Continuous Relaxation Annealing: Enhancing Learning for Combinatorial Optimization

Yuma Ichikawa [1, 2]

[1]Fujitsu Limited [2]The University of Tokyo

Introduction

Combinatorial Optimization (CO)

The objective is to find the discrete optimal value as follows:

$$\min_{\boldsymbol{x}\in\{0,1\}^N} f(\boldsymbol{x};C) \quad \text{subject to} \quad \boldsymbol{x}\in\left\{\boldsymbol{x}\in\{0,1\}^N \middle| \begin{matrix}\forall i\in[I],\ g_i(\boldsymbol{x};C)\leq 0\\ \forall j\in[J],\ h_j(\boldsymbol{x};C)=0\end{matrix}\right\} \tag{1}$$

$C\in\mathcal{C}$ represents instance-specific parameters (e.g., a graph $G=(V,E)$), and $f:\mathcal{X}\times\mathcal{C}\to\mathbb{R}$ denotes the cost function.

Learning-Based Methods for CO Problems

Learning-based methods have gained attention as general-purpose solvers due to their ability to learn problem-specific heuristics.

- Supervised Learning (SL)-based Solvers
  - **Method**: Predict solutions for unseen instances by training on optimal solutions as supervised labels.
  - **Challenges**: Limited availability of optimal solutions in real-world settings and poor generalization.
- Reinforcement Learning (RL)-based Solvers
  - **Method**: Learn a policy to solve CO problems by optimizing reward signals as feedback.
  - **Challenges**: Training is notoriously unstable due to noisy gradient estimations and the difficulty of exploration.

Unsupervised Learning (UL)-Based Solvers

Penalty Method

In UL-based solvers, Eq. (1) is redefined as an unconstrained CO problem:

$$\min_{\boldsymbol{x}\in\{0,1\}^N} l(\boldsymbol{x};C,\boldsymbol{\lambda}),\quad l(\boldsymbol{x};C,\boldsymbol{\lambda})\triangleq f(\boldsymbol{x};C)+\sum_{i=1}^{I+J}\lambda_i v_i(\boldsymbol{x};C),$$

where, for all $i\in[I+J]$, $v:\{0,1\}^N\times\mathcal{C}\to\mathbb{R}$ represents a penalty term that increases when constraints are violated and $\boldsymbol{\lambda}=(\lambda_i)_{1\leq i\leq I+J}\in\mathbb{R}^{I+J}$ denotes the penalty strengths.

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

Continuous Relaxation

UL-based solvers employ a continuous relaxation strategy as follows:

$$\min_{\boldsymbol{p}\in[0,1]^N} \hat{l}(\boldsymbol{p};C,\boldsymbol{\lambda}),\quad \hat{l}(\boldsymbol{p};C,\boldsymbol{\lambda})\triangleq \hat{f}(\boldsymbol{p};C)+\sum_{i=1}^{I+J}\lambda_i\hat{v}_i(\boldsymbol{p};C),$$

where $\boldsymbol{p}=(p_i)_{1\leq i\leq N}\in[0,1]^N$ represents a set of relaxed continuous variables.

UL-Based Solvers

The relaxed vector $\boldsymbol{p}$ is parameterized by a neural network, represented as $\boldsymbol{p}_\theta$. The parameters $\boldsymbol{\theta}$ are optimized by directly minimizing the following label-independent objective function:

$$\hat{l}(\boldsymbol{\theta};C,\boldsymbol{\lambda})\triangleq \hat{f}(\boldsymbol{p}_\theta(C);C)+\sum_{i=1}^{I+J}\lambda_i\hat{v}_i(\boldsymbol{p}_\theta(C);C). \tag{2}$$

After training, the relaxed solution $\boldsymbol{p}_\theta$ is converted into discrete variables using artificial rounding. Specifically, $\forall i\in[N]$, $x_i=\mathrm{int}(p_{\theta,i}(C))$ [2].

In summary, UL-based solvers reformulate the CO problem in Eq. (1) as an optimization problem over the higher-dimensional parameters $\boldsymbol{\theta}$ of a neural network, analogous to *kernel methods*. UL-based solvers that leverage Graph Neural Networks (GNNs) are referred to as *PI-GNN* [2].

Practical Issues of UL-Based Solvers

Ambiguity in Rounding

- UL-based solvers often produce the half integral values $1/2$, which undermines the robustness of the existing rounding methods.

Optimization Issue

- Angelini demonstrated that the PI-GNN solver falls short of achieving results comparable to those of greedy algorithms [1].
- Wang highlighted the importance of utilizing training or historical datasets, $\mathcal{D}=\{C_\mu\}_{1\leq\mu\leq P}$, which consist of various graphs, as well as initializing with outputs from greedy solvers [3].

Continuous Relaxation Annealing (CRA)

To balance discreteness and continuity, we introduce the following penalty term:

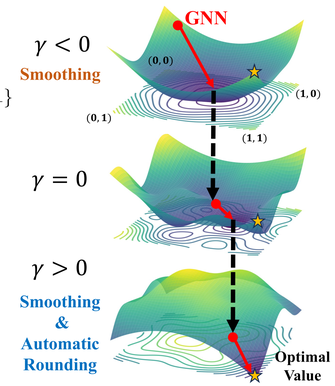
$$\hat{r}(\boldsymbol{\theta};C,\boldsymbol{\lambda},\gamma)=\hat{l}(\boldsymbol{\theta};C,\boldsymbol{\lambda})+\gamma\Phi(\boldsymbol{\theta};C),$$

$$\Phi(\boldsymbol{\theta};C)=\sum_{i=1}^{N}(1-(2p_{\boldsymbol{\theta},i}(C)-1)^\alpha),\quad \alpha\in\{2n\mid n\in\mathbb{N}_+\}$$

where $\gamma\in\mathbb{R}$ is the penalty parameter, and the even number α determines the curvature.

$\gamma<0$: Encourages the relaxed variables to favor continuous space, smoothing the non-convex objective function $\hat{l}(\boldsymbol{p};C,\boldsymbol{\lambda})$ due to the convexity of the penalty term $\Phi(\boldsymbol{p})$.

$\gamma>0$: Encourages the relaxed variables to favor discrete space, pushing continuous solutions toward discrete ones.



Continuous Relaxation Annealing

A technique to gradually anneal the parameter γ from a negative value to favor discreteness.

- **Exploration Phase** ($\gamma<0$): Promotes broad exploration by smoothing the non-convexity.
- **Rounding Phase**: Automatically rounds relaxed variables by transforming suboptimal continuous solutions oscillating between 1 and 0 into discrete solutions.
- **Early Stopping**: Monitors the penalty term $\Phi(\boldsymbol{p})$ and halts training when $\Phi(\boldsymbol{p})\approx 0$.
- **Scheduling**: Updates γ via $\gamma(\tau+1)\leftarrow\gamma(\tau)+\varepsilon$, where ε is a small constant.

Experimental Setting

All experiments adopt PI-GNN [2] as the baseline method.

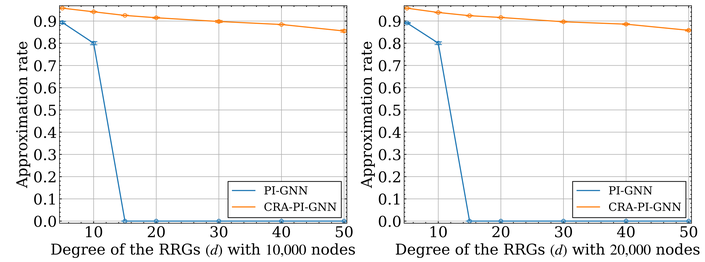
- **Architecture, Optimizer**: We use the same experimental configuration described in PI-GNN [2], employing a simple two-layer `GCV` and `GraphSAGE`.
- **Annealing**: $\gamma(0)=-20$, with a scheduling rate of $\varepsilon=10^{-3}$ and a curve rate $\alpha=2$.
- **Metric**: ApR is defined as $\mathrm{ApR}=f(\boldsymbol{x};C)/f(\boldsymbol{x}^*;C)$, where $\boldsymbol{x}^*$ denotes the optimal solution.

Experiments

We evaluate the performance of CRA-PI-GNN on benchmark problems, including MIS, MaxCut, and DBM. Below, we focus on summarizing the results for **MIS on regular random graphs** as a representative example. Similar qualitative improvements are observed for the other benchmarks.

Degree Dependency

We compare the performance of PI-GNN and CRA-PI-GNN using GCV. The following figure shows the ApR as a function of degree d for PI-GNN and CRA-PI-GNN solvers. Across all degrees d, CRA-PI-GNN solver consistently outperforms PI-GNN solver.



Overcoming Optimization Issues

Several studies [1, 3] have raised optimization concerns for UL-based solvers. However, CRA-PI-GNN substantially outperforms heuristics such as DGA and RGA on MIS for graphs with $d=20,100$, without relying on training or historical datasets $\mathcal{D}=\{G^\mu\}_{\mu=1}^p$.

Method	20-RRG	100-RRG
RGA	0.776 ± 0.001	0.663 ± 0.001
DGA	0.891 ± 0.001	0.848 ± 0.002
EGN	0.775 ([3])	–
META-EGN	0.887 ([3])	–
PI-GNN (GCV)	0.000 ± 0.000	0.000 ± 0.000
PI-GNN (SAGE)	0.745 ± 0.003	0.000 ± 0.000
CRA (GCV)	0.937 ± 0.002	0.855 ± 0.004
CRA (SAGE)	$\mathbf{0.963\pm0.001}$	$\mathbf{0.924\pm0.001}$

Computational Scaling

The computational scaling of CRA-PI-GNN solver for MIS problems on large-scale RRGs with a node degree of 100 exhibits moderate super-linear behavior. Specifically, total computational time scales approximately as $\sim N^{1.4}$ for `GCN` and $\sim N^{1.7}$ for `GraphSAGE`. This scaling is nearly identical to that of PI-GNN solver [2] for problems on RRGs with lower degrees.

References

[1] 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.

[2] 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, 2022.

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

ichikawa-yuma1@fujitsu.com