Graph neural networks for Ramsey graphs

Amur Ghose
Huawei
a3ghose@uwaterloo.ca

Amit Levi
Huawei
amit.levi@huawei.com

Yingxue Zhang
Huawei
yingxue.zhang@huawei.com

Abstract

Ramsey-like problems are ubiquitous in extremal combinatorics and occupy a central place in the field. In simple terms, Ramsey theory wishes to find the minimum size of a large graph structure such that some sought substructure - generally a clique or an independent set - is guaranteed to exist. Due to considerations of computational complexity, brute force approaches to solving these problems are usually not very feasible, as the substructures cannot be checked in polynomial time. At the same time, we seek extremal graphs that completely avoid such substructures to better understand the graph theory governing their occurrence. We investigate the feasibility of Graph Neural Networks (GNNs) in terms of indicating and refining search procedures for finding these special classes of Ramsey-extremal graphs, which are of interest to mathematicians.

1 Introduction

Neural networks often arrive at superhuman performance in domains governed entirely by logic such as Chess and Go [1, 2]. Simultaneously, they are also often able to achieve metrics that exceed the average human at tasks such as image recognition. Despite this, they are not always able to explain their route to such abilities. This dilemma makes neural methods somewhat unsuitable for mathematics, which requires writing rigorous proofs that explain every step of a result. Indeed, the veracity of a conjecture is often of secondary importance to the proof tools, frameworks, and constructions used to prove or disprove it. While recent advances in language models [3] have enabled neural nets to increasingly “show their work" [4], it remains largely true that the actual capabilities of a model usually exceed its legible and explainable capabilities.

Thus, for mathematics, neural networks may be expected to excel at providing counterexamples and edge cases more than at directly writing proofs. Nowhere is such an ability valued more than in combinatorics, and we focus on Ramsey theory [5], a key subarea. The key idea of Ramsey theory is to pose the question :

What is the minimal value of N such that any graph on $\geq N$ vertices must possess some property ?

The property of interest in this case is generally the existence of a clique (a subset of nodes which form between themselves a complete graph, i.e. a set of vertices V such that (v_i, v_j) is an edge in the graph for all $v_i, v_j \in V$) of some size **or** a corresponding maximum independent set, which is the “opposite" of a clique : a set of vertices V such that (v_i, v_j) is **not** an edge in the graph for all $v_i, v_j \in V$. For example, the classic illustration of Ramsey theory asks to show that :

Given 6 people who may be friends or enemies, there is a group of at least 3 friends or enemies.

Denoting “friend" as edge, and “enemy" as no edge, we can see that this is asking to show that any graph of ≥ 6 vertices has either a clique of size ≥ 3 or a maximum independent set of size ≥ 3. It can also be shown that 6 is the minimum number to satisfy such a property by finding a counterexample

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Workshop on MATH-AI.

on 5 vertices. We will denote this result as $R(3,3) = 6$. More generally, if Z is the minimum vertex count which ensures a clique of size m or a max independent set of size n, $R(m,n) = Z$.

The values of $R(\cdot,\cdot)$ are called Ramsey numbers. We will focus on the case $m = n$. These are called diagonal Ramsey numbers. The values of $R(m,m)$ are known exactly only upto $m = 4$. There exist ranges above this, for instance, $R(5,5)$ [6, 7] is known to be ≥ 43 and ≤ 48. These bounds become increasingly loose as m rises [8], and unlike the $m = 5$ case where the gap between the lower and upper value is 5, the corresponding gap becomes over $10,000$ even at $m = 10$. This reflects the difficulty in precisely finding $R(m,m)$.

Concurrently with this notion, we can discuss (m,n)-*free graphs*. A graph is (m,n)-free if its max clique size is $\leq m-1$ and its maximum independent set is $\leq n-1$. Of course, there can be no such graphs for node counts $\geq R(m,n)$. However, when $R(m,n)$ is not precisely known, these graphs, and their constructions, are beneficial in understanding the behaviour of the function R.

Given $N \in \mathbb{N}$, we define the problem of generating diagonal **Ramsey-extremal** graphs, i.e. generating (m,m)-free graphs on N vertices where m is minimal. For a concrete example, consider $N = 17$. We know that $R(4,4) = 18$. That implies, that at $N = 17$ it is possible to generate a graph whose max clique size and max independent set size are both ≤ 3. Indeed, there is only one such graph [9]. At $N = 16$, there are only two. As N grows, the number of possible graphs grows exponentially, which makes finding such graphs challenging when there are only 1 or 2 suitable candidates. Mathematical research has been conducted for decades on finding such graphs. Generally, exhaustive search is the way forward after some criteria such as degree, connectivity have been fulfilled. This is naturally often undesirable, and hard to scale. We would like to find out if neural approaches can help this area by generating such graphs en masse.

Unique suitabilities that motivate our choice of problem. There are many unsolved areas of mathematics of consequence, and one may question why we have chosen this particular area to focus on over, for instance, attempts to work on the Riemann hypothesis. Ramsey theory fulfills various criteria which are not easily simultaneously satisfied by other areas of mathematics as a test of machine learning capabilities. These are :

- **Very high legible difficulty.** Solving a mathematical statement with machine learning may reflect either the power of the method, or lack of attempts. Largely, we are interested in the former, and not the actual solution. Consider advances such as AlphaGo [10] or AlphaStar [1, 2], in the domain of games, which do not by themselves fulfill a purpose beyond entertainment. However, AlphaGo has tremendous **signal value** - it demonstrates progress in an area which attracts human competition of high skill levels. Ramsey theory fulfills this criterion - the problem has attracted interest from the top combinatorialists such as Erdos over a period spanning nearly a century and remains an active area of research in the present day by noted researchers in the field.
- **Sufficiency of low-dimensional concrete constructed instances.** Unlike most domains, the objects in Ramsey theory are relatively low-dimensional, concrete and not very abstract - graphs of somewhat low (< 100) vertex count. It is not clear, for instance, how to best encode a group or a vector space, let alone prove or disprove conjectures about them. Further, in Ramsey theory, counterexamples and extremal graphs have value by themselves, and are finite and enumerable. It would make no sense, for instance, to ask to generate all vector spaces which are a counterexample to some statement if the set of counterexamples was infinite. The parametrization would need to be suitably changed to generate classes of objects, instead of objects themselves. As another unsuitable scenario, we can consider number theory. Problems such as the collatz conjecture require working with very large integers, and encoding them becomes the question to solve.
- **Prior track record of computational approaches.** In Ramsey theory, we have a large bunch of extremal graphs already generated for us, ready to use [11, 12]. Most promisingly, these graphs are often generated by coding them up and simply enumerating them after reducing the problem search via degree sequences, triangle count and so on. This bruteforcing, based on “handcrafted" criteria, can be easily seen as a form of proto-data science, amenable to machine learning methods that can cut down on it.
- **Partial approaches have value.** In a lot of problems, the solution to a small sub-case has no value at all, and the community can learn nothing from it. Ramsey theory’s advances, on the

other hand, often come from extending previous solutions [13], even when these solutions are somewhat suboptimal. This is encouraging, as we do not expect neural approaches to be breakthroughs right away.

2 Machine learning approaches to combinatorics

Previously, there have been approaches utilizing Reinforcement Learning (RL) to prove or disprove conjectures in combinatorics [14]. In RL, the rules are set up and the agent tries to optimize a function, which would for our case be minimizing the maximum over the clique size and the independent set. Such approaches can especially find success when the objects are trees and can be Prufer coded (converted efficiently to sequences), allowing scalability to a relatively high number of nodes.

There is a key difference between solving an arbitrary conjecture in combinatorics, and trying to approach an already researched problem such as Ramsey graphs. In the latter case, we already have a well-studied problem, with previously found extremal graphs and bounds. We are not starting from scratch. This makes non-RL approaches more appropriate. We are interested in **gradient ascent**-like methods, i.e. offline optimization [15]. We view our work as moving from posing mathematical problems as supervised learning problems instead of reinforcement learning ones.

Concretely, our problem setting is as follows. We fix the number of nodes and create a train set of previously found extremal graphs, collated from combinatorics webpages. We then seek to train a model M which can reliably tell these extremal graphs apart from a contrasting set of Erdos-Renyi graphs by learning to predict the maximum of the clique and independent set. Then, we see how many unseen extremal graphs can be generated using this classifier/regressor, by finding the graph G_{opt} which minimizes $M(G_{opt})$ through backpropagation.

Recall that the Ramsey objective involves finding the max clique and the max independent set, and cannot be done in polynomial time. As such, we cannot hope to create a model that, in the general case, predicts the actual target perfectly. We can however, target to create a model that significantly improves on randomness and is correct upto some randomness. Note also that we do not need to predict the actual value, since our objective is gradient ascent. Predicting the correct rank order will do - and our performance will be measured by correlation statistics of the rank orders, i.e. Spearman and Kendall correlations. (Pearson correlations on respectively the rank and pairwise correctness.)

3 Approach, results and conclusion

We download the Ramsey-extremal graphs from : https://users.cecs.anu.edu.au/ bdm/data/ramsey.html. We focus on $(4, 4)$-free graphs on 14 nodes. We will see if upon creating a test and train dataset pair consisting of known extremal graphs from $R(4, 4)$ and Erdos-Renyi random graphs with $p = 0.5$ of equal size, a Graph Neural Network (GNN) can reliably predict the sought label (the maximum over the clique and the independent set), the correlation metrics, and the ability to generate graphs that are unseen which are also extremal by gradient ascent. We add an equal number of random graphs as the number of extremal graphs to both train and test. Note that for a given node count, if G is Ramsey-extremal, so is G', the complement of G. The probability of G, G' being generated from an Erdos-Renyi model is maximal at $p = 0.5$. Of all Erdos-Renyi models (i.e. models in which each edge is independent and has same probability p), this is the strongest benchmark to compare to.

Methods of parametrization and gradient ascent. We consider the following ways of initializing the data and model pipeline.

- Create for every graph G, the Laplacian embedding based feature of G as its node attributes to form the attributed graph G_{attr}, and predict the label using a GNN. This approach has the issue that the overall process is :

$$G \to G_{attr} \to GNN \to Y$$

  However, the $G \to G_{attr}$ step is not differentiable. This makes the gradient ascent difficult. This step can be circumvented by directly optimizing over G_{attr} and forming the adjacency matrix back from G_{attr} - this can be done as the node features are Laplacian eigenvectors, and thus if they are known, we also have a low-rank approximation of the Laplacian and then from that, the adjacency matrix, around which we can sample.

- Perform the process of gradient ascent after first learning an embedding z_G for every G through a graph VAE [16]. This process finds a candidate after ascent as z_C which can then be decoded via the VAE decoder. The advantage of this process is that the attribute step can be as non-differentiable as we like and we can encode G_{attr} to z. The disadvantage is that the VAE's fidelity begins to impact the optimization on top of the predictor network.
- Use a constant G_{attr}. The simplest example is using the attribute of a vector of 1s for all nodes in the graph. This allows direct optimization over G without running into backpropagation issues.

If a VAE is not used, we convert the adjacency matrix (Adj) entries to lie in $[0, 1]$ instead of $\{0, 1\}$, and optimize over it. In this case, the issue lies with the fact that we are training the GNN to learn a function - the maximum over clique and independent set - that is hard to define in the case where every adjacency matrix entry is a positive real number in $[0, 1]$. Nevertheless, we can try optimizing over the continuous case and apply randomized rounding (by replacing each A_{ij} with a Bernoulli r.v. of bias equal to A_{ij}) and try our results. We also try a case where we use no GIN at all and directly optimize over the matrix using a MLP-like model.

For our GNN, we used a 4-layer GIN [17] of hidden size 100. The Laplacian embedding used was of 4 dimensions, and $30,000$ extremal graphs each were used for train and testing, along with $30,000$ random graphs each in train and test to serve as a "contrast" to distinguish against. For the graph autoencoder, we built on a standard architecture[1] with 2 hidden layers of 40 units each. All our code was done in DGL+PyTorch. We also check the efficiency of Bayesian optimization [18] instead of naive gradient ascent at generation of examples. We add Erdos-Renyi (0.5) as a random benchmark.

3.1 Results

Table 1: Comparison of various methods described above to generate Ramsey-extremal graphs on 14 nodes. Standard deviations of the yield, the desired number, are shown by $\pm$. The first method (row) utilizes optimization over G_{attr} as described above in the methods. Last row is random benchmark.

Method	Accuracy	RMSE	Pearson	Spearman	Kendall	Yield
Laplacian + $[0, 1]$ Adj + GIN	93.8	1.98	0.61	0.64	0.50	18 $\pm$3
Constant + $[0, 1]$ Adj + GIN	94.2	3.84	0.84	0.85	0.69	42 $\pm$4
Laplacian + GNN-VAE + MLP	93.5	3.11	0.76	0.78	0.64	34 $\pm$4
$[0, 1]$ Adj + MLP (No GIN)	94.5	3.22	0.68	0.72	0.75	12 $\pm$3
Erdos-Renyi ($p = 0.5$)	N/A	N/A	N/A	N/A	N/A	$\leq 10^{-5}$

We present our results in table 3.1. We are able to generate many unseen Ramsey graphs, which we term **Yield**. Yield denotes how many non isomorphic and non complementary Ramsey graphs (to each other or to the train dataset) are generated from gradient ascent out of $10,000$ instances. The random method did not succeed in generating any random graphs which were extremal over $10,000$ instances, which shows the difficulty. We provide a calculated probability. **It can be seen clearly that our search approaches improve on a random method by many orders of magnitude.** We achieve high correlations and classification accuracies, while RMSE seems less relevant to yield.

We also attempted to generate extremal graphs using the setting of [14] and converting the cross-entropy method in it to an OpenAI environment and adding other methods such as SAC. Our best efforts yielded only 3 graphs for $N = 14$, over $\geq 100,000$ candidates. We consider RL to be not as attractive when train datasets are available. In the case of the VAE, we performed Bayesian optimization on top of the representations to see if active acquisition could improve the results. This led to no statistically significant improvement. We also attempted to generate extremal graphs for $N = 13$ and $N = 15$ using the model for $N = 14$. We succeeded in finding a yield ≥ 10 for $N = 13$, but none for $N = 15$. This reflects the number of candidate extremal graphs : $\approx 10^6, 130 \times 10^3, 640$ for $13, 14, 15$ respectively. It indicates generalizing over node counts is hard but not impossible.

Conclusion. We showcase neural models for generating Ramsey-extremal graphs that vastly outperform random baselines. Currently, although able to generate some new instances, they are hamstrung by isomorphism checking and the problem of being able to generate diverse sets at the scale of the

[1] https://github.com/shionhonda/gae-dgl

problem (potentially millions of graphs) even when the classifier is accurate upto a correlation. With the aid of better isomorph-avoiding bayesian optimization methods, we hope to generate diverse sets and make further progress in this domain, potentially even to solve unknown Ramsey numbers, and make headway for $R(5,5)$ in conjunction with RL to construct the train set from scratch.

References

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

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

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[5] Ronald L Graham, Bruce L Rothschild, and Joel H Spencer. *Ramsey theory*, volume 20. John Wiley & Sons, 1991.

[6] Vigleik Angeltveit and Brendan D McKay. $r(5,5) \leq 48$. *arXiv preprint arXiv:1703.08768*, 2017.

[7] Geoffrey Exoo. A lower bound for r (5, 5). *Journal of graph theory*, 13(1):97–98, 1989.

[8] Stanislaw Radziszowski. Small ramsey numbers. *The electronic journal of combinatorics*, 1000:DS1–Aug, 2011.

[9] Robert E Greenwood and Andrew Mattei Gleason. Combinatorial relations and chromatic graphs. *Canadian Journal of Mathematics*, 7:1–7, 1955.

[10] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[11] Brendan D McKay and Stanisław P Radziszowski. Subgraph counting identities and ramsey numbers. *journal of combinatorial theory, Series B*, 69(2):193–209, 1997.

[12] Gunnar Brinkmann, Kris Coolsaet, Jan Goedgebeur, and Hadrien Mélot. House of graphs: a database of interesting graphs. *Discrete Applied Mathematics*, 161(1-2):311–314, 2013.

[13] David Conlon, Jacob Fox, and Benny Sudakov. Recent developments in graph ramsey theory. *Surveys in combinatorics*, 424(2015):49–118, 2015.

[14] Adam Zsolt Wagner. Constructions in combinatorics via neural networks. *arXiv preprint arXiv:2104.14516*, 2021.

[15] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[16] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[17] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[18] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[19] Paul Erdős and András Hajnal. On chromatic number of graphs and set-systems. *Acta Math. Acad. Sci. Hungar*, 17(61-99):1, 1966.

[20] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 459–467, 2018.

[21] Béla Bollobás. Degree sequences of random graphs. *Discrete Mathematics*, 33(1):1–19, 1981.

[22] Christopher Cox, Michael Ferrara, Ryan M Martin, and Benjamin Reiniger. Chvatal-type results for degree sequence ramsey numbers. *arXiv preprint arXiv:1510.04843*, 2015.

[23] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

[24] Xiangyang Ju, Steven Farrell, Paolo Calafiura, Daniel Murnane, Lindsey Gray, Thomas Klijnsma, Kevin Pedro, Giuseppe Cerati, Jim Kowalkowski, Gabriel Perdue, et al. Graph neural networks for particle reconstruction in high energy physics detectors. *arXiv preprint arXiv:2003.11603*, 2020.

[25] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[26] Guo Zhang, Hao He, and Dina Katabi. Circuit-gnn: Graph neural networks for distributed circuit design. In *International conference on machine learning*, pages 7364–7373. PMLR, 2019.

[27] Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[28] Robert Kirby, Saad Godil, Rajarshi Roy, and Bryan Catanzaro. Congestionnet: Routing congestion prediction using deep graph neural networks. In *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 217–222. IEEE, 2019.

[29] Amur Ghose, Vincent Zhang, Yingxue Zhang, Dong Li, Wulong Liu, and Mark Coates. Generalizable cross-graph embedding for gnn-based congestion prediction. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[30] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Wenbin Xu, Sachin S Sapatnekar, Ramesh Harjani, and Jiang Hu. A customized graph neural network model for guiding analog ic placement. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2020.

[31] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[32] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[33] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[34] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. *arXiv preprint arXiv:1711.00740*, 2017.

[35] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L Dill. Learning a sat solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*, 2018.

[36] Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*, 2021.

A Appendix

A.1 Motivation behind (4,4)-free graphs on 14 nodes

The choice of focusing on 14 nodes might seem a bit peculiar. However, in general, for (m, n)-free extremal graphs, the most insightful constructions (from the mathematical point of view) tend to occur when they are constructed with a node count as large as possible (as for very small node counts, the property trivially holds - the difficulty and usefulness of such an instance grows as we increase node count). For $(4, 4)$-free cases, there are $\approx 130 \times 10^3$ instances for 14 nodes but only 640 for 15 nodes. This abrupt drop in the number of available samples greatly lowers the efficacy of training based methods and makes our approach unfeasible. Meanwhile, $R(5, 5)$ is as of yet open and lacks a catalogue of extremal graphs i.e. $(5, 5)$-free graphs which are already similarly enumerated and pre-computed. Therefore, focusing on $(4, 4)$-free graphs on 14 nodes is not ad hoc, but an active choice of the highest leverage open point of research.

A.2 How does any of this help us understand R?

We have mentioned earlier that constructing these (m, n)-free graphs helps us understand the behavior of the function R itself. This point needs further elaboration.

Generally, our present day understanding of $R(m, m)$ for $m \geq 5$ is of the form $[a_m, b_m]$ i.e. inclusive ranges. Now, $b_m >> a_m$ becomes increasingly common as we increase m. If we can construct, given m, (m, m)-free graphs that exceed a_m, the lower bound is violated and thus a_m is changed upwards.

However, the reverse is far more difficult. Constructing a c-node (m, m)-free graph makes the new $a_m \geq c + 1$, but if our method fails for a given c, this might indicate that the new $b_m \leq c$, **or** that our method is too weak (and the latter cannot be verified for certainty in the way a construction that changes a_m can). Therefore, the future roadmap from this work (from the human perspective) is to manually examine, through classical proof techniques, value ranges of c where the algorithm cannot find a (m, m)-free graph, while raising the lower bound through automated means.

Once the bounds are narrow enough, more sophisticated techniques can be employed in the sense that extremal graphs on nodes $N + 1$ often arise from extensions of the extremal graphs on N nodes [13], and RL [14] can be used here once the near-extremal graph is found to modify it into an extremal graph.

A.3 Why these particular tools ?

The motivation for the VAE requires some elaboration. Generally, given a graph G, the representation for it is done in two steps : give each vertex $n \in G$ a representation, then run a GNN on G, which creates node level representations h_v, which are then pooled (summed/averaged) to get h_g the graph representation.

In this entire process, if G is used as the object to perform gradient steps over, the issue is forming the representation. For example, suppose the representation for v is the degree of v - there is no problem here, because the degree is a differentiable row-wise sum of the adjacency matrix, and the overall process is differentiable.

However, suppose we wish to assign to each $v \in G$ an embedding akin to the random walk embedding over G, or its color under some coloring of the graph [19]. The former can be approximated by eigendecomposition [20] of suitable transformed matrices and the same is true for Laplacian embeddings, but these steps require some work to be put into differentiable forms. The latter - color under some coloring - is an example of a case where we cannot easily write the expression in terms of the adjacency. In both these cases, the step that breaks is the step of forming the node embedding. Everything else works. For certain cases, there is a workaround - if A - the adjacency itself - can be formed from the embedding approximately, we can optimize over the node embeddings directly. (This occurs with the Laplacian embedding.)

VAE helps here by creating a latent that works with the embedding even if the embedding was nontrivial to compute. The gradient ascent occurs in the latent space, and is decoded to the structure. The fact that non-trivially computed embeddings were utilized to learn the latent space does not

hinder the subsequent gradient ascent. In the place of the VAE, any suitable representation learner that allows a reconstruction loss is usable.

While this motivates the usage of the VAE, it can be seen that the best result is actually from a non-VAE method. We still however consider the VAE method worthy of study as it can utilize any embedding method and there will possibly emerge domain-specific embeddings based on e.g. degree sequences [21, 22] that can fully take advantage of the VAE setup.

The usage of the GNN on top is simply due to the fact that these tools have succeeded in modeling physics [23, 24, 25], circuits [26, 27, 28, 29, 30], molecules [31, 32], programs [33, 34] and other abstract domains [35, 36], and thus are a natural fit.

A.4 Implementation details

The data representations were converted to dgl. Optimization was done via standard RMSE and SGD, with the output as a continuous prediction (classification-like integer outputs were also tried but performed slightly worse). In the VAE case, there is also a reconstruction loss as per the original GAE [16].