OPT2022: 14th Annual Workshop on Optimization for Machine Learning

Annealed Training for Combinatorial Optimization on Graphs

Haoran Sun HSUN349@GATECH.EDU
Georgia Tech
Etash Guha ETASH@GATECH.EDU
Georgia Tech
Hanjun Dai HADAI@GOOGLE.COM
Google Brain

Abstract

Learning neural networks for CO problems is notoriously difficult given the lack of labeled data as the training gets trapped easily at local optima. However, the hardness of combinatorial optimization (CO) problems hinders collecting solutions for supervised learning. We propose a simple but effective unsupervised annealed training framework for CO problems in this work. In particular, we transform CO problems into unbiased energy-based models (EBMs). We carefully selected the penalties terms to make the EBMs as smooth as possible. Then we train graph neural networks to approximate the EBMs and we introduce an annealed loss function to prevent the training from being stuck at local optima near the initialization. An experimental evaluation demonstrates that our annealed training framework obtains substantial improvements. In four types of CO problems, our method achieves performance substantially better than other unsupervised neural methods on both synthetic and real-world graphs.

1. Introduction

Combinatorial Optimization (CO) problems occur whenever there is a requirement to select the best option from a finite set of alternatives. They arise in various application areas, like business, medicine, and engineering [37]. Many CO problems are NP-complete [16, 25]. Thus, excluding the use of exact algorithms to find the optimal solution [36, 56], different heuristic methods are employed to find suitable solutions in a reasonable time [14, 21, 27, 34].

Often, instances from the same combinatorial optimization problem family are solved repeatedly, giving rise to the opportunity for learning to improve the heuristic [7]. Recently, learning algorithms for CO problems has shown much promise, including supervised [17, 26, 30, 33, 42], unsupervised [24, 46], and reinforcement learning [9, 11, 44, 62] The success of supervised learning relies on labeled data. However, solving a hard problem could take several hours or even days and is computationally prohibitive [61]. Reinforcement learning, suffering from its larger state space and lack of full differentiability, tends to be more challenging and time-consuming to train.

Unsupervised learning usually transforms a CO problem into an optimization problem with a differentiable objective function f where the minima represent discrete solutions [21, 24, 43]. Although this framework allows for efficient learning on large, unlabeled datasets, it is not without challenges. The objective function is typically highly non-convex [31]. During learning, the model's parameters can easily get trapped near a local optimum close to the initialization, never reaching the optimal set of parameters. This makes unsupervised learning for CO problems extremely hard.

To address this challenge, we propose an annealed training framework. In detail, given a CO problem, we consider a tempered EBM $P_\tau \propto e^{-f(\mathbf{x})/\tau}$, where the energy function f unifies constrained or unconstrained CO problems via the big-M method, that is to say, adding large penalties for violated constraints. We derive the minimum values of the penalty coefficient in different CO problems that give us the smoothest, unbiased energy-based models. We train a graph neural network (GNN) that predicts a variational distribution Q_ϕ to approximate the energy-based model P_τ. During training, we set a high initial temperature τ and decrease it gradually during the training process. When τ is large, P_τ is close to a uniform distribution and only has shallow local optima, such that the parameter θ can traverse to distant regions. When τ decreases to values small enough, the unbiased model P_τ will concentrate on the optimal solutions to the original CO problem.

The experiments are evaluated on four NP-hard graph CO problems: MIS, maximum clique, MDS, and minimum cut. On both synthetic and real-world graphs, our annealed training framework achieves excellent performance compared to other unsupervised neural methods [24, 46], classical algorithms [1, 8], and integer solvers [18]. The ablation study demonstrates the importance of selecting proper penalty coefficients and cooling schedules.

In summary, our work has the following contributions:

- We propose an annealed learning framework for generic unsupervised learning on combinatorial optimization problems in Sec. 2. It is simple to implement yet effective in improving unsupervised learning across various problems on both synthetic and real graphs (Sec. 3).
- We conducted ablation studies in Sec. 4 that show: 1) annealed training enables the parameters to escape from local optima and traverse a longer distance, 2) selecting proper penalty coefficients is essential, 3) Using initial temperature large enough is critical.

2. Annealed Training for Combinatorial Optimization

We want to learn a graph neural network G_θ to solve combinatorial optimization problems. Given an instance I, the G_θ generates a feature $\phi = G_\theta(I)$ that determines a variational distribution Q_ϕ, from which we decode solutions. This section presents our annealed training framework for training G_θ. We first represent CO problems via an energy-based model. Then, we define the annealed loss function and explain how it helps in training. We give a toy example in Sec. 2.4 to help the understanding.

2.1. Energy Based Model

We denote the set of combinatorial optimization (CO) problems as $\mathcal{I}$. An instance $I \in \mathcal{I}$ is

$$I = (c(\cdot), \{\psi_i\}_{i=1}^m) := \arg\min_{\mathbf{x}\in\{0,1\}^n} c(\mathbf{x}) \quad \text{s.t. } \psi_i(\mathbf{x}) = 0, \quad i = 1, ..., m \tag{1}$$

where $c(\cdot)$ is the objective function and $\psi_i \in \{0, 1\}$ indicates if the i-th constraint is satisfied. We rewrite the constrained problem into an equivalent unconstrained form via the big M method:

$$\arg\min_{\mathbf{x}\in\{0,1\}^n} f^{(I)}(\mathbf{x}) := c(\mathbf{x}) + \sum_{i=1}^{m} \beta_i \psi_i(\mathbf{x}), \quad \beta_i \geq 0 \tag{2}$$

If $f^{(I)}$ has its smallest values on optimal solutions for (1), we refer it to unbiased. The selection of penalty coefficient β plays an important role in the success of training, and we will discuss our choice of β detailedly in section C. Using unbiased $f^{(I)}$ as an energy to measure the fitness of a solution

x, solving CO problems is converted to finding low energy states. Accordingly, we can define the unbiased energy-based models (EBMs):

$$P_\tau^{(I)}(\mathbf{x}) \propto e^{-f^{(I)}(\mathbf{x})/\tau} \tag{3}$$

where a state x is more likely being observed than another state x' if it has a lower energy $f^{(I)}(x) < f^{(I)}(x')$. The EBMs naturally introduce a temperature τ to control the smoothness of the distribution. When f is unbiased, it has the following property:

Proposition 1 *Assume f is unbiased, that's to say, all minimizers of (2) are feasible solutions for (1). When the temperature τ increases to infinity, the energy-based model P_τ converges to a uniform distribution over the whole state space $\{0,1\}^n$. When the temperature τ decreases to zero, the energy-based model P_τ converges to a uniform distribution over the optimal solutions for (1).*

The proposition above shows that the temperature τ in unbiased EBMs provides an interpolation between a flat uniform distribution and a sharp distribution concentrated on optimal solutions. This idea is the key to the success of simulated annealing [27] in inference tasks. We will show that the temperature also helps in learning.

2.2. Tempered Loss and Parameterization

We want to learn a graph neural network G_θ parameterized by θ. Given an instance $I \in \mathcal{I}$, $G_\theta(I) = \phi$ generates a vector ϕ that determines a variational distribution Q_ϕ to approximate the target distribution $P_\tau^{(I)}$. We want to minimize the KL-divergence:

$$D_{\mathrm{KL}}(Q_\phi || P_\tau^{(I)}) = \int Q_\phi(\mathbf{x}) \Big(\log Q_\phi(\mathbf{x}) - \log \frac{e^{-f^{(I)}(\mathbf{x})/\tau}}{\sum_{\mathbf{z}\in\{0,1\}^n} e^{-f^{(I)}(\mathbf{z})/\tau}} \Big) d\mathbf{x} \tag{4}$$

$$= \frac{1}{\tau} \mathbb{E}_{\mathbf{x}\sim Q_\phi(\cdot)}[f^{(I)}(\mathbf{x})] - H(Q_\phi) + \log \sum_{\mathbf{z}\in\{0,1\}^n} e^{-f^{(I)}(\mathbf{z})/\tau} \tag{5}$$

where $H(p) = -\sum_x p(x) \log p(x)$ denote the entropy of a distribution p. Removing the terms not involving ϕ and multiplying the constant τ, we define our annealed loss functions for ϕ and τ as:

$$L_\tau(\phi, I) = \mathbb{E}_{\mathbf{x}\sim Q_\phi(\cdot)}[f^{(I)}(\mathbf{x})] - \tau H(Q_\phi) \tag{6}$$

$$L_\tau(\theta) = \mathbb{E}_{I\sim\mathcal{I}} \Big[\mathbb{E}_{\mathbf{x}\sim Q_{G_\theta(I)}(\cdot)}[f^{(I)}(\mathbf{x})] - \tau H(Q_{G_\theta(I)}) \Big] \tag{7}$$

In this work, we consider the variational distribution as a product distribution:

$$Q_\phi(x) = \prod_{i=1}^n (1-\phi_i)^{1-x_i} \phi_i^{x_i} \tag{8}$$

where $\phi \in [0,1]^n$. Such a form is popular in learning graphical neural networks for combinatorial optimization [12, 24, 30] for its simplicity and effectiveness. However, directly applying it to unsupervised learning is challenging. Unlike supervised learning, where the loss function cross-entropy is convex for ϕ, $L_\tau(\phi, I)$ in unsupervised learning could be highly non-convex, especially when τ is small.

2.3. Annealed Training

To address the non-convexity in training, we employ annealed training. In particular, we use a large initial temperature τ_0 to smooth the loss function and reduce τ_t gradually to zero during training.

From proposition 1, it can be seen as a curriculum learning [6] along the interpolation path from the easier uniform distribution to a more challenging target distribution.

Why is it helpful? We need a thorough investigation of the training procedure to answer this. Since the loss function (7) is the expectation over the set of instances $\mathcal{I}$, we use a batch of instances $I_1, ..., I_B$ to calculate the empirical loss $\hat{L}_\tau(\theta)$ and perform stochastic gradient descent. It gives:

$$\nabla_\theta \hat{L}_\tau(\theta) = \sum_{i=1}^{B} \nabla_\theta L_\tau(G_\theta(I_i), I_i) = \sum_{i=1}^{B} \frac{\partial G_\theta(I_i)}{\partial \theta} \nabla_\phi L_\tau(\phi, I_i)|_{\phi=G_\theta(I_i)} \tag{9}$$

$$= \mathbb{E}_{I\sim\mathcal{I}} \left[\frac{\partial G_\theta(I)}{\partial \theta} \nabla_\phi L_\tau(\phi, I)|_{\phi=G_\theta(I)} \right] + \xi \tag{10}$$

$$\approx \mathbb{E}_{I\sim\mathcal{I}} \left[\frac{\partial G_\theta(I)}{\partial \theta} (\nabla_\phi L_\tau(\phi, I)|_{\phi=G_\theta(I)} + \zeta) \right] \tag{11}$$

In (10), we assume the batch introduces a stochastic term ξ in gradient w.r.t. θ. In (11), we incorporate the stochastic term into the gradient with respect to ϕ. When we assume ζ is a Gaussian noise, the inner term $g = \nabla_\phi L_\tau(\phi, I)|_{\phi=G_\theta(I)} + \zeta$ performs as a stochastic Langevin gradient with respect to ϕ [55]. Since the training data is sampled from a fixed distribution $I \sim \mathcal{I}$, the scale of the noise ζ is also fixed. When $L_\tau(\phi, i)$ is unsmooth, the randomness from ζ is negligible compared to the gradient $\nabla L_\tau(\phi, i)$ and can not bring ϕ out of local optima. By introducing the temperate τ, we smooth the loss function and reduce the magnitude of $\nabla L_\tau(\phi, i)$. During the training, the annealed training performs an implicit simulated annealing [27] for ϕ.

2.4. A Toy Example

We look at a toy example to have a more intuitive understanding of the annealed training. Consider a MIS problem on an undirected, unweighted graph $G = (V, E)$, the corresponding energy function $f(x)$ is:

$$f(x) = -\sum_{i=1}^{n} x_i + \sum_{(i,j)\in E} x_i x_j \tag{12}$$

Its correctness can be justified by proposition 2. When we use the variational distribution Q_ϕ in (8), the first term in $L_\tau(\phi, I)$ becomes to:

$$\mathbb{E}_{\mathbf{x}\sim Q_\phi(\cdot)}[f^{(I)}(\mathbf{x})] = -\sum_{i=1}^{n} \phi_i + \sum_{(i,j)\in E} \phi_i \phi_j \tag{13}$$

and accordingly, the gradient w.r.t. ϕ is:

$$g = -1 + 2 \sum_{j\in N(i)} \phi_j + \tau(\log \phi_i - \log(1 - \phi_i)) + \zeta \tag{14}$$

where we assume $\zeta \sim \mathcal{N}(0, \sigma^2)$ for a very small σ. When the temperature $\tau = 0$, ϕ_i will collapse to either 0 or 1 very fast. When $\phi_i = 1$, we have $g = -1 + \zeta$, when $\phi_i = 0$, we have $g \geq 1 + \zeta$. Since σ is small, the noise ζ can hardly have an effect, and ϕ will be stuck at local optima, i.e., any maximal independent set such as figure. 1 (a). In figure. 1, we simulate the input (a) at decreasing temperatures $\tau = 1.0, 0.5, 0.1$. When τ is large, all ϕ_i will be pushed to a neutral state, e.g., in the figure. 1 (b) where the difference of ϕ_i is at scale 10^{-3}. In this case, the noise ζ can significantly affect the sign of the gradient g and lead to phase transitions. By gradually decreasing the temperature, ϕ collapses to the global optimum and provides correct guidance to update θ.

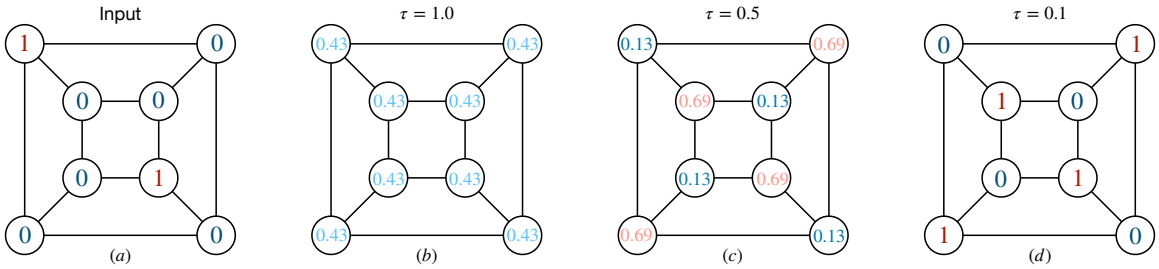

Figure 1: A toy example of maximum independent set

Table 1: Evaluation of Maximum Independent Set

Size	small		large		Collab		Twitter	
Method	*ratio*	time (s)	*ratio*	time (s)	*ratio*	time (s)	*ratio*	time (s)
Erdos	0.805 ± 0.052	0.156	0.781 ± 0.644	2.158	0.986 ± 0.056	0.010	0.975 ± 0.033	0.020
Our's	$\mathbf{0.898} \pm 0.030$	0.165	$\mathbf{0.848} \pm 0.529$	2.045	$\mathbf{0.997} \pm 0.020$	0.010	$\mathbf{0.986} \pm 0.012$	0.020
RUNCSP	0.823 ± 0.145	1.936	0.587 ± 0.312	7.282	0.912 ± 0.101	0.254	0.845 ± 0.184	4.429
RUNCSP(A)	0.851 ± 0.158	1.942	0.629 ± 0.451	7.268	0.923 ± 0.188	0.281	0.877 ± 0.209	4.438
Greedy	0.761 ± 0.058	0.002	0.720 ± 0.046	0.009	0.996 ± 0.017	0.001	0.957 ± 0.037	0.006
MFA	0.784 ± 0.058	0.042	0.747 ± 0.056	0.637	0.998 ± 0.007	0.002	0.994 ± 0.010	0.003
G(0.5s)	0.864 ± 0.169	0.723	0.632 ± 0.176	1.199	1.000 ± 0.000	0.029	0.950 ± 0.191	0.441
G(1.0s)	0.972 ± 0.065	1.063	0.635 ± 0.176	1.686	1.000 ± 0.000	0.029	1.000 ± 0.000	0.462

3. Experiments

3.1. Results

We report the results for MIS in Table 1. We also the results for maximum clique, MDS, and minimum cut in Table 2, Table 3, Table 4 in Appendix E.1, respectively. More results for comparison with supervised learning methods and evaluation on very large graphs are provided in E.2 and E.3. In the MIS and maximum clique, we report the ratios computed by dividing the optimal value by the obtained value (the larger, the better). In the MDS, we report the ratios computed from the obtained value by dividing optimal value (the larger, the better). Our method outperforms greedy heuristics, classical algorithms, and other neural based methods. Besides, with annealed training, the learned GNN outperforms MFA in most problems, with less number of iterations. It indicates that learning the shared patterns in graphs is helpful in solving CO problems. Comparing to integer solver, Gurobi is able to obtain good ratio on smaller graphs. On larger scale instances, our method can achieve comparable or even better results.

3.2. Parameter Change Distance

We want to stress that we use the same graph neural network as Erdos or RUNCSP, and the performance improvements come from our annealed training framework. In scatter plot 2, 3, we report the relative change for the parameters of GNN in MIS and MDS problems on the Twitter dataset. The relative change is calculated as $\frac{\|u-v\|_2}{\|v\|_2}$, where v and u are vectors flattened from the parameters of GNN before and after training. For each method, we run 20 seeds. After introducing the annealed training, we see that both the ratio and the relative change of the parameters have a systematic increase, meaning the parameters of GNN can traverse to more distant regions and

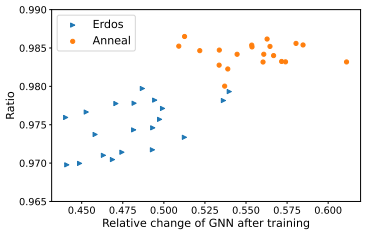

Figure 2: Distance in MIS

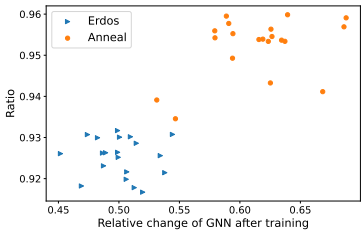

Figure 3: Distance in MDS

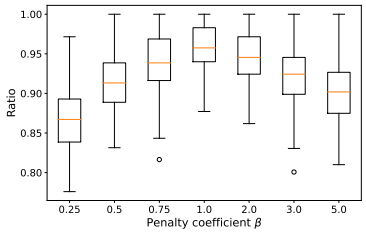

Figure 4: Ablation for β

find better optima in annealed learning. We believe this effectively supports that annealed training prevents the training from being stuck at local optima.

4. Ablation Study

We conduct an ablation study on the MDS problem on the small BA graphs to answer two questions:

1. How does the penalty coefficient β in (2) influence the performance?
2. How does the annealing schedule influence the performance?

4.1. Penalty Coefficient

In the MDS problem, we know that the minimum penalty coefficient β needed to ensure the EBMs unbiased on the unweighted BA graphs is $\beta = 1.0$. To justify the importance to use the minimum penalty, we evaluate the performance for $\beta = \{0.0, 0.25, 0.5, 0.75, 1.0, 2.0, 3.0, 5.0\}$. For each β, we run experiments with five random seeds, and we report the result in Figure 4. We can see that the minimum penalty $\beta = 1$ has the best ratio. When the penalty coefficient $\beta < 1$, the EBMs (3) are biased and have weights on infeasible solutions, thereby reducing the performance. When the penalty coefficient $\beta > 1$, the energy model (3) becomes less smooth and increases the difficulty in training. The penalty coefficient $\beta = 1$ gives the smoothest unbiased EBMs and has the best performance. We want to note that when $\beta = 0$, the loss function is non-informative, and the performance ratio can be as low as 0.3, so we do not plot its result in the figure.

4.2. Annealing Schedule

We report the results in Figure 5 in Appendix E.4 for different annealing schedule. We see that the performance is robust for whichever convex, linear, or concave schedule is used. The more important factor is the initial temperature τ_0. The performance is reduced when τ_0 is too small as the energy-based model (3) is not smooth enough, and the performance is robust when τ_0 is large.

References

[1] Emile Aarts, Emile HL Aarts, and Jan Karel Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[2] 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.

[3] C PETERSON J ANDERSON. Neural networks and np-complete optimization problems; a performance study on the graph bisection problem. *Complex Systems*, 2:58–59, 1988.

[4] 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.

[5] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[6] 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.

[7] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 2020.

[8] Griff Bilbro, Reinhold Mann, Thomas Miller, Wesley Snyder, David van den Bout, and Mark White. Optimization by mean field annealing. *Advances in neural information processing systems*, 1:91–98, 1988.

[9] Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. *Advances in Neural Information Processing Systems*, 32:6281–6292, 2019.

[10] Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. Compositional generalization via neural-symbolic stack machines. *Advances in Neural Information Processing Systems*, 33:1690–1701, 2020.

[11] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665*, 2017.

[12] Hanjun Dai, Xinshi Chen, Yu Li, Xin Gao, and Le Song. A framework for differentiable discovery of graph algorithms. 2020.

[13] Rina Dechter, David Cohen, et al. *Constraint processing*. Morgan Kaufmann, 2003.

[14] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[15] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018.

[16] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[17] 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.

[18] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2022. *URL http://www. gurobi. com*, 19.

[19] He He, Hal Daume III, and Jason M Eisner. Learning to search in branch and bound algorithms. *Advances in neural information processing systems*, 27:3293–3301, 2014.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

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

[22] Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Mayur Naik, Le Song, and Xujie Si. Scallop: From probabilistic deductive databases to scalable differentiable reasoning. *Advances in Neural Information Processing Systems*, 34, 2021.

[23] 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.

[24] Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *arXiv preprint arXiv:2006.10643*, 2020.

[25] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[26] Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[27] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[28] 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.

[29] Henrique Lemos, Marcelo Prates, Pedro Avelar, and Luis Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 879–885. IEEE, 2019.

[30] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. *arXiv preprint arXiv:1810.10659*, 2018.

[31] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[32] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[33] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[34] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[35] 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.

[36] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[37] Vangelis Th Paschos. *Applications of combinatorial optimization*. John Wiley & Sons, 2013.

[38] 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.

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

[40] J Ramanujam and P Sadayappan. Mapping combinatorial optimization problems onto neural networks. *Information sciences*, 82(3-4):239–255, 1995.

[41] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *International Conference on Machine Learning*, pages 8959–8970. PMLR, 2021.

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

[43] 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.

[44] Haoran Sun, Wenbo Chen, Hui Li, and Le Song. Improving learning to branch via reinforcement learning. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*, 2020. URL https://openreview.net/forum?id=z4D7-PTxTb.

[45] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *31st Conference on Neural Information Processing Systems (NIPS)*, volume 30, pages 1–18, 2017.

[46] Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3:98, 2021.

[47] 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.

[48] 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.

[49] Nate Veldt, David Gleich, and Michael Mahoney. A simple and strongly-local flow-based method for cut improvement. In *International Conference on Machine Learning*, pages 1938–1947. PMLR, 2016.

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

[51] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *arXiv preprint arXiv:1506.03134*, 2015.

[52] Di Wang, Kimon Fountoulakis, Monika Henzinger, Michael W Mahoney, and Satish Rao. Capacity releasing diffusion for speed and locality. In *International Conference on Machine Learning*, pages 3598–3607. PMLR, 2017.

[53] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3056–3065, 2019.

[54] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Combinatorial learning of robust deep graph matching: an embedding based approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[55] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[56] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

[57] 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.

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

[59] 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.

[60] 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.

[61] Gal Yehuda, Moshe Gabel, and Assaf Schuster. It's not what machines can learn, it's what we cannot teach. In *International Conference on Machine Learning*, pages 10831–10841. PMLR, 2020.

[62] Emre Yolcu and Barnabás Póczos. Learning local search heuristics for boolean satisfiability. In *NeurIPS*, pages 7990–8001, 2019.

Appendix A. Types of Problems Solvable

The current method uses conditional decoding [39] to sample solutions from the learned variational distributions, which requires monotonic post-processing to make sure the final solution is feasible. For example, in the maximum independent set, the monotonic post-processing is removing nodes when conflict happens, in the minimum dominant set, the monotonic post-processing is adding nodes when a node has not been covered. Such a framework can be applied to CO problems that have trivial solutions, such as set covering problems, but can not be applied to CO problems with complicated constraints, such as vehicle routing problems.

Appendix B. Related Work

Recently, there has been a surge of interest in learning algorithms for CO problems [7]. Supervised learning is widely used. Numerous works have combined GNNs with search procedures to solve classical CO problems, such as the traveling salesman problem [23, 38, 51], graph matching [53, 54], quadratic assignments [35], graph coloring [29], and MIS [30]. Another fruitful direction is combining learning with existing solvers. For example, in the branch and bound algorithm, Gasse et al. [17], He et al. [19], Khalil et al. [26], Nair et al. [33] learn the variable selection policy by imitating the decision of oracle or rules designed by human experts. However, the success of supervised learning relies on large labeled datasets, which is hard to efficiently generate in an unbiased and representative manner [61].

Many works, therefore, choose to use reinforcement learning instead. Dai et al. [11] combines Q-learning with greedy algorithms to solve CO problems on graphs. Q-learning is also used in [4] for maximum subgraph problem. Sun et al. [44] uses an evolutionary strategy to learn variable selection in the branch and bound algorithm. Yolcu and Póczos [62] employs REINFORCE algorithm to learn local heuristics for SAT problems. Chen and Tian [9] uses actor-critic learning to learn a local rewriting algorithm. Despite being a promising approach that avoids using labeled data, reinforcement learning is typically sample inefficient and notoriously unstable to train due to poor gradient estimations, correlations present in the sequence of observations, and hard explorations [15, 45].

Works in unsupervised learning show promising results. In initial attempts, Hopfield and Tank [21], Ramanujam and Sadayappan [40], Van den Bout and Miller [48] transform CO problems into optimization problem of neural networks with differentible objective functions. More recently, a series of deep learning approaches emerges. Yao et al. [60] train GNN for the max-cut problem by optimizing a relaxation of the cut objective, Toenshoff et al. [46] trains RNN for maximum-SAT via maximizing the probability of its prediction. Karalias and Loukas [24] use a GNN to predict the distribution and the graphical neural network to minimize the expectation of the objective function on this distribution. The probabilistic method provides a good framework for unsupervised learning. However, optimizing the distribution is typically non-convex [31], making the training very unstable.

Appendix C. Case Study

We consider four combinatorial optimization problems on graphs in this work: maximum independent set (MIS), maximum clique, minimum dominate set (MDS), and minimum cut. An undirected weighted graph can represent all problems $G = (V, E, w)$, where $V = \{1, ..., n\}$ is the set of nodes, E is the set of edges, and w is the weight function. For any $i \in V$, $w_i = w(i)$ is the weight of the

node. For any $(i,j) \in E$, $w_{ij} = w(i,j)$ is the weight of the edge. For each problem, we derive the minimum value of the penalty coefficient β such that the energy function has the lowest energy at optimal solutions, and we use the derived values to design the loss functions in our experiments.

C.1. Maximum Independent Set

In MIS, we use the energy function:

$$f(x) := -\sum_{i=1}^{n} w_i x_i + \sum_{(i,j)\in E} \beta_{ij} x_i x_j \tag{15}$$

We are going to prove the following proposition.

Proposition 2 *If $\beta_{ij} \geq \min\{w_i, w_j\}$ for all $(i,j) \in E$, then for any $x \in \{0,1\}^n$, there exists a $x' \in \{0,1\}^n$ that satisfies the constraints in (15) and has lower energy: $f(x') \leq f(x)$.*

Proof For arbitrary $x \in \{0,1\}^n$, if x satisfies all constraints, we only need to let $x' = x$. Else, there must exist an edge $(i,j) \in E$, such that $x_i x_j = 1$. Denote $k = \arg\min\{w_i, w_j\}$, we define $x'_i = x_i$ if $i \neq k$ and $x'_k = 0$. In this case, we have:

$$f(x') - f(x) = w_k - \sum_{j\in N(k)} \beta_{k,j} x_j \leq w_k(1 - \sum_{j\in N(k)} x_j) \leq 0 \tag{16}$$

Thus we show $f(x') \leq f(x)$.

On the other side, consider a graph $G = (V = \{1,2\}, E = \{(1,2)\})$ and $\beta_{12} < w_1 < w_2$. Then the maximum independent set is $\{2\}$, which can be represented by $x = (0,1)$. However, in this case, let $x' = (1,1)$ is feasible while $f(x') \leq f(x)$. This means the condition we just derived is sharp. ■

C.2. Maximum Clique

A clique is a subset of the vertices $S \subseteq V$, such that every two distinct $i, j \in S$ are adjacent: $(i,j) \in E$. The maximum problem is finding a clique S with the largest weight. Rigorously, if we denote $x_i = 1$ to indicate $i \in S$ and $x_i = 0$ to indicate $i \notin S$, the problem can be formulated as:

$$\arg\min_{x\in\{0,1\}^n} c(x) := -\sum_{i=1}^{n} w_i x_i, \quad \text{subject to } x_i x_j = 0, \forall (i,j) \in E^c \tag{17}$$

where $E^c = \{(i,j) \in V \times V : i \neq j, (i,j) \notin E\}$ is the set of complement edges on graph G. We define the corresponding energy function:

$$f(x) := -\sum_{i=1}^{n} w_i x_i + \sum_{(i,j)\in E^c} \beta_{ij} x_i x_j \tag{18}$$

We are going to prove the following proposition.

Proposition 3 *If $\beta_{ij} \geq \min\{w_i, w_j\}$ for all $(i,j) \in E^c$, then for any $x \in \{0,1\}^n$, there exists a $x' \in \{0,1\}^n$ that satisfies the constraints in (17) and has lower energy: $f(x') \leq f(x)$.*

Proof For arbitrary $x \in \{0,1\}^n$, if x satisfies all constraints, we only need to let $x' = x$. Else, there must exist an edge $(i,j) \in E^c$, such that $x_i x_j = 1$. Denote $k = \arg\min\{w_i, w_j\}$, we define $x'_i = x_i$ if $i \neq k$ and $x'_k = 0$. In this case, we have:

$$f(x') - f(x) = w_k - \sum_{j:(k,j)\in E^c} \beta_{k,j} x_j \leq w_k(1 - \sum_{j:(k,j)\in E^c} x_j) \leq 0 \tag{19}$$

Thus we show $f(x') \leq f(x)$.

On the other side, consider a graph $G = (V = \{1, 2\}, E = \{\})$ and $\beta_{12} < w_1 < w_2$. Then the maximum clique is $\{2\}$, which can be represented by $x = (0, 1)$. However, in this case, let $x' = (1, 1)$ is feasible while $f(x') \leq f(x)$. This means the condition we just derived is sharp. ■

C.3. Minimum Dominate Set

In MIS, we use the energy function:

$$f(x) := \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \beta_i (1 - x_i) \prod_{j \in N(i)} (1 - x_j) \tag{20}$$

We are going to prove the following proposition.

Proposition 4 *If $\beta_i \geq \min_k\{w_k : k \in N(i) \text{ or } k = i\}$, then for any $x \in \{0, 1\}^n$, there exists a $x' \in \{0, 1\}^n$ that satisfies the constraints in (17) and has lower energy: $f(x') \leq f(x)$.*

Proof For arbitrary $x \in \{0, 1\}^n$, if x satisfies all constraints, we only need to let $x' = x$. Else, there must exist a node $t \in V$, such that $x_t = 0$ and $x_j = 0$ for all $j \in N(t)$. Let $k = \arg\min\{w_j : j \in N(t), \text{ or } j = t\}$, we define $x'_i = x_i$ if $i \neq k$ and $x'_k = 1$. In this case, we have:

$$f(x') - f(x) = w_k - \beta_t + \sum_{i \neq t} \beta_i \Big[(1 - x'_i) \prod_{j \in N(i)} (1 - x'_j) - (1 - x_i) \prod_{j \in N(i)} (1 - x_j) \Big] \leq 0 \tag{21}$$

Thus, we prove $f(x') \leq f(x)$.

On the other side, consider a graph $G = (V = \{1\}, E = \{\})$ and $\beta_1 < w_1$. Then the maximum clique is $\{1\}$, which can be represented by $x = (1)$. However, in this case, let $x' = (0)$ is feasible while $f(x') \leq f(x)$. This means the condition we just derived is sharp. ■

C.4. Minimum Cut

In MIS, we use the energy function:

$$f(x) := \sum_{(i,j) \in E} x_i (1 - x_j) w_{ij} + \beta (\sum_{i=1}^{n} d_i x_i - D_1)_+ + \beta (D_0 - \sum_{i=1}^{n} d_i x_i)_+ \tag{22}$$

We are going to prove the following proposition.

Proposition 5 *If $\beta \geq \max_i\{\sum_{j \in N(i)} |w_{i,j}|\}$, then any $x \in \{0, 1\}^n$, there exists a $x' \in \{0, 1\}^n$ that satisfies the constraints in (17) and has lower energy: $f(x') \leq f(x)$.*

Appendix D. Experiment Details

D.1. Hardware

All methods were run on Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz, with 377GB of available RAM. The neural networks were executed on a single RTX6000 25GB graphics card. The code was executed on version 1.9.0 of PyTorch and version 1.7.2 of PyTorch Geometric.

D.2. Settings

Dataset: For MIS and maximum clique, problems on both real and random graphs are easy [12]. Hence, we follow Karalias and Loukas [24] to use RB graphs [57], designed to generate hard instances. We use a small dataset containing graphs with 200-300 nodes and a large dataset containing graphs with 800-1200 nodes. For MDS, we follow Dai et al. [12] to use BA graphs with 4 attaching edges [5]. We also use a small dataset containing graphs with 200-300 nodes and a large dataset containing graphs with 800-1200 nodes. We also use real graph datasets Collab, Twitter from TUdataset [32]. For minimum cut, we follow Karalias and Loukas [24] and use real graph datasets including SF-295 [59], Facebook [47], and Twitter [32]. For RB graphs, the optimal solution is known during the graph construction. For other problems, we generate the "ground truth" solution through Gurobi 9.5 [18] with a time limit of 3600 seconds. For synthetic datasets, we generate 2000 graphs for training, 500 for validation, and 500 for testing. For real datasets, we follow Karalias and Loukas [24] and use a 60-20-20 split for training, validating, and testing.

Implementation: We train our graph neural network on training data with 500 epochs. We choose the penalty coefficient β at the critical point for each problem type. We use the schedule:

$$\tau_k = \tau_0/(1+\alpha k) \tag{23}$$

where τ_0 is chosen as the Lipschitz constant of the energy function (2) and α is selected to make sure the final temperature $\tau_{500} = 0.001$. Since the contribution of this work focuses on the training framework, the architecture of the graph neural network is not important. Hence, we provide results from applying annealing training to Karalias and Loukas [24] and Toenshoff et al. [46] for fair comparison, denoted as "Annealed Erdos" and "Annealed RUNCSP" respectively. In particular, the architecture from Karalias and Loukas [24] consists of multiple layers of the Graph Isomorphism Network [58] and a graph Attention [50]. More details refer to Karalias and Loukas [24]. Moreover, the architecture from Toenshoff et al. [46] creates a network that approximates a Constraint Language [13] using a message-passing GNN using an LSTM for internal updates [20]. With both of these GNN architectures, after obtaining the variational distribution Q_ϕ (8), we generate the solution via conditional decoding [39].

Baselines: We compare our method with unsupervised neural methods, classical algorithms, and integer programming solvers. To establish a strong baseline for neural methods, we use the Erdos GNN [24], the state-of-the-art unsupervised learning framework for combinatorial optimization problems. For maximum clique and MIS, we transform the problem to constraint programming and compare them with RUNCSP [46]. We also implement the annealed training version of RUNCSP and denote it as RUNCSP(A). We followed Karalias and Loukas [24] for minimum cut and built the L1 GNN and L2 GNN. In classical algorithms, we consider greedy algorithms and mean field annealing (MFA) [8]. MFA also runs mean field approximation [3] to predict a variational distribution as our method. The difference is that the update rule of MFA is determined after seeing the current graph, while the parameters in GNN are trained on the whole dataset. Also, in minimum cut, we follow [24] to compare with well-known and advanced algorithms: *Pageran-Nibble* [2], Capacity Releasing Diffusion (CRD) [52], Max-flow Quotient-cut Improvement [28], and Simple-Local [49]. For integer programming solver, we use Gurobi 9.0 [18] and set different time limits t. We denote G(ts) as Gurobi 9.0 (t s). where t is the solving time limit. One needs to notice that Gurobi has proprocessing before solving, so the actual running time can be longer than the given time limit.

D.3. Greedy Algorithm

For MIS, greedy algorithm can be described in the following steps:

1. Pick the variable i has the smallest degree d_i in the candidate set.
2. Delete i and all its neighborhood $N(i) = \{j : (i, j) \in E\}$ on the current graph.
3. Repeat step 1-2, until the current graph is empty.

For maximum clique, we first transform the graph into its complementary, then apply the greedy algorithm for MIS.

For MDS, greedy algorithm can be described in the following steps:

1. For every node i, initialize its state $s_i = 1$ to indicate whether it has not been covered.
2. For every node i, initialize its covering number $c_i = s_i + \sum_{j \in N(i)} s_j$ to indicate how many nodes can be covered by selecting node i
3. Select the node i has the largest covering number c_i.
4. Mark $s_i = 0$, and $s_j = 0$ for $j \in N(i)$.
5. Repeat step 3-4, until all $s_i = 0$.

D.4. Datasets

For MIS and maximum clique, we follow Karalias and Loukas [24] and use RB graphs [57]. The construction of RB graphs has 4 parameters n, k, p. Following Karalias and Loukas [24], for small graphs, we use n uniformly sampled from the integers $[20, 25]$ and k uniformly sampled from $[5, 12]$; for large graphs, we use n uniformly sampled from the integers $[40, 55]$ and k uniformly sampled from $[20, 25]$.

For minimum dominant set, we follow Dai et al. [12] to Barabasi-Albert networks [5] with attachment 4.

Appendix E. More Results

E.1. Results on Max Clique, MDS, Min Cut

Here, we report the results for maximum clique, Minimum dominating set, and minimum cut. The analysis for maximum clique and minimum dominating set are given in the main text.

In minimum cut, we follow Karalias and Loukas [24] and evaluate the performance via local conductance: $\text{cut}(S)/\text{vol}(S)$ (the smaller the better). We can see that the annealed training substantially improves the performance of Erdos across all problem types and all datasets, except for SF-295 in minimum cut, by utilizing a better-unsupervised training framework. Our method also outperforms greedy heuristics, classical algorithms such as MFA, CRD, MQI, and other learning based approaches such as RUNCSP, L1/L2 GNN.

Table 2: Evaluation of Maximum Clique

Size	small		large		Collab		Twitter	
Method	*ratio*	time (s)	*ratio*	time (s)	*ratio*	time (s)	*ratio*	time (s)
Erdos	0.813 ± 0.067	0.279	0.735 ± 0.084	0.622	0.960 ± 0.019	0.139	0.822 ± 0.085	0.222
Our's	$\mathbf{0.901} \pm 0.055$	0.262	$\mathbf{0.831} \pm 0.078$	0.594	$\mathbf{0.988} \pm 0.011$	0.143	$\mathbf{0.920} \pm 0.083$	0.213
RUNCSP	0.821 ± 0.131	2.045	0.574 ± 0.299	7.332	0.887 ± 0.134	0.164	0.832 ± 0.153	4.373
RUNCSP(A)	0.860 ± 0.189	2.101	0.609 ± 0.381	7.294	0.895 ± 0.162	0.188	0.877 ± 0.221	4.442
Greedy	0.764 ± 0.064	0.002	0.727 ± 0.038	0.014	0.999 ± 0.002	0.001	0.959 ± 0.034	0.001
MFA	0.804 ± 0.064	0.144	0.710 ± 0.045	0.147	1.000 ± 0.000	0.005	0.994 ± 0.010	0.010
G(0.5s)	0.948 ± 0.076	0.599	0.812 ± 0.087	0.617	0.997 ± 0.035	0.061	0.976 ± 0.065	0.382
G(1.0s)	0.984 ± 0.042	0.705	0.847 ± 0.101	1.077	0.999 ± 0.015	0.062	0.997 ± 0.029	0.464

Table 3: Evaluation of Minimum Dominating Set

Size	small		large		Collab		Twitter	
Method	*ratio*	time (s)	*ratio*	time (s)	*ratio*	time (s)	*ratio*	time (s)
Erdos	0.909 ± 0.037	0.121	0.889 ± 0.017	0.449	0.982 ± 0.070	0.007	0.924 ± 0.098	0.015
Our's	$\mathbf{0.954} \pm 0.006$	0.120	$\mathbf{0.931} \pm 0.015$	0.453	$\mathbf{0.993} \pm 0.062$	0.006	$\mathbf{0.952} \pm 0.074$	0.016
Greedy	0.743 ± 0.053	0.254	0.735 ± 0.026	3.130	0.661 ± 0.406	0.028	0.741 ± 0.142	0.079
MFA	0.926 ± 0.032	0.213	0.910 ± 0.016	3.520	0.895 ± 0.210	0.030	0.952 ± 0.076	0.099
G(0.5s)	0.993 ± 0.014	0.381	0.994 ± 0.013	0.384	1.000 ± 0.000	0.042	1.000 ± 0.000	0.084
G(1.0s)	0.999 ± 0.005	0.538	0.999 ± 0.005	0.563	1.000 ± 0.000	0.042	1.000 ± 0.000	0.084

E.2. Comparison to Supervised Learning

In order to compare our unsupervised results to supervised results, we provide evaluation results for the MDS problem on BA-4 graphs, using the supervised learning result in [12]. As in Table 5, we can see that annealed training significantly improves the performance of unsupervised learning and has a ratio very close to supervised learning. We also provide a comparison for supervised learning for evaluation on small RB graphs for MIS. As a source of labels, we use Gurobi to solve the maximum independent set problem. Due to the computational limitations, we set a time limit as 10 seconds and some of the instances are not solved to optimal. In this case as in Table 6, we observe that, with the proper training algorithm, the unsupervised learning can even beat the supervised learning.

Table 4: Evaluation of Minimum Cut

Size	SF-295		Facebook		Twitter	
Method	*ratio*	time (s)	*ratio*	time (s)	*ratio*	time (s)
Erdos	$\mathbf{0.124} \pm 0.001$	0.22	0.156 ± 0.026	289.3	0.292 ± 0.009	6.17
Our's	0.135 ± 0.011	0.23	$\mathbf{0.151} \pm 0.045$	290.5	$\mathbf{0.201} \pm 0.007$	6.16
L1 GNN	0.188 ± 0.045	0.02	0.571 ± 0.191	13.83	0.318 ± 0.077	0.53
L2 GNN	0.149 ± 0.038	0.01	0.305 ± 0.082	13.83	0.388 ± 0.074	0.53
Pagerank-Nibble	0.375 ± 0.001	1.48	N/A	N/A	0.603 ± 0.005	20.62
CRD	0.364 ± 0.001	0.03	0.301 ± 0.097	596.46	0.502 ± 0.020	20.35
MQI	0.659 ± 0.000	0.03	0.935 ± 0.024	408.52	0.887 ± 0.007	0.71
Simple-Local	0.650 ± 0.024	0.05	0.955 ± 0.019	404.67	0.895 ± 0.006	0.84
G(10s)	0.105 ± 0.000	0.16	0.961 ± 0.010	1787.79	0.535 ± 0.006	52.98

Table 5: Evaluation of MDS on BA-4 compared to Supervised Learning

Graphs	256-300	512-600	1024-1100
Supervised	0.947	0.938	0.934
Erdos	0.909	0.905	0.898
Ours	0.937	0.935	0.926

Table 6: Evaluation of MIS on small RB graphs compared to Supervised Learning

Graphs	Small
Supervised	0.889
Erdos	0.805
Ours	0.898

E.3. Evaluation on very large graphs

We conduct extra experiments for MIS on large BA-4 Graphs following Dai et al. [12]. The model is trained on BA-4 graphs with size 1024-1100. For each larger size, we evaluate the methods on 100 graphs and report the mean, std, and average running time. We can see that the performance of Gurobi decreases with increasing the graph size and the learning based approaches. Another observation is that the learning based approaches have much smaller running time, as Dai et al. [12] has the conditional decoding implemented in cpp, while our conditional decoding is implemented in python.

E.4. Annealing Schedule

We use the schedule (23) so as to make sure the potential change $f/\tau_{k+1} - f/\tau_k \equiv C$ is a constant for all steps k. In fact, with the schedule (23), the potential $f/\tau_k = (1 + \alpha(k-1))f/\tau_0$ is a linear function w.r.t. k. Hence, we name it a linear schedule. It is possible to use other schedules, e.g. $f/\tau_k = (1 + \alpha(k-1))^{\frac{1}{2}} f/\tau_0$ and $f/\tau_k = (1 + \alpha(k-1))^3 f/\tau_0$, and we name them as concave and convex schedule. The visualization of the temperature schedule and the potential schedule is given in Figure 5. The initial temperature is also an important hyperparameter. We evaluate the initial temperature $\tau_0 = \{0.0, 0.1, 0.5, 1.0, 2.0, 5.0\}$. We report the results in Figure 5. We see that the performance is robust for whichever convex, linear, or concave schedule is used. The more important factor is the initial temperature τ_0. The performance is reduced when τ_0 is too small as the energy-based model (3) is not smooth enough, and the performance is robust when τ_0 is large.

Table 7: Evaluation of MIS on Very Large BA-4 Graphs

Method	2048-2200		4096-4400		8192-8800	
Size	Ratio	Time	Ratio	Time	Ratio	Time
Greedy	928 ± 26	$5.6e^{-3}$	1861 ± 50	$2.7e^{-2}$	3727 ± 102	$1.3e^{-1}$
Erdos	950 ± 26	$6.5e^{-2}$	1900 ± 49	$1.6e^{-1}$	3742 ± 118	$3.2e^{-1}$
Ours	$\mathbf{960} \pm 25$	$6.5e^{-2}$	$\mathbf{1923} \pm 48$	$1.6e^{-1}$	$\mathbf{3845} \pm 99$	$3.2e^{-1}$
Gurobi	919 ± 26	$1.2e^{0}$	1845 ± 59	$5.3e^{0}$	3165 ± 656	$5.5e^{0}$

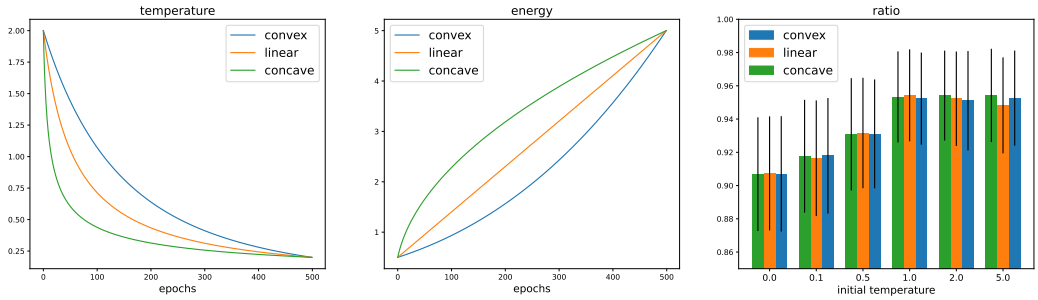

Figure 5: Ablation for annealing schedule

Appendix F. Discussion

This paper proposes a generic unsupervised learning framework for combinatorial optimization problems and substantially improves the performance of the state-of-the-art method. One restriction of the current method is that it relies on condtional decoding to samle solutions from the learned variational distributions. For problems with more complex constraints, the decoded solutions might be infeasible. Hence, we believe better decoding strategies should be considered in future work.

The framework's success relies on smoothing the loss function via critical penalty coefficients and annealed training as they effectively prevent the training from being stuck at local optima. The techniques introduced here can be potentially applied in a broader context beyond combinatorial optimization, especially in the weakly supervised learning setting like logic reasoning [22], program induction [10], question answering [41] where fine-grained supervisions are missing and required to be inferred.