NOT CONSTRUCTING RAMSEY GRAPHS USING DEEP REINFORCEMENT LEARNING

David Berghaus[1, 2]
Lamarr Institute[1], Fraunhofer IAIS[2]
david.berghaus@iais.fraunhofer.de

ABSTRACT

We consider the problem of constructing Ramsey graphs using deep reinforcement learning. We introduce a novel permutation invariant architecture that combines ideas from GNNs with self-attention algorithms over the cliques, which shows promising results in a related regression task. To generate graphs, we train our model using established reinforcement learning algorithms such as PPO and A2C. Our results are however very poor compared to traditional local-search algorithms, indicating that this problem is not well-suited for neural networks yet.

1 INTRODUCTION & RELATED WORK

Constructing counter-examples to mathematical conjectures using deep reinforcement learning has been an emerging field. Previous works have mostly focused on combinatorics Wagner (2021); Charton et al. (2024); Romera-Paredes et al. (2024); Mehrabian et al. (2024) and representation and knot theory Davies et al. (2021).

Ramsey graphs are graphs with no clique of size r and no cliques of size s in the complementary graph. The Ramsey number $R(r, s)$ is the smallest number of vertices for which such a graph exists. We refer to appendix A for a brief background on Ramsey graphs.

Discoveries of new Ramsey graphs have so far been based on classical local search methods such as simulated annealing and tabu search Exoo (2012). Deep learning methods have been significantly less successful: Ghose et al. (2022) used a graph neural network to construct Ramsey graphs for $R(4, 4) = 18$. Their approach is however severely limited because they trained their model in a supervised setting on isomorphisms of already known Ramsey graphs and thus cannot find new Ramsey numbers. Parczyk et al. (2024) reported that they used the approach of Wagner (2021) to try to construct Ramsey graphs with no success. Vott & Lehavi (2023) searched for Ramsey graphs with reinforcement learning using a neural network based heuristic - also with no success.

In this work, we aim to improve upon previous approaches by introducing a novel architecture that aims to exploit specific properties of Ramsey graphs.

2 METHODOLOGY

Contrary to previous approaches that generate the graphs auto-regressively Wagner (2021); Charton et al. (2024), we start with a random Erdős–Rényi graph with $p = 0.5$ and want the model to improve it by flipping one entry in the adjacency matrix at a time (that is, creating/removing an edge).

We developed a custom architecture that is specifically designed for Ramsey graphs which is highlighted in fig 1. A detailed description of the architecture can be found in appendix B. Our main idea has been to extend the node embeddings obtained by a GNN Kipf & Welling (2017) by performing self-attention Vaswani et al. (2017) over the cliques of sizes r and s respectively. These can be considered as *defects* that need to be resolved in order to obtain a Ramsey graph. We therefore expect it to be beneficial for the model to have information about these. Our implementation is available online[1].

[1] https://github.com/David-Berghaus/ICLR2025-ramsey

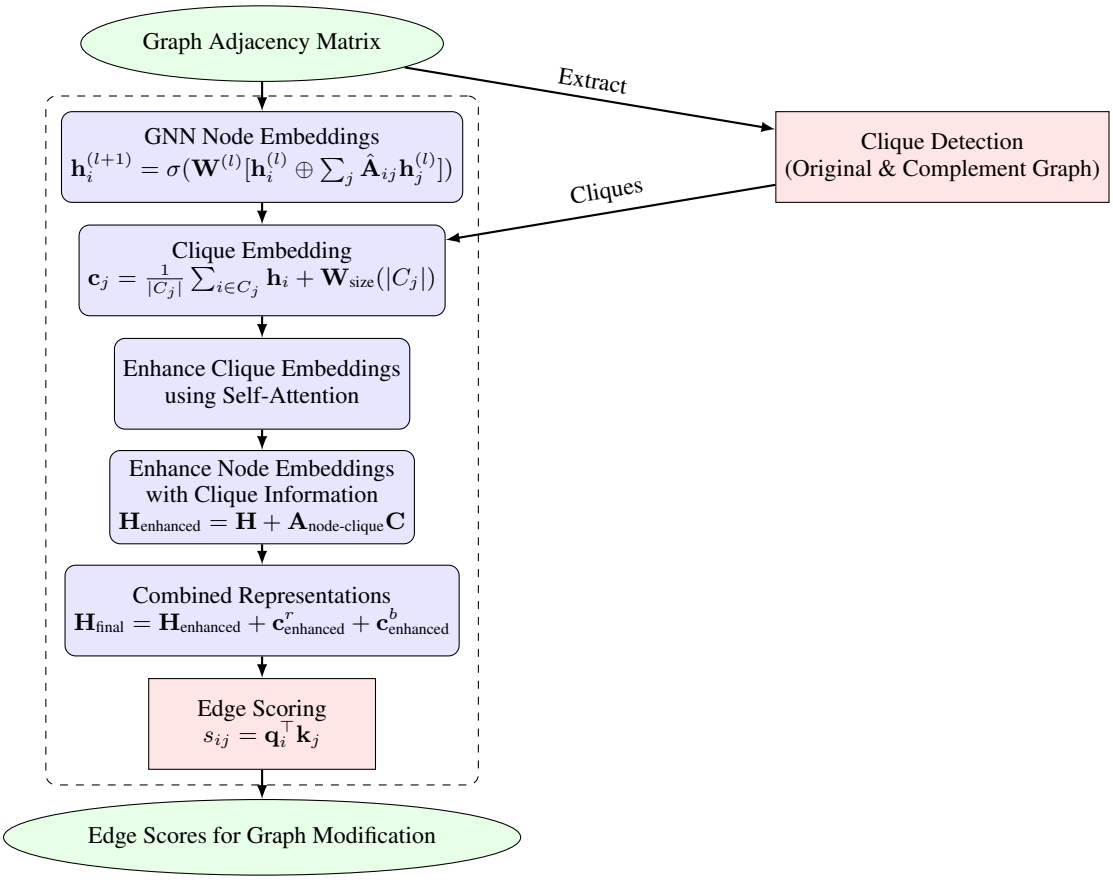


Figure 1: Architecture of the RamseyGNN. The model combines graph convolution for node embedding, explicit clique representation, and attention mechanisms to score potential edge modifications.

2.1 Observation Space

The observation space is a binary vector representing the flattened upper triangular part of the adjacency matrix of a graph with n nodes, excluding the diagonal. This results in a space of size $n(n-1)/2$.

2.2 Action Space

The action space is discrete and corresponds to the indices of the flattened adjacency matrix. Each action involves flipping a bit in the observation space, which effectively adds or removes an edge between a pair of nodes in the graph.

2.3 Num Steps

The maximum number of steps for each episode is set to $n(n-1)/2$, which corresponds to the total number of possible edges in the graph. This limit ensures that the agent has the opportunity to potentially flip each possible edge exactly once per episode.

2.4 Reward

We compute a score function of a given graph as follows

$$\text{score}(\mathcal{G}) = -\left(\text{num\_cliques}(\mathcal{G}, r) + \text{num\_cliques}(\overline{\mathcal{G}}, b)\right) , \tag{1}$$

where num_cliques denotes the number of cliques of specified size. Larger scores are better and a score of 0 corresponds to a Ramsey graph.

The reward is structured as follows:

- **Intermediate episode reward:** During each step of an episode, the reward is calculated based on the difference between the current score and the score from the previous step

$$\text{reward}_i = \text{score}(\mathcal{G}_i) - \text{score}(\mathcal{G}_{i-1})\,. \tag{2}$$

  This difference reflects the improvement (or deterioration) in the graph's structure due to the agent's action, encouraging the agent to make beneficial changes that maximize the score.

- **Disconnected graph penalty:** It is trivial to construct a graph of the best score (i.e., score 0) if it is not connected. For this reason, we punish the model if it creates disconnected graphs by giving it a strong negative reward. Additionally, the episode is truncated, meaning it is forcibly ended due to the undesirable state. This discourages the agent from creating disconnected graphs.

- **Final episode reward:** In the final step of an episode, the reward is not based on the score difference but rather on the absolute score achieved by the graph. This final reward provides a cumulative measure of the agent's performance over the entire episode, acknowledging the state of the graph at the end rather than the incremental change.

This reward structure aims to balance the immediate feedback from incremental improvements with the long-term goal of constructing high-quality Ramsey graphs, while ensuring that graph connectivity is maintained.

2.5 Reinforcement Learning Framework

We implemented our approach in the Stable Baselines 3 framework Raffin et al. (2021). For our reported results we made use of the PPO algorithm Schulman et al. (2017), although we also experimented with other algorithms such as A2C Mnih et al. (2016).

3 Results

We have investigated graphs with $n = 17$ nodes in this work to attempt to find a graph for $R(4, 4)$. Since this Ramsey number is already known, it provides a useful example to experiment with our algorithm. Note that $R(3, 3)$ can be easily obtained via random guessing, so $R(4, 4)$ can be considered as the first non-trivial example, though it is still comparatively small.

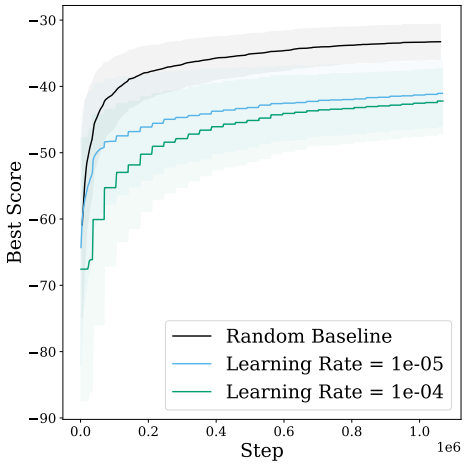
As an ablation study, we demonstrate in appendix C that our architecture is able to use the clique information provided to it to predict the number of cliques of size 4 in the graph and the complementary graph very accurately.

Yet, when it comes to the task of finding Ramsey graphs via reinforcement learning, our algorithms only perform worse than the random baseline, as shown in figure 2.

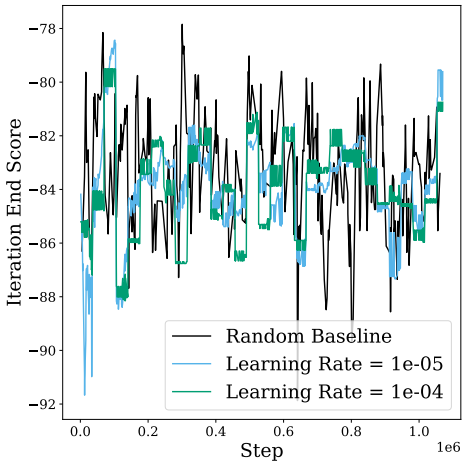
We do not expect this to be a bug in our code since we also tried the independent implementation of Wagner (2021) (which uses the deep-crossentropy-method) and also have been unable to achieve results that are better than random (with the added inconvenience that the model started to predict disconnected graphs, which is a disadvantage of the deep-crossentropy-method). Additionally, we have replaced our model with a simple MLP that takes as input the flattened off-diagonal adjacency matrix and obtained similar (arguably even slightly worse) results.

The construction of Ramsey graphs is a hard problem (for reference, out of the $2^{n(n-1)/2} \approx 9 \cdot 10^{40}$ dimensional solution space for $R(4, 4)$, there exists just one Ramsey graph isomorphism class McKay (2024)). Yet, our surprising result is that neural networks are even unable to learn a heuristic that performs better than random. We do not have a satisfying explanation on why this is the case. Maybe the *dual setup*, that destroying cliques in the graph can introduce new cliques in the complementary graph, causes problems for the neural network.

Potential improvements to our approach include:



(a) Averages of the best encountered scores per environment. We report the means and standard deviations.



(b) Averages of the scores of the graphs at the end of the episodes.

Figure 2: Performance of our models. The results have been averaged over 256 environments and the models have been trained for 48 hours.

- **Starting with a better initial state.** Our current method starts with Erdős–Rényi graphs. It might be beneficial to start with graphs that are *closer* to Ramsey graphs (in the sense that they have better initial scores and require fewer edge-flips to reach Ramsey configurations). The disadvantage of this approach is however that it can introduce an additional bias.
- **Coming up with a better reward function.** The reward function is crucial for the performance of reinforcement learning approaches. Finding a better function might help.
- **Augmenting the heuristic with MCTS.** It would be interesting to augment a neural network based heuristic with a Monte-Carlo tree search, as it has been done in AlphaZero Silver et al. (2017). Promising results in this direction have recently been obtained in Mehrabian et al. (2024).

4 Conclusion

In this work we have shown that neural network based methods perform very poorly on the construction of Ramsey graphs, not even beating random baselines. Further work is needed until they can become competitive to classical local search algorithms Exoo (2012) and can discover new unknown Ramsey graphs.

Acknowledgements

The author would like to thank Danylo Radchenko for useful discussions, Kostadin Cvejoski for proof-reading, and the anonymous referees for valuable suggestions. This research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence.

REFERENCES

Vigleik Angeltveit and Brendan D. McKay. $r(5,5) \leq 46$, 2024. URL https://arxiv.org/abs/2409.15709.

David Berghaus, Kostadin Cvejoski, Patrick Seifner, Cesar Ojeda, and Ramses J Sanchez. Foundation inference models for markov jump processes. In *Advances in Neural Information Processing Systems*, volume 38, pp. 129407–129442, 6 2024.

François Charton, Jordan S. Ellenberg, Adam Zsolt Wagner, and Geordie Williamson. Patternboost: Constructions in mathematics with a little help from ai, 2024. URL https://arxiv.org/abs/2411.00566.

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis, and Pushmeet Kohli. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, dec 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-04086-x. URL https://doi.org/10.1038/s41586-021-04086-x.

Geoffrey Exoo. A lower bound for r(5, 5). *Journal of Graph Theory*, 13(1):97–98, March 1989. doi: 10.1002/jgt.3190130113.

Geoffrey Exoo. On the ramsey number r(4,6). *The Electronic Journal of Combinatorics*, 19(1), 2012. doi: 10.37236/2102.

Amur Ghose, Amit Levi, and Yingxueff Zhang. Graph neural networks for ramsey graphs. In *NeurIPS 2022: MATH-AI: Toward Human-Level Mathematical Reasoning*, 2022. URL https://neurips.cc/virtual/2022/58289.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Brendan McKay. Ramsey graphs, 2024. URL https://users.cecs.anu.edu.au/~bdm/data/ramsey.html.

Abbas Mehrabian, Ankit Anand, Hyunjik Kim, Nicolas Sonnerat, Matej Balog, Gheorghe Comanici, Tudor Berariu, Andrew Lee, Anian Ruoss, Anna Bulanova, Daniel Toyama, Sam Blackwell, Bernardino Romera Paredes, Petar Veličković, Laurent Orseau, Joonkyung Lee, Anurag Murty Naredla, Doina Precup, and Adam Zsolt Wagner. Finding increasingly large extremal graphs with alphazero and tabu search. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 6985–6993. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/772. URL https://doi.org/10.24963/ijcai.2024/772. Main Track.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1928–1937. JMLR.org, 2016.

Olaf Parczyk, Sebastian Pokutta, Christoph Spiegel, and Tibor Szabó. New ramsey multiplicity bounds and search heuristics. *Foundations of Computational Mathematics*, August 2024. ISSN 1615-3383. doi: 10.1007/s10208-024-09675-6. URL https://doi.org/10.1007/s10208-024-09675-6.

Stanisław Radziszowski. Small ramsey numbers. *The Electronic Journal of Combinatorics*, 2021. doi: https://doi.org/10.37236/21.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22:1–8, 2021. Submitted 12/20; Revised 10/21; Published 11/21.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, jan 2024. ISSN 1476-4687. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Vera Rosta. Ramsey theory applications. *The Electronic Journal of Combinatorics*, DS13:43 p., electronic only–43 p., electronic only, 2004. URL http://eudml.org/doc/125408.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, October 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL https://doi.org/10.1038/nature24270.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Steve Vott and Adam M. Lehavi. Ramseyrl: A framework for intelligent ramsey number counterexample searching, 2023. URL https://arxiv.org/abs/2308.11943.

Adam Zsolt Wagner. Constructions in combinatorics via neural networks, 2021. URL https://arxiv.org/abs/2104.14516.

A Background on Ramsey Graphs

Ramsey graphs are a fundamental concept in the field of combinatorial mathematics and graph theory with applications in various branches of mathematics (such as number theory, harmonic analysis, convex geometry) and theoretical computer science (such as information theory and complexity theory). We refer to Rosta (2004) for an extensive overview of applications of Ramsey theory.

A Ramsey graph is often explained through the *party problem*. Imagine a party where you want to determine the minimum number of guests needed to ensure that there are either r mutual friends or s mutual strangers. This minimum number is known as the Ramsey number $R(r, s)$ and a graph for which this property does not hold is called a Ramsey graph.

Described more formally, a Ramsey graph $\mathcal{G}$ is a graph that does not exhibit the Ramsey property, meaning that for any given integers r and s, the graph will not contain a clique of size r nor a clique of size s in the complementary graph. (Note that a clique is a subset of vertices of $\mathcal{G}$ for which all vertices are mutually connected.) The Ramsey number $R(r, s)$ is the smallest number of vertices n needed in a complete graph such that, no matter how its edges are colored using two colors (commonly red and blue), it will always contain a monochromatic clique of size r in one color or a monochromatic anti-clique of size s in the other color.

The challenge is now to construct a graph in two colors such that no red cliques of size r and no blue cliques of size s exist to obtain a lower bound on $R(r, s)$. An example for $R(3, 3)$ is shown in figure 3.

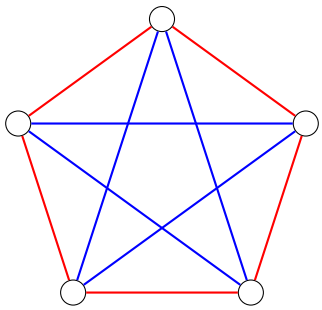


Figure 3: This Ramsey graph with 5 nodes contains no blue clique of size 3 and no red clique of size 3. It can be shown that no such configuration exists for graphs with 6 nodes, resulting in $R(3, 3) = 6$.

The list of all known Ramsey numbers (or the known bounds) is being maintained and continuously updated in Radziszowski (2021). McKay (2024) hosts a list of some already known Ramsey graphs. Arguably the most famous unknown Ramsey number is $R(5, 5)$ which is known to be between 43 Exoo (1989) and 46 Angeltveit & McKay (2024).

B Architecture

In this section, we present our neural network architecture for constructing Ramsey graphs through reinforcement learning. The architecture employs Graph Neural Networks (GNNs), transformer-inspired attention mechanisms, and specialized components for modeling both local and global graph structures. An overview of the architecture can be found in fig 1.

B.1 Node Feature Extraction

The foundation of our model is a graph neural network that extracts meaningful node embeddings from the adjacency matrix using normalized graph convolution Kipf & Welling (2017):

$$\hat{\mathbf{A}} = \mathbf{D}^{-1/2}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-1/2}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix with added self-loops, and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is the degree matrix. Node features are propagated through multiple GNN layers:

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{W}^{(l)}[\mathbf{H}^{(l)} \oplus \hat{\mathbf{A}}\mathbf{H}^{(l)}])$$

This normalized graph convolution enables the model to effectively capture the local structural properties of the graph. By incorporating information from neighboring nodes, the GNN creates representations that reflect each node's position and connectivity patterns within the graph, which is crucial for identifying potential Ramsey substructures.

B.2 Clique Representation

Our architecture explicitly models important substructures by identifying r-cliques in the original graph and s-cliques (independent sets) in the complement graph. For each clique C_j, we compute a representation by pooling node embeddings and augmenting with size information:

$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{H}_i$$

$$\mathbf{c}'_j = \mathbf{c}_j + \mathbf{W}_{\text{size}}(|C_j|)$$

The explicit representation of cliques allows the model to directly reason about the forbidden substructures that define Ramsey graphs. The size embeddings are crucial because they preserve information about clique cardinality that would otherwise be lost during mean pooling, enabling the model to differentiate between cliques of different sizes (a similar trick as been employed in the path attention in Berghaus et al. (2024)).

B.3 Node-Clique Cross-Attention

A key innovation in our architecture is the node-clique cross-attention mechanism, which facilitates bidirectional information flow between nodes and the cliques they participate in. Given a set of node embeddings $\mathbf{H} \in \mathbb{R}^{n \times d}$ and clique sets $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$, the cross-attention operates as follows:

First, we construct a binary membership matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ where:

$$\mathbf{M}_{ij} = \begin{cases} 1 & \text{if node } i \text{ belongs to clique } j \\ 0 & \text{otherwise} \end{cases}$$

We then apply a softmax operation along the clique dimension to obtain attention weights:

$$\mathbf{A}_{\text{node-clique}} = \text{softmax}(\mathbf{M})$$

For each clique C_j, we create a representation by mean-pooling the embeddings of its member nodes:

$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \mathbf{h}_i$$

These representations are collected into a clique embedding matrix $\mathbf{C} \in \mathbb{R}^{m \times d}$. Finally, we enhance each node's embedding by adding a weighted sum of clique embeddings:

$$\mathbf{H}_{\text{enhanced}} = \mathbf{H} + \mathbf{A}_{\text{node-clique}}\mathbf{C}$$

This mechanism enables nodes to incorporate information about the larger structural patterns they belong to, which is crucial for understanding the graph's structure in relation to the Ramsey problem constraints.

B.4 CLIQUE ATTENTION MECHANISM

For processing collections of cliques, we employ a multi-head attention mechanism similar to the transformer encoder Vaswani et al. (2017). This mechanism is applied separately to red cliques (in the original graph) and blue cliques (in the complement graph).

Given a set of clique embeddings $\mathbf{C} \in \mathbb{R}^{m \times d}$ where m is the number of cliques and d is the embedding dimension, we compute queries, keys, and values:

$$\mathbf{Q} = \mathbf{W}_q \mathbf{C}, \quad \mathbf{K} = \mathbf{W}_k \mathbf{C}, \quad \mathbf{V} = \mathbf{W}_v \mathbf{C}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

The attention output is processed through layer normalization and feed-forward networks following the transformer architecture:

$$\mathbf{C}' = \text{LayerNorm}(\mathbf{C} + \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}))$$

$$\mathbf{C}'' = \text{LayerNorm}(\mathbf{C}' + \text{FFN}(\mathbf{C}'))$$

of which we compute the means to obtain $\mathbf{c}_{\text{enhanced}} \in \mathbb{R}^d$.

While the Node-Clique Cross-Attention focuses on enhancing node representations with clique information, this Clique Attention Mechanism serves a complementary purpose: it allows cliques to interact with each other and produces a fixed-dimension graph-level representation. This interaction enables the model to understand relationships between different substructures, such as overlapping cliques or nearly-complete cliques.

B.5 COMBINED REPRESENTATIONS

The enhanced node embeddings and processed clique embeddings are combined to form a comprehensive representation:

$$\mathbf{H}_{\text{final}} = \mathbf{H}_{\text{enhanced}} + \mathbf{c}^r_{\text{enhanced}} + \mathbf{c}^s_{\text{enhanced}}$$

This integration step merges local structural information (from node embeddings) with global pattern recognition (from clique attention), creating representations that capture both perspectives. The additive combination allows the model to preserve both types of information, rather than forcing them to compete through concatenation or other mechanisms.

B.6 EDGE SCORING FOR REINFORCEMENT LEARNING

The architecture culminates in an edge scoring mechanism that evaluates potential graph modifications:

$$\mathbf{q}_i = \mathbf{W}_{\text{query}} \mathbf{H}_{\text{final},i}, \quad \mathbf{k}_j = \mathbf{W}_{\text{key}} \mathbf{H}_{\text{final},j}$$
$$s_{ij} = \mathbf{q}_i^\top \mathbf{k}_j$$

This attention-based scoring approach allows the model to evaluate each potential edge modification based on the learned representations of its endpoint nodes. By projecting node representations into query and key spaces, the model can assess the compatibility of nodes for forming or removing connections. This mechanism effectively translates the model's understanding of graph structure into actionable decisions for reinforcement learning, while maintaining invariance under the labeling of nodes and cliques.

C Ablation Study: Predicting the Clique Counts

As an ablation study, we test our model on a related task, that is, estimating the number of cliques of size r (resp. s) in the graph (resp. complementary graph).

Following the outputs of section B.3 and B.4 we construct embeddings for the clique count regression task as follows:

$$\mathbf{h}_{\text{graph,r}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_{\text{enhanced},i} + \mathbf{c}^{r}_{\text{enhanced}}$$

$$\mathbf{h}_{\text{graph,s}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_{\text{enhanced},i} + \mathbf{c}^{s}_{\text{enhanced}}$$

C.1 Regression Heads

The final stage of the architecture consists of two separate regression heads that predict the number of r-cliques in the original graph and s-cliques in the complement graph:

$$\hat{y}_r = \text{MLP}_r(\mathbf{h}_{\text{graph,r}})$$

$$\hat{y}_s = \text{MLP}_s(\mathbf{h}_{\text{graph,s}})$$

Each MLP consists of a sequence of fully-connected layers with ReLU activations.

C.2 Training and Evaluation

For the regression task, the model is trained to minimize the mean squared error (MSE) between predicted and actual clique counts:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left((y_{r,i} - \hat{y}_{r,i})^2 + (y_{s,i} - \hat{y}_{s,i})^2\right)$$

where $y_{r,i}$ and $y_{s,i}$ are the ground truth counts of r-cliques and s-cliques respectively for the i-th graph in a batch of N graphs.

C.3 Results

We sampled 50000 Erdős–Rényi graphs and performed the clique count prediction using three baselines:

1. A simple MLP model that takes the adjacency matrix as input
2. A GNN where we took the mean node embedding as a graph embedding
3. Our custom RamseyGNN

All models were trained with a learning rate of 10^{-4}. Our results are shown in table 1 and figs 4, 5, 6. The results show that the RamseyGNN architecture is able to make use of the additional information about the cliques provided to it.

	MLP	GNN	RamseyGNN
MAE	12.07	22.04	**0.22**

Table 1: Mean absolute error of the regression models.

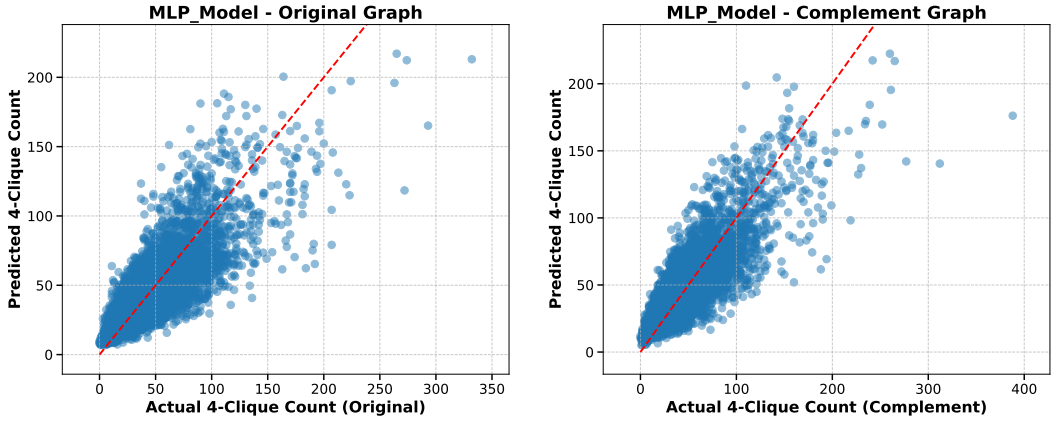


Figure 4: Predicted clique counts for $R(4,4)$ using a simple MLP that takes the adjacency matrix as input.

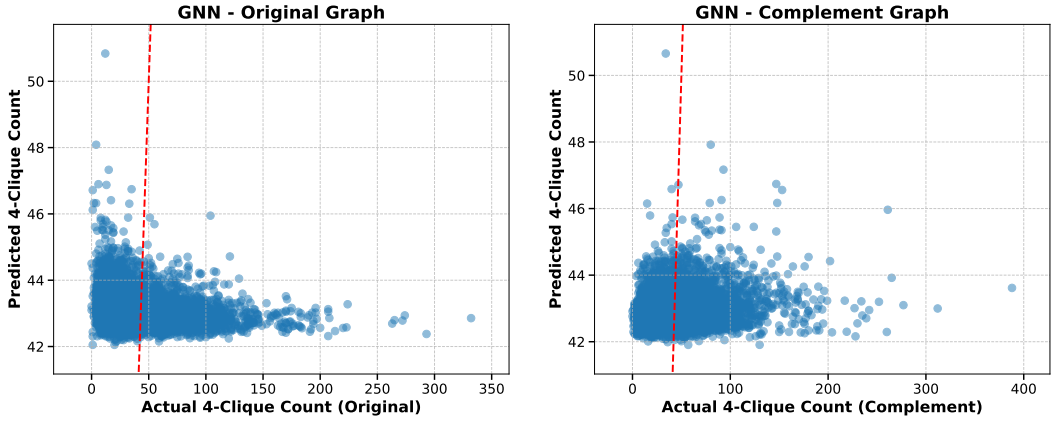


Figure 5: Predicted clique counts for $R(4,4)$ using the mean node embedding of the GNN. This model did not perform well and only returned values close to the mean.

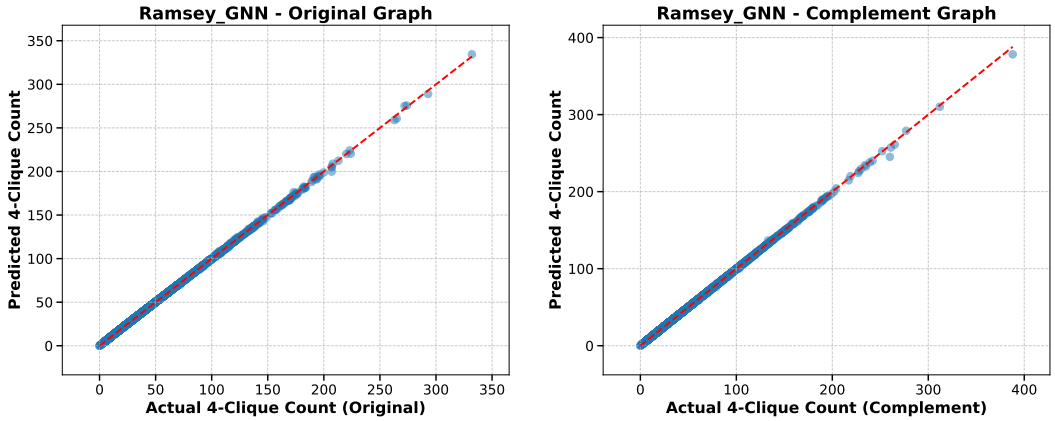


Figure 6: Predicted clique counts for $R(4,4)$ using our RamseyGNN architecture.