Appendix

Checklist

1. Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
2. Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
3. Did you discuss any potential negative societal impacts of your work? [No]
4. Did you describe the limitations of your work? [No]
5. Did you state the full set of assumptions of all theoretical results? [Yes]
6. Did you include complete proofs of all theoretical results? [Yes]
7. Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] A link to the code is provided in Appendix A.
8. Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
9. Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report the standard deviation of our model.
10. Did you include the amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]

A Implementation of ScatteringClique Model

Our code can be found at
`https://github.com/yimengmin/GeometricScatteringMaximalClique`.

B Dataset Statistics

Table 3: Dataset statistics for the discussed datasets.

Dataset	# Graphs	Avg. # Nodes	Avg. # Edges
IMDB	$1,000$	19.8 ± 10.0	96.5 ± 105.6
COLLAB	$5,000$	74.5 ± 62.3	2457.2 ± 6438.9
TWITTER	973	131.8 ± 64.4	1709.3 ± 1559.7
SMALL-easy	1,000	209.0 ± 49.9	2293.6 ± 1147.8
SMALL-medium	1,000	187.6 ± 62.7	3378.2 ± 2077.9
SMALL-hard	1,000	181.3 ± 62.2	4165.3 ± 2780.2
LARGE-easy	1,000	1346.7 ± 222.9	48631.9 ± 12327.7
LARGE-medium	1,000	1334.7 ± 207.7	79689.8 ± 22527.2
LARGE-hard	1,000	1324.5 ± 209.0	105939.9 ± 32414.7

C Further Discussion of Loss Function

Here, we further discuss the relationship between our loss function (Eq. 10) and the loss function proposed in Karalias and Loukas [2020]. Their loss function is stated in Corollary 1 of their paper and, using our notations, reads

$$L(\mathbf{p}) = \gamma - (\beta' + 1) \sum_{(u,v) \in E} w_{u,v}\, \mathbf{p}_u\, \mathbf{p}_v + \frac{\beta'}{2} \sum_{u \neq v} \mathbf{p}_u\, \mathbf{p}_v, \tag{11}$$

Table 4: MC test approx. score (mean $\pm$ std.) and avg. prediction time measured in seconds per graph (in brackets) on SMALL dataset for our model compared to the different baselines. In our decoder, we set κ equal to 1, 1 and 10 for the easy, medium and hard subclass, respectively. As explained in Sec. 4.3, the inference time of Gurobi can exceed the indicated time budget.

Dataset	SMALL-easy	SMALL-medium	SMALL-hard
ScatteringClique	0.993 $\pm$0.017 (0.050)	0.935 $\pm$ 0.102 (0.021)	0.846 $\pm$ 0.177 (0.184)
GCN (low-pass)	0.951 $\pm$0.010 (0.042)	0.873 $\pm$ 0.145 (0.019)	0.776 $\pm$ 0.194 (0.156)
Erdős (fast)	0.954 $\pm$ 0.087 (0.28)	0.826 $\pm$ 0.135 (0.31)	0.734 $\pm$ 0.128 (0.39)
Erdős (accurate)	0.977 $\pm$ 0.053 (0.47)	0.899 $\pm$ 0.103 (0.62)	0.784 $\pm$ 0.163 (0.59)
RUN-CSP (fast)	0.948 $\pm$ 0.031 (0.34)	0.849 $\pm$ 0.137 (0.35)	0.753 $\pm$ 0.175 (0.51)
RUN-CSP (acc.)	0.967 $\pm$ 0.079 (0.72)	0.883 $\pm$ 0.069 (0.69)	0.787 $\pm$ 0.136 (0.89)
Gurobi 9.0 (0.1s)	0.991 $\pm$ 0.000 (0.42)	0.806 $\pm$ 0.101 (0.41)	0.742 $\pm$ 0.258 (0.42)
Gurobi 9.0 (0.2s)	0.998 $\pm$ 0.000 (0.49)	0.878 $\pm$ 0.035 (0.51)	0.843 $\pm$ 0.019 (0.48)
Gurobi 9.0 (0.5s)	1.000 $\pm$ 0.000 (0.55)	0.977 $\pm$ 0.015 (0.68)	0.962 $\pm$ 0.085 (0.60)
Gurobi 9.0 (1s)	1.000 $\pm$ 0.000 (0.65)	0.991$\pm$ 0.043 (0.91)	0.989 $\pm$ 0.047 (0.70)
$\mathrm{Heuristic}(5, 10)$	0.710$\pm$ 0.147 (0.19)	0.657 $\pm$ 0.210 (0.15)	0.559 $\pm$ 0.206 (0.13)
$\mathrm{Heuristic}(5, 20)$	0.995$\pm$ 0.046 (0.33)	0.829 $\pm$ 0.166 (0.31)	0.772 $\pm$ 0.171 (0.27)
$\mathrm{Heuristic}(5, 30)$	0.996$\pm$ 0.037 (0.46)	0.879 $\pm$ 0.146 (0.44)	0.841 $\pm$ 0.163 (0.39)
$\mathrm{Heuristic}(20, 20)$	0.995 $\pm$ 0.045 (1.59)	0.902 $\pm$ 0.135 (1.19)	0.849 $\pm$ 0.159 (1.06)
$\mathrm{Heuristic}(1, 100)$	0.952 $\pm$ 0.098 (0.11)	0.755 $\pm$ 0.198 (0.11)	0.689 $\pm$ 0.201 (0.14)
$\mathrm{Heuristic}(5, 100)$	0.996 $\pm$ 0.036 (0.54)	0.888 $\pm$ 0.145 (0.52)	0.891 $\pm$ 0.142 (0.67)

with positive constants $\gamma, \beta' \geq 0$. Setting $\gamma = 0$, $\beta' = 1$ and then dividing Eq. 11 by 2 corresponds to our loss function (Eq. 10) with the hyperparameter $\beta = 1/4$. We note that our loss function does not satisfy Corollary 1 from Karalias and Loukas [2020], which provides a lower bound on the probability of finding a clique of at least a certain size. Details can be found in the original paper. However, our loss function has proven viable in practice and, in conjunction with our hybrid GNN framework and decoder, is able to outperform the approach from Karalias and Loukas [2020] in both accuracy and inference time.

D Additional Results on Small Graphs

We now present results on the proposed SMALL dataset where graphs were generated following the approach in Xu et al. [2005]. Notably, in contrast to the LARGE dataset discussed in Sec. 4 of the main paper, we now have access to the ground truth MC, enabling a more precise evaluation of the performance. The results are presented in Table 4.

With respect to approximation score, our model clearly outperforms all other neural network approaches across all three difficulties. The closest competing method, i.e., Erdős (accurate) on easy and medium; RUN-CSP (acc.) on hard, is outperformed by 1.6, 3.6 and 5.9 percentage points, respectively, suggesting our method holds up particularly well for increasing difficulties of the task. When given sufficiently large time budgets, Gurobi can achieve higher approximation scores, however at a significantly higher time cost. For instance, Gurobi 9.0 (0.5s) takes 32 times longer for the medium difficulty graphs. The heuristic [Grosso et al., 2008] also performs well, outperforming our model for the easy and hard samples, while on the other hand requiring more than three times the inference time.

E Additional Figures

In this Section, we provide additional figures, analogous to Figure 3 of the main text which compare the probability vector $\mathbf{p}$ output by the hybrid scattering model to the vector output by a pure low-pass model. As with Figure 3, these figures demonstrate that the output of the scattering model is much less smooth and therefore better able to discriminate which vertices are members of the MC.

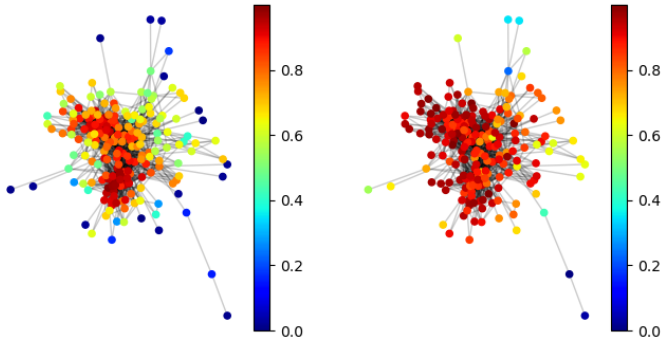

Figure 1: Comparison of the output probability vector $\mathbf{p}$ for our hybrid scattering model (left) and the low-pass model (right) on a graph taken from the TWITTER dataset with MC size 11. Our model returns a clique of size 9, while the low-pass model returns a clique of 5 nodes.

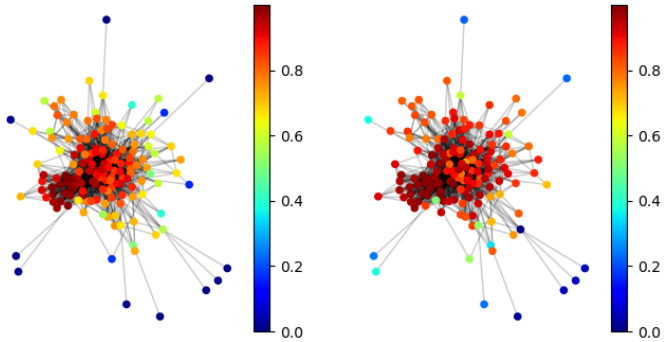

Figure 2: Comparison of the output probability vector $\mathbf{p}$ for our hybrid scattering model (left) and the low-pass model (right) on a graph taken from the TWITTER dataset with MC size 14. Our model returns the correct clique, while the low-pass model returns a clique of size 12.

References

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33: 6659–6672, 2020.

Ke Xu, Frederic Boussemart, Fred Hemery, and Christophe Lecoutre. A simple model to generate hard satisfiable instances. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 337–342, 2005.

Andrea Grosso, Marco Locatelli, and Wayne Pullan. Simple ingredients leading to very efficient heuristics for the maximum clique problem. *Journal of Heuristics*, 14(6):587–612, 2008.

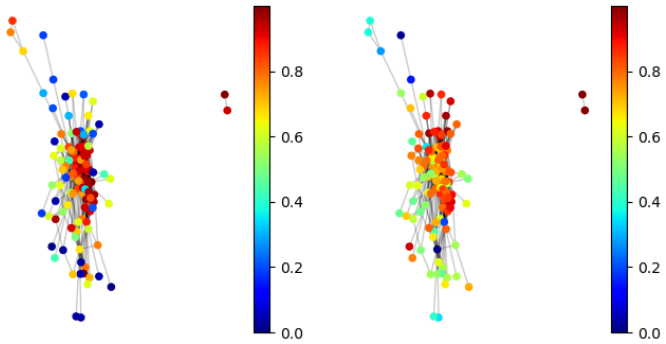

Figure 3: Comparison of the output probability vector $\mathbf{p}$ for our hybrid scattering model (left) and the low-pass model (right) on a graph taken from the TWITTER dataset with MC size 12. Our model returns a clique of size 11, while the low-pass model returns a clique of size 8.

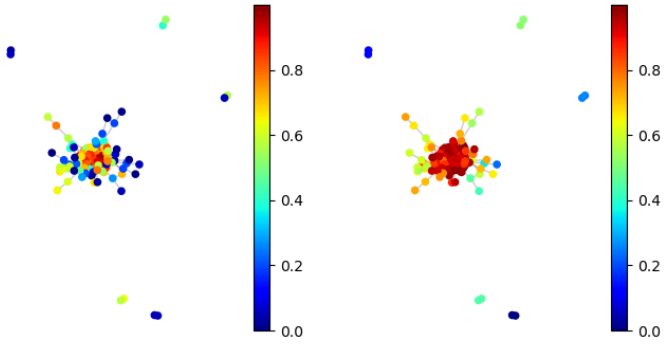

Figure 4: Comparison of the output probability vector $\mathbf{p}$ for our hybrid scattering model (left) and the low-pass model (right) on a graph taken from the TWITTER dataset with MC size 11. Our model returns a clique of size 10, while the low-pass model returns a clique of size 6.