CADO: Cost-Aware Diffusion Solvers for Combinatorial Optimization through RL Fine-Tuning

Jeong-Hyun Kim

AIOPT
Incheon National University

2025. 4. 25

Overview

1. Introduction
2. Problem Formulation
3. Proposed Method: CADO
4. Experiments
5. Conclusion

Motivation and Background

- **Combinatorial Optimization (CO)** problems arise in scheduling, routing, network design, etc.
- Classic examples include:
  - **TSP (Traveling Salesman Problem):** Visit all cities once with minimal cost.
  - **MIS (Maximum Independent Set):** Choose largest set of non-adjacent vertices in a graph.
- These problems are **NP-hard**, making exact solvers infeasible at large scales.
- **ML Trend:** Deep generative models (e.g., Transformer, Diffusion) are explored to generate approximate but high-quality solutions.
- However, most models focus on output **structure fidelity**, not cost optimization.

Limitations of Existing Approaches (1/2)

(1) Supervised Learning (SL)

- Learns to imitate optimal solutions using labeled data.
- Used in early neural CO solvers (e.g., Pointer Networks, GCNs).
- **Limitations:**
  - Two solutions with similar structure can differ significantly in cost.
  - Model minimizes *prediction error*, not *solution quality*.

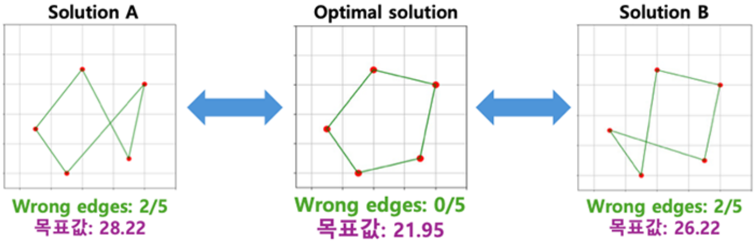
(2) Reinforcement Learning (RL)

- Trains via interaction with environment to directly minimize cost.
- **Limitations:**
  - Sparse and delayed rewards make learning unstable and slow.
  - Hard to scale to large instances.

Limitations of Existing Approaches (2/2)

(3) Diffusion-based Models (e.g., DIFUSCO)

- Uses forward noise process and reverse denoising to learn solution generation.
- Good at modeling discrete combinatorial structures.
- **Limitations:**
  - Ignores **cost** during training – structure is learned, but not cost-aware.
  - Post-processing decoder (e.g., feasibility repair) is used during inference but **not reflected in training objective**.



Why CADO?

- **SL:** Learns structure, but ignores true cost.
- **RL:** Optimizes cost, but unstable and data-inefficient.
- **Diffusion models:** Flexible, but unaware of feasibility-decoder.
- **CADO bridges the gap:**
  - Combines SL training with cost-aware RL fine-tuning.
  - Incorporates decoder directly into training.
  - Produces low-cost, feasible solutions with strong generalization.

CADO: Two-Stage Learning

1. SL pretraining (structure imitation)
2. RL fine-tuning (decoder-aware cost optimization)

Combinatorial Optimization Objective

- Problem instance: $g \in \mathcal{G}$
- Solution space: $x \in X_g = \{0,1\}^N$
- Objective function:

$$c_g(x) = \text{cost}(x, g) + \text{valid}(x, g)$$

- Where the validity term is defined as:

$$\text{valid}(x, g) = \begin{cases} 0 & \text{if } x \text{ is feasible} \\ \infty & \text{otherwise} \end{cases}$$

The Decoder Issue

- The sampled solution x_0 may be infeasible.
- A post-processing decoder $f_g(x_0)$ is used to obtain a feasible solution.
- The actual evaluation is based on $c_g(f_g(x_0))$.
- However, SL training is based solely on x_0, ignoring the decoder effect.
- $\Rightarrow$ This can result in suboptimal performance with respect to cost.

SL vs RL: Training Objectives in CO

- **Supervised Learning (SL)**
  - Objective:

$$\mathcal{L}_{SL}(\theta) = \mathbb{E}_{g \sim P(g)}[-\log p_\theta(x_g^\star | g)]$$

- **Reinforcement Learning (RL)**
  - **Basic Objective:**

$$\mathcal{R}_{RL}(\theta) = \mathbb{E}_{g, x \sim p_\theta(x|g)}[-c_g(x)]$$

  - **Decoder-aware Extension:**

$$\mathcal{R}_{\text{decoder-aware}}(\theta) = \mathbb{E}_{g, x \sim p_\theta(x|g)}[-\text{cost}(f_g(x_0), g)]$$

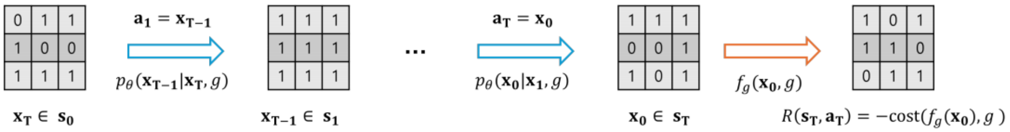
- **Takeaway:** Decoder-aware RL aligns training with true evaluation objective in CO.

MDP Formulation for Diffusion

MDP Definition

- State: $s_t = (g, t, x_t)$, Action: $a_t = x_{t-1}$
- Policy: $\pi_\theta(a_t, s_t) = p_\theta(x_{t-1}|x_t, G)$
- Reward: $R(s_t, a_t) = -\text{cost}(f_g(x_0), g)$ at $t = 0$
- RL objective:

$$\nabla_\theta J = \mathbb{E}\left[\sum_t \nabla_\theta \log p_\theta(x_{t-1}|x_t, g) \cdot (-\text{cost}(f_g(x_0), g))\right]$$



MDP Formulation for Diffusion

Fine-Tuning Strategies and Comparison

- **CADO fine-tuning:**
  - Freeze first 11 GNN layers, fine-tune only the last layer.
  - Efficient even with small updates (supports **LoRA** if needed).
  - Works well with post-processing decoders (e.g., 2-OPT).
- **Compared to T2T:**
  - T2T: Requires differentiable cost functions (limits applicability).
  - CADO: Uses **policy gradients** — compatible with discrete, heuristic, or black-box objectives.
- **Conclusion:**
  - CADO is more flexible, general, and efficient across a wide range of CO problems.

Experiment Settings

- **Hardware:** NVIDIA Tesla A40 GPU, 2 cores of AMD EPYC 7413 CPU
- **Problems:**
  - **TSP:** Shortest round-trip tour visiting all nodes.
  - **MIS:** Largest set of non-adjacent nodes in a graph.
- **Cost Functions:**
  - $\text{cost}_{TSP}(x, G) = \sum_{i,j} x_{i,j} \cdot w_{i,j}$
  - $\text{cost}_{MIS}(x, G) = \sum_{i} (1 - x_i)$
- **Validity:** Ensures that solutions follow TSP or MIS constraints.
- **Key Objectives:**
  - Evaluate whether RL fine-tuning improves cost-optimization performance.
  - Analyze generalization to larger instances (e.g., TSP-500 / TSP-1000).
  - Test robustness under suboptimal training datasets (e.g., LKH-3 vs. Concorde).

Datasets, Metrics, and Baselines

- **TSP Data:**
  - Training from DIFUSCO (Concorde, LKH-3)
  - Test from Joshi et al. (TSP-50/100) and Fu et al. (TSP-500/1000)
- **MIS Data:**
  - SATLIB and Erdős–Rényi graphs
  - Test instances from Qiu et al. (2022)
- **Metrics:**
  - **Length/Size:** Tour length or MIS size (better = lower/higher)
  - **Drop:** Gap from optimal solutions
  - **Time:** Inference runtime
- **Baselines:**
  - Classical: Concorde, LKH-3, HGS, OR-Tools
  - NCO Models: POMO, MDAM, EAS, SGBS, BQ
  - Heatmap: Att-GCN + MCTS

TSP Results

Algorithm	Type	TSP-50		TSP-100	
		Length ↓	Drop ↓	Length ↓	Drop ↓
Concorde (Applegate et al., 2006)	Exact	5.69*	0.00%	7.76*	0.00%
2OPT (Lin & Kernighan, 1973)	Heuristics	5.86	2.95%	8.03	3.54%
Farthest Insertion	Heuristics	6.12	7.50%	8.72	12.36%
AM (Kool et al., 2019b)	RL+Grdy	5.80	1.76%	8.12	4.53%
GCN (Joshi et al., 2019a)	SL+Grdy	5.87	3.10%	8.41	8.38%
Transformer (Bresson & Laurent, 2021)	RL+Grdy	5.71	0.31%	7.88	1.42%
POMO (Kwon et al., 2020)	RL+Grdy	5.73	0.64%	7.84	1.07%
Sym-NCO (Kim et al., 2022)	RL+Grdy	-	-	7.84	0.94%
Image Diffusion (Graikos et al., 2022b)	SL+Grdy	5.76	1.23%	7.92	2.11%
BQ† (Drakulic et al., 2023)	SL+Grdy	-	-	7.79	0.35%
LEHD† (Luo et al., 2023)	SL+Grdy	-	-	7.81	0.58%
ICAM† (Zhou et al., 2024)	RL+Grdy	-	-	7.83	0.90%
DIFUSCO (Sun & Yang, 2023)	SL+Grdy	5.72	0.48%	7.84	1.01%
T2T (Sun & Yang, 2023)	SL+Grdy	5.69	0.04%	7.77	0.18%
CADO (Ours)	SL+RL+Grdy	**5.69**	**0.01%**	**7.77**	**0.08%**
AM (Kool et al., 2019b)	RL+Grdy+2OPT	5.77	1.41%	8.02	3.32%
GCN (Joshi et al., 2019a)	SL+Grdy+2OPT	5.70	0.12%	7.81	0.62%
Transformer (Bresson & Laurent, 2021)	RL+Grdy+2OPT	5.70	0.16%	7.85	1.19%
POMO (Kwon et al., 2020)	RL+Grdy+2OPT	5.73	0.63%	7.82	0.82%
Sym-NCO (Kim et al., 2022)	RL+Grdy+2OPT	-	-	7.82	0.76%
BQ† (Drakulic et al., 2023)	-	-	-	-	-
LEHD† (Luo et al., 2023)	SL+Grdy+RRC	-	-	7.76	0.01%
ICAM† (Zhou et al., 2024)	RL+Grdy+RRC	-	-	7.79	0.41%
DIFUSCO (Sun & Yang, 2023)	SL+Grdy+2OPT	5.69	0.09%	7.78	0.22%
T2T (Li et al., 2023)	SL+Grdy+2OPT	5.69	0.02%	7.76	0.06%
CADO (Ours)	SL+RL+Grdy+2OPT	**5.69**	**0.00%**	**7.76**	**0.01%**

TSP-50 and TSP-100 Results

Algorithm	Type	TSP-500			TSP-1000		
		Length ↓	Drop ↓	Time	Length ↓	Drop ↓	Time
Concorde (Applegate et al., 2006)	Exact	16.55*	-	37.66m	23.12*	-	6.65h
Gurobi (Gurobi Optimization, 2020)	Exact	16.55	0.00%	45.63h	-	-	-
LKH-3 (default) (Helsgaun, 2017)	Heuristics	16.55	0.00%	46.28m	23.12	0.00%	2.57h
Farthest Insertion	Heuristics	18.30	10.57%	0s	25.72	11.25%	0s
AM (Kool et al., 2019b)	RL+Grdy	20.02	20.99%	1.51m	31.15	34.75%	3.18m
GCN (Joshi et al., 2019a)	SL+Grdy	29.72	79.61%	6.67m	48.62	110.29%	28.52m
POMO+EAS-Emb (Hottung et al., 2021)	RL+AS+Grdy	19.24	16.25%	12.80h	-	-	-
POMO+EAS-Tab (Hottung et al., 2021)	RL+AS+Grdy	24.54	48.22%	11.61h	49.56	114.36%	63.45h
DIMES (Qiu et al., 2022)	RL+Grdy	18.93	14.38%	0.97m	26.58	14.97%	2.08m
DIMES (Qiu et al., 2022)	RL+AS+Grdy	17.81	7.61%	2.10h	24.91	7.74%	4.49h
DIMES (Qiu et al., 2022)	RL+Grdy+2OPT	17.65	6.62%	1.01m	24.83	7.38%	2.29m
DIMES (Qiu et al., 2022)	RL+AS+Grdy+2OPT	17.31	4.57%	2.10h	24.33	5.22%	4.49h
BQ† (Drakulic et al., 2023)	SL+Grdy	16.72	1.18%	0.77m	23.65	2.27%	1.9m
LEHD† (Luo et al., 2023)	SL+Grdy	16.78	1.56%	0.27m	23.85	3.17%	1.6m
LEHD† (Luo et al., 2023)	SL+Grdy+RRC	16.58	0.34%	8.7m	23.40	1.20%	48.6m
ICAM† (Zhou et al., 2024)	RL+Grdy	16.78	1.56%	0.03	23.80	2.93%	0.03m
ICAM† (Zhou et al., 2024)	RL+Grdy+RRC	16.69	1.01%	2.4m	23.55	1.86%	16.8m
DIFUSCO (Sun & Yang, 2023)	SL+Grdy	18.11	9.41%	5.70m	25.72	11.24%	17.33m
DIFUSCO (Sun & Yang, 2023)	SL+Grdy+2OPT	16.81	1.55%	5.75m	23.55	1.86%	17.52m
T2T (Li et al., 2023)	SL+Grdy	17.69	6.92%	4.90m	25.39	9.83%	17.93m
T2T (Li et al., 2023)	SL+G+2OPT	16.68	0.83%	4.83m	23.41	1.26%	18.37m
CADO (Ours)	SL+RL+Grdy	16.97	2.56%	2.52m	24.92	7.78 %	18.31m
CADO (Ours)	SL+RL+Grdy+2OPT	**16.64**	**0.58**%	2.67m	**23.35**	**1.02** %	7.67m
EAN (Deudon et al., 2018)	RL+S+2OPT	23.75	43.57%	57.76m	47.73	106.46%	5.39h
AM (Kool et al., 2019b)	RL+BS	19.53	18.03%	21.99m	29.90	29.23%	1.64h
GCN (Joshi et al., 2019a)	SL+BS	30.37	83.55%	38.02m	51.26	121.73%	51.67m
DIMES (Qiu et al., 2022)	RL+S	18.84	13.84%	1.06m	26.36	14.01%	2.38m
DIMES (Qiu et al., 2022)	RL+AS+S	17.80	7.55%	2.11h	24.89	7.70%	4.53h
DIMES (Qiu et al., 2022)	RL+S+2OPT	17.64	6.56%	1.10m	24.81	7.29%	2.86m
DIMES (Qiu et al., 2022)	RL+AS+S+2OPT	17.29	4.48%	2.11h	24.32	5.17%	4.53h
BQ† (Drakulic et al., 2023)	SL+BS	16.62	0.58%	11.9m	23.43	1.36%	29.4m
ICAM† (Zhou et al., 2024)	RL+BS	16.69	1.01%	1.5m	23.54	1.83%	10.5m
ICAM† (Zhou et al., 2024)	RL+S	16.65	0.78%	0.63m	23.49	1.58%	3.8m
DIFUSCO (Sun & Yang, 2023)	SL+S	17.48	5.65%	19.02m	25.11	8.61%	59.18m
DIFUSCO (Sun & Yang, 2023)	SL+S +2OPT	16.69	0.37%	19.05m	23.42	1.30%	59.53m
T2T (Li et al., 2023)	SL+S	17.14	3.60%	17.05m	24.85	7.51%	1.12h
T2T (Li et al., 2023)	SL+S +2OPT	16.62	0.46%	17.02m	23.31	0.85%	1.17h
CADO (Ours)	SL+RL+S	16.75	1.27%	6.83m	24.47	5.88 %	24.73m
CADO (Ours)	SL+RL+S+2OPT	**16.60**	0.34%	6.90m	**23.28**	0.69 %	25.78m

TSP-500 and TSP-1000 Results

MIS Results

Algorithm	Type	SATLIB			ER-[700-800]		
		Size ↑	Drop ↓	Time	Size ↑	Drop ↓	Time
KaMIS (Lamm et al., 2016)	Heuristics	425.96*	-	37.58m	44.87*	-	52.13m
Gurobi (Gurobi Optimization, 2020)	Exact	425.95	0.00%	26.00m	41.28	7.78%	50.00m
Intel (Li et al., 2018a)	SL+Grdy	420.66	1.48%	23.05m	34.86	22.31%	6.06m
DIMES (Qiu et al., 2022)	RL+Grdy	421.24	1.11%	24.17m	38.24	14.78%	6.12m
DIFUSCO (Sun & Yang, 2023)	SL+Grdy	424.56	0.33%	8.25m	36.55	18.53%	8.82m
T2T (Li et al., 2023)	SL+Grdy	**425.02**	**0.22%**	8.12m	39.56	11.83%	8.53m
CADO (Ours)	SL+RL+Grdy	425.01	0.22%	9.52m	**42.96**	**4.25%**	9.50m
Intel (Li et al., 2018a)	SL+TS	-	-	-	38.80	13.43%	20.00m
DGL (Böther et al., 2022)	SL+TS	-	-	-	37.26	16.96%	22.71m
LwD (Ahn et al., 2020a)	RL+S	422.22	0.88%	18.83m	41.17	8.25%	6.33m
GFlowNets (Zhang et al., 2023)	UL+S	423.54	0.57%	23.22m	41.14	8.53%	2.92m
DIFUSCO (Sun & Yang, 2023)	SL+S	425.13	0.19%	26.32m	40.35	10.07%	32.98m
T2T (Li et al., 2023)	SL+S	**425.22**	**0.17%**	23.80m	41.37	7.81%	29.73m
CADO (Ours)	SL+RL+S	425.14	0.19%	16.57m	**43.53**	**2.998%**	11.90m

Robustness to Low-Quality Training Data

- We compare training results using:
  - An optimal dataset (Drop 0%) and
  - A suboptimal dataset (Drop 1.36%), created by limiting LKH-3 to 1s per instance.
- **CADO** achieves the best performance under both conditions.
- DIFUSCO suffers a large drop in performance when trained on poor data.
- In contrast, CADO and T2T remain robust due to cost-based training.
- **Takeaway:** RL fine-tuning enables higher-quality solution generation even from weak data.

Algorithm	**Drop 0%** Drop ↓	**Drop 1.36%** Drop ↓
DIFUSCO	0.48 %	2.298%
T2T	0.04%	1.001%
CADO(ours)	0.01 %	0.911%

Transfer Learning Results

- We evaluated transferability across tasks:
  - Train on TSP100 $\to$ fine-tune on TSP500
  - Train on TSP500 $\to$ fine-tune on TSP1000
- **Without fine-tuning:** Significant performance degradation observed.
- **SL $\to$ RL fine-tuning:**
  - Matches SL on TSP500 and outperforms on TSP1000.
  - Requires no additional optimal labels for the target task.
- **Conclusion:** RL fine-tuning is more cost-effective and scalable for CO tasks.

Fine-tuning	**100→500** Drop ↓	**500→1000** Drop ↓
SL $\to$ $\times$	3.2%	2.12%
SL $\to$ SL	1.55%	1.86%
SL $\to$ RL	1.59%	1.04%

Conclusion

- **CADO:** A two-stage framework combining SL and RL for combinatorial optimization (CO).
- **Key Innovations:**
  - **Cost-aware:** Optimizes solution quality, not just structure.
  - **Decoder-aware:** Considers post-processing effects during training.
- **Strong Performance:**
  - SOTA results on TSP and MIS benchmarks.
  - Closes the optimality gap—even without heuristics.
- **Practical Benefits:**
  - Robust to low-quality data.
  - Generalizes to larger, unseen problem sizes.
  - Efficient via partial fine-tuning (e.g., LoRA).
- **Takeaway:** *Cost-aware + decoder-aware training makes diffusion-based CO solvers scalable, robust, and effective.*