Controlling Continuous Relaxation for Combinatorial Optimization

Yuma Ichikawa
Fujitsu Limited, Kanagawa, Japan
Department of Basic Science, University of Tokyo

Abstract

Unsupervised learning (UL)-based solvers for combinatorial optimization (CO) train a neural network that generates a soft solution by directly optimizing the CO objective using a continuous relaxation strategy. These solvers offer several advantages over traditional methods and other learning-based methods, particularly for large-scale CO problems. However, UL-based solvers face two practical issues: (I) an optimization issue, where UL-based solvers are easily trapped at local optima, and (II) a rounding issue, where UL-based solvers require artificial post-learning rounding from the continuous space back to the original discrete space, undermining the robustness of the results. This study proposes a Continuous Relaxation Annealing[1] (**CRA**) strategy, an effective rounding-free learning method for UL-based solvers. CRA introduces a penalty term that dynamically shifts from prioritizing continuous solutions, effectively smoothing the non-convexity of the objective function, to enforcing discreteness, eliminating artificial rounding. Experimental results demonstrate that CRA significantly enhances the performance of UL-based solvers, outperforming existing UL-based solvers and greedy algorithms in complex CO problems. Additionally, CRA effectively eliminates artificial rounding and accelerates the learning process.

1 Introduction

The objective of combinatorial optimization (CO) problems is to find the optimal solution from a discrete space, and these problems are fundamental in many real-world applications [Papadimitriou and Steiglitz, 1998]. Most CO problems are NP-hard or NP-complete; making it challenging to solve large-scale problems within feasible computational time. Traditional methods frequently depend on heuristics to find approximate solutions; however, considerable insights into the specific problems are required. Alternatively, problems can be formulated as integer linear programming (ILP) and solved using ILP solvers. However, ILP lacks scalability when applied to problems with graph structures.

Recently, several studies have used machine learning methods to handle CO problems by learning heuristics. Most of these studies focus on supervised learning (SL)-based solvers [Hudson et al., 2021, Joshi et al., 2019, Gasse et al., 2019, Selsam et al., 2018, Khalil et al., 2016], which require optimal solutions to CO problems as supervision during training. However, obtaining optimal solutions is challenging in practice, and SL-based solvers often fail to generalize well [Yehuda et al., 2020]. Reinforcement learning (RL)-based solvers [Yao et al., 2019, Chen and Tian, 2019, Yolcu and Póczos, 2019, Nazari et al., 2018, Khalil et al., 2017, Bello et al., 2016] avoid the need for optimal solutions but often suffer from notoriously unstable training due to poor gradient estimation and hard explorations [Mnih et al., 2015, Tang et al., 2017, Espeholt et al., 2018]. Unsupervised learning (UL)-based solvers [Schuetz et al., 2022a, Karalias and Loukas, 2020, Amizadeh et al., 2018]

[1]The code is available at https://github.com/Yuma-Ichikawa/CRA4CO.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

have recently attracted much attention. UL-based solvers follow a continuous relaxation approach, training a UL model to output a *soft solution* to the relaxed CO problem by directly optimizing a differentiable objective function, offering significantly stable and fast training even for large-scale CO problems. Notably, the physics-inspired GNN (PI-GNN) solver [Schuetz et al., 2022a] employs graph neural networks (GNN) to automatically learn instance-specific heuristics and performs on par with or outperforms existing solvers for CO problems with millions of variables without optimal solutions. While these offer some advantages over traditional and other machine learning-based solvers, they face two practical issues. The first issue is an optimization issues where UL-based solvers are easily trapped at local optima. Due to this issue, Angelini and Ricci-Tersenghi [2023] demonstrated that the PI-GNN solver [Schuetz et al., 2022a] could not achieve results comparable to those of the degree-based greedy algorithm (DGA) [Angelini and Ricci-Tersenghi, 2019] on maximum independent set (MIS) problems in random regular graphs (RRG). Wang and Li [2023] also pointed out the importance of using dataset or history, and initializing the GNN with outputs from greedy solvers to help the PI-GNN solver overcome optimization challenges. This issue is a crucial bottleneck to the applicability of this method across various real-world applications. The second issue relates to the inherent ambiguity of the continuous relaxation approach. This approach necessitates artificial rounding from the soft solution, which may include continuous values, back to the original discrete solution, potentially undermining the robustness of the results. While linear relaxation can provide an optimal solution for original discrete problems on bipartite graphs [Hoffman and Kruskal, 2010], it typically leads to solutions with $1/2$ values, which is known to half-integrality [Nemhauser and Trotter Jr, 1974], in which existing rounding methods [Schuetz et al., 2022b, Wang et al., 2022] completely lose their robustness. For NP-hard problems with graph structures, such as the MIS and MaxCut, semidefinite programming (SDP) relaxations have been proposed as effective approximation methods [Lovász, 1979, Goemans and Williamson, 1995]. However, these approaches rely on rounding techniques, such as spectral clustering [Von Luxburg, 2007], to transform relaxed solutions into feasible ones, which often fails to obtain optimal solutions.

To address these issues, we propose the Continuous Relaxation Annealing (**CRA**). CRA introduces a penalty term to control the continuity and discreteness of the relaxed variables, with a parameter γ to regulate the intensity of this penalty term. When the parameter γ is small, the relaxed variable tends to favor continuous solutions, whereas a large γ biases them toward discrete values. This penalty term also effectively eliminates local optimum. Moreover, a small γ forces the loss function to approach a simple convex function, encouraging active exploration within the continuous space. CRA also includes an annealing process, where γ is gradually increased until the relaxed variables approach discrete values, eliminating the artificial rounding from the continuous to the original discrete space after learning. In this study, the solver that applies the CRA to the PI-GNN solver is referred to as the CRA-PI-GNN solver. We also demonstrate the benefits of the CRA through experiments on benchmark CO problems, including MIS, maximum cut (MaxCut), and diverse bipartite matching (DBM) problems across graphs of varying sizes and degrees. The experimental results show that the CRA significantly enhances the performance of the PI-GNN solver, outperforming the original PI-GNN solver, other state-of-the-art learning-based baselines, and greedy algorithms. This improvement is achieved by directly optimizing each instance without any history, e.g., previous optimal solutions and the information of other instances. Additionally, these experiments indicate that the CRA accelerates the learning process of the PI-GNN solver. Notably, these results overcome the limitations pointed out by Angelini and Ricci-Tersenghi [2023], Wang and Li [2023], highlighting the further potential of UL-based solvers.

Notation We use the shorthand expression $[N] = \{1, 2, \ldots, N\}$, where $N \in \mathbb{N}$. $I_N \in \mathbb{R}^{N \times N}$ denotes an $N \times N$ identity matrix, $\mathbf{1}_N$ denotes the vector $(1, \ldots, 1)^\top \in \mathbb{R}^N$, and $\mathbf{0}_N$ denotes the vector $(0, \ldots, 0)^\top \in \mathbb{R}^N$. $G(V, E)$ represents an undirected graph, where V is the set of nodes with cardinality $|V| = N$, and $E \subseteq V \times V$ denotes the set of edges. For a graph $G(V, E)$, A_{ij} denotes the adjacency matrix, where $A_{ij} = 0$ if an edge (i, j) does not exist and $A_{ij} > 0$ if the edge is present.

2 Background

Combinatorial optimization The goal of this study is to solve the following CO problem.

$$\min_{\boldsymbol{x} \in \{0,1\}^N} f(\boldsymbol{x}; C) \quad \text{s.t.} \quad \boldsymbol{x} \in \mathcal{X}(C), \tag{1}$$

where $C \in \mathcal{C}$ denotes instance-specific parameters, such as a graph $G = (V, E)$, and $\mathcal{C}$ represents the set of all possible instances. $f : \mathcal{X} \times \mathcal{C} \to \mathbb{R}$ denotes the cost function. Additionally, $\boldsymbol{x} = (x_i)_{1 \le i \le N} \in \{0,1\}^N$ is a binary vector to be optimized, and $\mathcal{X}(C) \subseteq \{0,1\}^N$ denotes the feasible solution space, typically defined by the following equality and inequality constraints.

$$\mathcal{X}(C) = \{\boldsymbol{x} \in \{0,1\}^N \mid \forall i \in [I],\ g_i(\boldsymbol{x}; C) \le 0,\ \forall j \in [J],\ h_j(\boldsymbol{x}; C) = 0\},\ \ I, J \in \mathbb{N},$$

Here, for $i \in [I]$, $g_i : \{0,1\}^N \times \mathcal{C} \to \mathbb{R}$ denotes the inequality constraint, and for $j \in [J]$, $h_j : \{0,1\}^N \times \mathcal{C} \to \mathbb{R}$ denotes the equality constraint. Following UL-based solvers [Wang et al., 2022, Schuetz et al., 2022a, Karalias and Loukas, 2020], we reformulate the constrained problem into an equivalent unconstrained form using the penalty method [Smith et al., 1997]:

$$\min_{\boldsymbol{x}} l(\boldsymbol{x}; C, \boldsymbol{\lambda}),\ \ l(\boldsymbol{x}; C, \boldsymbol{\lambda}) \triangleq f(\boldsymbol{x}; C) + \sum_{i=1}^{I+J} \lambda_i v_i(\boldsymbol{x}; C). \tag{2}$$

Here, for all $i \in [I + J]$, $v : \{0,1\}^N \times \mathcal{C} \to \mathbb{R}$ is the penalty term, which increases when the constraints are violated. For example, the penalty term is defined as follows:

$$\forall i \in [I], j \in [J],\ v_i(\boldsymbol{x}; C) = \max(0, g_i(\boldsymbol{x}; C)),\ \ \forall j \in [J],\ v_j(\boldsymbol{x}; C) = (h_j(\boldsymbol{x}; C))^2$$

and $\boldsymbol{\lambda} = (\lambda_i)_{1 \le i \le I+J} \in \mathbb{R}^{I+J}$ denotes the penalty parameters that control the trade-off between constraint satisfaction and cost optimization. Note that, as $\boldsymbol{\lambda}$ increases, the penalty for constraint violations becomes more significant. In the following, we provide an example of this formulation.

Example: MIS problem The MIS problem is a fundamental NP-hard problem [Karp, 2010], defined as follows. Given an undirected graph $G(V, E)$, an independent set (IS) is a subset of nodes $\mathcal{I} \in V$ where any two nodes are not adjacent. The MIS problem aims to find the largest IS, denoted as $\mathcal{I}^*$. In this study, ρ denotes the IS density, defined as $\rho = |\mathcal{I}|/|V|$. Following Schuetz et al. [2022a], a binary variable x_i is assigned to each node $i \in V$. The MIS problem can be formulated as follows:

$$f(\boldsymbol{x}; G, \lambda) = -\sum_{i \in V} x_i + \lambda \sum_{(i,j) \in E} x_i x_j, \tag{3}$$

where the first term maximizes the number of nodes assigned a value of 1, and the second term penalizes adjacent nodes assigned 1 according to the penalty parameter λ.

2.1 Unsupervised learning based solvers

Learning for CO problems involves training an algorithm $\mathcal{A}_{\boldsymbol{\theta}}(\cdot) : \mathcal{C} \to \{0,1\}^N$ parameterized by a neural network (NN), where $\boldsymbol{\theta}$ denotes the parameters. For a given instance $C \in \mathcal{C}$, this algorithm generates a valid solution $\hat{\boldsymbol{x}} = \mathcal{A}_\theta(C) \in \mathcal{X}(C)$ and aims to minimize $f(\hat{\boldsymbol{x}}; C)$. Several approaches have been proposed to train $\mathcal{A}_\theta$. This study focuses on UL-based solvers, which do not use a labeled solution $\boldsymbol{x}^* \in \operatorname{argmin}_{\boldsymbol{x} \in \mathcal{X}(C)} f(\boldsymbol{x}; C)$ during training [Wang et al., 2022, Schuetz et al., 2022a, Karalias and Loukas, 2020, Amizadeh et al., 2018]. In the following, we outline the details of the UL-based solvers.

The UL-based solvers employ a continuous relaxation strategy to train NN. This continuous relaxation strategy reformulates a CO problem into a continuous optimization problem by converting discrete variables into continuous ones. A typical example of continuous relaxation is expressed as follows:

$$\min_{\boldsymbol{p}} \hat{l}(\boldsymbol{p}; C, \boldsymbol{\lambda}),\ \ \hat{l}(\boldsymbol{p}; C, \boldsymbol{\lambda}) \triangleq \hat{f}(\boldsymbol{p}; C) + \sum_{i=1}^{m+p} \lambda_i \hat{v}_i(\boldsymbol{p}; C),$$

where $\boldsymbol{p} = (p_i)_{1 \le i \le N} \in [0,1]^N$ represents a set of relaxed continuous variables, where each binary variable $x_i \in \{0,1\}$ is relaxed to a continuous counterpart $p_i \in [0,1]$, and $\hat{f} : [0,1]^N \times \mathcal{C} \to \mathbb{R}$ denotes the relaxed form of f such that $\hat{f}(\boldsymbol{x}; C) = f(\boldsymbol{x}; C)$ for $\boldsymbol{x} \in \{0,1\}^N$. The relation between each constraint v_i and its relaxation $\hat{v}_i$ is similar for $i \in [I+J]$, meaning that $\forall i \in [I+J],\ \hat{v}_i(\boldsymbol{x}; C) = v_i(\boldsymbol{x}; C)$ for $\boldsymbol{x} \in \{0,1\}^N$. Wang et al. [2022] and Schuetz et al. [2022a] formulated $\mathcal{A}_\theta(C)$ as the relaxed continuous variables, defined as $\mathcal{A}_\theta(\cdot) : \mathcal{C} \to [0,1]^n$. In the following discussions, we denote

$\mathcal{A}_\theta$ as $\boldsymbol{p}_\theta$ to make the parametrization of the relaxed variables explicit. Then, $\boldsymbol{p}_\theta$ is optimized by directly minimizing the following label-independent function:

$$\hat{l}(\boldsymbol{\theta}; C, \boldsymbol{\lambda}) \triangleq \hat{f}(\boldsymbol{p}_\theta(C); C) + \sum_{i=1}^{I+J} \lambda_i \hat{v}_i(\boldsymbol{p}_\theta(C); C). \tag{4}$$

After training, the relaxed solution $\boldsymbol{p}_{\boldsymbol{\theta}}$ is converted into discrete variables using artificial rounding $\boldsymbol{p}_{\boldsymbol{\theta}}$, where $\forall i \in [N],\ x_i = \mathrm{int}(p_{\boldsymbol{\theta},i}(C))$ based on a threshold [Schuetz et al., 2022a], or alternatively, a greedy method [Wang et al., 2022]. Two types of schemes for UL-based solvers have been developed based on this formulation.

(Type I) Learning generalized heuristics from history/data One approach, proposed by Karalias and Loukas [2020], aims to automatically learn effective heuristics from historical dataset instances $\mathcal{D} = \{C_\mu\}_{\mu=1}^{P}$ and then apply these learned heuristics to a new instance C^*, through inference. Note that this method assumes that either the training dataset is easily obtainable or that meaningful data augmentation is feasible. Specifically, given a set of training instances $\mathcal{D} = (C_\mu)$, sampled independently and identically from a distribution $P(C)$, the goal is to minimize the average loss function $\min_{\boldsymbol{\theta}} \sum_{\mu=1}^{P} l(\boldsymbol{\theta}; C_\mu, \boldsymbol{\lambda})$. However, this method does not guarantee quality for a test instance, C^*. Even if the training instances $\mathcal{D}$ are extensive and the test instance C follows $P(C)$, low average performance $\mathbb{E}_{C\sim P(C)}[\hat{l}(\theta; C)]$ may not guarantee a low $l(\theta; C)$ for on a specific C. To address this issue, Wang and Li [2023] introduced a meta-learning approach where NNs aim to provide good initialization for future instances rather than direct solutions.

(Type II) Learning effective heuristics on a specific single instance Another approach, known as the PI-GNN solver [Schuetz et al., 2022a,b], automatically learns instance-specific heuristics for a single instance using the instance parameter C by directly applying Eq. (4). This approach addresses CO problems on graphs, where $C = G(V, E)$, and employs GNNs for the relaxed variables $p_\theta(G)$. Here, an L-layered GNN is trained to directly minimize $\hat{l}(\boldsymbol{\theta}; C, \boldsymbol{\lambda})$, taking as input a graph G and the embedding vectors on its nodes, and outputting the relaxed solution $\boldsymbol{p}_{\boldsymbol{\theta}}(G) \in [0, 1]^N$. A detailed description of GNNs is provided in Appendix E.2. Note that this setting is applicable even when the training dataset $\mathcal{D}$ is difficult to obtain. The overparameterization of relaxed variables is expected to smooth the objective function by introducing additional parameters to the optimization problem, similar to the kernel method. However, minimizing Eq. 4 for a single instance can be time-consuming compared to the inference process. Nonetheless, for large-scale CO problems, this approach has been reported to outperform other solvers in terms of both computational time and solution quality [Schuetz et al., 2022a,b].

Note that, while both UL-based solvers for multiple instances (Type I) and individual instances (Type II) are valuable, this study focuses on advancing the latter: a UL-based solver for a single instance. Both types of solvers are applicable to cost functions that meet a particular requirement due to their reliance on a gradient-based algorithm to minimize Eq (4).

Assumption 2.1 (Differentiable cost function)**.** The relaxed loss function $\hat{l}(\boldsymbol{\theta}; C, \boldsymbol{\lambda})$ and its partial derivative $\partial \hat{l}(\boldsymbol{\theta}; C, \boldsymbol{\lambda}) / \partial \boldsymbol{\theta}$ are accessible during the optimization process.

These requirements encompass a nonlinear cost function and interactions involving many-body interactions, extending beyond simple two-body interactions.

3 Continuous relaxation annealing for UL-based solvers

In this section, we discuss the practical issues associated with UL-based solvers and then introduce continuous relaxation annealing (CRA) as a proposed solution.

3.1 Motivation: practical issues of UL-based solvers

UL-based solvers (Type II) [Schuetz et al., 2022a,b] are effective in addressing large-scale CO problems. However, these solvers present following two practical issues, highlighted in several recent studies [Wang and Li, 2023, Angelini and Ricci-Tersenghi, 2023]. Additionally, we numerically validate these issues; see Appendix F.1 for detailed results.

(I) Ambiguity in rounding method after learning UL-based solvers employ a continuous relaxation strategy to train NNs and then convert the relaxed continuous variables into discrete binary values through artificial rounding as discussed in Section 2.1. This inherent ambiguity in continuous relaxation strategy often results in potential discrepancies between the optimal solutions of the original discrete CO problem and those of the relaxed continuous one. Continuous relaxation expands the solution space, often producing continuous values that lower the cost compared to an optimal binary value. Indeed, while linear relaxation can provide an optimal solution for discrete problems on bipartite graphs [Hoffman and Kruskal, 2010], it typically results in solutions with $1/2$ values, which is known to half-integrality [Nemhauser and Trotter Jr, 1974]. Existing rounding methods [Schuetz et al., 2022b, Wang et al., 2022] often lose robustness in these scenarios. In practice, PI-GNN solver often outputs values near $1/2$, underscoring the limitations of current rounding techniques for UL-based solvers.

(II) Difficulty in optimizing NNs Recently, Angelini and Ricci-Tersenghi [2023] demonstrated that PI-GNN solver falls short of achieving results comparable to those of the degree-based greedy algorithm (DGA) [Angelini and Ricci-Tersenghi, 2019] when solving the MIS problems on RRGs. Angelini and Ricci-Tersenghi [2023] further emphasized the importance of evaluating UL-based solvers on complex CO problems, where greedy algorithms typically perform worse. A representative example is the MIS problems on RRGs with a constant degree $d > 16$, where a clustering transition in the solution space creates barriers that impede optimization. Moreover, Wang and Li [2023] emphasized the importance of using training/historical datasets, $\mathcal{D} = \{C_\mu\}_{1 \leq \mu \leq P}$, which contain various graphs and initialization using outputs from greedy solvers, such as DGA and RGA for MIS problems. Their numerical analysis indicated that PI-GNN solver tends to get trapped in local optima when directly optimized directly for a single instance without leveraging a training dataset $\mathcal{D}$. However, in a practical setting, systematic methods for generating or collecting training datasets $\mathcal{D}$ to effectively avoid local optima remains unclear. Additionally, training on instances that do not contribute to escaping local optima is time-consuming. Therefore, it is crucial to develop an effective UL-based solver that can operate on a single instance without relying on training data, $\mathcal{D}$. Our numerical experiments, detailed in Appendix F.1, also confirmed this optimization issue. They demonstrated that as problem complexity increases, the PI-GNN solver is often drawn into trivial local optima, $\boldsymbol{p}_{\boldsymbol{\theta}} = \mathbf{0}_N$, in certain problems. This entrapment results in prolonged plateaus that significantly slow down the learning process and, in especially challenging cases, can render learning entirely infeasible. Our numerical experiments, detailed in Appendix F.1, also validated this optimization issue, demonstrating that as the problem complexity increases, PI-GNN solver tends to be absorbed into the trivial local optima $\boldsymbol{p}_{\boldsymbol{\theta}} = \mathbf{0}_N$ in some problems, resulting in prolonged plateaus which significantly decelerates the learning process and, in particularly challenging cases, can render learning entirely infeasible.

3.2 Continuous relaxation annealing

Penalty term to control discreteness and continuity To address these issues, we propose a penalty term to control the balance between discreteness and continuity in the relaxed variables, formulated as follows:

$$\hat{r}(\boldsymbol{p}; C, \boldsymbol{\lambda}, \gamma) = \hat{l}(\boldsymbol{p}; C, \boldsymbol{\lambda}) + \gamma \Phi(\boldsymbol{p}), \;\; \Phi(\boldsymbol{p}) \triangleq \sum_{i=1}^{N} (1 - (2p_i - 1)^\alpha), \;\; \alpha \in \{2n \mid n \in \mathbb{N}_+\}, \quad (5)$$

where $\gamma \in \mathbb{R}$ is a penalty parameter, and the even number α denote a curve rate. When γ is negative, i.e., $\gamma < 0$, the relaxed variables tend to favor the continuous space, smoothing the non-convexity of the objective function $\hat{l}(\boldsymbol{p}; C, \boldsymbol{\lambda})$ due to the convexity of the penalty term $\Phi(\boldsymbol{p})$. In contrast, when γ is positive, i.e., $\gamma > 0$, the relaxed variables tend to favor discrete space, smoothing out the continuous solution into discrete solution. Formally, the following theorem holds as λ approaches $\pm\infty$.

Theorem 3.1. *Assuming the objective function $\hat{l}(\boldsymbol{p}; C)$ is bounded within the domain $[0,1]^N$, as $\gamma \to +\infty$, the relaxed solutions $\boldsymbol{p}^* \in \operatorname{argmin}_{\boldsymbol{p}} \hat{r}(\boldsymbol{p}; C, \boldsymbol{\lambda}, \gamma)$ converge to the original solutions $\boldsymbol{x}^* \in \operatorname{argmin}_{\boldsymbol{x}} l(\boldsymbol{x}; C, \boldsymbol{\lambda})$. Moreover, as $\gamma \to -\infty$, the loss function $\hat{r}(\boldsymbol{p}; C, \boldsymbol{\lambda}, \gamma)$ becomes convex, and the relaxed solution $\mathbf{1}_N/2 = \operatorname{argmin}_{\boldsymbol{p}} \hat{r}(\boldsymbol{p}, C, \boldsymbol{\lambda}, \gamma)$ is unique.*

For the detailed proof, refer to Appendix B.1. Theorem 3.1 can be generalized for any convex function $\Phi(\boldsymbol{p}; C)$ that has a unique maximum at $\mathbf{1}_N/2$ and achieves a global minimum for all $\boldsymbol{p} \in \{0,1\}^N$; an

example is binary cross entropy $\Phi_{\text{CE}}(\boldsymbol{p}) = \sum_{i=1}^{N}(p_i \log p_i + (1-p_i)\log(1-p_i))$, introduced by Sun et al. [2022], Sanokowski et al. [2024] for the UL-based solvers (Type I). Additionally, the penalty term eliminates the stationary point $\boldsymbol{p}^* = \mathbf{0}_N$ described in Section 3.1, preventing convergence to a plateau. For UL-based solvers, the penalty term is expressed as follows:

$$\hat{r}(\boldsymbol{\theta}; C, \boldsymbol{\lambda}, \gamma) = \hat{l}(\boldsymbol{\theta}; C, \boldsymbol{\lambda}) + \gamma\Phi(\boldsymbol{\theta}; C), \tag{6}$$

where $\Phi(\boldsymbol{\theta}; C) \triangleq \Phi(\boldsymbol{p}_\theta(C))$. According to Theorem 3.1, setting a sufficiently large γ value cases the relaxed variables to approach nearly discrete values. We can also generalize this penalty term $\Phi(\boldsymbol{\theta}; C)$, to Potts variables optimization, including coloring problems [Schuetz et al., 2022b], and mixed-integer optimization; refer to Appendix C.1.

Annealing penalty term We propose an annealing strategy that gradually anneals the penalty parameter γ in Eq. (6). Initially, a negative gamma value, i.e., $\gamma < 0$, is chosen to leverage the properties, facilitating broad exploration by smoothing the non-convexity of $\hat{l}(\boldsymbol{\theta}; C, \boldsymbol{\lambda})$ and eliminating the stationary point $\boldsymbol{p}^* = \mathbf{0}_N$ to avoid the plateau, as discussed in Section 3.1. Subsequently, the penalty parameter γ is gradually increased to a positive value, $\gamma > 0$, with each update of the trainable parameters (one epoch), until the penalty term approaches zero, i.e., $\Phi(\boldsymbol{\theta}, C) \approx 0$, to automatically round the relaxed variables by smoothing out suboptimal continuous solutions oscillating between 1 or 0. A conceptual diagram of this annealing process is shown in Fig. 1.

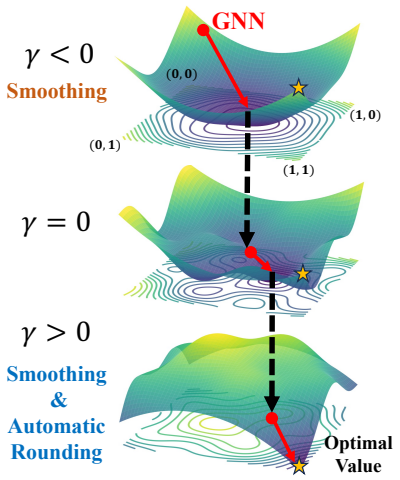

Figure 1: Annealing strategy. When $\gamma < 0$, it facilitates exploration by reducing the non-convexity of the objective function. As γ increases, it promotes optimal discrete solutions by smoothing away suboptimal continuous ones.

Note that employing the binary cross-entropy $\Phi_{\text{CE}}(\boldsymbol{p})$ is infeasible for UL-based solvers when $\gamma > 0$, as the gradient $\partial\Phi_{\text{CE}}(\boldsymbol{p})/\partial p_i$ diverges to $\pm\infty$ at 0 or 1. In deed, when $\gamma = 0$, most relaxed variables typically approach binary values, with a relatively small number of variables oscillating between 0 and 1. This gradient divergence issue in $\Phi_{\text{CE}}(\boldsymbol{p})$ makes the learning infeasible without additional techniques, such as gradient clipping. In contrast, the gradient of the penalty term in Eq. 5, $\partial\Phi(\boldsymbol{p})/\partial p_i$, is bounded within $[-2\alpha, 2\alpha]$ for any γ, preventing the gradient divergence issue seen in $\Phi_{\text{CE}}(\boldsymbol{p})$. Additionally, by increasing α, the absolute value of the gradient near $1/2$ becomes smaller, allowing for control over the smoothing strength toward a discrete solution near $1/2$.

We also propose an early stopping strategy that monitors both the loss function and the penalty term, halting the annealing and learning processes when the penalty term approaches zero, i.e., $\Phi(\boldsymbol{\theta}; C) \approx 0$. Various annealing schedules can be considered; in this study, we employ the following scheduling: $\gamma(\tau + 1) \leftarrow \gamma(\tau) + \varepsilon$, where the scheduling rate $\varepsilon \in \mathbb{R}_+$ is a small constant, and τ denotes the update iterations of the trainable parameters. We refer to the PI-GNN solver with this continuous relaxation annealing as CRA-PI-GNN solver. Here, two additional hyperparameters are introduced: the initial scheduling value $\gamma(0)$ and the scheduling rate ε. Numerical experiments suggest that better solutions are obtained when $\gamma(0)$ is set to a small negative value and ε is kept low. The ablation study are presented in Appendix F.5.

4 Related Work

Previous works on UL-based solvers have addressed various problems, such as MaxCut problems [Yao et al., 2019] and traveling salesman problems [Hudson et al., 2021], using carefully tailored problem-specific objectives. Some studies have also explored constraint satisfaction problems [Amizadeh et al., 2018, Toenshoff et al., 2019], but applying these approaches to broader CO problems often requires problem-specific reductions. Karalias and Loukas [2020] proposed Erdős Goes Neural (EGN) solver, an UL-based solver for general CO problems based on Erdős' probabilistic method. This solver generate solutions through an inference process using training instances. Subsequently, Wang et al.

[2022] proposed an entry-wise concave continuous relaxation, broadening the EGN solver to a wide range of CO problems. In contrast, Schuetz et al. [2022a,b] proposed PI-GNN solver, an UL-based solver for a single CO problems that automatically learns problem-specific heuristics during the training process. However, Angelini and Ricci-Tersenghi [2023], Boettcher [2023] pointed out the optimization difficulties where PI-GNN solver failed to achieve results comparable to those of greedy algorithms. Wang and Li [2023] also claimed optimization issues with PI-GNN solver, emphasizing the importance of learning from training data and history to overcome local optima. They then proposed Meta-EGN solvers, a meta-learning approach that updates NN network parameters for individual CO problem instances. Furthermore, to address these optimization issue, Lin et al. [2023], Sun et al. [2022], Sanokowski et al. [2024] proposed annealing strategy similar to simulated annealing [Kirkpatrick et al., 1983].

5 Experiments

We begin by evaluating the performance of CRA-PI-GNN solver on the MIS and the MaxCut benchmark problems across multiple graphs of varying sizes, demonstrating that CRA effectively overcomes optimization challenges without relying on data/history $\mathcal{D}$. We then extend the evaluation to the DBM problems, showing the applicability to more practical CO problems. For the objective functions and the detailed explanations, refer to Appendix E.1.

5.1 Experimental settings

Baseline methods In all experiments, the baseline methods include the PI-GNN solver [Schuetz et al., 2022a] as the direct baseline of a UL-based solver for a single instance. For the MIS problems, we also consider the random greedy algorithm (RGA) and DGA [Angelini and Ricci-Tersenghi, 2019] as heuristic baselines. For the MaxCut problems, RUN-CSP solver [Toenshoff et al., 2019] is considered as an additional baseline, and a standard greedy algorithm and SDP based approximation algorithm [Goemans and Williamson, 1995] are considered as an additional classical baseline. The parameters for the Goemans-Williamson (GW) approximation are all set according to the settings in Schuetz et al. [2022b]. The implementation used the open-source CVXOPT solver with CVXPY [Mehta, 2019] as the modeling interface. Note that we do not consider UL-based solvers for learning generalized heuristics [Karalias and Loukas, 2020, Wang et al., 2022, Wang and Li, 2023], which rely on training instances $\mathcal{D} = \{C_\mu\}_{\mu=1}^{P}$. The primary objective of this study is to evaluate whether CRA-PI-GNN solver can surpass the performance of both PI-GNN solver and greedy algorithms. However, for the MIS problem, EGN solver [Karalias and Loukas, 2020] and Meta-EGN solver [Wang and Li, 2023] are considered to confirm that CRA can overcome the optimization issues without training instances.

Implementation The objective of the numerical experiments is to compare the CRA-PI-GNN solver with the PI-GNN solver. Thus, we follow the same experimental configuration described as the experiments in Schuetz et al. [2022a], employing a simple two-layer `GCV` and `GraphSAGE` [Hamilton et al., 2017] implemented by the Deep Graph Library [Wang et al., 2019]; Refer to Appendix D.1 for the detailed architectures. We use the AdamW [Kingma and Ba, 2014] optimizer with a learning rate of $\eta = 10^{-4}$ and weight decay of 10^{-2}. The GNNs are trained for up to 5×10^4 epochs with early stopping, which monitors the summarized loss function $\sum_{s=1}^{S} \hat{l}(P_{:,s})$ and the entropy term $\Phi(P; \gamma, \alpha)$ with tolerance 10^{-5} and patience 10^3 epochs; Further details are provided in Appendix D.2. We set the initial scheduling value to $\gamma(0) = -20$ for the MIS and matching problems, and we set $\gamma(0) = -6$ for the MaxCut problems with the scheduling rate $\varepsilon = 10^{-3}$ and curve rate $\alpha = 2$ in Eq. (6). These values are not necessarily optimal, and refining these parameters can lead to better solutions; Refer to Appendix F.5 and Appendix F.6 for an ablation study of these parameters. Once the training process is complete, we apply projection heuristics to map the obtained soft solutions back to discrete solutions using simple projection, where for all $i \in [N]$, we map $p_{\theta,i}$ into 0 if $p_{\theta,i} \leq 0.5$ and $p_{\theta,i}$ into 1 if $p_{\theta,i} > 0.5$. However, due to the early stopping in Section 3.2, the CRA-PI-GNN solver ensures that for all benchmark CO problems, the soft solution at the end of the training process became 0 or 1 within the 32-bit Floating Point range in Pytorch GPU; thus, it is robust against a given threshold, which we set to 0.5 in our experiments. Additionally, no violations of the constraints were observed in our numerical experiments. Thus, following results presented in are feasible solutions.

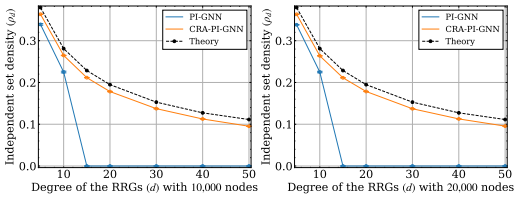

Figure 2: Independent set density of the MIS problem on d-RRG. Results for graphs with $N = 10{,}000$ nodes (Left) and $N = 20{,}000$ nodes (Right). the dashed lines represent the theoretical results [Barbier et al., 2013].

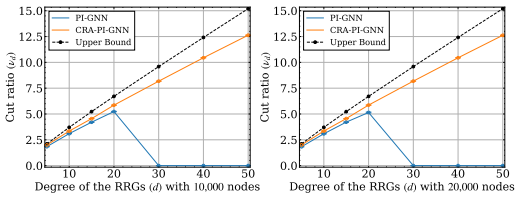

Figure 3: Cut ratio of the MaxCut problem on d-RRG as a function of the degree d Results for $N = 10{,}000$ (Left) and $N = 20{,}000$ (Right). The dashed lines represents the theoretical upper bounds [Parisi, 1980].

Table 1: ApR in MIS problems on RRGs with 10,000 nodes and node degree $d = 20, 100$. "CRA" represents the CRA-PI-GNN solver.

Instance	Method	ApR
20-RRG	RGA	0.776 ± 0.001
	DGA	0.891 ± 0.001
	EGN	0.775 (2023)
	META-EGN	0.887 (2023)
	PI-GNN (GCV)	0.000 ± 0.000
	PI-GNN (SAGE)	0.745 ± 0.003
	CRA (GCV)	0.937 ± 0.002
	CRA (SAGE)	$\mathbf{0.963 \pm 0.001}$
100-RRG	RGA	0.663 ± 0.001
	DGA	0.848 ± 0.002
	PI-GNN (GCV)	0.000 ± 0.000
	PI-GNN (SAGE)	0.000 ± 0.000
	CRA (GCV)	0.855 ± 0.004
	CRA (SAGE)	$\mathbf{0.924 \pm 0.001}$

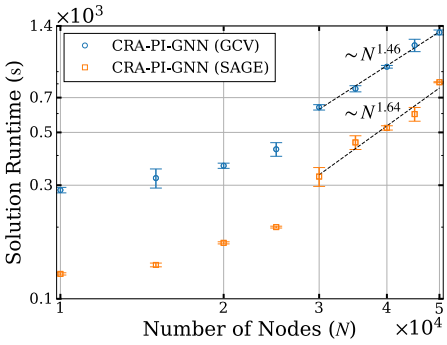

Figure 4: (Right) computational runtime (in seconds) of the CRA-PI-GNN solvers with the GraphSage and Conv architectures on 100-RRG with varying numbers of nodes N.Error bars represent the standard deviations of the results.

Evaluation metrics Following Karalias and Loukas [2020], We use the approximation ratio (ApR), formulated as $\mathrm{ApR} = f(\boldsymbol{x}; C)/f(\boldsymbol{x}^*; C)$, where $\boldsymbol{x}^*$ is optimal solution. For the MIS problems, we evaluate the ApRs using the theoretical optimal cost [Barbier et al., 2013] and the independent set density ρ relative to the theoretical results. For the MaxCut problems on RRGs, we adopt the cut ratio ν against the theoretical upper bound [Parisi, 1980, Dembo et al., 2017]; see Appendix E.1 for the details. All the results for the MIS and MaxCut problems are summarized based on 5 RRGs with different random seeds. In the case of the MaxCut Gset problem, the ApR is calculated compared to the known best cost functions. Regarding the DBM problems, we calculate the ApR against the global optimal, identified using Gurobi 10.0.1 solver with default settings.

5.2 MIS problems

Degree dependency of solutions using CRA First, we compare the performance of the PI-GNN and CRA-PI-GNN solvers using GCV, as in Schuetz et al. [2022a]. Fig. 2 shows the independent set density ρ_d as a function of degree d obtained by the PI-GNN and CRA-PI-GNN solvers compared with the theoretical results [Barbier et al., 2013]. Across all degrees d, the CRA-PI-GNN solver outperformed the PI-GNN solver and approached the theoretical results, whereas the PI-GNN solver fail to find valid solutions, especially for $d \geq 15$, as pointed by the previous studies [Angelini and Ricci-Tersenghi, 2023, Wang and Li, 2023].

Response to Angelini and Ricci-Tersenghi [2023] and Wang and Li [2023] MIS problems on RRGs with a degree d larger than 16 is known to be hard problems [Barbier et al., 2013]. As discussed in Section 3.1, Angelini and Ricci-Tersenghi [2023], Wang and Li [2023] have posted the optimization concerns on UL-based solvers. However, we call these claim into question by substantially outperforming heuristics DGA and RGA for the MIS on graphs with $d = 20, 100$,

without training/historical instances $\mathcal{D} = \{G^\mu\}_{\mu=1}^n$, as shown in Table 1. See Appendix 6 for the results of solving all other Gsets, where consistently, CRA-PI-GNN provides better results as well. A comparison of the sampling-based solvers, RL-based solvers, SL-based solvers, Gurobi, and MIS-specific solvers is presented in Appendix F.2.

Acceleration of learning speed We also compared the learning curve between PI-GNN and CRA-PI-GNN solver to confirmed that the CRA-PI-GNN solver does not become trapped in the plateau, $\boldsymbol{p}_N = \mathbf{0}_N$, as discussed in Section 3.1. Fig. 5 shows the dynamics of the cost functions for the MIS problems with $N = 10{,}000$ across $d = 3, 5, 20, 100$. Across all degrees, CRA-PI-GNN solver achieves a better solution with fewer epochs than PI-GNN solver. Specifically, PI-GNN solver becomes significantly slower due to getting trapped in the plateau even for graphs with low degrees, such as $d = 3, 5$. In contrast, CRA-PI-GNN solver can effectively escape from plateaus through the smoothing and automatic rounding of the penalty term when the negative parameter $\gamma > 0$.

Computational scaling We next evaluate the computational scaling of the CRA-PI-GNN solver for MIS problems with large-scale RRGs with a node degree of 100 in Fig. 4, following previous studies [Schuetz et al., 2022a, Wang and Li, 2023]. Fig. 4 demonstrated a moderate super-linear scaling of the total computational time, approximately $\sim N^{1.4}$ for GCN and $\sim N^{1.7}$ for GraphSage. This performance is nearly identical to that of the PI-GNN solver [Schuetz et al., 2022a] for problems on RRGs with lower degrees. It is important note that the runtimes of CRA-PI-GNN solver heavily depend on the optimizer for GNNs and annealing rate ε; thus this scaling remains largely unchanged for problems other than the MIS on 100 RRG. Additionally, CRA demonstrate that the runtime remains nearly constant as graph order and density increase, indicating effective scalability with denser graphs which is presented in Appendix F.2.

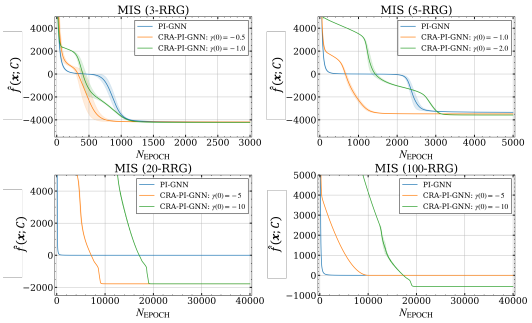

Figure 5: The dynamics of cost function for MIS problems on RRGs with $N = 10{,}000$ nodes varying degrees d as a function of the number of parameters updates N_{EPOCH}.

5.3 MaxCut problem

Degree dependency of solutions We first compare the performances of PI-GNN and CRA-PI-GNN solvers with GCV, following Schuetz et al. [2022a]. Fig. 3 shows the cut ratio ν_d as a function of the degree d compared to the theoretical upper bound [Parisi, 1980, Dembo et al., 2017]. Across all degrees d, CRA-PI-GNN solver also outperforms PI-GNN solver, approaching the theoretical upper bound. In contrast, PI-GNN solver fails to find valid solutions for $d > 20$ as with the case of the MIS problems in Section 5.2.

Standard MaxCut benchmark test Following Schuetz et al. [2022a], we next conducted additional experiments on standard MaxCut benchmark instances based on the publicly available Gset dataset [Ye, 2003], which is commonly used to evaluate MaxCut algorithms. Here, we provide benchmark results for seven distinct graphs with thousands of nodes, including Erdös-Renyi graphs with uniform edge probability, graphs in which the connectivity decays gradually from node 1 to N, 4-regular toroidal graphs, and a very large Gset instance with $N = 10{,}000$ nodes. Table 2 shows, across all problems, CRA-PI-GNN solver outperforms both the PI-GNN, RUN-CSP solvers and other greedy algorithm. See Appendix 6 for the results of solving all other Gsets, where consistently, CRA-PI-GNN provides better results as well.

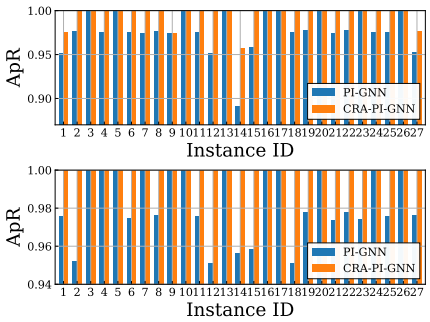

Figure 6: ApR on DBM problems.

Table 2: ApR for MaxCut on Gset

GRAPH	(NODES, EDGES)	GREEDY	SDP	RUN-CSP	PI-GNN	**CRA**
G14	(800, 4,694)	0.946	0.970	0.960	0.988	**0.994**
G15	(800, 4,661)	0.939	0.958	0.960	0.980	**0.992**
G22	(2,000, 19,990)	0.923	0.77	0.975	0.987	**0.998**
G49	(3,000, 6,000)	**1.000**	**1.00**	**1.000**	0.986	**1.000**
G50	(3,000, 6,000)	**1.000**	**1.000**	**1.000**	0.990	**1.000**
G55	(5,000, 12,468)	0.892	–	0.982	0.983	**0.991**
G70	(10,000, 9,999)	0.886	–	0.970	0.982	**0.992**

5.4 Diverse bipartite matching

To evaluate the applicability of the CRA-PI-GNN solver to more practical problems not on graphs, we conducted experiments on DBM problems [Ferber et al., 2020, Mulamba et al., 2020, Mandi et al., 2022]; refer to Appendix E.1 for details. This problems consists of 27 distinct instances with varying properties, and each instance comprises 100 nodes representing scientific publications, divided into two groups of 50 nodes N_1 and N_2. The optimization is formulated as follows:

$$l(\boldsymbol{x}; C, M, \boldsymbol{\lambda}) = -\sum_{ij} C_{ij}x_{ij} + \lambda_1 \sum_i \mathrm{ReLU}\Big(\sum_j x_{ij} - 1\Big) + \lambda_2 \sum_j \mathrm{ReLU}\Big(\sum_i x_{ij} - 1\Big)$$
$$+ \lambda_3 \mathrm{ReLU}\Big(p\sum_{ij} x_{ij} - \sum_{ij} M_{ij}x_{ij}\Big) + \lambda_4 \mathrm{ReLU}\Big(q\sum_{ij} x_{ij} - \sum_{ij}(1-M_{ij})x_{ij}\Big),$$

where $C \in \mathbb{R}^{N_1 \times N_2}$ represents the likelihood of a link between each pair of nodes, an indicator M_{ij} is set to 0 if article i and j share the same subject field (1 otherwise) $\forall i \in N_1$, and $j \in N_2$. The parameters $p, q \in [0, 1]$ represent the probability of pairs sharing their field and of unrelated pairs, respectively. As in Mandi et al. [2022], we explore two variations of this problem, with $p = q =$ being 25% and 5%, respectively, and these variations are referred to as Matching-1 and Matching-2, respectively. In this experiment, we set $\lambda_1 = \lambda_2 = 10$ and $\lambda_3 = \lambda_4 = 25$. Fig 6 shows that the CRA-PI-GNN solver can find better solutions across all instances.

6 Conclusion

This study proposes CRA strategy to address the both optimization and rounding issue in UL-based solvers. CRA strategy introduces a penalty term that dynamically shifts from prioritizing continuous solutions, where the non-convexity of the objective function is effectively smoothed, to enforcing discreteness, thereby eliminating artificial rounding. Experimental results demonstrate that CRA-PI-GNN solver significantly outperforms both PI-GNN solver and greedy algorithms across various complex CO problems, including MIS, MaxCut, and DBM problems. CRA approach not only enhances solution quality but also accelerates the learning process.

Limitation In this numerical experiments, most hyperparameters were fixed to their default values, as outlined in Section 5.1, with minimal tuning. However, tuning may be necessary for certain problems or to further enhance performance.

References

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. *arXiv preprint arXiv:2110.05291*, 2021.

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.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

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.

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.

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.

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, pages 242–251. SPIE, 2019.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Emre Yolcu and Barnabás Póczos. Learning local search heuristics for boolean satisfiability. *Advances in Neural Information Processing Systems*, 32, 2019.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

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

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.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

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.

Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022a.

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.

Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. In *International Conference on Learning Representations*, 2018.

Maria Chiara Angelini and Federico Ricci-Tersenghi. Modern graph neural networks do worse than classical greedy algorithms in solving combinatorial optimization problems like maximum independent set. *Nature Machine Intelligence*, 5(1):29–31, 2023.

Maria Chiara Angelini and Federico Ricci-Tersenghi. Monte carlo algorithms are very effective in finding the largest independent set in sparse random graphs. *Physical Review E*, 100(1):013302, 2019.

Haoyu Wang and Pan Li. Unsupervised learning for combinatorial optimization needs meta-learning. *arXiv preprint arXiv:2301.03116*, 2023.

Alan J Hoffman and Joseph B Kruskal. Integral boundary points of convex polyhedra. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 49–76, 2010.

George L Nemhauser and Leslie E Trotter Jr. Properties of vertex packing and independence system polyhedra. *Mathematical programming*, 6(1):48–61, 1974.

Martin JA Schuetz, J Kyle Brubaker, Zhihuai Zhu, and Helmut G Katzgraber. Graph coloring with physics-inspired graph neural networks. *Physical Review Research*, 4(4):043131, 2022b.

Haoyu Peter Wang, Nan Wu, Hang Yang, Cong Hao, and Pan Li. Unsupervised learning for combinatorial optimization with principled objective relaxation. *Advances in Neural Information Processing Systems*, 35:31444–31458, 2022.

László Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1): 1–7, 1979.

Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6): 1115–1145, 1995.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

Alice E Smith, David W Coit, Thomas Baeck, David Fogel, and Zbigniew Michalewicz. Penalty functions. *Handbook of evolutionary computation*, 97(1):C5, 1997.

Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010.

Haoran Sun, Etash K Guha, and Hanjun Dai. Annealed training for combinatorial optimization on graphs. *arXiv preprint arXiv:2207.11542*, 2022.

Sebastian Sanokowski, Wilhelm Berghammer, Sepp Hochreiter, and Sebastian Lehner. Variational annealing on graphs for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Run-csp: unsupervised learning of message passing networks for binary constraint satisfaction problems. *CoRR, abs/1909.08387*, 2019.

Stefan Boettcher. Inability of a graph neural network heuristic to outperform greedy algorithms in solving combinatorial optimization problems. *Nature Machine Intelligence*, 5(1):24–25, 2023.

Xi Lin, Zhiyuan Yang, Xiaoyuan Zhang, and Qingfu Zhang. Continuation path learning for homotopy optimization. In *International Conference on Machine Learning*, pages 21288–21311. PMLR, 2023.

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

Hermish Mehta. Cvx graph algorithms. `https://github.com/hermish/cvx-graph-algorithms`, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

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

Jean Barbier, Florent Krzakala, Lenka Zdeborová, and Pan Zhang. The hard-core model on random graphs revisited. In *Journal of Physics: Conference Series*, volume 473, page 012021. IOP Publishing, 2013.

Giorgio Parisi. A sequence of approximated solutions to the sk model for spin glasses. *Journal of Physics A: Mathematical and General*, 13(4):L115, 1980.

Amir Dembo, Andrea Montanari, and Subhabrata Sen. Extremal cuts of sparse random graphs. 2017.

Y. Ye. The gset dataset. `https://web.stanford.edu/~yyye/yyye/Gset/`, 2003.

Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. Mipaal: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1504–1511, 2020.

Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, and Tias Guns. Contrastive losses and solution caching for predict-and-optimize. *arXiv preprint arXiv:2011.05354*, 2020.

Jayanta Mandi, Vıctor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. Decision-focused learning: through the lens of learning to rank. In *International Conference on Machine Learning*, pages 14935–14947. PMLR, 2022.

Mohsen Bayati, David Gamarnik, and Prasad Tetali. Combinatorial approach to the interpolation method and scaling limits in sparse random graphs. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 105–114, 2010.

Amin Coja-Oghlan and Charilaos Efthymiou. On independent sets in random graphs. *Random Structures & Algorithms*, 47(3):436–486, 2015.

Bahram Alidaee, Gary A Kochenberger, and Ahmad Ahmadian. 0-1 quadratic programming approach for optimum solutions of two scheduling problems. *International Journal of Systems Science*, 25 (2):401–408, 1994.

Hartmut Neven, Geordie Rose, and William G Macready. Image recognition with an adiabatic quantum computer i. mapping to quadratic unconstrained binary optimization. *arXiv preprint arXiv:0804.4457*, 2008.

Michel Deza and Monique Laurent. Applications of cut polyhedra—ii. *Journal of Computational and Applied Mathematics*, 55(2):217–247, 1994.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

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.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. DIMES: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems 35*, 2022.

Haoran Sun, Katayoon Goshvadi, Azade Nova, Dale Schuurmans, and Hanjun Dai. Revisiting sampling for combinatorial optimization. In *International Conference on Machine Learning*, pages 32859–32874. PMLR, 2023.

Katayoon Goshvadi, Haoran Sun, Xingchao Liu, Azade Nova, Ruqi Zhang, Will Grathwohl, Dale Schuurmans, and Hanjun Dai. Discs: A benchmark for discrete sampling. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 79035–79066. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/f9ad87c1ebbae8a3555adb31dbcacf44-Paper-Datasets_and_Benchmarks.pdf`.

Holger H Hoos and Thomas Stützle. Satlib: An online resource for research on sat. *Sat*, 2000: 283–292, 2000.

A Overview

This supplementary material provides extended explanations, implementation details, and additional results.

B Derivation

B.1 Proof of Theorem 3.1

First, we present three lemmas, and then we demonstrate Theorem 3.1 based on these lemmas.

Lemma B.1. *For any even natural number $\alpha \in \{2n \mid n \in \mathbb{N}_+\}$, the function $\phi(p) = 1 - (2p-1)^\alpha$ defined on $[0,1]$ achieves its maximum value of 1 when $p = 1/2$ and its minimum value of 0 when $p = 0$ or $p = 1$.*

Proof. The derivative of $\phi(p)$ relative to p is ${}^{d\phi(p)}/_{dp} = -2\alpha(2p-1)$, which is zero when $p = 1/2$. This is a point where the function is maximized because the second derivative ${}^{d^2\phi(p)}/_{dp^2} = -4\alpha \leq 0$. In addition, this function is concave and symmetric relative to $p = 1/2$ because α is an even natural number, i.e., $\phi(p) = \phi(1-p)$, thereby achieving its minimum value of 0 when $p = 0$ or $p = 1$. □

Lemma B.2. *For any even natural number $\alpha \in \{2n \mid n \in \mathbb{N}_+\}$, if $\gamma \to +\infty$, minimizing the penalty term $\gamma\Phi(\boldsymbol{p}) = \gamma\sum_{i=1}^N (1-(2p_i-1)^\alpha) = \gamma\sum_{i=1}^N \phi(p_i)$ enforces that the for all $i \in [N]$, p_i is either 0 or 1 and, if $\gamma \to -\infty$, the penalty term enforces $\boldsymbol{p} = \mathbf{1}_N/2$.*

Proof. From Lemma B.1, as $\gamma \to +\infty$, $\phi(p)$ is minimal value when, for any $i \in [N]$, $p_i = 0$ or $p_i = 1$. As $\gamma \to -\infty$, $\phi(p;\alpha,\gamma)$ is minimal value when, for any $i \in [N]$, $p_i = 1/2$. □

Lemma B.3. *For any even number $\alpha \in \{2n \mid n \in \mathbb{N}_+\}$, $\gamma\Phi(\boldsymbol{p})$ is concave when $\lambda > 0$ and is a convex function when $\lambda < 0$.*

Proof. Note that $\gamma\Phi(\boldsymbol{p}) = \gamma\sum_{i=1}^N \phi(p_i) = \gamma\sum_{i=1}^N (1-(2p_i-1)^\alpha)$ is separable across its components p_i. Thus, it is sufficient to prove that each $\gamma\phi_i(p_i;\alpha)$ is concave or convex in p_i because the sum of the concave or convex functions is also concave (and vice versa). Thus, we consider the second derivative of $\gamma\phi_i(p_i)$ with respect to p_i:

$$\gamma\frac{d^2\phi_i(p_i)}{dp_i^2} = -4\gamma\alpha.$$

If $\gamma > 0$, the second derivative is negative for all $p_i \in [0,1]$, and this completes the proof that $\gamma\Phi(\boldsymbol{p})$ is a concave function when γ is positive (and vice versa). □

Combining Lemma B.1, Lemma B.2 and Lemma B.3, one can show the following theorem.

Theorem B.4. *Under the assumption that the objective function $\hat{l}(\boldsymbol{p};C)$ is bounded within the domain $[0,1]^N$, as $\gamma \to +\infty$, the soft solutions $\boldsymbol{p}^* \in \mathrm{argmin}_{\boldsymbol{p}}\hat{r}(\boldsymbol{p};C,\boldsymbol{\lambda},\gamma)$ converge to the original solutions $\boldsymbol{x}^* \in \mathrm{argmin}_{\boldsymbol{x}} l(\boldsymbol{x};C,\boldsymbol{\lambda})$. In addition, as $\gamma \to -\infty$, the loss function $\hat{r}(\boldsymbol{p};C,\boldsymbol{\lambda},\gamma)$ becomes convex, and the soft solution ${}^{\mathbf{1}_N}/_2 = \mathrm{argmin}_{\boldsymbol{p}}\hat{r}(\boldsymbol{p},C,\boldsymbol{\lambda},\gamma)$ is unique.*

Proof. As $\lambda \to +\infty$, the penalty term $\Phi(\boldsymbol{p})$ dominates the loss function $\hat{r}(\boldsymbol{p};C,\boldsymbol{\lambda},\gamma)$. According to Lemma B.2, this penalty term forces the optimal solution $\boldsymbol{p}^*$ to a binary vector whose components, for all $i \in [N]$ p_i^* that are either 0 or 1 because any non-binary value results in an infinitely large penalty. This effectively restricts the feasible region to the vertices of the unit hypercube, which correspond to the binary vector in $\{0,1\}^N$. Thus, as $\lambda \to \infty$, the solutions to the relaxed problem converge to those of the original problem. As $\lambda \to -\infty$, the penalty term $\Phi(\boldsymbol{p})$ also dominates the loss function $\hat{r}(\boldsymbol{p};C,\boldsymbol{\lambda},\gamma)$ and the $\hat{r}(\boldsymbol{p};C,\boldsymbol{\lambda})$ convex function from Lemma B.3. According to Lemma B.2, this penalty term forces the optimal solution $\boldsymbol{p}^* = \mathbf{1}_N/2$. □

The theorem holds for the cross entropy penalty given by

$$\Phi(\boldsymbol{p}) = \sum_{i=1}^{N} \left(p_i \log(p_i) + (1-p_i)\log(1-p_i)\right) \tag{7}$$

in the UL-based solver using data or history [Sun et al., 2022, Sanokowski et al., 2024] because $\Phi(\boldsymbol{p})$ can similarly satisfy Lemma B.1, Lemma B.2 and Lemma B.3.

Corollary B.5. *Theorem B.4 holds for the following penalty term:*

$$\Phi(\boldsymbol{p}) = \sum_{i=1}^{N} \left(p_i \log(p_i) + (1-p_i)\log(1-p_i)\right). \tag{8}$$

C Generalization of CRA

C.1 Generalization for to Potts variable optimization

This section generalize the penalty term $\Phi(\boldsymbol{\theta}; C)$ introduced for binary variables to K-Potts variables. K-Potts variable is the Kronecker delta $\delta(x_i, x_j)$ which equas one whenever $x_i = x_j$ and zero otherwise and a decision variable takes on K different values, $\forall i \in [N],\ x_i = 1, \ldots, K$. For example, graph K-coloring problems can be expressed as

$$f(\boldsymbol{x}; G(V,E)) = -\sum_{(i,j)\in E} \delta(x_i, x_j) \tag{9}$$

For Potts variables, the output of the GNN is $\boldsymbol{h}_{\boldsymbol{\theta}}^{L} = P(\boldsymbol{\theta}; C) \in \mathbb{R}^{N\times K}$ and the penalty term can be generalized as follows.

$$\Phi(\boldsymbol{\theta}; C) = \sum_{i=1}^{N} \left(1 - \sum_{k=1}^{K} \left(K P_{i,k}(\boldsymbol{\theta}; C) - 1\right)^{\alpha}\right). \tag{10}$$

D Additional implementation details

D.1 Architecture of GNNs

We describe the details of the GNN architectures used in all numerical experiments. The first convolutional layer takes H_0-dimensional node embedding vectors, $\boldsymbol{h}_{\boldsymbol{\theta}}^{0}$ for each node, as input, yielding H_1-dimensional feature vectors $\boldsymbol{h}_{\boldsymbol{\theta}}^{1}$. Then, the ReLU function is applied as a component-wise nonlinear transformation. The second convolutional layer takes the H_1-dimensional feature vectors, $\boldsymbol{h}_{\boldsymbol{\theta}}^{1}$, as input, producing a $H^{(2)}$-dimensional vector $\boldsymbol{h}_{\boldsymbol{\theta}}^{2}$. Finally, a sigmoid function is applied to the $H^{(2)}$-dimensional vector $\boldsymbol{h}_{\boldsymbol{\theta}}^{2}$, and the output is the soft solution $\boldsymbol{p}_{\boldsymbol{\theta}} \in [0,1]^N$. As in [Schuetz et al., 2022a], we set $H_0 = \mathrm{int}(N^{0.8})$ or , $H_1 = \mathrm{int}(N^{0.8}/2)$ and $H^2 = 1$ for both `GCN` and `GraphSAGE` .

D.2 Training settings

For all numerical experiments, we use the AdamW [Kingma and Ba, 2014] optimizer with a learning rate of $\eta = 10^{-4}$ and weight decay of 10^{-2}, and we train the GNNs for up to 10^5 epochs with early stopping set to the absolute tolerance 10^{-5} and patience 10^3. As discussed in Schuetz et al. [2022a], the GNNs are initialized with five different random seeds for a single instance because the results are dependent on the initial values of the trainable parameters; thus selecting the best solution.

E Experiment details

E.1 Benchmark problems

MIS problems There are some theoretical results for MIS problems on RRGs with the node degree set to d, where each node is connected to exactly d other nodes. The MIS problem is a

fundamental NP-hard problem [Karp, 2010] defined as follows. Given an undirected graph $G(V, E)$, an independent set (IS) is a subset of nodes $\mathcal{I} \in V$ where any two nodes in the set are not adjacent. The MIS problem attempts to find the largest IS, which is denoted $\mathcal{I}^*$. In this study, ρ denotes the IS density, where $\rho = |\mathcal{I}|/|V|$. To formulate the problem, a binary variable x_i is assigned to each node $i \in V$. Then the MIS problem is formulated as follows:

$$f(\boldsymbol{x}; G, \lambda) = -\sum_{i \in V} x_i + \lambda \sum_{(i,j) \in E} x_i x_j, \tag{11}$$

where the first term attempts to maximize the number of nodes assigned 1, and the second term penalizes the adjacent nodes marked 1 according to the penalty parameter λ. In our numerical experiments, we set $\lambda = 2$, following Schuetz et al. [2022a], no violation is observed as in [Schuetz et al., 2022a]. First, for every d, a specific value ρ_d^*, which is dependent on only the degree d, exists such that the independent set density $|\mathcal{I}^*|/|V|$ converges to ρ_d^* with a high probability as N approaches infinity [Bayati et al., 2010]. Second, a statistical mechanical analysis provides the typical MIS density ρ_d^{Theory}, as shown in Fig. 2, and we clarify that for $d > 16$, the solution space of $\mathcal{I}$ undergoes a clustering transition, which is associated with hardness in sampling [Barbier et al., 2013] because the clustering is likely to create relevant barriers that affect any algorithm searching for the MIS $\mathcal{I}^*$. Finally, the hardness is supported by analytical results in a large d limit, which indicates that, while the maximum independent set density is known to have density $\rho_{d\to\infty}^* = 2\log(d)/d$, to the best of our knowledge, there is no known algorithm that can find an independent set density exceeding $\rho_{d\to\infty}^{\mathrm{alg}} = \log(d)/d$ [Coja-Oghlan and Efthymiou, 2015].

MaxCut problems The MaxCut problem is also a fundamental NP-hard problem [Karp, 2010] with practical application in machine scheduling [Alidaee et al., 1994], image recognition [Neven et al., 2008] and electronic circuit layout design [Deza and Laurent, 1994]. The MaxCut problem is also a fundamental NP-hard problem [Karp, 2010] Given an graph $G = (V, E)$, a cut set $\mathcal{C} \in E$ is defined as a subset of the edge set between the node sets dividing $(V_1, V_2 \mid V_1 \cup V_2 = V,\ V_1 \cap V_2 = \emptyset)$. The MaxCut problems aim to find the maximum cut set, denoted $\mathcal{C}^*$. Here, the cut ratio is defined as $\nu = |\mathcal{C}|/|\mathcal{V}|$, where $|\mathcal{C}|$ is the cardinality of the cut set. To formulate this problem, each node is assigned a binary variable, where $x_i = 1$ indicates that node i belongs to V_1, and $x_i = 0$ indicates that the node belongs to V_2. Here, $x_i + x_j - 2x_i x_j = 1$ holds if the edge $(i, j) \in \mathcal{C}$. As a result, we obtain the following:

$$f(\boldsymbol{x}; G) = \sum_{i<j} A_{ij}(2x_i x_j - x_i - x_j). \tag{12}$$

This study has also focused on the MaxCut problems on d-RRGs, for which several theoretical results have been established. Specifically, for each d, the maximum cut ratio is given by $\nu_d^* \approx d/4 + P_*\sqrt{d/4} + \mathcal{O}(\sqrt{d})$, where $P_* = 0.7632\ldots$ with a high probability as N approaches infinity [Parisi, 1980, Dembo et al., 2017]. Thus, we take $\nu_d^{\mathrm{UB}} = d/4 + P_*\sqrt{d/4}$ as an upper bound for the maximum cut ratio in the large n limit.

DBM problems Here, the topologies are taken from the Cora citation network [Sen et al., 2008], where each node has 1,433 bag-of-words features, and each edge represents likelihood, as predicted by a machine learning model. Mandi et al. [2022] focused on disjoint topologies within the given topology, and they created 27 distinct instances with varying properties. Each instance comprises 100 nodes representing scientific publications, divided into two groups of 50 nodes N_1 and N_2. The optimization task is to find the maximum matching, where diversity constraints ensure connections among papers in the same field and between papers of different fields. This is formulated using a penalty method as follows.

$$\begin{aligned} l(\boldsymbol{x}; C, M, \boldsymbol{\lambda}) = &-\textstyle\sum_{ij} C_{ij}x_{ij} + \lambda_1 \sum_i \mathrm{ReLU}\Big(\sum_j x_{ij} - 1\Big) + \lambda_2 \sum_j \mathrm{ReLU}\Big(\sum_i x_{ij} - 1\Big) \\ &+ \lambda_3 \mathrm{ReLU}\Big(p\textstyle\sum_{ij} x_{ij} - \sum_{ij} M_{ij}x_{ij}\Big) + \lambda_4 \mathrm{ReLU}\Big(q\textstyle\sum_{ij} x_{ij} - \sum_{ij}(1 - M_{ij})x_{ij}\Big), \end{aligned} \tag{13}$$

where $C \in \mathbb{R}^{N_1 \times N_2}$ represents the likelihood of a link between each pair of nodes, an indicator M_{ij} is set to 0 if article i and j share the same subject field (1 otherwise) $\forall i \in N_1$, and $j \in N_2$. The parameters $p, q \in [0, 1]$ represent the probability of pairs sharing their field and of unrelated pairs, respectively. As in Mandi et al. [2022], we explore two variations of this problem, with $p = q =$ being 25% and 5%, respectively, and these variations are referred to as Matching-1 and Matching-2, respectively. In this experiment, we set $\lambda_1 = \lambda_2 = 10$ and $\lambda_3 = \lambda_4 = 25$.

E.2 GNNs

A GNN [Gilmer et al., 2017, Scarselli et al., 2008] is a specialized neural network for representation learning of graph-structured data. GNNs learn a vectorial representation of each node in two steps, i.e., the aggregate and combine steps. The aggregate step employs a permutation-invariant function to generate an aggregated node feature, and in the combine step, the aggregated node feature is passed through a trainable layer to generate a node embedding, known as "message passing" or the "readout phase." Formally, for a given graph $G = (V, E)$, where each node feature $\boldsymbol{h}_v^0 \in \mathbb{R}^{H_0}$ is attached to each node $v \in V$, the GNN updates the following two steps iteratively. First, the aggregate step at each l-th layer is defined as follows:

$$\boldsymbol{a}_v^l = \mathrm{Aggregate}_{\boldsymbol{\theta}}^l \left(\{ h_u^{l-1}, \forall u \in \mathcal{N}_v \} \right), \tag{14}$$

where the neighborhood of $v \in V$ is denoted $\mathcal{N}_v = \{u \in V \mid (v, u) \in E\}$, $\boldsymbol{h}_u^{l-1}$ is the node feature of the neighborhood, and $\boldsymbol{a}_v^l$ is the aggregated node feature of the neighborhood. Then, the combined step at each l-th layer is defined as follows:

$$\boldsymbol{h}_v^l = \mathrm{Combine}_{\boldsymbol{\theta}}^l (\boldsymbol{h}_v^{l-1}, \boldsymbol{a}_v^l), \tag{15}$$

where $\boldsymbol{h}_v^l \in \mathbb{R}^{H_l}$ denotes the node representation at the l-th layer. Here, the hyperparameters for the total number of layers L and the intermediate vector dimension N^l are determined empirically. Although numerous implementations of GNN architectures have been proposed to date, the most basic and widely used architecture is the GCN [Scarselli et al., 2008], which is given as follows:

$$\boldsymbol{h}_v^l = \sigma \left(W^l \sum_{u \in \mathcal{N}(v)} \frac{\boldsymbol{h}_u^{l-1}}{|\mathcal{N}(v)|} + B^l \boldsymbol{h}_v^{l-1} \right), \tag{16}$$

where W^l and B^l are trainable parameters, $|\mathcal{N}(v)|$ serves as a normalization factor, and $\sigma : \mathbb{R}^{H_l} \to \mathbb{R}^{H_l}$ is some component-wise nonlinear activation function, e.g., the sigmoid or ReLU function.

F Additional experiments

F.1 Numerical validation of practical issues presented in Section 3.1

In this sectioin, we will examine the issue (I) with continuous relaxations and the issue (II), the difficulties of optimization, as pointed out by previous studies [Wang and Li, 2023, Angelini and Ricci-Tersenghi, 2023], in the NP-hard problems of MIS and the MaxCut problem. Therefere, we conducted numerical experiments using the PI-GNN solver for MIS and MaxCut problems on RRGs with higher degrees. To ensure the experimental impartiality, we adhered to the original settings of the PI-GNN solver [Schuetz et al., 2022b]. Refer to Section E for the detailed experimental settings. Fig. 7 (top) shows the solutions obtained by the PI-GNN solver as a function of the degree d for the MIS and MaxCut problems with varying system sizes N. These results indicate that finding independent and cut sets becomes unfeasible as the RRG becomes denser. In addition, to clarify the reasons for these failures, we analyzed the dynamics of the cost function for MIS problems with $N = 10{,}000$, with a specific focus on a graph with degrees $d = 5$ and $d = 20$, as depicted in Fig. 7 (bottom). For the $d = 5$ case, the cost function goes over the plateau of $\hat{l}(\boldsymbol{\theta}; G, \boldsymbol{\lambda}) = 0$ with $\boldsymbol{p}_\theta(G) = \mathbf{0}_N$, as investigated in the histogram, eventually yielding a solution comparable to those presented by Schuetz et al. [2022a]. Conversely, in the $d = 20$ case, the cost function remains stagnant on the plateau of $\hat{l}(\boldsymbol{\theta}; G, \boldsymbol{\lambda}) = 0$ with $\boldsymbol{p}_\theta(G) = \mathbf{0}_N$, thereby failing to find any independent nodes. Interpreting this phenomenon, we hypothesize that the representation capacity of the GNN is sufficiently large, leading us to consider the optimization of $\hat{L}_{\mathrm{MIS}}(\boldsymbol{\theta}; G, \lambda)$ and $\hat{L}_{\mathrm{MaxCut}}(\boldsymbol{\theta}; G)$ as a variational optimization problem relative to $\boldsymbol{p}_\theta$. In this case, $\boldsymbol{p}_\theta^* = \mathbf{0}_N$ satisfies the first-order variational optimality conditions $\delta \hat{l}_{\mathrm{MIS}} / \delta \boldsymbol{p}_\theta |_{\boldsymbol{p}_\theta = \boldsymbol{p}^*} = \delta \hat{l}_{\mathrm{MaxCut}} / \delta \boldsymbol{p}_\theta |_{\boldsymbol{p}_\theta = \boldsymbol{p}^*} = \mathbf{0}_N$, which implies a potential reason for absorption into the plateau. However, this does not reveal the conditions for the convergence to the fixed point $\boldsymbol{p}^*$ during the early learning stage or the condition to escape from the fixed point $\boldsymbol{p}^*$. Thus, an extensive theoretical evaluation through stability analysis remains an important topic for future work.

In summary, UL-based solver, minimizing $\boldsymbol{\theta}$ can be challenging and unstable. In particular, the PI-GNN solver, which is one of the UL-based solvers employing GNNs, fails to optimize $\boldsymbol{\theta}$ due to a

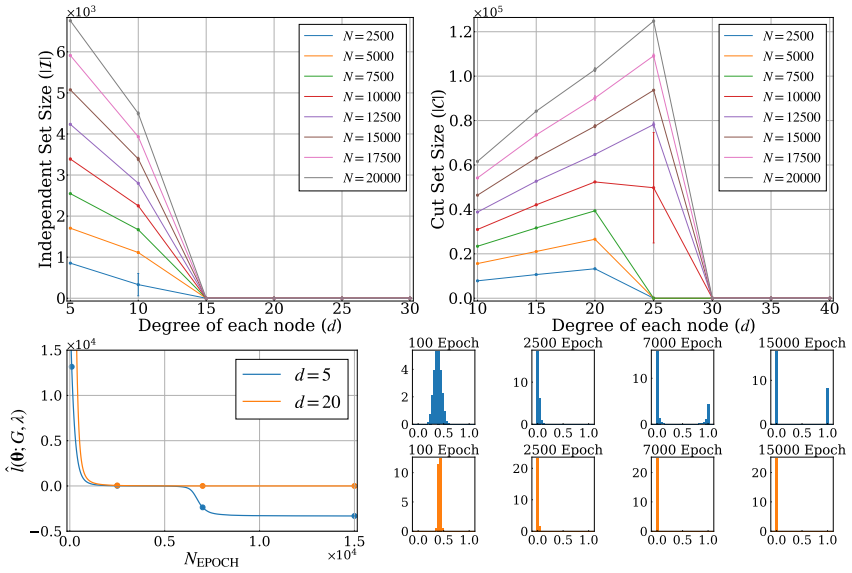

Figure 7: The top graph shows the independent set density for MIS problems (left) and the cut ratio for MaxCut problems (right) as a function of degree d using the PI-GNN solver with varying system size N. Each data point represents the average result of five different graph instances, with the error bars indicating the standard deviation of those results. The bottom graph shows the cost as a function of the number of parameter updates N_{EPOCH}, for $N = 10000$ MIS problems on 5-RRG and 20-RRG. The histogram represents the relaxed vector distribution with varying numbers of parameter updates N_{EPOCH}. Each point in the bottom-left plot is linked to the corresponding bottom-right histogram.

local solution in complex CO problems on relatively dense graphs where the performance of greedy algorithms worsens. This issues can be potential bottleneck for more practical and relatively dense problems, making it challenging to employ the PI-GNN solver confidently.

F.2 Additional results of MIS

We evaluate our method using the MIS benchmark dataset from recent studies [Goshvadi et al., 2023, Qiu et al., 2022], which includes graphs from SATLIB [Hoos and Stützle, 2000] and Erdős–Rényi graphs (ERGs) of varying sizes. Following Sun et al. [2023], our test set consists of 500 SATLIB graphs, each containing between 403 and 449 clauses with up to 1,347 nodes and 5,978 edges, 128 ERGs with 700 to 800 nodes each, and 16 ERGs with 9,000 to 11,000 nodes each. We conducted numerical experiments on PQQA using four different configurations: parallel runs with $S = 100$ or $S = 1000$ and shorter steps (3000 steps) or longer steps (30000 steps), similar to the approach in iSCO [Sun et al., 2023]. Table F.2 presents the solution quality and runtime results. The results show that CRA, which optimizes the relaxed variables as an optimization of GNN parameters, takes extra time for smaller ER-[700-800] instances due to the smaller number of decision variables. However, for larger instances, CRA achieves results comparable to iSCO. Although limited space makes it difficult to present other benchmark results employed by iSCO, such as MaxCut and MaxClique, numerical experiments on these benchmarks also show that CRA is less effective for small problems. However, for larger problems, the results are comparable to or slightly inferior to those of iSCO.

We also investigated the relationship between the order of the graph and the solving time of the solver, and the results are shown in Table F.2 and F.2. The results demonstrate that the runtime remains nearly constant as graph order and density increase, indicating effective scalability with denser graphs.

F.3 Additional results of Gset

We conducted experiments across the additional GSET collection to further validate that including CRA enhances PI-GNN results beyond previously achievable in Table 6.

Table 3: ApR and runtime are evaluated on three benchmarks provided by DIMES [Qiu et al., 2022]. The ApR is assessed relative to the results obtained by KaMIS. Runtime is reported as the total clock time, denoted in seconds (s), minutes (m), or hours (h). The runtime and solution quality are sourced from iSCO [Sun et al., 2023]. The baselines include solvers from the Operations Research (OR) community, as well as data-driven approaches utilizing Reinforcement Learning (RL), Supervised Learning (SL) combined with Tree Search (TS), Greedy decoding (G), or sampling (S). Methods that fail to produce results within 10 times the time limit of DIMES are marked as N/A.

Method	Type	ER-[700-800]		ER-[9000-11000]	
		ApR	Time	ApR	Time
KaMIS	OR	1.000	52.13m	1.000	7.6h
Gurobi	OR	0.922	50.00m	N/A	N/A
Intel	SL+TS	0.865	20.00m	N/A	N/A
	SL+G	0.777	6.06m	0.746	5.02m
DGL	SL+TS	0.830	22.71m	N/A	N/A
LwD	RL+S	0.918	6.33m	0.907	7.56m
DIMES	RL+G	0.852	6.12m	0.841	5.21m
	RL+S	0.937	12.01m	0.873	12.51m
iSCO	fewer steps	0.998	1.38m	0.990	9.38m
	more steps	1.006	5.56m	1.008	1.25h
CRA	UL-based	0.928	47.30m	0.963	1.03h

Table 4: ApR of the MIS problem on $\mathrm{RRG}(N, d)$. All the results are averaged based on 5 RRGs with different random seeds.

Problem	ApR (CRA)	ApR (PI)	Time (CRA)	Time (PI)
$\mathrm{RRG}(1{,}000, 10)$	0.95	0.78	108 (s)	98 (s)
$\mathrm{RRG}(1{,}000, 20)$	0.95	0.56	103 (s)	92 (s)
$\mathrm{RRG}(1{,}000, 30)$	0.94	0.00	102 (s)	88 (s)
$\mathrm{RRG}(1{,}000, 40)$	0.93	0.00	101 (s)	82 (s)
$\mathrm{RRG}(1{,}000, 50)$	0.92	0.00	102 (s)	82 (s)
$\mathrm{RRG}(1{,}000, 60)$	0.91	0.00	101 (s)	91 (s)
$\mathrm{RRG}(1{,}000, 70)$	0.91	0.00	101 (s)	86 (s)
$\mathrm{RRG}(1{,}000, 80)$	0.91	0.00	102 (s)	93 (s)
$\mathrm{RRG}(5{,}000, 10)$	0.93	0.77	436 (s)	287 (s)
$\mathrm{RRG}(5{,}000, 20)$	0.95	0.74	413 (s)	280 (s)
$\mathrm{RRG}(5{,}000, 30)$	0.95	0.00	419 (s)	283 (s)
$\mathrm{RRG}(5{,}000, 40)$	0.94	0.00	429 (s)	293 (s)
$\mathrm{RRG}(5{,}000, 50)$	0.94	0.00	418 (s)	324 (s)
$\mathrm{RRG}(5{,}000, 60)$	0.93	0.00	321 (s)	302 (s)
$\mathrm{RRG}(5{,}000, 70)$	0.92	0.00	321 (s)	325 (s)
$\mathrm{RRG}(5{,}000, 80)$	0.92	0.00	330 (s)	305 (s)

F.4 Additional results of TSP

We conducted additional experiments on several TSP problems from the TSPLIB dataset[2], presenting results that illustrate the α-dependency.

Experiments calculated the ApR as the ratio of the optimal value to the CRA result, with the ApR representing the average and standard deviation over 3 seeds. The "–" symbol in PI-GNN denotes cases where most variables are continuous values and where no solution satisfying the constraint was found within the maximum number of epochs. The same GNN and optimizer settings were used as in the main text experiments in Section 5.1.

[2] `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`

Table 5: The ApR of the MIS problem on $\mathrm{ERG}(N, d)$ is evaluated against the results of KaMIS. Due to time limitations, the maximum running time for KaMIS was constrained. The results below present the average ApRs and runtimes across five different random seeds.

Problem	CRA (ApR)	PI (ApR)	Time (CRA)	Time (PI)	Time (KaMIS)
$\mathrm{ERG}(1{,}000, 0.05)$	0.97	0.01	103 (s)	98 (s)	100 (s)
$\mathrm{ERG}(1{,}000, 0.10)$	0.95	0.00	100 (s)	98 (s)	210 (s)
$\mathrm{ERG}(1{,}000, 0.15)$	0.94	0.00	100 (s)	92 (s)	315 (s)
$\mathrm{ERG}(1{,}000, 0.20)$	0.91	0.00	99 (s)	88 (s)	557 (s)
$\mathrm{ERG}(1{,}000, 0.25)$	0.93	0.00	98 (s)	82 (s)	733 (s)
$\mathrm{ERG}(1{,}000, 0.30)$	0.90	0.00	98 (s)	82 (s)	1000 (s)
$\mathrm{ERG}(1{,}000, 0.35)$	0.92	0.00	99 (s)	91 (s)	1000 (s)
$\mathrm{ERG}(1{,}000, 0.40)$	0.91	0.00	97 (s)	86 (s)	1000 (s)

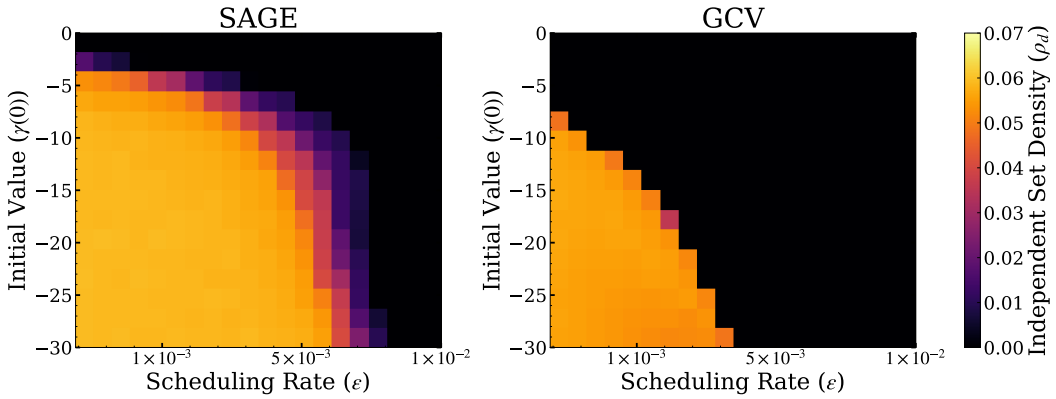

Figure 8: (Top) IS density of $N = 10{,}000$ MIS problems on 100-RRG as a function of initial scheduling $\gamma(0)$ and scheduling rate ε values obtained by the CRA-PI-GNN solver using GraphSage (Left) and GCV (Right). The color of the heat map represents the average IS over five different instances.

Table F.4 shows that the CRA approach yielded solutions with an ApR exceeding 0.9 across various instances. Notably, for the burma14 problem, our method identified the global optimal solution (3,323) multiple times. However, the optimal value can vary based on the specific GNN architecture and problem structure, indicating that a more comprehensive ablation study could provide valuable insights in future work.

F.5 Ablation over initial scheduling value and scheduling rate

We conducted an ablation study focusing on the initial scheduling value $\gamma(0)$ and scheduling rate ε. This numerical experiment was conducted under the configuration described in Section 5 and E. Fig. 8 shows the IS density of $N = 10000$ MIS problems on a 100-RRG as a function of the initial scheduling value $\gamma(0)$ and the scheduling rate ε using the CRA-PI-GNN with both GraphSage and GCV. As can be seen, smaller initial scheduling $\gamma(0)$ and scheduling rate ε values typically yield better solutions. However, the convergence time increases progressively as the initial scheduling $\gamma(0)$ and scheduling rate ε values become smaller. In addition, GraphSage consistently produces better solutions even with relatively larger initial scheduling $\gamma(0)$ and scheduling rate ε values, which implies that the GNN architecture influences both the solution quality and the effective regions of the initial scheduling $\gamma(0)$ and scheduling rate ε values for the annealing process.

F.6 Ablation over curve rate

Next, we investigated the effect of varying the curvature α in Eq. 5. Numerical experiments were performed on MIS problems with 10,000 nodes and the degrees of 5 and 20, as well as MaxCut problems with 10,000 nodes and the degrees of 5 and 35. The GraphSAGE architecture was employed,

Table 6: ApR for MaxCut on Gset

GRAPH	(NODES, EDGES)	GREEDY	SDP	RUN-CSP	PI-GNN	CRA
G1	(800, 19,176)	0.942	0.970	0.979	0.978	**1.000**
G2	(800, 19,176)	0.951	0.970	0.981	0.976	**0.998**
G3	(800, 19,176)	0.945	0.972	0.982	0.972	**1.000**
G4	(800, 19,176)	0.949	0.971	0.980	0.978	**0.999**
G5	(800, 19,176)	0.949	0.970	0.980	0.978	**1.000**
G14	(800, 4,694)	0.946	0.952	0.956	0.988	**0.994**
G15	(800, 4,661)	0.939	0.958	0.952	0.980	**0.992**
G16	(800, 4,672)	0.948	0.958	0.953	0.965	**0.990**
G17	(800, 4,667)	0.946	–	0.956	0.967	**0.990**
G22	(2,000, 19,990)	0.923	–	0.972	0.987	**0.998**
G23	(2,000, 19,990)	0.927	–	0.973	0.968	**0.997**
G24	(2,000, 19,990)	0.927	–	0.973	0.959	**0.998**
G25	(2,000, 19,990)	0.929	–	0.974	0.974	**0.998**
G26	(2,000, 19,990)	0.924	–	0.974	0.965	**0.998**
G35	(2,000, 11,778)	0.942	–	0.953	0.968	**0.990**
G36	(2,000, 11,766)	0.942	–	0.953	0.966	**0.991**
G37	(2,000, 11,785)	0.946	–	0.950	0.966	**0.997**
G38	(2,000, 11,779)	0.943	–	0.949	0.966	**0.991**
G43	(1,000, 9,990)	0.928	0.968	0.976	0.966	**0.995**
G44	(1,000, 9,990)	0.920	0.955	0.978	0.968	**0.998**
G45	(1,000, 9,990)	0.930	0.950	0.979	0.961	**0.998**
G46	(1,000, 9,990)	0.930	0.960	0.976	0.974	**0.998**
G47	(1,000, 9,990)	0.931	0.956	0.976	0.972	**0.997**
G48	(3,000, 6,000)	**1.000**	**1.000**	**1.000**	0.912	**1.000**
G49	(3,000, 6,000)	**1.000**	**1.000**	**1.000**	0.986	**1.000**
G50	(3,000, 6,000)	**1.000**	**1.000**	0.999	0.990	**1.000**
G51	(1,000, 5,909)	0.949	0.960	0.954	0.964	**0.991**
G52	(1,000, 5,916)	0.944	0.957	0.955	0.961	**0.990**
G53	(1,000, 5,914)	0.945	0.956	0.950	0.966	**0.992**
G54	(1,000, 5,916)	0.900	0.956	0.952	0.970	**0.988**
G55	(5,000, 12,498)	0.892	–	0.978	0.983	**0.991**
G58	(5,000, 29,570)	0.945	–	0.980	0.966	**0.989**
G60	(7,000, 17,148)	0.889	–	0.980	0.945	**0.991**
G63	(7,000, 41,459)	0.948	–	0.947	0.968	**0.989**
G70	(10,000, 9,999)	0.886	–	0.970	0.982	**0.992**

Table 7: Comparison of ApR performance across different values of p for various TSPLIB instances.

	burma14	ulysses22	st70	gr96
ApR ($p = 2$)	0.91 ± 0.08	0.89 ± 0.03	0.96 ± 0.01	0.81 ± 0.05
ApR ($p = 4$)	0.98 ± 0.10	0.92 ± 0.02	0.85 ± 0.03	0.82 ± 0.03
ApR ($p = 6$)	0.97 ± 0.14	0.88 ± 0.07	0.88 ± 0.04	0.90 ± 0.05
ApR ($p = 8$)	0.99 ± 0.06	0.89 ± 0.05	0.80 ± 0.02	0.86 ± 0.05
ApR (PI)	0.736 ± 1.21	–	–	–
Optimum	3,323	7,013	675	5,5209

with other parameters set as Section 5 adn E. As shown in Fig. 9, we observed that $\alpha = 2$ consistently yielded the best scores across these problems.

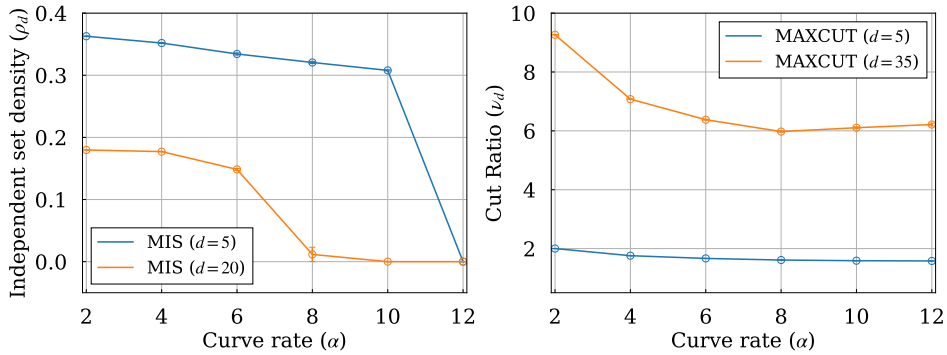

Figure 9: (Left) Independent set density as a function of curveture rate α in Eq. (5). (Right) Cut ratio as a function of curveture rate α in Eq. (5). Error bars represent the standard deviations of the results

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main claims of the paper, including the advantages of the proposed CRA strategy over greedy algorithm without training dataset and the experimental results demonstrating its effectiveness.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the existing UL-based solvers, including optimization and rounding issues. It also mentions the limitations of the proposed CRA strategy, such as the need for parameter tuning and the reliance on specific graph structures for testing.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper provides a full set of assumptions and complete proofs for the theoretical results presented, particularly for Theorem 3.1 and its supporting lemmas in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper includes detailed descriptions of the experimental setups, including the types of graphs used, the parameters for the algorithms, and the performance metrics. This information is sufficient for reproducing the main experimental results (Section 5 and Appendix E).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same

dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: This paper is accompanied by the code, along with detailed instructions to reproduce the experimental results. This will ensure that other researchers can faithfully reproduce the findings reported in the paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

Justification: The paper specifies all the necessary details about the experimental settings, including data splits, hyperparameters, optimizer types, and training procedures (Section 5 and Appendix E).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The paper reports error bars and provides information about the statistical significance of the experimental results, including confidence intervals and standard deviations.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The paper provides information on the computational resources used, including the type of compute workers (e.g., GPU) in Appendix E.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conforms to the NeurIPS Code of Ethics, ensuring responsible conduct and consideration of ethical implications.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses the potential positive impacts of the proposed CRA strategy, such as improving the efficiency of solving CO problems, as well as the potential negative impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper does not involve the release of high-risk data or models.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper does not use any existing assets, so this question is not applicable.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.