A Visual demonstration

Figure 1 provides a visual demonstration of the input and output of Erdős' GNN in a simple instance of the maximum clique problem.

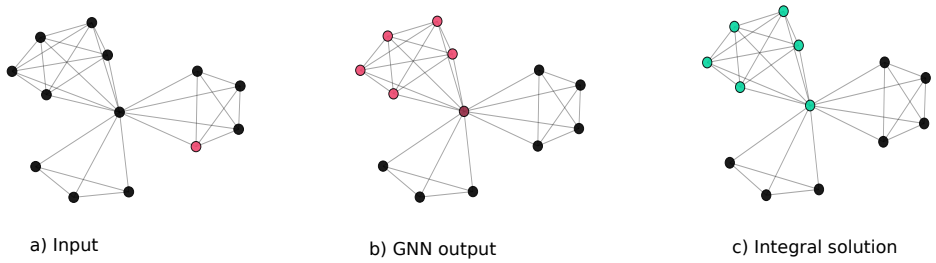

Figure 1: Illustration of our approach in a toy instance of the maximum clique problem from the IMDB dataset. a) A random node is selected to act as a 'seed'. b) Erdős' GNN outputs a probability distribution over the nodes (color intensity represents the probability magnitude) by exploring the graph in the vicinity of the seed. c) A set is sequentially decoded by starting from the node whose probability is the largest and iterating with the method of conditional expectation. The identified solution is guaranteed to obey the problem constraints, i.e., to be a clique.

We would like to make two observations. The first has to do with the role of the starting seed in the probability assignment produced by the network. In the maximum clique problem we did not require the starting seed to be included in the solutions. This allowed the network to flexibly detect maximum cliques within its receptive field without being overly constrained by the random seed selection. This is illustrated in the example provided in the figure, where the seed is located inside a smaller clique and yet the network is able to produce probabilities that focus on the largest clique. On the other hand, in the local graph partitioning problem we forced the seed to always lie in the identified solution—this was done to ensure a fair comparison with previous methods. Our second observation has to do with the sequential decoding process. It is encouraging to notice that, even though the central hub node has a considerably lower probability than the rest of the nodes in the maximum clique, the method of conditional expectation was able to reliably decode the full maximum clique.

B Experimental details

B.1 Datasets

The following table presents key statistics of the datasets that were used in this study:

	IMDB	COLLAB	TWITTER	RB (Train)	RB (Test)	RB (Large Inst.)	SF-295	FACEBOOK
nodes	19.77	74.49	131.76	216.673	217.44	1013.25	26.06	7252.71
edges	96.53	2457.78	1709.33	22852	22828	509988.2	28.08	276411.19
reduction time	0.0003	0.006	0.024	0.018	0.018	0.252	–	–
number of test graphs	200	1000	196	2000	500	40	8055	14

Table 4: Average number of nodes and edges for the considered datasets. Reduction time corresponds to the average number of seconds needed to reduce a maximum clique instance to a maximum independent instance. Number of test graphs refers to the number of graphs that the methods were evaluated on, in a given dataset.

To speed up computation and training, for the Facebook dataset, we kept graphs consisting of at most 15000 nodes (i.e., 70 out of the total 100 available graphs of the dataset).

The RB test set can be downloaded from the following link: `https://www.dropbox.com/s/9bdq1y69dw1q77q/cliques_test_set_solved.p?dl=0`. The latter was generated using the procedure described by Xu [75]. We used a python implementation by Toenshoff et al. [65] that is available on the RUN-CSP repository: `https://github.com/RUNCSP/RUN-CSP/blob/master/`

`generate_xu_instances.py`. Since the parameters of the original training set were not available, we selected a set of initial parameters such that the generated dataset resembles the original training set. As seen in Table 4, the properties of the generated test set are close to those of the training set. Specifically, the training set contained graphs whose size varied between 50 and 500 nodes and featured cliques of size 5 to 25. The test set was made out of graphs whose size was between 50 and 475 nodes and contained cliques of size 10 to 25. These minor differences provide a possible explanation for the drop in test performance of all methods (larger cliques tend to be harder to find).

All other datasets are publicly available.

B.2 Neural network architecture

In both problems, Erdős' GNN and our own neural baselines were given as node features a one-hot encoding of a random node from the input graph. For the local graph partitioning setting, our networks consisted of 6 GIN layers followed by a multi-head GAT layer. This depth is consistent across all datasets. We employed skip connections and batch-normalization at every layer. For the maximum clique problem, we also incorporated graph size normalization for each convolution that we found to improve optimization stability. The networks in this setting did not use a GAT layer, as we found that multi-head GAT had a significant impact on the speed/memory of the network without any significant benefits in accuracy to match that cost. Furthermore, locality was enforced after each layer by masking the receptive field. That is, after 1 layer of convolution only 1-hop neighbors were allowed to have nonzero values, after 2 layers only 2-hop neighbors could have nonzero values, etc. The output of the final GNN layer was passed through a two layer perceptron giving as output one value per node. The aforementioned numbers were re-scaled to lie in $[0, 1]$ (using a graph-wide min-max normalization) and were interpreted as probabilities $p_1, \ldots, p_n$. In the case of local graph partitioning, the forward-pass was concluded by the appropriate re-scaling of the probabilities (as described in Section 3.1.3).

B.3 Local graph partitioning setup

Following the convention of local graph clustering algorithms, for each graph in the test set we randomly selected d nodes of the input graph to act as cluster *seeds*, where $d = 10, 30$, and 100 for SF-295, TWITTER, and FACEBOOK, respectively. Each method was run once for each seed resulting in d sets per graph. We obtained one number per seed by averaging the conductances of the graphs. Table 3 reports the mean and standard deviation of these numbers.[1]

The volume-constrained graph partitioning formulation can be used to minimize conductance as follows: Perform grid search over the range of feasible volumes and create a small interval around each target volume. Then, solve a volume-constrained partitioning problem for each interval, and return the set of smallest conductance identified.

We used a fast and randomized variant of the above procedure with all neural approaches and Gurobi (see Section C.2 for more details). Specifically, for each seed node we generated a random volume interval within the receptive field of the network, and solved the corresponding constrained partitioning problem. Our construction ensured that the returned sets always contained the seed node and had a controlled volume. For L1 and L2 GNN, we obtained the set by sampling from the output distribution. We drew 10 samples and kept the best. We found that in contrast to flat thresholding (like in the maximum clique), sampling yielded better results in this case.

For the parameter search of local graph clustering methods, we found the best performing parameters on a validation set via grid search when that was appropriate. For CRD, we searched for all the integer values in the [1,20] interval for all 3 of the main parameters of the algorithm. For Simple Local, we searched in the [0,1] interval for the locality parameter. Finally, for Pagerank-Nibble we set a lower bound on the volume that is 10 % of the total graph volume. It should be noted, that while local graph clustering methods achieved inferior conductance results, they do not require explicit specification of a receptive field which renders them more flexible.

[1] Please note that the caption of Table 3 incorrectly reports that $d = 25$ seeds were used for the TWITTER dataset and that the best conductance was kept for each graph. The correct procedure is the one described here.

B.4 Hardware and software

All methods were run on an Intel Xeon Silver 4114 CPU, with 192GB of available RAM. The neural networks were executed on a single RTX TITAN 25GB graphics card. The code is executed on version 1.1.0 of PyTorch, and version 1.2.0 of PyTorch Geometric.

B.5 Pre-trained Models

Pre-trained models of Erdős' GNN for the maximum clique and the constrained minimum cut respectively can be downloaded from the following links:
`https://www.dropbox.com/sh/mdsjrcg9gch8dti/AADW0UUcQMUkChz8SZNXYGnVa?dl=0`
`https://www.dropbox.com/sh/z00ictftyxx3ipf/AADirtiMIwI3_sxCep5GzJf_a?dl=0`

C Additional results

C.1 Maximum clique problem

The following experiments provide evidence that both the learning and decoding phases of our framework are important in obtaining valid cliques of large size.

C.1.1 Constraint violation

Table 5 reports the percentage of instances in which the clique constraint was violated in our experiments. Neural baselines optimized according to penalized continuous relaxations struggle to detect cliques in the COLLAB and TWITTER datasets, whereas Erdős' GNN always respected the constraint.

	IMDB	COLLAB	TWITTER	RB (all datasets)
Erdős' GNN (fast)	**0%**	**0%**	**0%**	**0%**
Erdős' GNN (accurate)	**0%**	**0%**	**0%**	**0%**
Bomze GNN	**0%**	11.8%	78.1%	–
MS GNN	1%	15.1%	84.7%	–

Table 5: Percentage of test instances where the clique constraint was violated.

Thus, decoding solutions by the method of conditional expectation is crucial to ensure that the clique constraint is always satisfied.

C.1.2 Importance of learning

We also tested the efficacy of the learned probability distributions produced by our GNN on the Twitter dataset. We sampled multiple random seeds and produced the corresponding probability assignments by feeding the inputs to the GNN. These were then decoded with the method of conditional expectation and the best solution was kept. To measure the contribution of the GNN, we compared to random uniform probability assignments on the nodes. In that case, instead of multiple random seeds, we had the same number of multiple random uniform probability assignments. Again, these were decoded with the method of conditional expectation and the best solution was kept. The results of the experiment can be found in Table 6.

	Erdős' GNN	U $\sim$ [0,1]
1 sample	0.821 ± 0.222	0.513 ± 0.266
3 samples	0.875 ± 0.170	0.694 ± 0.210
5 samples	0.905 ± 0.139	0.760 ± 0.172

Table 6: Approximation ratios with sequential decoding using the method of conditional expectation on the twitter dataset. The second column represents decoding with the probabilities produced by the GNN. The third column shows the results achieved by decoding random uniform probability assignments on the nodes.

As observed, the cliques identified by the trained GNN were significantly larger than those obtained when decoding a clique from a random probability assignment.

C.2 Local graph partitioning

We also attempted to find sets of small conductance using Gurobi. To ensure a fair comparison, we mimicked the setting of Erdős' GNN and re-run the solver with three different time-budgets, making sure that the largest budget exceeded our method's running time by approximately one order of magnitude. We used the following integer-programming formulation of the constrained graph partitioning problem:

$$\min_{x_1,\dots,x_n\in\{0,1\}} \sum_{(v_i,v_j)\in E} (x_i - x_j)^2 \tag{6}$$
$$\text{subject to} \quad \left(1-\frac{1}{4}\right)\text{vol} \leq \sum_{v_i\in V} x_i d_i \leq \left(1+\frac{1}{4}\right)\text{vol} \quad \text{and} \quad x_s = 1.$$

Above, vol is a target volume and s is the index of the seed node (see explanation in Section B.3). Each binary variable x_i is used to indicate membership in the solution set. In order to encourage local solutions on a global solver like Gurobi, the generated target volumes were set to lie in an interval that is attainable within a fixed receptive field (identically to the neural baselines). Additionally, the seed node v_s was also required to be included in the solution. The above choices are consistent with the neural baselines and the local graph partitioning setting.

The results are shown in Table 7. Due to its high computational complexity, Gurobi performed poorly in all but the smallest instances. In the FACEBOOK dataset, which contains graphs of 7k nodes on average, Erdős' GNN was impressively able to find sets of more than 6× smaller conductance, while also being 6× faster.

	SF-295	FACEBOOK	TWITTER
Gurobi (0.1s)	0.107 ± 0.000 (0.16 s/g)	0.972 ± 0.000 (799.508 s/g)	0.617 ± 0.012 (3.88 s/g)
Gurobi (1s)	0.106 ± 0.000 (0.16 s/g)	0.972 ± 0.000 (893.907 s/g)	0.544 ± 0.007 (12.41 s/g)
Gurobi (10s)	**0.105 ± 0.000 (0.16 s/g)**	0.961 ± 0.010 (1787.79 s/g)	0.535 ± 0.006 (52.98 s/g)
Erdős' GNN	0.124 ± 0.001 (0.22 s/g)	**0.156 ± 0.026 (289.28 s/g)**	**0.292 ± 0.009 (6.17 s/g)**

Table 7: Average conductance of sets identified by Gurobi and Erdős' GNN (these results are supplementary to those of Table 3).

It should be noted that the time budget allowed for Gurobi only pertains to the *optimization time* spent (for every seed). There are additional costs in constructing the problem instances and their constraints for each graph. These costs become particularly pronounced in larger graphs, where setting up the problem instance takes more time than the allocated optimization budget. We report the total time cost in seconds per graph (s/g).

D Deferred technical arguments

D.1 Proof of Theorem 1

In the constrained case, the focus is on the probability $P(\{f(S;G) < \epsilon\} \cap \{S \in \Omega\})$. Define the following probabilistic penalty function:

$$f_p(S;G) = f(S;G) + \mathbf{1}_{S\notin\Omega}\,\beta, \tag{7}$$

where β is any number larger than $\max_S\{f(S;G)\}$. The key observation is that, if $\ell(\mathcal{D},G) = \epsilon < \beta$, then there must exist a valid solution of cost ϵ. It is a consequence of $f(S;G) > 0$ and β being an upper bound of f that

$$P(f_p(S;G) < \epsilon) = P(f(S;G) < \epsilon \cap S \in \Omega). \tag{8}$$

Similar to the unconstrained case, for a non-negative f, Markov's inequality can be utilized to bound this probability:

$$\begin{aligned} P(\{f(S;G) < \epsilon\} \cap \{S \in \Omega\}) &= P(f_p(S;G) < \epsilon) \\ &> 1 - \frac{1}{\epsilon}\mathbb{E}\left[f_p(S;G)\right] \\ &= 1 - \frac{1}{\epsilon}\left(\mathbb{E}\left[f(S;G)\right] + \mathbb{E}\left[\mathbf{1}_{S\notin\Omega}\,\beta\right]\right) \\ &= 1 - \frac{1}{\epsilon}\left(\mathbb{E}\left[f(S;G)\right] + P(S \notin \Omega)\,\beta\right). \end{aligned} \tag{9}$$

The theorem claim follows from the final inequality.

D.2 Iterative scheme for non-linear re-scaling

Denote by $\mathcal{D}^0$ the distribution of sets predicted by the neural network and let $p_1^0, \ldots, p_n^0$ be the probabilities that parameterize it. We aim to re-scale these probabilities such that the constraint is satisfied in expectation:

$$\sum_{v_i \in V} a_i p_i = \frac{b_l + b_h}{2}, \quad \text{where} \quad p_i = \text{clamp}\big(c\, p_i^0, 0, 1\big) \quad \text{and} \quad c \in \mathbb{R}.$$

This can be achieved by iteratively applying the following recursion:

$$p_i^{\tau+1} \leftarrow \text{clamp}(c^\tau p_i^\tau, 0, 1), \quad \text{with} \quad c^\tau = \frac{b - \sum_{v_i \in Q^\tau} a_i}{\sum_{v_i \in V \setminus Q^\tau} a_i p_i^\tau} \quad \text{and} \quad Q^\tau = \{v_i \in V \,:\, p_i^\tau = 1\},$$

where $b = \frac{b_l + b_h}{2}$.

The fact that convergence occurs can be easily deduced. Specifically, consider any iteration τ and let Q^τ be as above. If $p_i^{\tau+1} < 1$ for all $v_i \in V \setminus Q^\tau$, then the iteration has converged. Otherwise, we will have $Q^\tau \subset Q^{\tau+1}$. From the latter, it follows that in every τ (but the last), set Q^τ must expand until either $\text{clamp}(c^\tau p_i^\tau, 0, 1) = b$ or $Q^\tau = V$. The latter scenario will occur if $\sum_{v_i \in V} a_i \leq b$.

D.3 Proof of Theorem 2

Set $b = (b_l + b_h)/2$ and $\delta = (b_h - b_l)/2$. By Hoeffding's inequality, the probability that a sample of $\mathcal{D}$ will lie in the correct interval is:

$$P\left(\left|\sum_{v_i \in S} a_i - \mathbb{E}\left[\sum_{v_i \in S} a_i\right]\right| \leq \delta\right) = P\left(\left|\sum_{v_i \in S} a_i - b\right| \leq \delta\right) \geq 1 - 2\exp\left(-\frac{2\delta^2}{\sum_i a_i^2}\right).$$

We can combine this guarantee with the unconstrained guarantee by taking a union bound over the two events:

$$\begin{aligned} &P\left(f(S;G) < \ell(\mathcal{D},G) \text{ AND } \sum_{v_i \in S} a_i \in [b_l, b_h]\right) \\ &\qquad = 1 - P\left(f(S;G) \geq \ell(\mathcal{D},G) \text{ OR } \sum_{v_i \in S} a_i \notin [b_l, b_h]\right) \\ &\qquad \geq 1 - P\left(f(S;G) \geq \ell(\mathcal{D},G)\right) - P\left(\sum_{v_i \in S} a_i \notin [b_l, b_h]\right) \\ &\qquad \geq t - 2\exp\left(-\frac{2\delta^2}{\sum_i a_i^2}\right) \end{aligned}$$

The previous is positive whenever $t > 2\exp\left(-2\delta^2/(\sum_i a_i^2)\right)$.

D.3.1 Proof of Corollary 1

To ensure that the loss function is non-negative, we will work with the translated objective function $f(S;G) = \gamma - w(S)$, where the term γ is any upper bound of $w(S)$ for all S.

Theorem 1 guarantees that if

$$\mathbb{E}\left[f(S;G)\right] + P(S \notin \Omega)\,\beta \leq \ell_{\text{clique}}(\mathcal{D}, G) \leq \epsilon \tag{10}$$

and as long as $\max_S f(S;G) = \gamma - \min_S w(S) \leq \gamma \leq \beta$, then with probability at least t, set $S^* \sim \mathcal{D}$ satisfies $\gamma - \epsilon/(1-t) < w(S^*)$.

Denote by x_i a Bernoulli random variable with probability p_i. It is not difficult to see that

$$\mathbb{E}\left[w(S)\right] = \mathbb{E}\left[\sum_{(v_i,v_j)\in E} w_{ij}x_ix_j\right] = \sum_{(v_i,v_j)\in E} w_{ij}p_ip_j \tag{11}$$

We proceed to bound $P(S \notin \Omega_{\text{clique}})$. Without loss of generality, suppose that the edge weights have been normalized to lie in $[0,1]$. We define $\bar{w}(S)$ to be the volume of S on the complement graph:

$$\bar{w}(S) \triangleq \sum_{v_i,v_j\in S} \{(v_i,v_j) \notin E\}$$

By definition, we have that $P\left(S \notin \Omega_{\text{clique}}\right) = P\left(\bar{w}(S) \geq 1\right)$. Markov's inequality then yields

$$\begin{aligned}
P\left(S \notin \Omega_{\text{clique}}\right) &\leq \mathbb{E}\left[\bar{w}(S)\right] = \mathbb{E}\left[\frac{|S|(|S|-1)}{2}\right] - \mathbb{E}\left[w(S)\right] \\
&= \frac{1}{2}\mathbb{E}\left[\left(\sum_{v_i\in V} x_i\right)^2 - \sum_{v_i\in V} x_i\right] - \mathbb{E}\left[w(S)\right] \\
&= \frac{1}{2}\sum_{v_i\neq v_j} \mathbb{E}\left[x_ix_j\right] + \frac{1}{2}\sum_{v_i\in V} \mathbb{E}\left[x_i^2\right] - \sum_{v_i\in V} \mathbb{E}\left[x_i\right] - \frac{1}{2}\mathbb{E}\left[w(S)\right] \\
&= \frac{1}{2}\sum_{v_i\neq v_j} p_ip_j + \frac{1}{2}\sum_{v_i\in V} p_i - \frac{1}{2}\sum_{v_i\in V} p_i - \mathbb{E}\left[w(S)\right] = \frac{1}{2}\sum_{v_i\neq v_j} p_ip_j - \mathbb{E}\left[w(S)\right].
\end{aligned} \tag{12}$$

It follows from the above derivations that

$$\begin{aligned}
\gamma - \mathbb{E}\left[w(S)\right] + P(S \notin \Omega)\,\beta &\leq \gamma - \mathbb{E}\left[w(S)\right] + \frac{\beta}{2}\sum_{v_i\neq v_j} p_ip_j - \beta\mathbb{E}\left[w(S)\right] \\
&= \gamma - (1+\beta)\mathbb{E}\left[w(S)\right] + \frac{\beta}{2}\sum_{v_i\neq v_j} p_ip_j \\
&= \gamma - (1+\beta)\sum_{(v_i,v_j)\in E} w_{ij}p_ip_j + \frac{\beta}{2}\sum_{v_i\neq v_j} p_ip_j.
\end{aligned} \tag{13}$$

The final expression is exactly the probabilistic loss function for the maximum clique problem.

D.4 Proof of Corollary 2

Denote by S the set of nodes belonging to the cut, defined as $S = \{v_i \in V, \text{ such that } x_i = 1\}$. Our first step is to re-scale the probabilities such that, in expectation, the following is satisfied:

$$\mathbb{E}\left[\text{vol}(S)\right] = \frac{v_l + v_h}{2}.$$

This can be achieved by noting that the expected volume is

$$\mathbb{E}\left[\text{vol}(S)\right] = \mathbb{E}\left[\sum_{v_i\in V} d_ix_i\right] = \sum_{v_i\in V} d_ip_i$$

and then using the procedure described in Section D.2.

With the probabilities $p_1, \ldots, p_n$ re-scaled, we proceed to derive the probabilistic loss function corresponding to the min cut.

The cut of a set $S \sim \mathcal{D}$ can be expressed as

$$\text{cut}(S) = \sum_{v_i \in S} \sum_{v_j \notin S} w_{ij} = \sum_{(v_i, v_j) \in E} w_{ij} z_{ij}, \tag{14}$$

where z_{ij} is a Bernoulli random variable with probability p_i which is equal to one if exactly one of the nodes v_i, v_j lies within set S. Formally,

$$z_{ij} = |x_i - x_j| = \begin{cases} 1 & \text{with probability } p_i - 2p_i p_j + p_j \\ 0 & \text{with probability } 2p_i p_j - (p_i + p_j) + 1 \end{cases} \tag{15}$$

It follows that the expected cut is given by

$$\begin{aligned} \mathbb{E}\left[\text{cut}(S)\right] &= \sum_{(v_i, v_j) \in E} w_{ij} \, \mathbb{E}\left[z_{ij}\right] \\ &= \sum_{(v_i, v_j) \in E} w_{ij}(p_i - 2p_i p_j + p_j) \\ &= \sum_{(v_i, v_j) \in E} w_{ij}(p_i + p_j) - 2 \sum_{(v_i, v_j) \in E} p_i p_j w_{ij} = \sum_{v_i \in V} d_i p_i - 2 \sum_{(v_i, v_j) \in E} p_i p_j w_{ij}. \end{aligned}$$

We define, accordingly, the min-cut probabilistic loss as

$$\ell_{\text{cut}}(\mathcal{D}; G) = \sum_{v_i \in V} d_i p_i - 2 \sum_{(v_i, v_j) \in E} p_i p_j w_{ij}$$

Then, for any $t \in (0, 1]$, Markov's inequality yields:

$$P\left(\text{cut}(S) < \frac{\ell_{\text{cut}}(\mathcal{D}; G)}{1 - t}\right) > t$$

The proof then concludes by invoking Theorem 2

References

[1] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. 2018.

[2] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Pdp: A general neural framework for learning constraint satisfaction solvers, 2019.

[3] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006.

[4] Konstantin Andreev and Harald Racke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6): 929–939, 2006.

[5] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Graph edit distance computation via graph neural networks. *arXiv preprint arXiv:1808.05689*, 2018.

[6] Yunsheng Bai, Derek Xu, Alex Wang, Ken Gu, Xueqing Wu, Agustin Marinovic, Christopher Ro, Yizhou Sun, and Wei Wang. Fast detection of maximum common subgraph via deep q-learning. *arXiv preprint arXiv:2002.03129*, 2020.

[7] Pablo Barceló, Egor V Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2019.

[8] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[9] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Mincut pooling in graph neural networks, 2019.

[10] Immanuel M Bomze. Evolution towards the maximum clique. *Journal of Global Optimization*, 10(2): 143–164, 1997.

[11] Immanuel M Bomze, Marco Budinich, Panos M Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[12] Ravi Boppana and Magnús M Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196, 1992.

[13] Xavier Bresson, Thomas Laurent, David Uminsky, and James Von Brecht. Multiclass total variation clustering. In *Advances in Neural Information Processing Systems*, pages 1421–1429, 2013.

[14] Maurizio Bruglieri, Francesco Maffioli, and Matthias Ehrgott. Cardinality constrained minimum cut problems: complexity and algorithms. *Discrete Applied Mathematics*, 137(3):311–341, 2004.

[15] Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, pages 6278–6289, 2019.

[16] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

[17] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[18] Jean-Baptiste Cordonnier and Andreas Loukas. Extrapolating paths with graph neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2187–2194. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/303. URL `https://doi.org/10.24963/ijcai.2019/303`.

[19] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 170–181. Springer, 2018.

[20] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[21] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[22] Kimon Fountoulakis, David F Gleich, and Michael W Mahoney. A short introduction to local graph clustering methods and software. *arXiv preprint arXiv:1810.07324*, 2018.

[23] Lei Gao, Mingxiang Chen, Qichang Chen, Ganzhong Luo, Nuoyi Zhu, and Zhixin Liu. Learn to design the heuristics for vehicle routing problem. *arXiv preprint arXiv:2002.08539*, 2020.

[24] Vikas K Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. *arXiv preprint arXiv:2002.06157*, 2020.

[25] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *arXiv preprint arXiv:1906.01629*, 2019.

[26] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[27] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020. URL `http://www.gurobi.com`.

[28] John J Hopfield and David W Tank. "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

[29] Alex Irpan. Deep reinforcement learning doesn't work yet. `https://www.alexirpan.com/2018/02/14/rl-hard.html`, 2018.

[30] Rishabh Iyer, Stefanie Jegelka, and Jeff Bilmes. Fast semidifferential-based submodular function optimization: Extended version. In *ICML*, 2013.

[31] JQ James, Wen Yu, and Jiatao Gu. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3806–3817, 2019.

[32] johnjforrest, Stefan Vigerske, Haroldo Gambini Santos, Ted Ralphs, Lou Hafer, Bjarni Kristjansson, jpfasano, EdwinStraver, Miles Lubin, rlougee, jpgoncal1, h-i gassmann, and Matthew Saltzman. coin-or/cbc: Version 2.10.5, March 2020. URL `https://doi.org/10.5281/zenodo.3700700`.

[33] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

[34] Chaitanya K. Joshi, Thomas Laurent, and Xavier Bresson. On learning paradigms for the travelling salesman problem, 2019.

[35] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2020. URL `http://www.graphlearning.io/`.

[36] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

[37] Elias Boutros Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[40] Kevin Lang and Satish Rao. A flow-based method for improving the expansion or conductance of graph cuts. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 325–337. Springer, 2004.

[41] Henrique Lemos, Marcelo Prates, Pedro Avelar, and Luis Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems, 2019.

[42] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[43] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*, pages 539–548, 2018.

[44] Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1l2bp4YwS`.

[45] Andreas Loukas. How hard is graph isomorphism for graph neural networks? *arXiv preprint arXiv:2005.06649*, 2020.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[47] Theodore S Motzkin and Ernst G Straus. Maxima for graphs and a new proof of a theorem of turán. *Canadian Journal of Mathematics*, 17:533–540, 1965.

[48] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, pages 9839–9849, 2018.

[49] Evgenii Nikishin, Pavel Izmailov, Ben Athiwaratkun, Dmitrii Podoprikhin, Timur Garipov, Pavel Shvechikov, Dmitry Vetrov, and Andrew Gordon Wilson. Improving stability in deep reinforcement learning with weight averaging. In *Uncertainty in Artificial Intelligence Workshop on Uncertainty in Deep Learning*, volume 5, 2018.

[50] Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. *stat*, 1050:22, 2017.

[51] Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In *Advances in Neural Information Processing Systems*, pages 3368–3378, 2018.

[52] Bo Peng, Jiahai Wang, and Zizhen Zhang. A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems. In *International Symposium on Intelligence Computation and Applications*, pages 636–650. Springer, 2019.

[53] Marcelo Prates, Pedro HC Avelar, Henrique Lemos, Luis C Lamb, and Moshe Y Vardi. Learning to solve np-complete problems: A graph neural network for decision tsp. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4731–4738, 2019.

[54] Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988.

[55] Ryoma Sato. A survey on the expressive power of graph neural networks, 2020.

[56] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems, 2019.

[57] 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.

[58] Daniel Selsam and Nikolaj Bjørner. Guiding high-performance sat solvers with unsat-core predictions. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 336–353. Springer, 2019.

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

[60] Younjoo Seo, Andreas Loukas, and Nathanaël Perraudin. Discriminative structural graph classification, 2019.

[61] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[62] Kate A Smith. Neural networks for combinatorial optimization: a review of more than a decade of research. *INFORMS Journal on Computing*, 11(1):15–34, 1999.

[63] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6):1715–1737, 2011.

[64] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[65] Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Run-csp: Unsupervised learning of message passing networks for binary constraint satisfaction problems. *arXiv preprint arXiv:1909.08387*, 2019.

[66] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

[67] David E Van den Bout and TK Miller. Improving the performance of the hopfield-tank neural network through normalization and annealing. *Biological cybernetics*, 62(2):129–139, 1989.

[68] Nate Veldt, David F. Gleich, and Michael W. Mahoney. A simple and strongly-local flow-based method for cut improvement. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1938–1947. JMLR.org, 2016.

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

[70] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[71] 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.

[72] Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *arXiv preprint arXiv:1912.02175*, 2019.

[73] Di Wang, Kimon Fountoulakis, Monika Henzinger, Michael W Mahoney, and Satish Rao. Capacity releasing diffusion for speed and locality. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3598–3607. JMLR. org, 2017.

[74] Po-Wei Wang, Priya L Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. *arXiv preprint arXiv:1905.12149*, 2019.

[75] K BHOSLIB Xu. Benchmarks with hidden optimum solutions for graph problems. *URL http://www. nlsde. buaa. edu. cn/kexu/benchmarks/graph-benchmarks. htm*, 2007.

[76] Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artificial intelligence*, 171(8-9):514–534, 2007.

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

[78] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

[79] Xifeng Yan, Hong Cheng, Jiawei Han, and Philip S Yu. Mining significant graph patterns by leap search. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 433–444, 2008.

[80] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.

[81] Weichi Yao, Afonso S Bandeira, and Soledad Villar. Experimental performance of graph neural networks on random instances of max-cut. In *Wavelets and Sparsity XVIII*, volume 11138, page 111380S. International Society for Optics and Photonics, 2019.

[82] Gal Yehuda, Moshe Gabel, and Assaf Schuster. It's not what machines can learn, it's what we cannot teach. *arXiv preprint arXiv:2002.09398*, 2020.

[83] Emre Yolcu and Barnabas Poczos. Learning local search heuristics for boolean satisfiability. In *Advances in Neural Information Processing Systems*, pages 7990–8001, 2019.