Finding Increasingly Large Extremal Graphs with AlphaZero and Tabu Search*

Abbas Mehrabian[1], **Ankit Anand**[1], **Hyunjik Kim**[1], **Nicolas Sonnerat**[1], **Matej Balog**[1], **Gheorghe Comanici**[1], **Tudor Berariu**[2], **Andrew Lee**[1], **Anian Ruoss**[1], **Anna Bulanova**[1], **Daniel Toyama**[1], **Sam Blackwell**[1], **Bernardino Romera Paredes**[1], **Petar Veličković**[1], **Laurent Orseau**[1], **Joonkyung Lee**[3], **Anurag Murty Naredla**[4], **Doina Precup**[1] and **Adam Zsolt Wagner**[5]

[1]Google DeepMind
[2]Imperial College London
[3]Yonsei University
[4]University of Bonn
[5]Worcester Polytechnic Institute

Abstract

This work proposes a new learning-to-search benchmark and uses AI to discover new mathematical knowledge related to an open conjecture of Erdős (1975) in extremal graph theory. The problem is to find graphs with a given size (number of nodes) that maximize the number of edges without having 3- or 4-cycles. We formulate this as a sequential decision-making problem and compare AlphaZero, a neural network-guided tree search, with tabu search, a heuristic local search method. Using either method, by introducing a curriculum—jump-starting the search for larger graphs using good graphs found at smaller sizes—we improve the state-of-the-art lower bounds for several sizes. We also propose a flexible graph-generation environment and a permutation-invariant network architecture for learning to search in the space of graphs.

1 Introduction

With the recent advances in neural networks, artificial intelligence (AI) methods have achieved tremendous success in multiple domains like game playing [Silver *et al.*, 2018], biology [Jumper *et al.*, 2021], mathematics [Davies *et al.*, 2021], and robotics [Peng *et al.*, 2018]. Mathematics is of particular interest to AI researchers due to its challenging multistep reasoning structure, open-ended problems, and limited data. While automated theorem proving has always been of interest to AI researchers as a reasoning benchmark [Abdelaziz *et al.*, 2022; Aygün *et al.*, 2022; Kovács and Voronkov, 2013; Lample *et al.*, 2022; Polu and Sutskever, 2020; Schulz, 2002], some recent work have used machine learning to solve research problems across the fields of representation theory, knot theory, graph theory, and matrix algebra [Davies *et al.*, 2021; Fawzi *et al.*, 2022; Wagner, 2021].

Many mathematical problems can be modeled as searching for an object or a structure of desired characteristics in an extremely large space. Indeed, automated theorem proving is often modeled as searching for a sequence of operations—a proof—in an ever-growing space of operands with a few operators. Another example is counterexample generation [Wagner, 2021], where the object of interest is a counterexample to a particular conjecture or a mathematical construction that improves the bounds for a problem. The recent work of AlphaTensor [Fawzi *et al.*, 2022] also relies on neural network-guided tree search to find novel tensor decompositions that result in faster matrix multiplication algorithms.

Inspired by these, we focus on a classical extremal graph theory problem, studied by [Erdős, 1975], which is to find, for any given number of nodes, a graph that maximizes its number of edges but is constrained not to have a 3-cycle or a 4-cycle. While the problem is simple to state, mathematicians have not found optimal constructions for all sizes: the maximum number of edges is known for up to 53 nodes [Inc., 2023], and lower bounds using local search have been reported for up to 200 nodes [Bong, 2017; Garnick *et al.*, 1993]. Because strong local-search methods have been developed for this problem, it provides a challenging benchmark for learning-based search methods. We believe that developing new methods for graph problems could inspire methods for related fields such as drug discovery and chip design, where the goal is to find a graph object minimizing or maximizing a given objective function. Discovering new optimal solutions to this problem could lead to more efficient designs in other problems, such as data center organization and optimization as well as game theory.

In this paper, we use reinforcement learning (RL) and formulate graph generation as a sequential decision making process. In contrast to the graph-generation RL environment used by [Wagner, 2021], which starts from an empty graph and adds edges one by one in a fixed order, we start from an arbitrary graph and add/remove edges in an arbitrary order. This RL environment, called the *edge-flipping* environment, has at least two advantages: (a) we can start from known "good" graphs

*The full version is available at https://arxiv.org/abs/2311.03583.

to find even better graphs, and (b) we can scale to larger sizes by using a curriculum of starting from slightly smaller graphs. As our RL agent, we use the state-of-the-art AlphaZero [Silver *et al.*, 2017] algorithm, which has shown impressive success in a variety of domains such as tensor decomposition [Fawzi *et al.*, 2022], discovering new sorting algorithms [Mankowitz *et al.*, 2023], and game playing [Silver *et al.*, 2017].

A novel neural network architecture. Since AlphaZero is guided by a neural network, we also need a representation that aligns with the invariants of the search space. We deal with simple undirected graphs, so graph neural networks naturally come to mind [Veličković, 2023]; but we go beyond them and introduce a novel representation, the *Pairformer*: unlike traditional graph neural networks, which pass messages between nodes, Pairformers pass messages between *pairs of nodes*. Pairformers burden us with additional computational cost but are significantly better at detecting cycles.

The Pairformer has edge features for all existing and non-existing edges between nodes, whereas standard GNNs have edge features only for existing edges in the graph. This property of Pairformers allows the network to directly reason about non-existing edges, which is important for the policy network to understand which edges should be added or removed in the graph. Combined with triangle self-attention updates, this enables effective processing of neighboring edge features that can capture presence or absence of cycles with only few Pairformer layers. The Pairformer network architecture is novel and provides significant improvements over the ResNet architecture. We hope that this architecture will be useful to tackle other graph problems, too.

Incremental learning. When searching over all graphs, one challenge is that the number of graphs with a given number of nodes n increases exponentially with n; thus, finding optimal graphs becomes significantly harder as n grows. Interestingly, known optimal graphs for this problem have a *substructure property*: in many cases, optimal graphs of a given size are near-subgraphs of optimal graphs of larger sizes (see, e.g., [Backelin, 2015, Theorem 3]). Thus, finding near-optimal graphs for smaller n can serve as a stepping stone to find good graphs for larger values of n. This property can be used to construct a curriculum: start from discovered graphs of a given size, generate novel solutions of a larger size, and repeat. Our edge-flipping environment provides the flexibility to start from any graph and add or drop edges arbitrarily, so we are not restricted to supergraphs of the starting graph and can reach all graphs of that size. Deploying these ideas in AlphaZero, we develop *Incremental AlphaZero* and improve the lower bounds for sizes 64 to 136.

This way of scaling to larger sizes is related to the idea of curriculum learning [Soviany *et al.*, 2022], a widely-used method in RL, especially for solving hard exploration problems in many domains, including robotics [Akkaya *et al.*, 2019] and automated theorem proving [Aygün *et al.*, 2022] as well as solving the Rubik's cube and other difficult puzzles [Agostinelli *et al.*, 2019; Orseau *et al.*, 2023]. Note that the term "curriculum learning" has been used with different meanings in machine learning literature; in this paper, by *curriculum* we mean solving the problem on the smaller size first and then using the solution of the smaller problem to solve the same problem on the larger size.

Incremental local search. The substructure property can enhance other types of search as well. We develop an incremental version of tabu search, a known local search method [Glover, 1989], where the initial graph for each size is sampled from a previously-discovered "good" graph of a smaller size. This algorithm also improves over the state of the art. Our ablation shows that both search strategies improve significantly by this idea of incremental learning where we scale to larger sizes by using high scoring graphs of slightly smaller size.

Summary of contributions. We introduce a challenging benchmark for learning-to-search in large state spaces, inspired by an open problem in extremal graph theory, whose best solutions thus far are achieved by local search. We formulate graph generation as an edge-flipping RL environment and introduce the novel representation Pairformer, which is well-suited for detecting cycles in undirected graphs. We introduce the idea of incremental search (curriculum) to local search methods as well as AlphaZero, and show that kickstarting from solutions of smaller size is a key ingredient for improving the results on this extremal graph theory problem. We improve the lower bounds for the problem for all graph sizes from 64 to 134, and we release these graphs to the research community to aid further research: https://storage.googleapis.com/gdm_girth5_graphs/girth5_graphs.zip.

2 Problem Description

A *k-cycle* is a cycle with k nodes. Let G be a simple, undirected n-node graph that has no 3-cycles. What is the maximum number of edges that G can have? Mantel [Mantel, 1907] proved that the answer is precisely $\lfloor n^2/4 \rfloor$, initiating the field of extremal graph theory. Turán [Turán, 1941] generalized this result to cliques and found, for any k, the maximum number of edges that an n-node graph without k-cliques can have.

Generally, for a set $\mathcal{H}$ of graphs, let $\mathrm{ex}(n, \mathcal{H})$ denote the maximum number of edges in an n-node graph that does not contain any member of $\mathcal{H}$ as a subgraph (the symbol ex stands for "extremal"). Calculating $\mathrm{ex}(n, \mathcal{H})$ for various graph classes $\mathcal{H}$ is a central problem in extremal graph theory. In this paper, we study

$$f(n) \coloneqq \mathrm{ex}(n, \{C_3, C_4\}). \tag{1}$$

We know $\mathrm{ex}(n, \{C_3\}) = \lfloor n^2/4 \rfloor$ by Mantel's theorem and $\lim_{n\to\infty} \mathrm{ex}(n, \{C_4\})/(n\sqrt{n}) = 1/2$ by [Brown, 1966; Erdős *et al.*, 1966], but no formula has been found for $f(n)$, and even its asymptotic behavior is not understood well.

[Erdős, 1975] conjectured that $\lim_{n\to\infty} \frac{f(n)}{n\sqrt{n}} = \frac{1}{2\sqrt{2}}$. This conjecture has remained open since 1975. The tightest bounds are due to [Garnick *et al.*, 1993], who proved

$$\frac{1}{2\sqrt{2}} \leq \lim_{n\to\infty} \frac{f(n)}{n\sqrt{n}} \leq \frac{1}{2}. \tag{2}$$

Motivated by this conjecture, we want to estimate the value of $f(n)$ for specific values of n. It is known that $f(n) \leq n\sqrt{n-1}/2$ for all n [Garnick *et al.*, 1993]. The exact value

of $f(n)$ is known when $n \leq 53$ [Inc., 2023], and constructive lower bounds have been reported for all $n \leq 200$ [Bong, 2017; Garnick *et al.*, 1993]. Our goal is to improve these lower bounds for $54 \leq n \leq 200$. Hence, we want to find n-node graphs without 3-cycles or 4-cycles that have as many edges as possible.

The *size* of a graph is its number of nodes. For any graph G, we denote its number of edges, 3-cycles, and 4-cycles by $e(G)$, $\triangle(G)$, and $\square(G)$, respectively. We say that a graph G is *feasible* if it has no 3-cycles and no 4-cycles. The *score* of a graph G is defined as

$$\mathrm{s}(G) \coloneqq e(G) - \triangle(G) - \square(G).$$

The following lemma (proof in the full version) implies that, for any given number of nodes n, proving lower bounds for $f(n)$ is equivalent to maximizing the score over all n-node graphs.

Lemma 1. *For any n-node graph G, we have $\mathrm{s}(G) \leq f(n)$; and there exists at least one n-node feasible graph for which equality holds.*

In light of Lemma 1, we can formulate the problem of maximizing $f(n)$ in two ways: we can maximize $e(G)$ over *feasible* n-node graphs or maximize $\mathrm{s}(G)$ over *all* n-node graphs. The two formulations have the same optimal value, but their search space differs. The first formulation has a smaller search space, but the second one, which we use, allows us to define a convenient *neighborhood function*, which helps our algorithms navigate the space of graphs more smoothly.

Definition 1 ($\bigoplus$, flipping)**.** *Let G be a graph and let u and v be two of its nodes. If uv is an edge in G, then $G \bigoplus uv$ is obtained by removing the edge uv from G; otherwise, $G \bigoplus uv$ is obtained by adding the edge uv. In either case, we say $G \bigoplus uv$ is obtained by* flipping *uv.*

The flipping operation has two desirable properties: first, any n-node graph assumes exactly $\binom{n}{2}$ flips, a technical convenience for RL agents' action space; second, any n-node graph can be reached from any other n-node graph by doing up to $\binom{n}{2}$ many flips; that is, there are no "dead ends."

3 Graph Generation as an RL Environment

We define *graph generation* as a sequential decision making process, where we start from an n-node graph G and, at each step, modify it by adding or removing an edge e to obtain $G' = G \bigoplus e$. More formally, graph generation is a deterministic finite-horizon Markov Decision Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, R, \mathcal{T}, H\}$, where the state space, $\mathcal{S}$, consists of all simple undirected graphs of size n; the action space, $\mathcal{A} \coloneqq \{(i,j) : 1 \leq i < j \leq n\}$, consists of all edges of the complete graph of size n; the one-step reward function (defined below) is $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; the deterministic transition function, $\mathcal{T}$, is defined as $\mathcal{T}(G, e) \coloneqq G \bigoplus e$; and the horizon, H, denotes the number of steps in each episode.

We define the reward function as $R(G, e) \coloneqq \mathrm{s}(G \bigoplus e) - \mathrm{s}(G)$, i.e., the reward equals the change in the score after taking the action. We call this *the telescopic reward*, as the rewards accumulated over time form a telescoping series, making the episode return equal to $\mathrm{s}(G_H) - \mathrm{s}(G_0)$. Note that $\mathrm{s}(G_H)$ is precisely the objective value that we want to maximize. (Often in RL, a discount factor is introduced, and subsequent rewards are discounted when computing the return; but we do not introduce a discount factor here, as then the return would have been different from the actual objective function.) In our experiments, we found that the telescoping reward performs much better than the *non-telescoping* reward, where the reward is given at the end of each episode and equals $\mathrm{s}(G_H)$.

This *edge-flipping environment* provides more flexibility than environments in which the graph is built, for instance, by deciding about the edges one by one in a fixed order [Wagner, 2021]. An advantage of the edge-flipping environment is that every action is reversible: in this MDP, any n-node graph can be reached from any other n-node graph. This property is particularly useful when using a curriculum, as one can use *state resetting* [Florensa *et al.*, 2017; Hosu and Rebedea, 2016; Salimans and Chen, 2018] to warm up exploration from high-scoring initial graphs; e.g., start from high quality graphs of size $n-k$ (garnished with k isolated nodes) to build the desired graph of size n. On the other hand, reversibility can cause learning instability: since there is no termination action, an agent could end up flipping one of the edges indefinitely. To avoid such issues, we set a fixed horizon length.

4 AlphaZero for Graph Generation

AlphaZero is a reinforcement learning algorithm that demonstrated superhuman performance on Go, through self-play, without using any human knowledge [Silver *et al.*, 2017]. It was then adapted to show superhuman performance in other games such as chess and shogi [Silver *et al.*, 2018]. Recently, a version of AlphaZero was adapted to find faster algorithms for matrix multiplication. This new model, named AlphaTensor, improved Strassen's matrix-multiplication algorithm for some sizes for the first time after 50 years [Fawzi *et al.*, 2022]. The AlphaZero algorithm combines Monte Carlo Tree Search (MCTS), a heuristic search algorithm, with deep neural networks to represent the state space, e.g., a chessboard position. In our application of AlphaZero, each state is a simple undirected graph and each action is adding or removing an edge.

The edge-flipping environment is a deterministic MDP, where both the transition matrix and the reward function are fully known and deterministic, thus most search and planning algorithms are applicable. MCTS builds a finite tree rooted at the current state and, based on the statistics gathered from the neighboring states, selects the next action. Many successful works using MCTS use some variant of the upper confidence bound rule [Kocsis and Szepesvári, 2006] to balance exploration and exploitation when expanding the tree. While traditional approaches used Monte Carlo rollouts to estimate the value of a leaf state, in the last decade this has largely been replaced by a neural network, called the *value network*. Another neural network, called the *policy network*, determines which child to expand next. Often, the policy and value networks share the same first few layers. (They have the same latent representation, or torso, but they have different heads.) In AlphaZero, both the policy and value networks are trained using previously observed trajectories—see [Silver *et*

al., 2018] for details.

For updating the value of a state—which is a node in the MCTS tree—standard MCTS expands the node and uses the average value of the children. Since we want to maximize the best-case return rather than the expected return, it may appear more suitable to use the maximum value of the children to update the value of the node. We attempted this approach but it did not yield improvements.

One common issue with AlphaZero is encouraging it to diversely explore the space of possible trajectories. We attempted a few ideas to diversify, such as increasing UCB exploration parameter and also for each trajectory, if same graph (s_i) is encountered which has been seen in the previous timesteps ($t < i$) within the trajectory, we discourage this behavior by giving a small negative reward; but none of these approaches improved the result on top of starting from good graphs of smaller size.

4.1 Network Representation

To find a good representation for this problem of avoiding short cycles, we tested different architectures on the supervised learning problem of cycle detection, using this as a proxy for our RL problem for fast experiment turnaround. In particular, we compared ResNets [He *et al.*, 2016], Pointer Graph Networks [Veličković *et al.*, 2020], Graph Attention Networks [Veličković *et al.*, 2017], and a novel architecture called the *Pairformer*, described below. We studied node and edge level binary classification tasks (whether a node or an edge is part of a short cycle) as well as graph level tasks (whether a graph contains a short cycle). The Pairformer gave the best performance, hence this is the architecture we used in the RL setting.

An intuitive understanding for the Pairformer can be gained by recognising its main difference with standard GNNs: the Pairformer has edge features for all existing and non-existing edges between nodes, whereas standard GNNs have edge features only for existing edges in the graph. This property of Pairformers allows the network to directly reason about non-existing edges, which is important for the policy network to understand which edges should be added or removed in the edge-flipping environment. Combined with triangle self-attention updates (explained below), this enables effective processing of neighboring edge features that can capture presence or absence of cycles with only few Pairformer layers. We found that the additional computational cost is worth it.

The Pairformer is a simplified version of Evoformer, used in AlphaFold [Jumper *et al.*, 2021]. Each Evoformer block has two branches of computation: one processes the multiple sequence alignment (MSA) representation and the other one processes the pair representation. The Pairformer only uses the pair representation branch, which processes per-edge features and has shape (n, n, c). We set $c = 64$ in our implementations. Within the pair representation branch, each Pairformer block is composed of triangle self-attention blocks (row-wise multihead self-attention followed by column-wise multihead self-attention) followed by fully-connected layers with LayerNorm [Ba *et al.*, 2016]. We omitted the triangle multiplicative updates in the original Evoformer as they had minimal effect on performance for our tasks. A key difference with standard graph neural networks is that instead of only having features for existing edges, the Pairformer has features for all $\binom{n}{2}$ pairs of nodes, whether they correspond to existing edges or not. We believe that considering non-existing edges is crucial for the Pairformer to inform the policy for deciding whether to add new edges to the graph or not. This architecture is used as the torso network, which receives the current graph as input and outputs a representation that is consumed by the policy and value heads.

The current graph is given input as an $n \times n$ adjacency matrix. Since we use a single network for multiple sizes, we condition the torso and the policy head on the graph size n by concatenating each input with a matrix of 1s on the principal $m \times m$ (where $m < n$) submatrix and 0s everywhere else (concatenate along the channel dimension). This lets us use a shared set of parameters for multiple graph sizes without a separate network for each size.

A good model architecture should not only be expressive but also have fast inference in order for acting to be fast enough to quickly generate lots of data for the learner to optimize the model. The downside of the Pairformer is its $O(n^3)$ runtime, while ResNet's runtime is $O(n^2)$. Hence there exists a trade-off between expressiveness and speed, and we experimentally found that combining a small Pairformer torso with a larger ResNet policy head provides the best balance. Using a ResNet for the torso performs much worse, implying that the expressiveness that the Pairformer brings to the torso's representation of the input graph is indispensable—see Figure 3. For the value head, we used a feed-forward network over the representation provided by the Pairformer.

Another important detail is that although the environment supports only $\binom{n}{2}$ many actions, the last layer of our policy network has twice as many logits: for each edge, there is one logit for adding that edge and another logit for removing that edge. This means half of the logits correspond to invalid actions (e.g., adding an edge for an existing edge). We mask these invalid actions so a valid probability distribution is induced on the valid set of $\binom{n}{2}$ actions.

4.2 Distributed Implementation and Joint Learning Across Multiple Sizes

We use a distributed implementation of AlphaZero with multiple processing units: in each run, there are multiple actors, one replay buffer, and one learner. Each actor has a copy of the networks (supplied by the learner) and generates episodes, which are inserted into the replay buffer. The learner repeatedly samples an episode from the replay buffer to update the policy and value networks. The policy network is trained using the cross-entropy loss, where the ground truth label is assumed to be the decision taken at the root of the MCTS tree. The value network is trained using regression on the future return (sum of the future rewards) at each state of an episode.

For efficiency and transfer-learning across multiple sizes, we jointly train a single network for multiple sizes: each run of AlphaZero is provided with a list of target sizes, and each actor samples a target size uniformly at random from this list. The network input is modified in this case by padding it by 0s to turn it into a $target \times target$ matrix, while appending another plane to the observation, each entry of whose principal

$size \times size$ submatrix is 1 and the rest are 0. This helps to run experiments for multiple sizes jointly and ensures transfer-learning across different sizes.

4.3 Incremental AlphaZero

A key observation about our problem is that, in many cases, the optimal graph for a given size is nearly a subgraph of an optimal graph for a larger size. For example, by [Backelin, 2015, Theorem 3], all the optimal graphs for sizes 40, 45, 47, 48, 49 are subgraphs of the optimal graph for size 50. While this is not strictly true for all the sizes, it can be used as a heuristic to guide the search. As all the optimal graphs up to size 52 are known [McKay, 2023], we use high-scoring graphs of smaller sizes (garnished with a suitable number of isolated nodes) as the initial graph and iterate. Hence if the results for smaller sizes improve, this hopefully leads to subsequent improvements for larger sizes as well. This not only exploits the approximate substructure property for the problem but is also related the well-known idea of curriculum learning in machine learning [Bengio *et al.*, 2009].

Specifically, starting episodes from the empty graph leads to a difficult credit assignment problem for the RL agent, as the horizon should be long. Instead, we start from a high-scoring graph of size $n - k$ to build the target graph of size n. We observe that this choice of initial graph is critical for the performance of AlphaZero. It leads to a shorter horizon and more effective credit assignment in each episode. We call the resulting algorithm as *Incremental AlphaZero*.

5 Tabu Search for Graph Generation

Tabu search is a well-known iterative local search method [Glover, 1989]: given an objective function and a neighborhood structure over a set of states, it repeatedly moves from the current state to the neighboring state with the highest objective value, until some stopping condition is met. To avoid getting stuck at local minima, tabu search bans revisiting recently-visited states—hence the name "tabu" search.

In our case, the states are the graphs of a given size, the graphs obtained by flipping a single edge are the neighbors of the current graph, and the objective function is $\mathrm{s}(G) = e(G) - \triangle(G) - \square(G)$. Our tabu search algorithm (Algorithm 1) slightly differs from the typical definition; instead of banning visiting *states* that were recently visited, we ban playing the recently-played *actions*. Namely, we ban re-flipping edges that were flipped recently. This idea, inspired by [Parczyk *et al.*, 2023], results in a slightly faster algorithm than the usual tabu search. Note that the algorithm needs an initial graph G_0—we will describe later how it's chosen—and has a single hyperparameter: the history size, denoted by h in Algorithm 1, which determines the number of iterations flipping an edge is banned once it is flipped. Recall that $\bigoplus$ denotes flipping an edge.

5.1 Incremental Tabu Search

The *incremental* tabu search algorithm is inspired by the idea mentioned in section 4.3: we let the tabu search at each size start its search from one of the best graphs found at smaller sizes. Say we want to find lower bounds for $f(n)$ for some range $n \in \{a, \dots, b\}$. Incremental tabu search is a distributed algorithm with $b - a + 1$ parallel workers (processing units), indexed from a to b, where the worker with index n searches for graphs of size n. The workers need a common memory to share the graphs they have found: suppose that $BestGraphs[n]$, for $n \in \{a, \dots, b\}$, is a set of graphs that all workers have access to. (We can initialize it to contain just the empty graph of size n.) The algorithm for the size-n worker appears in Algorithm 2.

Algorithm 1 Our version of tabu search

Require: G_0 is an n-node graph with nodes indexed from 1 to n, and $0 \le h < \binom{n}{2}$
Ensure: $BestGraph$ is the highest-scoring graph found during search

$Tabu \leftarrow$ a first-in-first-out queue of fixed size h
$Actions \leftarrow \{(i, j) : 1 \le i < j \le n\}$
$BestGraph \leftarrow G_0$
for $i \leftarrow 1, 2, \dots, iterations$ **do**
  $ValidActions \leftarrow Actions \setminus Tabu$
  $BestActions \leftarrow \arg\max_e \{\mathrm{s}(G_{i-1} \bigoplus e) : e \in ValidActions\}$
  $Action \leftarrow$ random action chosen from $BestActions$
  $G_i \leftarrow G_{i-1} \bigoplus Action$
  Insert $Action$ into $Tabu$
  if $\mathrm{s}(G_i) > \mathrm{s}(BestGraph)$ **then**
    $BestGraph \leftarrow G_i$
  end if
end for

Algorithm 2 Incremental tabu search (worker for size n)

Require: $0 \le K$ and $a \le n \le b$ and $BestGraphs[n]$ is a set of graphs of size n
Ensure: $BestGraphs[n]$ contains the set of highest-score graphs found during the search

while True **do**
  Sample k randomly from $\{1, \dots, K\}$
  Sample G_0 randomly from $BestGraphs[n - k]$
  Add k isolated nodes to G_0
  Run tabu search starting from G_0
  $BestFoundGraph \leftarrow$ best graph found by tabu search
  Choose $ExistingGraph$ arbitrarily from $BestGraphs[n]$
  if $\mathrm{s}(BestFoundGraph) > \mathrm{s}(ExistingGraph)$ **then**
    $BestGraphs[n] \leftarrow \{BestFoundGraph\}$
  else
    Add $BestFoundGraph$ to $BestGraphs[n]$
  end if
end while

6 Experiments and Results

We compare five methods: tabu search starting from the empty graph; incremental tabu search, which uses a curriculum to use high-scoring graphs found at each size as the starting point for larger sizes; AlphaZero starting from the empty graph; incremental AlphaZero, which uses a curriculum; and Wagner's cross-entropy method [Wagner, 2021], the first machine-learning method to find counterexamples for mathematical conjecture. For AlphaZero, we also perform ablations on the choice of network representation. Since $f(n) = \Theta(n\sqrt{n})$ (see (2)) we have normalized the scores by $n\sqrt{n}$ in the plots. The hyperparameters are provided in the full version.

For each size n, we started the episodes in incremental AlphaZero with one of the high-scoring graphs found by incremental tabu search at size $n - k$, where k is chosen randomly between 1 and 4. (We did this for technical convenience, but we believe similar results are achievable if we sample from graphs of smaller sizes generated by previous runs of AlphaZero.)

A key hyperparameter is the episode length (the horizon). An overly long episode length is not only wasteful but also hinders learning, as the agent may find an optimal graph in the middle of an episode but still has to flip edges to reach the end of the episode. On the other hand, an overly short episode length would hinder the exploration as the agent is limited to the vicinity of the initial graph. We ran AlphaZero (without incremental search) for sizes 5 to 100, split these sizes in five nearly-equal buckets of 5–20, 21–40, 41–60, 61–80, and 81–100, and set horizons to 80, 160, 240, 320, and 434, respectively. With incremental search, since we start from a good graph of smaller size, we can choose shorter horizons; we experimented with horizon lengths of 30, 50 and 100 but found the results don't change much beyond 30.

For tabu search, we tried various history sizes but size 5 worked best. For each size, we ran 32 parallel copies of tabu search for seven days, restarting every 1000 iterations and merging the results. For incremental tabu search, we initialized $BestGraphs[n]$ (see Algorithm 2) to contain the set of graphs published by [McKay, 2023] (for $n = 1, 2, \ldots, 64$), set the history size to 5, and the K in incremental tabu search to 4—it is important this is greater than 1.

Comparison with the state-of-the-art lower bounds. As Figure 1 shows, incremental tabu search improves the state-of-the-art lower bounds[1] when $n \in \{64, \ldots, 134\} \cup \{138, \ldots, 160\} \cup \{176, \ldots, 186\} \cup \{188, \ldots, 190\}$. For a concrete example, see the full version, where we have also listed the lower bounds achieved by incremental tabu search for $n = 1, 2, \ldots, 200$. Incremental AlphaZero also improves over the state-of-the-art lower bounds on many sizes, and is exactly on par with incremental tabu search on all sizes between 54 to 100, except $n = 56$, 57, 64, 66, 77, and 96, where incremental tabu search leads by one edge. We observe that Wagner's cross entropy method [Wagner, 2021] performs much worse than both tabu search and AlphaZero—see Figure 4.

Benefits of using incremental search Without curriculum and incremental search, the agent must build graphs of a given size starting from the empty graph, while with incremental search, the agent starts from a previously-found graph of a smaller size and flips some edges to obtain a graph of the desired size. Figure 2 (left) illustrates, for sizes 54 to 100, the benefit of using a curriculum for tabu search, which increases significantly as the number of nodes increases. Figure 2 (right) shows a similar plot for AlphaZero: AlphaZero matches Incremental AlphaZero up to size around 30 but deteriorates afterwards. We believe the reasons are large episode lengths and the difficulties of exploration and credit assignment. We conclude that using incremental learning (and curriculum) is a vital ingredient for applying RL to this problem and presumably for any optimization problem with a substructure property and a huge state space.

Comparing representations in AlphaZero. We compare our novel representation, Pairformer, with ResNet [He *et al.*, 2016], which has been extensively used in literature, especially in environments where the observation is a matrix or an image. Since a graph can be naturally expressed as an adjacency matrix, we compare the ResNet architecture, which is oblivious to the graph structure, with the Pairformer architecture. We use a ResNet with 10 layers and 256 output channels. Figure 3 (left) shows the policy's cross-entropy loss during training, and Figure 3 (right) shows the average episode return. For this experiment, we focus on joint training for graph sizes [80, 100] with Incremental AlphaZero. We observe that Pairformer performs better on both metrics. Nevertheless, the final scores obtained by ResNet and Pairformer are equal except on sizes 80, 86 and 93, where Pairformer leads by one score point. It should be noted that the above experiments are for parameters sizes (number of layers, attention heads) beyond which we didn't see a performance improvement for either representation and the exact number of parameters may differ for both representations.

Comparing the cross-entropy method with incremental tabu search. We compare incremental tabu search with the cross-entropy method [Wagner, 2021] in Figure 4. (The hyperparameters are provided in the full version.) We observe that incremental tabu search outperforms cross-entropy method by a big margin as the size of the graph grows. (Incremental AlphaZero performs similarly to incremental tabu search, so we haven't plotted it.)

While it would have been useful to compare the resources used by each method, the different nature of the algorithms hinders a fair comparison, especially because some are sequential and others are parallel. AlphaZero has a distributed implementation with multiple actors. We ran tabu search for seven days to make a fair comparison to AlphaZero. Crucially, we ran all the methods until their results plateaued, and the cross-entropy method was run for large time frames for small sizes until no improvement was observed.

[1] In Figure 1, state-of-the-art lower bounds are from [Garnick *et al.*, 1993; Abajo *et al.*, 2010; Garnick, 2023; McKay, 2023] and theoretical upper bounds are from [Garnick *et al.*, 1993]. We have not compared against the lower bounds reported in [Bong, 2017], as neither the graphs achieving those bounds nor the method for generating them are presented in [Bong, 2017]; still, incremental tabu search improves over the lower bounds reported in [Bong, 2017] for $n \in \{64, \ldots, 76\}$.

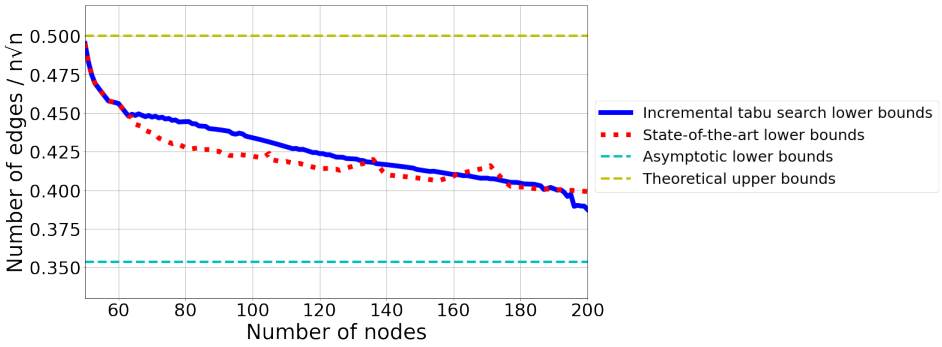

Figure 1: Normalized scores, given by $\frac{\text{number of edges}}{n\sqrt{n}}$, are plotted versus size, n. AlphaZero with curriculum (not plotted) achieves the same score as incremental tabu search for 41 of the sizes from 54 to 100. Erdős conjectured that both the red and blue curves converge to the cyan horizontal line as $n \to \infty$.

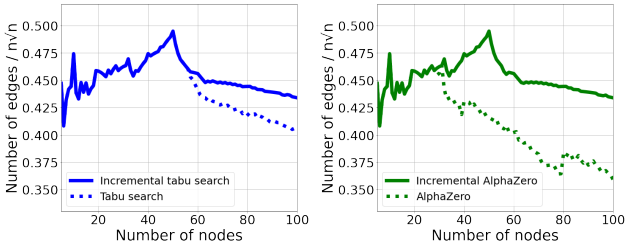

Figure 2: Left: Incremental tabu search, which uses a curriculum, performs increasingly better than tabu search without curriculum, for larger problem sizes. Right: Adding a curriculum improves the performance of AlphaZero significantly, especially on larger sizes.

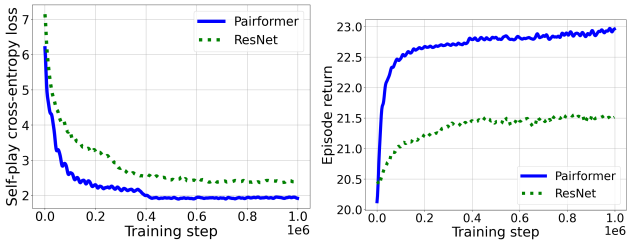

Figure 3: Left: The policy cross-entropy loss of Pairformer and ResNet during online training of AlphaZero on joint training for graph sizes [80, 100] with curriculum. Pairformer minimizes the loss faster as it captures invariances and other graph structures. Right: Average episode return of Pairformer and ResNet during training of AlphaZero using the edge-flipping environment on joint training for graph sizes [80, 100] with curriculum. In both plots, the average is taken over 3 seeds and Gaussian smoothing with $\sigma = 2$ is applied.

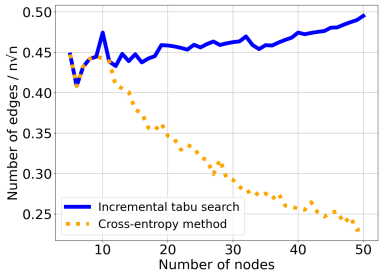

Figure 4: Incremental tabu search versus the cross-entropy method.

7 Discussion

We studied a challenging learning-to-search benchmark inspired by an open problem in extremal graph theory, compared a neural network-guided MCTS with tabu search, and observed that using a curriculum is crucial for improving the state of the art, but introducing learning did not yield improvements (a similar phenomenon was observed for another extremal graph theory problem [Parczyk *et al.*, 2023]). This could be because the problem has lots of local optima and the search space is hard to explore: for some sizes, there is only one feasible graph with the optimal score [Backelin, 2015], and during our experiments, for size $n = 96$, only one of our runs found a score of 411; all other runs found smaller scores. Also, in contrast to problems on which RL has improved the state of the art and do not have strong local search baselines (e.g., the tensor decomposition problem [Fawzi *et al.*, 2022]), for this problem, natural, fast, and strong local search algorithms exist. Finally, in contrast to typical RL problems, where the goal is to maximize the *expected return* in a non-deterministic environment, here we want to maximize the *best-case* return in a deterministic environment—i.e., we need only find a good solution once. So, it's unclear whether the classical RL objective is the right approach here; finding better objectives for learning-to-search is an open research problem.

Some of the ideas in this work—the curriculum, the edge-flipping environment, the novel representation Pairformer, and incremental local search—could be used in similar problems. In particular, our edge-flipping environment allows a flexible curriculum approach, where the high-scoring graph of the smaller size need not be an exact subgraph of the optimal larger graphs. This could prove useful in other mathematical problems that have a similar structure.

Finally, while we have brought the state-of-the-art ML algorithm AlphaZero on par with tabu search, the main improvement came from using an incremental approach. Hence one may ask: What improvements to ML approaches are required to outpace classical heuristics on search problems?

Contribution Statement

Abbas Mehrabian, Ankit Anand, and Hyunjik Kim contributed equally as joint first authors to the paper. Doina Precup and Adam Zsolt Wagner contributed equally as senior authors and are listed in alphabetical order. Tudor Berariu was doing an internship at Google DeepMind while this work was done.

Acknowledgements

We are grateful to Eser Aygün for advising us throughout this project and his helpful comments on the first draft of the paper. We also thank Brendan McKay for releasing the best-known graphs up to size 64, which helped us jump-start our incremental tabu search algorithm.

We appreciate several useful discussions with David Applegate, Charles Blundell, Alex Davies, Matthew Fahrbach, Alhussein Fawzi, Michael Figurnov, David Garnick, Xavier Glorot, Harris Kwong, Felix Lazebnik, Shibl Mourad, Sébastien Racaniere, Tara Thomas, and Theophane Weber.

Joonkyung Lee is supported by Samsung STF Grant SSTF-BA2201-02.

References

[Abajo *et al.*, 2010] E. Abajo, C. Balbuena, and A. Diánez. New families of graphs without short cycles and large size. *Discrete Appl. Math.*, 158(11):1127–1135, 2010.

[Abdelaziz *et al.*, 2022] Ibrahim Abdelaziz, Maxwell Crouse, Bassem Makni, Vernon Austil, Cristina Cornelio, Shajith Ikbal, Pavan Kapanipathi, Ndivhuwo Makondo, Kavitha Srinivas, Michael Witbrock, and Achille Fokoue. Learning to guide a saturation-based theorem prover. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2022.

[Agostinelli *et al.*, 2019] Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the Rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1, 07 2019.

[Akkaya *et al.*, 2019] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[Aygün *et al.*, 2022] Eser Aygün, Ankit Anand, Laurent Orseau, Xavier Glorot, Stephen M Mcaleer, Vlad Firoiu, Lei M Zhang, Doina Precup, and Shibl Mourad. Proving theorems using incremental learning and hindsight experience replay. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1198–1210. PMLR, 2022.

[Ba *et al.*, 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[Backelin, 2015] Jörgen Backelin. Sizes of the extremal girth 5 graphs of orders from 40 to 49. *arXiv preprint, arXiv:1511.08128v1 [math.CO]*, 2015.

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[Bong, 2017] Novi H. Bong. Some new upper bounds of ex(n;C3,C4). *AKCE International Journal of Graphs and Combinatorics*, 14(3):251–260, 2017.

[Brown, 1966] W. G. Brown. On graphs that do not contain a Thomsen graph. *Can. Math. Bull.*, 9:281–285, 1966.

[Davies *et al.*, 2021] 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, 2021.

[Erdős *et al.*, 1966] Paul Erdős, Alfréd Rényi, and Vera T. Sós. On a problem of graph theory. *Stud. Sci. Math. Hung.*, 1:215–235, 1966.

[Erdős, 1975] Paul Erdős. Some recent progress on extremal problems in graph theory. *Congr. Numer*, 14:3–14, 1975.

[Fawzi *et al.*, 2022] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 2022.

[Florensa *et al.*, 2017] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 482–495. PMLR, 13–15 Nov 2017.

[Garnick *et al.*, 1993] David K Garnick, YH Harris Kwong, and Felix Lazebnik. Extremal graphs without three-cycles or four-cycles. *Journal of Graph Theory*, 17(5):633–645, 1993.

[Garnick, 2023] David K Garnick. Extremal graphs without three-cycles or four-cycles. Personal communication, 2023.

[Glover, 1989] Fred Glover. Tabu search. I. *ORSA J. Comput.*, 1(3):190–206, 1989.

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

[Hosu and Rebedea, 2016] Ionel-Alexandru Hosu and Traian Rebedea. Playing Atari games with deep reinforcement learning and human checkpoint replay. *arXiv preprint arXiv:1607.05077*, 2016.

[Inc., 2023] OEIS Foundation Inc. Maximal number of edges in n-node graph of girth at least 5. Entry A006856 in the on-line encyclopedia of integer sequences. https://oeis.org/A006856, 2023. Accessed: September 13, 2023.

[Jumper *et al.*, 2021] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Kovács and Voronkov, 2013] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In *International Conference on Computer Aided Verification*, pages 1–35. Springer, 2013.

[Lample *et al.*, 2022] Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem proving. *Advances in Neural Information Processing Systems*, 35:26337–26349, 2022.

[Mankowitz *et al.*, 2023] Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.

[Mantel, 1907] Willem Mantel. Vraagstuk xxviii. *Wiskundige Opgaven met de Oplossingen*, 10(2):60–61, 1907.

[McKay, 2023] Brendan McKay. Extremal graphs and Turan numbers. https://users.cecs.anu.edu.au/~bdm/data/extremal.html, 2023. Accessed on 7 August 2023.

[Orseau *et al.*, 2023] Laurent Orseau, Marcus Hutter, and Levi H. S. Lelis. Levin tree search with context models. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 5622–5630. International Joint Conferences on Artificial Intelligence Organization, 2023.

[Parczyk *et al.*, 2023] Olaf Parczyk, Sebastian Pokutta, Christoph Spiegel, and Tibor Szabó. New Ramsey multiplicity bounds and search heuristics. *arXiv preprint, arXiv:2206.04036v2 [math.CO]*, 2023. Conference version in Proceedings of the AAAI Conference on Artificial Intelligence, 2023.

[Peng *et al.*, 2018] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018.

[Polu and Sutskever, 2020] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

[Salimans and Chen, 2018] Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

[Schulz, 2002] Stephan Schulz. E–a brainiac theorem prover. *AI Communications*, 15(2, 3):111–126, 2002.

[Silver *et al.*, 2017] 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, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140–1144, 2018.

[Soviany *et al.*, 2022] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *Int. J. Comput. Vision*, 130(6):1526–1565, jun 2022.

[Turán, 1941] Pál Turán. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Veličković *et al.*, 2020] Petar Veličković, Lars Buesing, Matthew Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. Pointer graph networks. *Advances in Neural Information Processing Systems*, 33:2232–2244, 2020.

[Veličković, 2023] Petar Veličković. Everything is connected: Graph neural networks. *arXiv preprint arXiv:2301.08210*, 2023.

[Wagner, 2021] Adam Zsolt Wagner. Constructions in combinatorics via neural networks. *Preprint, arXiv:2104.14516 [math.CO]*, 2021.