

High variance in cross-entropy. The cross-entropy method exhibits high variance throughout training. This behavior is primarily due to its inherently exploratory nature, which is preserved across all generations. However, the instability is also exacerbated by the limitations of the MLP network architecture, which struggles to capture and extract graph features effectively. In contrast, GIN-based models show more stable learning curves—particularly in the average reward—even while maintaining strong exploratory behavior.

Impact on learning speed. This high variance has a direct effect on the learning dynamics, as it introduces significant oscillations in the learnt reward curve, ultimately slowing down convergence.

PPO efficiency and stability. Compared to the cross-entropy method, PPO demonstrates both faster convergence and greater stability. Each PPO epoch consists of

2,048 timesteps, which corresponds to approximately 37 matches on $\mathcal{G}_{11}$. All PPO-based models achieved satisfactory performance before epoch 100 (i.e., around 3,700 matches), whereas the cross-entropy method required up to 500,000 matches to stabilize. The average reward curve confirms this enhanced consistency.

Influence of the environment. The structure of the environment has a notable impact on training. In particular, the *Linear* environment is highly sensitive to the order in which edges are visited. As a result, the agent must coordinate its actions precisely to construct a specific graph. In contrast, environments with greater flexibility in edge visitation accelerate the discovery of promising configurations.

Role of the reward function. The reward function proves fundamental in shaping the agent's learning trajectory. The *central reward* leads to a slower and more gradual increase in performance compared to the *complete reward*. Comparing the two may help identify structural factors more relevant to Conjecture 1.

5.2 Testing Phase

PPO with MLP setup. Notably, the highest-scoring configurations are achieved by PPO combined with the MLP architecture, consistently producing graphs with scores close to zero, apart from negligible variations attributable to numerical precision. Among these, the solution found in the Linear-complete reward setting stands out for its sparsity, contrasting with the denser block-structured adjacency matrices observed in the other cases. Analyzing the structure of the adjacency matrices, it can be observed that for both Local and Global, the reward functions drive edge elimination in opposite directions. This leads to larger zero blocks in the north-west corners for Local-central and Global-complete combinations, and in the south-east corners for Local-complete and Global-central combinations.

Wagner with MLP setup. The Wagner-MLP setup tends to produce denser graphs compared to the previous configurations, with the Linear-complete setting reaching a score of zero under the densest condition. Overall, this setup exhibits a less refined exploratory capability in uncovering sparser structures, as reflected by the significantly lower scores observed in the Local and Global environments.

Wagner with GNN setup. The Wagner-GNN setup reveals the most diverse range of configurations with scores approaching zero, with both the Local-central and Global-central settings achieving the best performance with identical scores. Notably, the Linear-complete configuration, although not reaching zero, results in a highly sparse graph structure. Overall, this setup outperforms the Wagner-MLP combination, demonstrating that the inclusion of a GNN significantly enhances the exploratory capability of the Wagner algorithm.

Building upon the encouraging outcomes observed with PPO, we extended our experiments to larger graph sizes, training agents on instances with $n \in \{12, 13, 14, 15\}$. This extension serves to explore how structural patterns evolve across different configurations. The results are given in Appendix B.

5.3 Structural properties

All best-performing graphs obtained after the train phase were subsequently tested to determine whether they belong to graph classes already known to satisfy Brouwer's Conjecture. The results are summarized in Figure 4, which reports the structural properties analyzed—namely, trees, unicyclic and bicyclic graphs, regular graphs, split graphs, and cographs. Interestingly, all graphs achieving higher scores (with the sole exception of the Linear–Complete–Wagner–GNN configuration) fall exclusively within the split and cograph families. In contrast, graphs with lower scores do not belong to any of the known categories for which Brouwer's Conjecture has been proven. This observation raises the question of whether training such models from an initially empty graph could yield different outcomes, particularly since identifying split or cograph structures may pose a greater challenge. Moreover, our experimental findings—at least in the case of 11-node graphs—provide supporting evidence that Brouwer's Conjecture may still hold. We believe this represents a meaningful contribution that could support more in-depth, mathematically grounded investigations.

Visualizations of these graphs can be found in Appendix A.

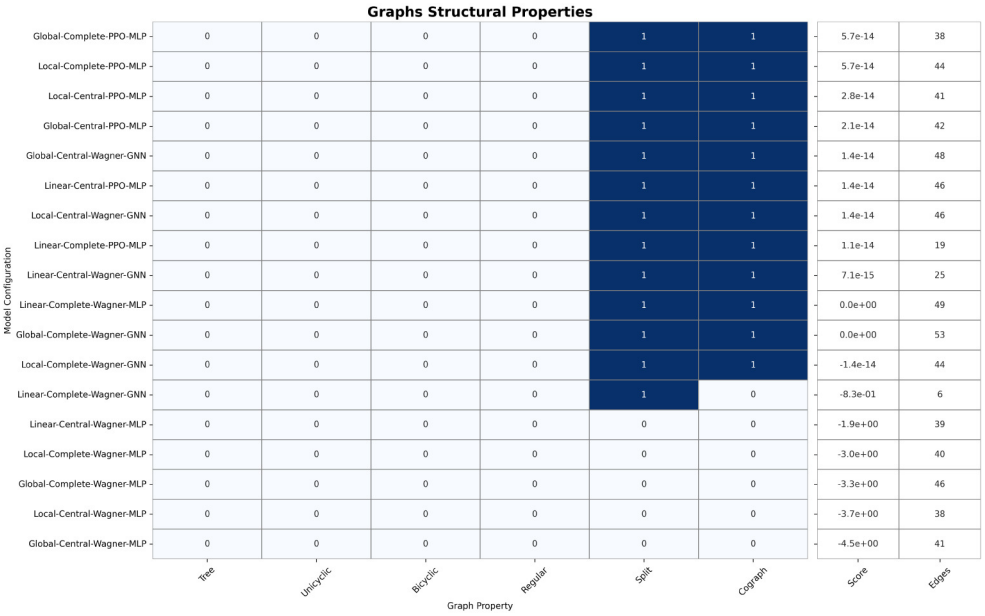

Fig. 4: Plot showing the presence (1) or absence (0) of structural properties across best graph configurations for 11 nodes. Each row corresponds to a specific model configuration (environment, reward function, algorithm, and network type). The properties include being a Tree, Unicyclic, Bicyclic, Regular, Split, or Cograph. To the right of the structural heatmap, the numerical values of the final score and the number of edges are displayed. Configurations are approximately sorted in descending order, as values of order 1e-14 or smaller may result from numerical errors

6 Future Work and Conclusions

A natural next step will be to integrate the GIN architecture [30], which has been proven to enhance stability and learning by capturing graph topological properties when used in conjunction with the Cross-Entropy method [15], into the PPO algorithm [37]. Considering the greater stability and efficiency of PPO compared to the Cross-Entropy method observed in our experiments (albeit with MLP architectures), we expect that the PPO+GIN combination could further improve the framework's exploratory capabilities and performance in identifying relevant graph configurations for the conjecture.

In this paper we have tested two GNN flavours, GCN and GIN. One could investigate other graph-based architectures, like for instance GAT [38] or GraphSAGE [39], augmented with GloNet [40], SwitchPath [41], DAC [42, 43].

Furthermore, the best-performing graphs generated during the experiments constitute a valuable dataset. We intend to subject these results to a more in-depth structural analysis. We will employ various graph metrics - such as the number of connected components, expansion properties, node degree distribution, and various centrality measures - to identify potential recurring topological patterns or salient features. This analysis aims to understand which structural properties of graphs are most correlated with high scores in Brouwer's Conjecture formulation, potentially guiding future theoretical investigations.

References

[1] Amato, G., Amelio, A., Caroprese, L., *et al.*: AI for Sustainability: Research at Ud'A Node. CEUR Workshop Proceedings **3762**, 494–498 (2024)

[2] Angileri, F., Lombardi, G., Fois, A., *et al.*: A Systematization of the Wagner Framework: Graph Theory Conjectures and Reinforcement Learning. Lecture Notes in Computer Science **15243 LNAI**, 325–338 (2025) https://doi.org/10.1007/978-3-031-78977-9_21

[3] Wagner, A.Z.: Constructions in combinatorics via neural networks. https://arxiv.org/abs/2104.14516 (2021)

[4] Tao, T.: Machine-Assisted Proofs. Simons Foundation / American Mathematical Society. AMS Presidential Address (2025). https://www.simonsfoundation.org/event/machine-assisted-proofs/

[5] Tao, T.: Machine-assisted proof. Notices of the American Mathematical Society **72** (2025)

[6] Davies, A., Gupta, P., Racaniere, S., Swirszcz, G., Wagner, A.Z., Weber, T., Williamson, G.: Drums of high width (2025). https://arxiv.org/abs/2503.09919

[7] Swirszcz, G., Wagner, A.Z., et al.: Advancing Geometry with AI: Multi-agent Generation of Polytopes (2025). https://arxiv.org/abs/2502.05199

[8] Charton, F., Ellenberg, J.S., Wagner, A.Z., Williamson, G.: PatternBoost: Constructions in Mathematics with a Little Help from AI (2024). https://arxiv.org/abs/2411.00566

[9] Stevanović, Đ., Damnjanović, I., Stevanović, D.: Finding counterexamples for a conjecture of Akbari, Alazemi and Anđelić (2021). https://arxiv.org/abs/2111.15303

[10] Davies, A., Velickovic, P., *et al.*: Advancing mathematics by guiding human intuition with AI. Nat. **600**(7887), 70–74 (2021) https://doi.org/10.1038/s41586-021-04086-x

[11] Parczyk, O., Pokutta, S., Spiegel, C., Szabó, T.: New Ramsey multiplicity bounds and search heuristics. Foundations of Computational Mathematics (2024)

[12] Vito, V., Stefanus, L.Y.: Adaptive Monte Carlo Search for Conjecture Refutation in Graph Theory (2023). https://arxiv.org/abs/2306.07956

[13] Roucairol, M., Cazenave, T.: Refutation of Spectral Graph Theory Conjectures with Search Algorithms (2024). https://arxiv.org/abs/2409.18626

[14] Roucairol, M., Cazenave, T.: Refutation of Spectral Graph Theory Conjectures with Monte Carlo Search. In: Computing and Combinatorics, pp. 162–176 (2022)

[15] Ghebleh, M., Al-Yakoob, S., Kanso, A., Stevanović, D.: Reinforcement learning for graph theory, I. Reimplementation of Wagner's approach (2024). https://arxiv.org/abs/2403.18429

[16] Ghebleh, M., Al-Yakoob, S., Kanso, A., Stevanović, D.: Reinforcement learning for graph theory, II. Small Ramsey numbers (2024). https://arxiv.org/abs/2403.20055

[17] Mehrabian, A., Anand, A., *et al.*: Finding Increasingly Large Extremal Graphs with AlphaZero and Tabu Search. In: IJCAI-24, pp. 6985–6993 (2024). https://doi.org/10.24963/ijcai.2024/772

[18] Vott, S., Lehavi, A.M.: RamseyRL: A Framework for Intelligent Ramsey Number Counterexample Searching (2023). https://arxiv.org/abs/2308.11943

[19] Gladkov, N., Pak, I., Zimin, A.: The Bunkbed Conjecture is false (2024). https://arxiv.org/abs/2410.02545

[20] Brouwer, A.E., Haemers, W.H.: Spectra of Graphs, (2012). https://doi.org/10.1007/978-1-4614-1939-6

[21] Bai, H.: The Grone-Merris Conjecture. Transactions of the American Mathematical Society **363**(8), 4463–4474 (2011) https://doi.org/10.1090/S0002-9947-2011-05393-6

[22] Haemers, W.H., Mohammadian, A., Tayfeh-Rezaie, B.: On the sum of laplacian eigenvalues of graphs. Linear Algebra and its Applications **432**(9), 2214–2221 (2010) https://doi.org/10.1016/j.laa.2009.03.038

[23] Du, Z., Zhou, B.: Upper bounds for the sum of laplacian eigenvalues of graphs. Linear Algebra and its Applications **436**(9), 3672–3683 (2012) https://doi.org/10.1016/j.laa.2012.01.007

[24] Berndsen, J., Blokhuis, A.: Three problems in algebraic combinatorics. Dissertação de Mestrado, Eindhoven University of Technology (2012)

[25] Ganie, H.A., Pirzada, S., Rather, B.A., Trevisan, V.: Further developments on brouwer's conjecture for the sum of laplacian eigenvalues of graphs. Linear Algebra and its Applications **588**, 1–18 (2020)

[26] Chen, X.: On brouwer's conjecture for the sum of k largest laplacian eigenvalues of graphs. Linear Algebra and its Applications **578**, 402–410 (2019)

[27] Torres, G.S., Trevisan, V.: Brouwer's conjecture for the cartesian product of graphs. Linear Algebra and its Applications **685**, 66–76 (2024)

[28] Chen, X.: On Brouwer's conjecture for the sum of k largest Laplacian eigenvalues of graphs (2019)

[29] Rocha, I.: Brouwer's conjecture holds asymptotically almost surely. Linear Algebra and its Applications **597**, 198–205 (2020) https://doi.org/10.1016/j.laa.2020.03.019

[30] Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2019). https://openreview.net/forum?id=ryGs6iA5Km

[31] Hu, W., Liu, B., *et al.*: Strategies for Pre-training Graph Neural Networks. In: International Conference on Learning Representations (2020)

[32] Godsil, C., Royle, G.F.: Algebraic Graph Theory. Graduate Texts in Mathematics, vol. 207 (2001). https://doi.org/10.1007/978-1-4613-0163-9

[33] Zhang, X.D.: The Laplacian eigenvalues of a graph: a survey (2011)

[34] Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)

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

[36] Xu, K., Li, C., *et al.*: Representation learning on graphs with jumping knowledge networks. In: ICML-18, pp. 5453–5462 (2018)

[37] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR **abs/1707.06347** (2017) arXiv:1707.06347

[38] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)

[39] Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS, pp. 1024–1034 (2017)

[40] Di Cecco, A., Metta, C., Fantozzi, M., *et al.*: GloNets: Globally Connected Neural Networks. In: IDA 2024. Lecture Notes in Computer Science, vol. 14641, pp. 53–64 (2024). https://doi.org/10.1007/978-3-031-58547-0_5

[41] Di Cecco, A., Papini, A., Metta, C., *et al.*: SwitchPath: Enhancing Exploration in Neural Networks Learning Dynamics. Lecture Notes in Computer Science **15243 LNAI**, 275–291 (2025) https://doi.org/10.1007/978-3-031-78977-9_18

[42] Metta, C., Fantozzi, M., Papini, A., *et al.*: Increasing biases can be more efficient than increasing weights. In: IEEE/CVF, Winter Conference on Applications of Computer Vision WACV, pp. 2798–2807 (2024). https://doi.org/10.1109/WACV57701.2024.00279

[43] Metta, C., Fantozzi, M., Papini, A., *et al.*: Increasing biases can be more efficient than increasing weights. Adv Data Anal Classif (2025) https://doi.org/10.1007/s11634-025-00649-2

Appendix A Best 11-Node Graphs

Figure A1 presents the best-performing graphs and their corresponding adjacency matrices obtained by evaluating all trained models—PPO with MLP, Wagner with MLP, and Wagner with GNN—under both reward functions (*central* and *complete*).

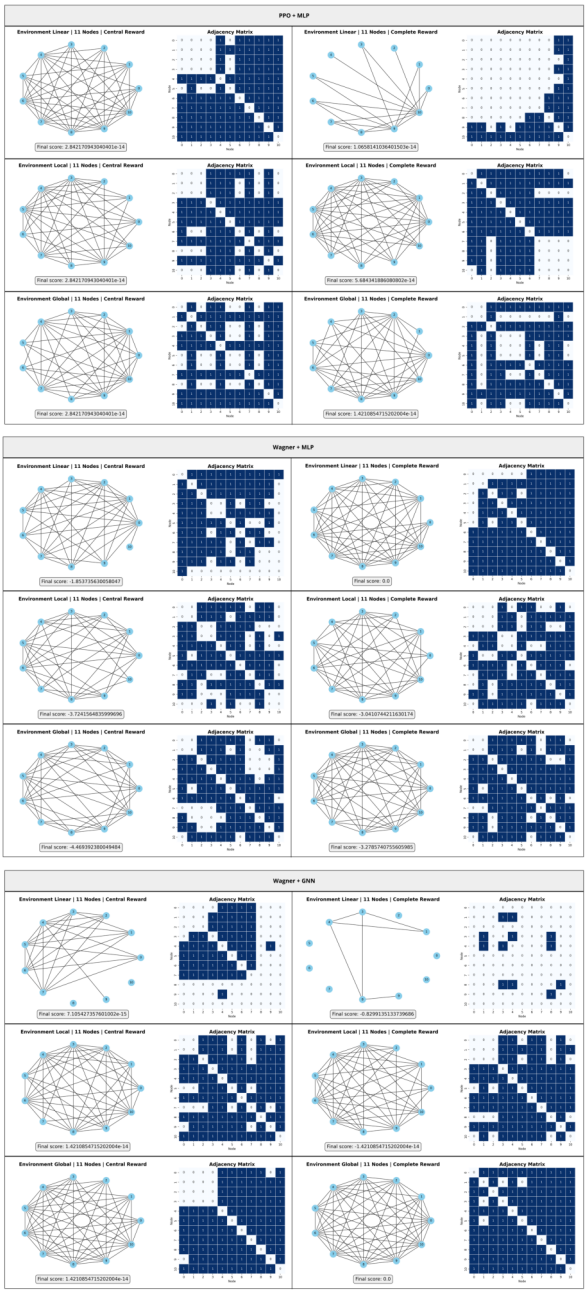

Fig. A1: Comparison of the best graphs generated for 11 nodes under different configurations: the first plot corresponds to PPO with MLP, the second to Wagner with MLP, and the third to Wagner with GNN. Columns represent the two reward functions: central (left) and complete (right).

Appendix B Extensive PPO Results

Due to its promising results, the PPO configuration was examined in greater detail. PPO agents were trained on graphs of various sizes, ranging from 12 to 15 nodes. Figure B2 shows the progression of the average reward per epoch throughout the training process for each graph size.

Furthermore, Figures B3, B4, B5, B6 display the best graphs obtained for each configuration, along with the corresponding score evolution over time that led to their final structures.

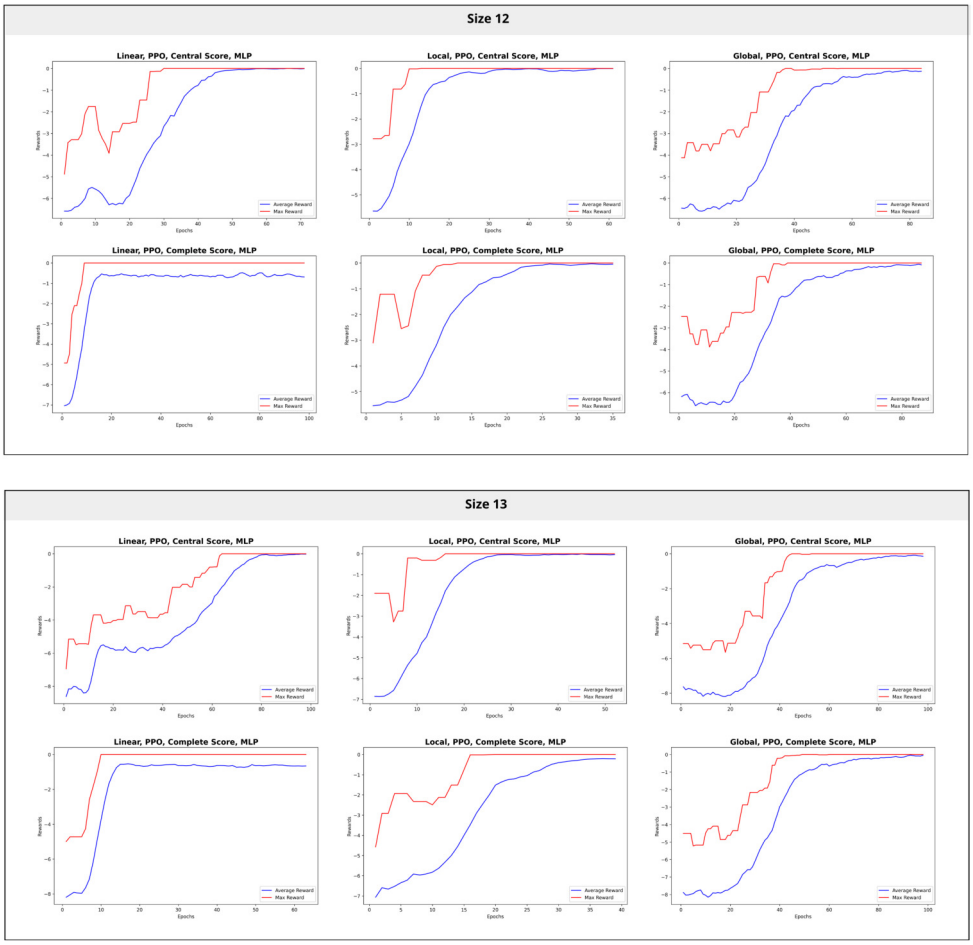

Fig. B2: PPO training performance for graph sizes 12 and 13. Each block shows average and maximum rewards over training epochs. Columns represent the three environments (*Linear*, *Local*, *Global*), and rows compare the two reward functions (central vs. complete).

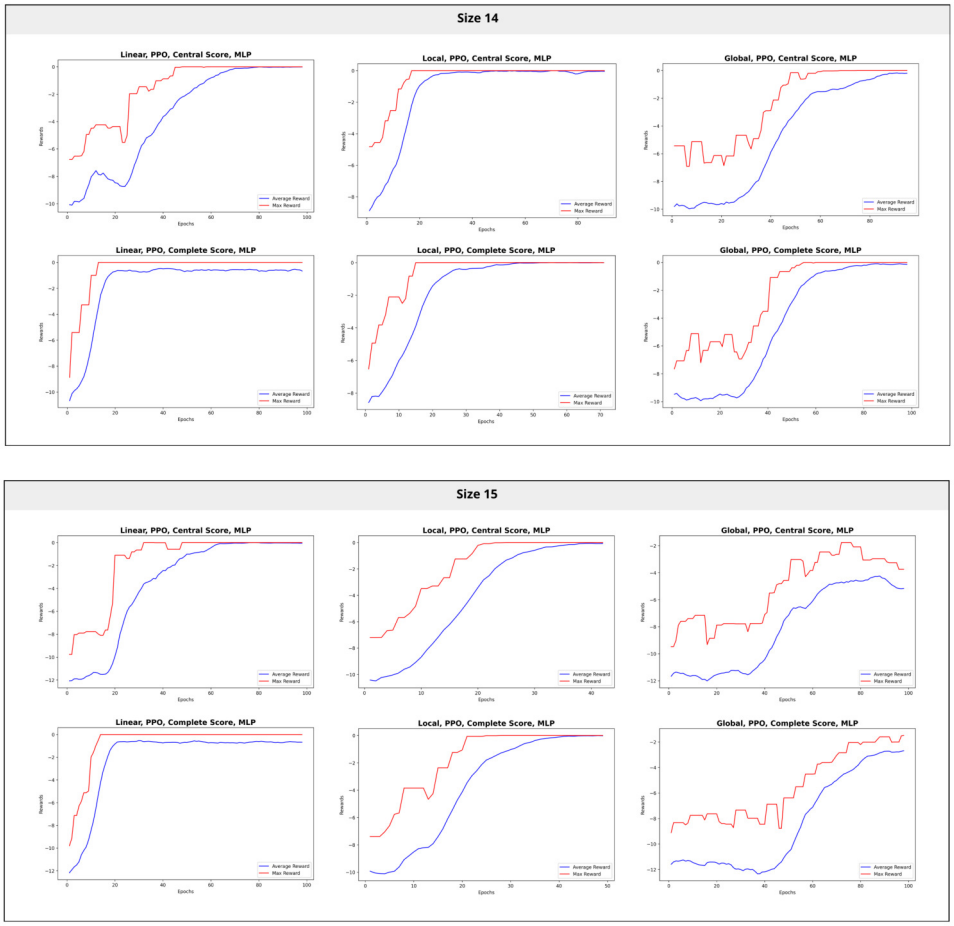

Fig. B2: PPO training performance for graph sizes 14 and 15.

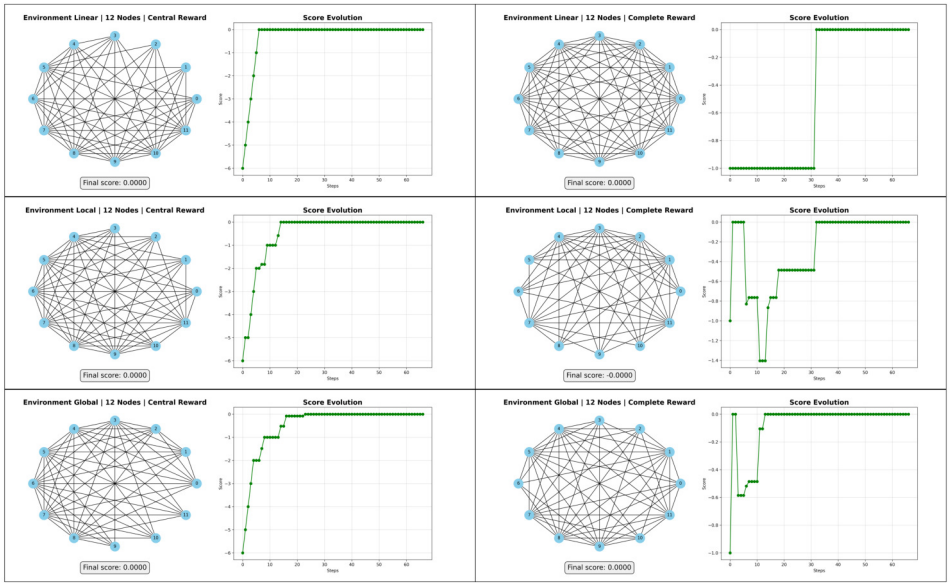

Fig. B3: Best-performing graphs (12 nodes) generated by trained PPO agents with MLP policy, along with score evolution over the episode. Rows correspond to environments (*Linear*, *Local*, *Global*); columns to reward functions (central vs. complete)

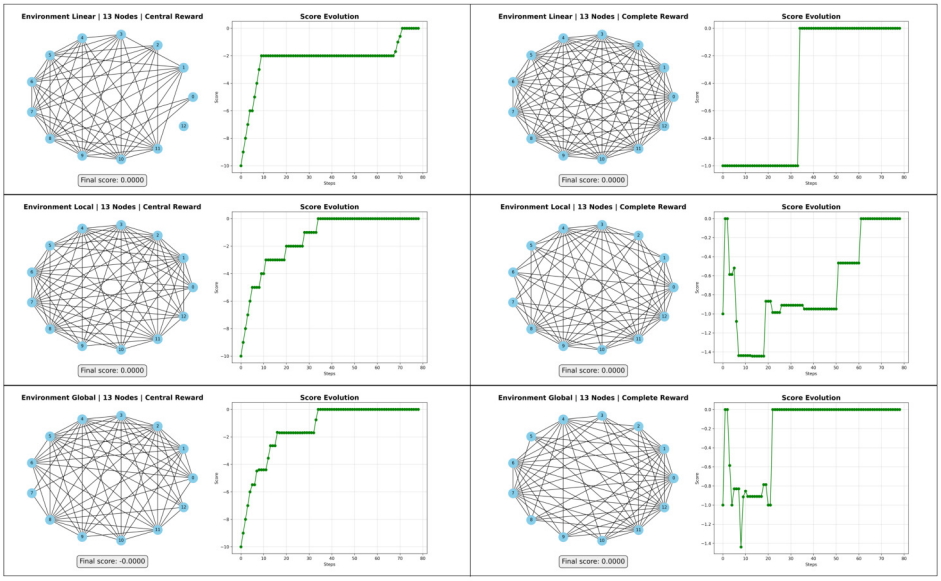

Fig. B4: Best graphs obtained by trained PPO agents for each environment and reward function (13 nodes).

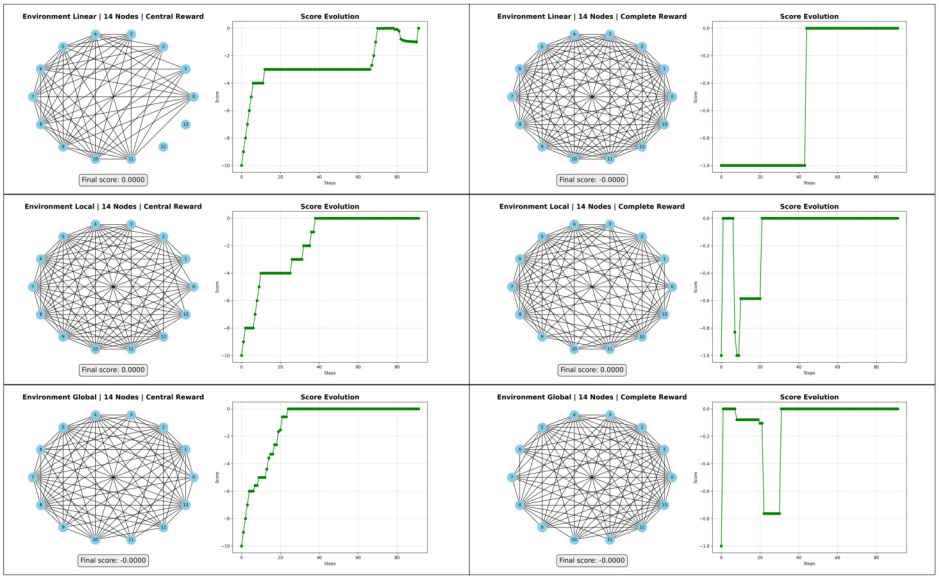

Fig. B5: Best graphs obtained by trained PPO agents for each environment and reward function (14 nodes).

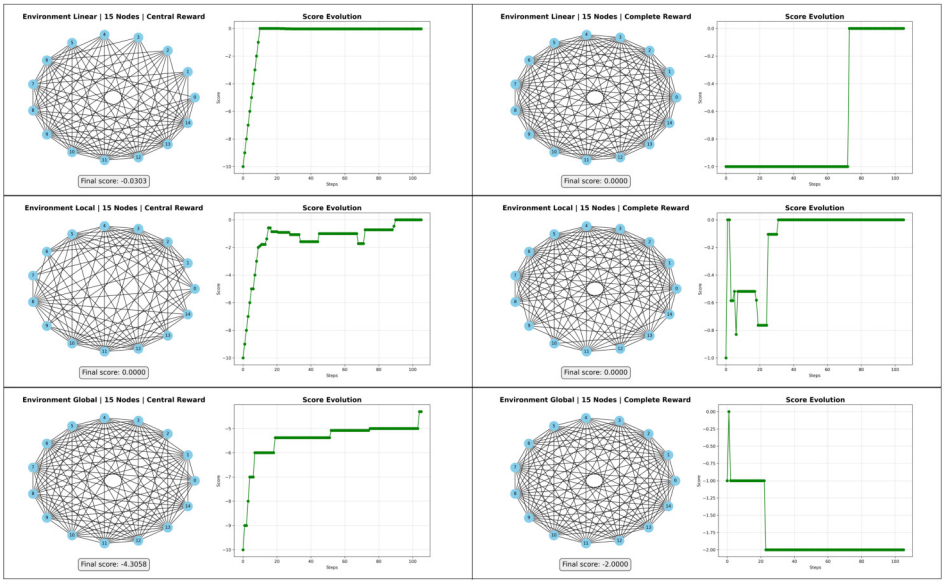

Fig. B6: Best graphs obtained by trained PPO agents for each environment and reward function (15 nodes).

Appendix C Theorems

Definition 1 (Condition 1, [29]). *Let* $\{G_1, G_2, G_3 \dots\}$ *be independent random weighted graphs, where weights* $\xi_{ij}^{(n)}$ *are bounded r.v.s on the same probability space and independent for each* n*, with*

$$\begin{aligned}
&\xi_{ij}^{(n)} = \xi_{ji}^{(n)} \\
&\mathbb{E}\big[\xi_{ij}^{(n)}\big] = \mu_n \\
&Var\big[\xi_{ij}^{(n)}\big] = \sigma_n^2 \\
&\sup_{i,j,n} \mathbb{E}\big[|(\xi_{ij}^{(n)} - \mu_n)/\sigma_n|^p\big] < \infty \quad \textit{for some } p > 6
\end{aligned}$$

Theorem 1 (1 and 2, [29]). *Assume* $\{G_1, G_2, G_3 \dots\}$ *are independent weighted random graphs as in definition 1. If* $\mu_n \in (0, 1-\gamma]$ *for some* $\gamma \in (0,1)$ *and* $\frac{\mu_n}{\sigma_n}\big(\frac{n}{\log n}\big)^{1/2} \longrightarrow \infty$ *as* $n \longrightarrow \infty$*, then for* n *large enough, we have that*

$$\mathbb{P}\Big[\sum_{i=1}^{k} \lambda_i(G_n) \leq m(G_n) + \binom{k+1}{2}\Big] \geq 1 - exp(-C\mu_n n) \tag{C1}$$

for some constant $C > 0$ *that depends only on* γ*. Otherwise, if* $\frac{\mu_n}{\sigma_n}\big(\frac{n}{\log n}\big)^{1/2} \longrightarrow 0$ *and* $\sigma_n^2 \frac{\log n}{\mu_n n} \longrightarrow 0$ *as* $n \longrightarrow \infty$*, then for* n *large enough, we have that*

$$\mathbb{P}\Big[\sum_{i=1}^{k} \lambda_i(G_n) \leq m(G_n) + \binom{k+1}{2}\Big] \geq 1 - exp(-C\mu_n n) \tag{C2}$$

for some constant $C > 0$*.*