CCTC: CONSTRUCTIVE–CORRECTIVE TRANSITION CALIBRATION FOR UNSUPERVISED COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models have recently shown promise for unsupervised combinatorial optimization (CO), learning to generate high-quality solutions without labeled optimal solutions. However, we find that their training signals can exhibit substantially different discriminative quality across transition directions, a phenomenon we term *Constructive–Corrective Transition Asymmetry*. Specifically, learned policies exhibit substantially weaker target-conditioned discrimination for constructive transitions than for corrective ones, which our analysis relates to a directional scale asymmetry in the underlying transition signals. These findings highlight a gap between effective trajectory-level credit assignment and discriminative transition-level supervision. To address this problem, we propose *Constructive–Corrective Transition Calibration (CCTC)*, which calibrates transition-level supervision according to signal magnitude and transition direction. CCTC combines bounded calibration with direction-conditioned weighting to selectively strengthen target-favored constructive transitions. Experiments on four node- and edge-selection CO problems across eight settings show that CCTC consistently outperforms existing unsupervised state-of-the-art methods while achieving stronger constructive transition discrimination.

1 INTRODUCTION

Combinatorial optimization problems arise widely in routing (Dantzig & Ramser, 1959; Clarke & Wright, 1964), scheduling (Cordeau & Laporte, 2003; Hemmelmayr et al., 2012), network design, and graph analysis. Many of these problems can be represented as binary decision-making tasks, yet finding high-quality solutions remains challenging due to their combinatorial complexity. Learning-based methods provide an alternative to classical solvers (Lawler & Wood, 1966) and handcrafted heuristics (Vidal et al., 2013; Helsgaun, 2017) by amortizing the search process into neural networks (Bengio et al., 2021; Bogyrbayeva et al., 2024; Ba et al., 2026). In particular, diffusion models have emerged as a promising framework for combinatorial optimization (Sun & Yang, 2023; Li et al., 2024; Wang et al., 2026b; Sanokowski et al., 2024). By generating solutions through multiple stochastic transitions, they allow discrete decisions to be iteratively revised over time. However, most diffusion-based CO solvers rely on high-quality reference solutions for supervised training, motivating unsupervised approaches that learn directly from problem objectives and constraints.

Recent work has explored different mechanisms for propagating optimization feedback through unsupervised diffusion training. DiffUCO (Sanokowski et al., 2024) formulates training through entropy-regularized reinforcement learning, propagating terminal rewards through the diffusion trajectory. SDDS (Sanokowski et al., 2025) reformulates this objective through importance sampling to improve scalability. In contrast, RLNN (Feng & Yang, 2025) adopts a local objective based on regularized Langevin dynamics, improving training stability at the cost of limited long-range credit assignment. More recently, CAM (Feng et al., 2026) introduces discrete adjoint dynamics to propagate terminal optimization information into structured trajectory-level supervision with lower-variance optimization signals. These methods provide increasingly effective mechanisms for estimating and propagating optimization feedback. However, they largely leave open a complementary question. *Does the resulting supervision effectively discriminate target-favored from target-suppressed transitions at the individual transition level?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this work, we study unsupervised diffusion training from a transition-level perspective and identify a **Constructive-Corrective Transition Asymmetry Problem**. A constructive transition activates an unselected variable through $0 \to 1$, whereas a corrective transition removes a selected variable through $1 \to 0$. Our analysis reveals a directional scale asymmetry in the underlying training signals. As illustrated in Fig. 1 (a), constructive encouragement is bounded, whereas constructive suppression can increase with constraint conflicts, with the opposite pattern appearing in the corrective direction. Fig. 1 (b, c) shows substantially stronger target-conditioned discrimination for corrective than constructive transitions, making favorable additions more difficult to prioritize and potentially degrading training stability and solution quality. These results show that effective trajectory-level credit assignment does not guarantee discriminative transition-level supervision.

To improve transition-level supervision while preserving trajectory-level credit assignment, we propose **Constructive-Corrective Transition Calibration (CCTC)**, which calibrates optimization signals according to their magnitude and transition direction. CCTC combines bounded calibration to limit excessively large signals with direction-conditioned weighting to selectively strengthen target-favored constructive transitions, and applies to both node- and edge-selection problems. We evaluate CCTC on four representative CO problems, including Maximum Independent Set (MIS), Max-Cut, Traveling Salesman Problem (TSP), and Capacitated Vehicle Routing Problem (CVRP), across eight settings. CCTC consistently improves solution quality over CAM across all evaluated settings while achieving stronger constructive transition discrimination. **The contributions of this paper are:**

- We identify the **Constructive-Corrective Transition Asymmetry Problem** in unsupervised diffusion training for CO and show that constructive transitions exhibit substantially weaker target-conditioned discrimination than corrective transitions.
- We propose **Constructive-Corrective Transition Calibration**, which combines bounded calibration with direction-conditioned weighting to strengthen target-favored constructive transition learning while retaining the underlying diffusion policy.
- We evaluate CCTC on four node- and edge-selection CO problems across eight settings, demonstrating consistent improvements over existing unsupervised state-of-the-art methods together with stronger constructive transition discrimination.

2 Background and Related Work

2.1 Binary Combinatorial Optimization

We consider combinatorial optimization problems represented by binary decision variables $\mathbf{x} \in \{0,1\}^N$ for an instance G. The constrained problem is written as

$$\min_{\mathbf{x}\in\{0,1\}^N} a(\mathbf{x}) \quad \text{s.t.} \quad b(\mathbf{x}) = 0, \tag{1}$$

where $a(\mathbf{x})$ denotes the objective and $b(\mathbf{x}) \geq 0$ measures constraint violation. We use the minimization convention throughout, with maximization objectives negated accordingly. Since generated states are not guaranteed to satisfy $b(\mathbf{x}) = 0$, we consider the penalized unconstrained problem

$$\min_{\mathbf{x}\in\{0,1\}^N} g(\mathbf{x}) = a(\mathbf{x}) + \beta b(\mathbf{x}), \qquad \beta > 0, \tag{2}$$

which extends the optimization objective to both feasible and infeasible states. For node-selection problems, a continuous extension of $g(\mathbf{x})$ provides the gradients used to construct training targets. For edge-selection problems, CAM constructs surrogate flip targets from feasible solutions obtained through greedy decoding and subsequent local refinement.

2.2 CAM Diffusion Policy and Training Objective

Recent unsupervised diffusion methods formulate combinatorial optimization through different learning objectives, including reinforcement learning (Sanokowski et al., 2024), importance sampling (Sanokowski et al., 2025), local energy-based optimization (Feng & Yang, 2025), and trajectory-level credit assignment (Feng et al., 2026). We focus below on CAM because its trajectory-level supervision forms the basis of our analysis, while a broader discussion of learning-based CO and discrete diffusion methods is provided in Appendix A.

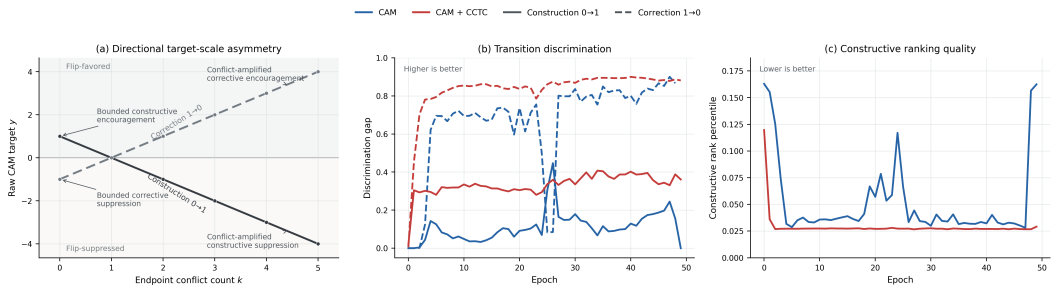


Figure 1: **Constructive–corrective transition asymmetry.** (a) Directional scale asymmetry of the transition signal under increasing conflict counts. (b) Constructive and corrective transition discrimination during training. (c) Percentile rank of greedily decoded constructive candidates.

CAM considers coordinate-wise transitions over binary states. Let $\mathbf{x}^{(i)}$ denote the state obtained by flipping the i-th coordinate of $\mathbf{x}$, and let $u_t^\theta(\mathbf{x}^{(i)}, \mathbf{x})$ denote the corresponding transition probability. We collect these coordinate-wise probabilities into the vector $\mathbf{u}_t^\theta(\mathbf{x}) = \big(u_t^\theta(\mathbf{x}^{(1)}, \mathbf{x}), \dots, u_t^\theta(\mathbf{x}^{(N)}, \mathbf{x})\big)^\top$. A flip with $x_i = 0$ induces a **constructive** transition $0 \to 1$, whereas a flip with $x_i = 1$ induces a **corrective** transition $1 \to 0$. These terms specify transition directions rather than objective improvement. CAM characterizes each transition through a discrete adjoint, which measures the difference in future cost between the flipped and current states. Under the one-Hamming transition structure, this signal can be estimated from a sampled trajectory as

$$\tilde{a}_t^u\big(\mathbf{X}_t^{(i)}; \mathbf{X}_t\big) \approx (-1)^{X_{t,i} \oplus X_{1,i}} \Big[g(\mathbf{X}_1^{(i)}) - g(\mathbf{X}_1) \Big]. \tag{3}$$

where $\mathbf{X}_1$ denotes the endpoint used to construct the trajectory-level signal. Defining the flip-gradient as

$$[\nabla_{\text{flip}} g(\mathbf{x})]_i = g(\mathbf{x}^{(i)}) - g(\mathbf{x}), \tag{4}$$

the CAM matching term couples the transition policy with the endpoint flip-gradient propagated along the trajectory. After time discretization, the CAM objective takes the form

$$\mathcal{L}_{\text{CAM}} = \sum_t \Big[-\tau H\big(\mathbf{u}_t^\theta(\mathbf{X}_t)\big) + \lambda \big\|\mathbf{u}_t^\theta(\mathbf{X}_t)\big\|_1 + \mathbf{u}_t^\theta(\mathbf{X}_t)^\top \big((-1)^{\mathbf{X}_t \oplus \mathbf{X}_1} \circ \nabla_{\text{flip}} g(\mathbf{X}_1)\big) \Big]. \tag{5}$$

where $\mathbf{u}_t^\theta(\mathbf{X}_t)$ collects the coordinate-wise flip probabilities, $H(\cdot)$ denotes the Bernoulli entropy, τ controls the entropy regularization, λ weights the transition-rate penalty, and $\circ$ denotes element-wise multiplication. The final adjoint matching term provides the principal trajectory-level optimization signal in CAM and forms the basis of our subsequent transition-level analysis.

3 Transition Asymmetry Problem

We next examine how CAM's *trajectory-level* optimization signals affect constructive and corrective transitions. Although both transition directions are governed by the same diffusion policy, they play distinct roles in modifying a candidate solution. Therefore, we analyze the *transition-level* learning behavior induced by the CAM training objective and characterize how its optimization signals act across the two transition directions.

3.1 Directional Transition Asymmetry

We first analyze the directional structure of the adjoint matching signal introduced in Section 2.2. For maximum independent set (MIS), we consider the penalized objective

$$g(\mathbf{x}) = -\sum_i x_i + \beta \sum_{\{i,j\} \in E} x_i x_j, \tag{6}$$

where $\beta > 0$ penalizes conflicts between selected neighboring nodes. Let $k_i(\mathbf{X}_1)$ denote the number of selected neighbors of node i at the rollout endpoint $\mathbf{X}_1$. The corresponding endpoint flip-gradient is

$$[\nabla_{\text{flip}} g(\mathbf{X}_1)]_i = \begin{cases} -1 + \beta k_i(\mathbf{X}_1), & X_{1,i} = 0, \\ 1 - \beta k_i(\mathbf{X}_1), & X_{1,i} = 1. \end{cases} \tag{7}$$

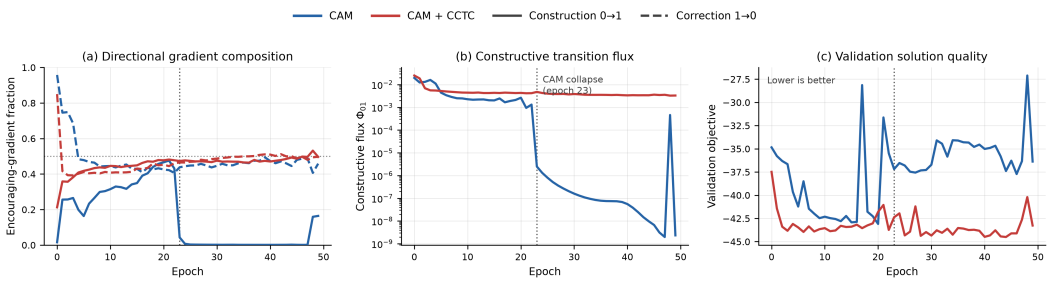


Figure 2: **Training dynamics of construction collapse.** (a) Fraction of matching-gradient contributions that encourage constructive and corrective transitions. (b) Constructive transition flux during training. (c) Validation solution quality. The dotted line marks the detected onset of construction collapse for CAM. The aligned changes in gradient composition, constructive activity, and validation quality characterize a coupled low-construction regime.

Following the adjoint propagation in Eq. 3, we define the signed transition signal

$$y_{t,i} = -(-1)^{X_{t,i} \oplus X_{1,i}} [\nabla_{\text{flip}} g(\mathbf{X}_1)]_i, \tag{8}$$

such that $y_{t,i} > 0$ favors increasing the corresponding transition probability, whereas $y_{t,i} < 0$ suppresses it. Substituting Eq. 7 into Eq. 8 yields

$$y_{t,i} = \begin{cases} 1 - \beta k_i(\mathbf{X}_1), & X_{t,i} = 0, \\ \beta k_i(\mathbf{X}_1) - 1, & X_{t,i} = 1. \end{cases} \tag{9}$$

As illustrated in Fig. 1 (a), Eq. 9 yields distinct target-scale patterns across the two transition directions as the endpoint conflict count $k_i(\mathbf{X}_1)$ increases. For constructive transitions, positive signals are bounded by one, whereas negative signals can increase in magnitude with the conflict count. Conversely, corrective transitions admit conflict-amplified positive signals, while their negative signals remain bounded. Thus, the CAM signal exhibits a directional scale asymmetry between constructive and corrective transitions.

To examine whether this signal asymmetry is reflected in the learned policy, we measure the separation between target-favored and target-suppressed transitions within each direction as

$$D_d(\eta) = \mathbb{E}[p_{t,i} \mid d_{t,i} = d,\, y_{t,i} > \eta] - \mathbb{E}[p_{t,i} \mid d_{t,i} = d,\, y_{t,i} \le \eta], \qquad d \in \{0{\to}1, 1{\to}0\}, \tag{10}$$

where $d_{t,i}$ denotes the transition direction and η is the diagnostic threshold. As shown in Fig. 1 (b), CAM exhibits substantially weaker separation between target-favored and target-suppressed transitions in the constructive direction than in the corrective direction. Moreover, Fig. 1 (c) evaluates the corresponding ranking quality among constructive candidates, providing a complementary view of the policy's ability to prioritize favorable additions.

3.2 Construction Collapse as an Extreme Failure Mode

The weak constructive discrimination identified above does not necessarily imply the disappearance of constructive transitions. We use *construction collapse* to describe a more severe dynamic regime in which constructive transition activity becomes persistently negligible. Specifically, we quantify constructive activity by the expected constructive transition flux

$$\Phi_{01} = \mathbb{E}\left[(1 - X_{t,i}) p_{t,i}\right], \tag{11}$$

where $p_{t,i}$ denotes the i-th coordinate of the transition policy $\mathbf{u}_t^\theta(\mathbf{X}_t)$. This distinction is important because weak constructive discrimination can occur while the policy still maintains substantial constructive activity.

Fig. 2 illustrates an extreme instance of this behavior during CAM training. As shown in Fig. 2 (a), the fraction of constructive matching gradients that encourage a transition drops sharply near epoch 23, while the corresponding corrective fraction remains relatively stable. At approximately the same stage, Fig. 2 (b) shows that the constructive transition flux decreases by several orders of magnitude and remains near zero for most subsequent epochs. Together, these observations characterize a transition to a low-construction regime rather than merely a reduction in constructive discrimination.

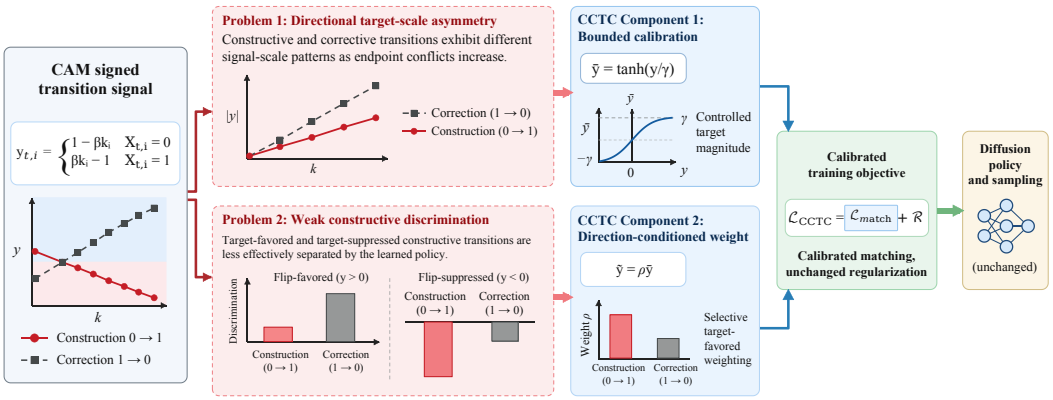


Figure 3: **Overview of CCTC.** Bounded calibration controls the scale of transition-level supervision, while direction-conditioned weighting selectively strengthens target-favored constructive transitions. The calibrated supervision is then used in the task-specific matching objective.

This low-construction regime can further attenuate the optimization of constructive transitions. For the signed transition signal $y_{t,i}$ defined in Eq. 8, the gradient of the corresponding matching term with respect to the transition logit $z_{t,i}$ is

$$\frac{\partial \mathcal{L}_{\text{match},t,i}}{\partial z_{t,i}} = -y_{t,i} p_{t,i}(1 - p_{t,i}), \tag{12}$$

where $p_{t,i} = \sigma(z_{t,i})$. As constructive transition probabilities approach zero, the factor $p_{t,i}(1 - p_{t,i})$ also decreases. Therefore, even when target-favored constructive signals remain present, their direct influence on the corresponding logits can become increasingly attenuated. Moreover, because CAM evaluates endpoint flip-gradients on states sampled from the evolving diffusion policy, changes in constructive activity can also alter the optimization signals encountered in subsequent trajectories.

The corresponding validation behavior is shown in Fig. 2 (c), where CAM exhibits poorer and more variable validation solution quality after entering the low-construction regime. In contrast, CCTC maintains active constructive transitions together with more stable validation performance. These observations associate construction collapse with coupled changes in gradient composition, constructive activity, and solution quality. To assess its reproducibility, we further analyze five CAM seeds, all of which exhibit at least one detected low-construction episode, although the onset and persistence vary across trajectories, as detailed in Appendix D.

3.3 Implications for Transition Learning

The preceding analysis shows that effective transition learning requires selective discrimination rather than uniformly increasing constructive activity. In particular, the directional scale asymmetry in CAM can weaken the distinction between target-favored and target-suppressed constructive transitions. Addressing this problem therefore requires controlling excessive target magnitudes while selectively strengthening target-favored constructive signals. These considerations motivate the bounded calibration and direction-conditioned weighting introduced in CCTC.

4 Methodology

Motivated by the *transition-level* asymmetry identified above, we propose **Constructive–Corrective Transition Calibration**. CCTC modifies the optimization signals used to train the diffusion policy while retaining CAM's transition parameterization and sampling process. Its central principle is to calibrate transition supervision according to both signal magnitude and transition direction.

4.1 Constructive–Corrective Transition Calibration

CCTC calibrates the transition-level supervision in CAM through two complementary operations. First, bounded calibration compresses the magnitude of the signed transition signal while preserv-

ing its direction. Second, direction-conditioned weighting selectively increases the contribution of target-favored constructive transitions while leaving the remaining transition categories unchanged. Together, these operations are designed to strengthen weak constructive discrimination without uniformly increasing constructive transition probabilities or forcing symmetric behavior across the two transition directions. As illustrated in Fig. 3, CCTC leaves the diffusion policy, model architecture, and transition sampling procedure unchanged and modifies only the training objective.

4.2 Node-Selection Instantiation

For node-selection problems, CCTC first applies bounded calibration to the signed transition signal defined in Eq. 8,

$$\bar{y}_{t,i} = \tanh\left(\frac{y_{t,i}}{\gamma}\right), \qquad \gamma > 0, \tag{13}$$

where γ controls the calibration scale. This transformation preserves the sign and ordering of $y_{t,i}$ while bounding its magnitude by one. For MIS, it limits the conflict-amplified signal magnitudes identified in Eq. 9. CCTC then applies direction-conditioned weighting to selectively strengthen target-favored constructive transitions,

$$\rho_{t,i} = \begin{cases} \alpha_{\mathrm{con}}, & X_{t,i} = 0 \text{ and } y_{t,i} > 0, \\ 1, & \text{otherwise}, \end{cases} \qquad \alpha_{\mathrm{con}} \geq 1, \tag{14}$$

where α_{con} increases the contribution of target-favored constructive transitions while leaving corrective and target-suppressed constructive transitions unchanged.

With $\widetilde{y}_{t,i} = \rho_{t,i}\bar{y}_{t,i}$, the calibrated node-selection objective is

$$\mathcal{L}_{\mathrm{node}} = \mathbb{E}\Big[\underbrace{-\sum_i p_{t,i}\widetilde{y}_{t,i}}_{\mathcal{L}_{\mathrm{match}}} + \mathcal{R}(\mathbf{p}_t)\Big], \tag{15}$$

where $\mathcal{R}(\mathbf{p}_t)$ denotes the regularization terms inherited from CAM in Eq. 5, which remain unchanged. The resulting objective bounds the magnitude of the matching signal while selectively increasing the contribution of target-favored constructive transitions.

4.3 Edge-Selection Instantiation

For edge-selection problems, CAM formulates transition supervision as a binary matching objective. We denote the corresponding probabilistic target for edge e by $q_{t,e} \in [0, 1]$. To preserve its probabilistic semantics, CCTC calibrates the target around the neutral value 0.5 as

$$\bar{q}_{t,e} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{2q_{t,e} - 1}{\gamma}\right), \tag{16}$$

which preserves both the ordering of the targets and their side relative to 0.5.

The direction-conditioned weight is defined as

$$\rho_{t,e} = \begin{cases} \alpha_{\mathrm{con}}, & X_{t,e} = 0 \text{ and } q_{t,e} > 0.5, \\ 1, & \text{otherwise}, \end{cases} \qquad \alpha_{\mathrm{con}} \geq 1. \tag{17}$$

Unlike the signed node-selection signal, the edge target retains a probabilistic interpretation after calibration. We therefore use $\rho_{t,e}$ to weight the binary matching loss rather than multiplying it into $\bar{q}_{t,e}$. The calibrated edge-selection objective is

$$\mathcal{L}_{\mathrm{edge}} = \mathbb{E}\Big[\underbrace{\frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} \rho_{t,e}\mathrm{BCE}(p_{t,e}, \bar{q}_{t,e})}_{\mathcal{L}_{\mathrm{match}}} + \mathcal{R}(\mathbf{p}_t)\Big], \tag{18}$$

where BCE denotes binary cross-entropy and the regularization term follows Eq. 5. Thus, the node- and edge-selection formulations share the same calibration principle while respecting the different semantics of their matching targets.

Table 1: Comparative results on MIS. The best neural results are **bolded**. Gaps are computed against KaMIS, except on ER-[9000–11000], where the best reported result is used.

		RB-[200–300]			RB-[800–1200]		
Type	**Method**	**Size↑**	**Gap↓**	**Time↓**	**Size↑**	**Gap↓**	**Time↓**
OR	Gurobi	19.98	0.60%	47.57m	40.90	5.21%	2.17h
	KaMIS (Lamm et al., 2017)	20.10	0.00%	1.40h	43.15	0.00%	2.05h
SL	INTEL (Li et al., 2018)	18.47	8.11%	13.07m	34.47	20.12%	20.28m
	DGL (Böther et al., 2022)	17.36	13.93%	12.78m	34.50	20.05%	23.90m
	DIFUSCO (Sun & Yang, 2023)	18.74	6.77%	5.33m	37.32	13.51%	8.46m
UL	DiffUCO (Sanokowski et al., 2024)	19.88	1.09%	4.97m	40.52	6.10%	6.61m
	SDDS (Sanokowski et al., 2025)	19.75	1.74%	4.97m	39.76	7.86%	6.61m
	RLNN (Feng & Yang, 2025)	19.65	2.24%	4.97m	39.96	7.39%	6.61m
	CAM (Feng et al., 2026)	19.91	0.95%	4.97m	41.25	4.40%	6.61m
	CCTC	**19.92**	**0.89%**	4.97m	**41.56**	**3.68%**	6.61m

		ER-[700–800]			ER-[9000–11000]		
Type	**Method**	**Size↑**	**Gap↓**	**Time↓**	**Size↑**	**Gap↓**	**Time↓**
OR	Gurobi	41.38	7.78%	50.00m	–	–	–
	KaMIS (Lamm et al., 2017)	44.87	0.00%	52.13m	381.31	1.95%	7.60h
SL	INTEL (Li et al., 2018)	34.86	22.31%	6.06m	284.63	26.81%	5.02m
	DGL (Böther et al., 2022)	37.26	16.96%	22.71m	–	–	–
	DIFUSCO (Sun & Yang, 2023)	41.85	6.73%	1.34m	347.00	10.77%	5.89m
UL	DIMES (Qiu et al., 2022)	42.06	6.26%	12.01m	332.80	14.42%	12.72m
	DiffUCO (Sanokowski et al., 2024)	43.98	1.98%	55s	373.31	4.00%	5.53m
	SDDS (Sanokowski et al., 2025)	43.31	3.48%	55s	350.63	9.84%	5.53m
	RLNN (Feng & Yang, 2025)	43.14	3.86%	55s	374.38	3.73%	5.53m
	CAM (Feng et al., 2026)	44.16	1.58%	55s	384.38	1.16%	5.53m
	CCTC	**44.47**	**0.89%**	55s	**388.88**	**0.00%**	5.53m

5 EXPERIMENTS

5.1 PROTOCOLS

Benchmarks. We evaluate CCTC on four combinatorial optimization problems covering both node- and edge-selection settings, including MIS, Max-Cut, TSP, and CVRP. The benchmarks span different graph families and problem scales. We compare against classical optimization methods, supervised neural solvers, and unsupervised or reinforcement-learning-based approaches, with CAM serving as the primary baseline for isolating the effect of transition calibration.

Setup. CCTC applies bounded calibration and direction-conditioned weighting only during training, while retaining CAM's model architecture, transition-signal construction, and sampling procedure without additional inference-time computation (Feng et al., 2026). Unless otherwise specified, we use $\gamma = 1$ and $\alpha_{\text{con}} = 3$, while keeping the remaining task-specific configurations consistent with CAM. For each task, diffusion-based methods use the same sampling budget, decoder, and post-processing procedure. We report solution objective and optimality gap, selecting the best solution from the independent samples generated for each instance. CCTC can be trained either from random initialization (Sec. 5.2) or by continuing optimization from a CAM checkpoint (Sec. 5.5). In both settings, transition signals are constructed from trajectories sampled by the current policy and treated as detached training targets. Details about datasets, baselines, implementation, and hyperparameters are shown in C.

5.2 MAIN RESULTS

Tables 1–4 summarize the main results. CCTC consistently improves over CAM across all eight evaluated settings and remains competitive with strong OR, supervised learning, and unsupervised learning baselines. It achieves the best neural results on all MIS settings, Max-Cut, TSP-1000, and CVRP-100, while remaining competitive with the strongest supervised methods on TSP-500. The improvement also persists under substantial size generalization, where the MIS model trained on

Table 2: Comparative results on TSP. The best neural results are **bolded**.

TSP		TSP-500			TSP-1000		
Type	**Method**	**Length↓**	**Gap↓**	**Time↓**	**Length↓**	**Gap↓**	**Time↓**
OR	Concorde	16.55	0.00%	37.66m	23.12	0.00%	6.65h
	LKH-3 (Helsgaun, 2017)	16.55	0.00%	46.88m	23.12	0.00%	2.57h
SL	GCN (Joshi et al., 2019)	30.37	83.55%	38.02m	51.26	121.73%	51.67m
	DIFUSCO (Sun & Yang, 2023)	16.85	1.81%	11.95m	23.85	3.16%	47.01m
	FMIP (Li et al., 2026a)	**16.83**	**1.69%**	11.95m	23.81	2.98%	47.01m
UL	AM (Kool et al., 2019)	19.53	18.03%	21.99m	29.90	29.23%	1.64m
	POMO (Kwon et al., 2020)	19.24	16.25%	12.80h	49.56	114.36%	63.45h
	DIMES (Qiu et al., 2022)	17.80	7.55%	2.11h	24.89	7.70%	4.53h
	SDDS (Sanokowski et al., 2025)	16.98	2.60%	11.95m	23.90	3.37%	47.01m
	CAM (Feng et al., 2026)	16.91	2.18%	11.95m	23.70	2.51%	47.01m
	CCTC	16.89	2.03%	11.95m	**23.65**	**2.29%**	47.01m

Table 3: Comparative results on CVRP-100. The best neural result is **bolded**.

Type	Method	Length↓	Gap↓	Time↓
OR	HGS (Vidal et al., 2013)	15.56	0.00%	55.64h
SL	DIFUSCO (Sun & Yang, 2023)	15.97	2.63%	33.75m
	FMIP (Li et al., 2026a)	16.04	3.08%	33.70m
UL	SDDS (Sanokowski et al., 2025)	16.16	3.86%	33.70m
	CAM (Feng et al., 2026)	15.98	2.70%	33.70m
	CCTC	**15.90**	**2.18%**	33.70m

Table 4: Comparative results on Max-Cut BA-[800–1200]. The best neural result is **bolded**.

Type	Method	Size↑	Gap↓	Time↓
OR	MQLib (Dunning et al., 2018)	2961.38	0.95%	8.33h
SL	DIFUSCO (Sun & Yang, 2023)	2946.23	1.46%	6.08m
	FMIP (Li et al., 2026a)	2939.58	1.68%	5.49m
UL	DiffUCO (Sanokowski et al., 2024)	2961.07	0.96%	5.49m
	SDDS (Sanokowski et al., 2025)	2963.95	0.87%	5.49m
	RLNN (Feng & Yang, 2025)	2956.81	1.11%	5.49m
	CAM (Feng et al., 2026)	2964.20	0.86%	5.49m
	CCTC	**2989.91**	**0.00%**	5.49m

ER-[700–800] improves over both CAM and the reported OR and learning-based baselines on ER-[9000–11000] without retraining. Together, the results demonstrate consistent gains across different problem scales and both node- and edge-selection formulations.

5.3 Component Ablation

We ablate bounded calibration and direction-conditioned weighting on MIS ER-[700–800]. Fig. 4 compares CAM, each component alone, and full CCTC using one matched training trajectory per configuration. As shown in Fig. 4 (a), bounding substantially reduces the early-stage imbalance between suppressive and encouraging constructive gradients, while weighting provides a complementary reduction. Fig. 4 (b) further shows that both components improve constructive discrimination, with their combination achieving the strongest late-training separation. The corresponding validation curves in Fig. 4 (c) show that CCTC maintains competitive and stable solution quality throughout training. These results indicate that bounded calibration and direction-conditioned weighting provide complementary benefits for constructive transition learning.

5.4 Effect of Hyperparameters

We analyze the sensitivity of CCTC to the calibration scale γ and constructive weight α_{con} on MIS ER-[700–800]. When varying one parameter, the other is fixed to its default value, with $\gamma = 1$ and $\alpha_{\text{con}} = 3$. Fig. 5 reports the best validation size together with the late-stage discrimination ratio $D_{0\to1}/D_{1\to0}$ averaged over the last ten epochs. As γ increases, bounded calibration becomes weaker, accompanied by a progressive decrease in relative constructive discrimination and degraded validation quality at larger values. Increasing α_{con} produces stronger relative constructive discrimination, whereas validation quality improves up to $\alpha_{\text{con}} = 3$ and does not further improve under stronger weighting. These results indicate that overly weak calibration or overly strong constructive weighting can be unfavorable to solution quality, while the default setting provides a favorable balance between constructive discrimination and validation performance.

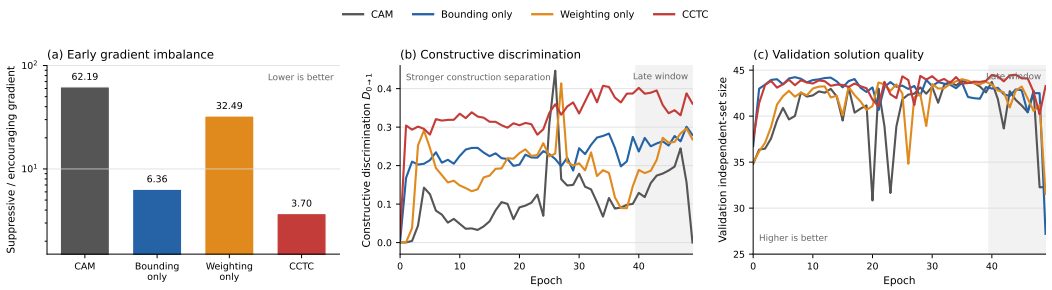


Figure 4: **Component ablation on MIS ER-[700–800].** (a) Early-stage imbalance between suppressive and encouraging constructive gradients. (b) Constructive discrimination during training. (c) Validation independent-set size. Bounding and direction-conditioned weighting provide complementary improvements, with full CCTC achieving the strongest late-training constructive discrimination while maintaining stable validation quality. Curves are shown without smoothing, and the shaded region denotes epochs 40–49. Each configuration shows one matched training trajectory.

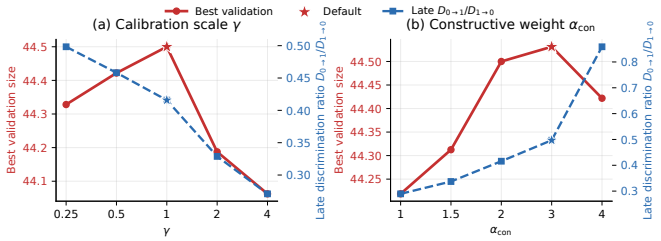


Figure 5: **Hyperparameter sensitivity of CCTC on MIS ER-[700–800].** We vary the calibration scale γ and constructive weight α_{con} while keeping other settings fixed. The red curve reports the best validation size, and the blue curve reports the late-stage discrimination ratio $D_{0\to1}/D_{1\to0}$ averaged over the last ten epochs. Stars denote the optimal settings.

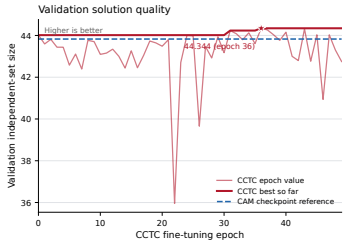


Figure 6: **Continued training from a CAM checkpoint.** Raw validation results and the best-so-far independent-set size during CCTC training, together with the CAM checkpoint reference.

5.5 Continued Training from CAM

Beyond training from random initialization, CCTC can also be applied by continuing optimization from an existing CAM checkpoint. We initialize CCTC from the best CAM checkpoint while reinitializing the optimizer and learning-rate scheduler, with all model parameters inherited from the checkpoint. As shown in Fig. 6, the best validation independent-set size improves from 43.828 at the source CAM checkpoint to 44.344 during continued CCTC training. This result shows that the transition-level calibration introduced by CCTC can further improve an already trained CAM.

6 Conclusion

We identify the Constructive–Corrective Transition Asymmetry Problem in unsupervised diffusion training for combinatorial optimization, showing that constructive transitions can receive substantially less discriminative supervision than corrective transitions. To address this problem, we propose Constructive–Corrective Transition Calibration (CCTC), which combines bounded calibration with direction-conditioned weighting without modifying the model architecture or inference procedure. Experiments across node- and edge-based CO problems demonstrate consistent improvements in solution quality, constructive transition discrimination, and training dynamics. Our findings highlight the importance of transition-level supervision for effective unsupervised diffusion-based combinatorial optimization.

AI USE STATEMENT

In this work, we used generative AI tools solely to edit and polish author-written text to improve the readability, clarity, and linguistic quality of the manuscript. All AI-assisted edits were reviewed and verified by the authors to ensure that they preserved the intended technical meaning and did not introduce unsupported claims or alter the scientific content. The authors take full responsibility for the final content of this work, including any text produced or modified with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of CCTC and its node- and edge-selection instantiations in Section 4. Section 5.1 and Appendix C specify the benchmarks, training protocols, hyperparameters, sampling budgets, and evaluation procedures used in our experiments. Code and configurations will be made publicly available upon acceptance.

REFERENCES

Yunpeng Ba, Xi Lin, Changliang Zhou, Ruihao Zheng, Zhenkun Wang, Xinyan Liang, Zhichao Lu, Jianyong Sun, Yuhua Qian, and Qingfu Zhang. Survey on neural routing solvers. *arXiv preprint arXiv:2602.21761*, 2026.

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

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, 2021.

Aigerim Bogyrbayeva, Meraryslan Meraliyev, Taukekhan Mustakhov, and Bissenbay Dauletbayev. Machine learning to solve vehicle routing problems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 25(6):4754–4772, 2024.

Maximilian Böther, Otto Kißig, Martin Taraz, Sarel Cohen, Karen Seidel, and Tobias Friedrich. What's wrong with deep learning in tree search for combinatorial optimization. *arXiv preprint arXiv:2201.10494*, 2022.

Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

Jean-François Cordeau and Gilbert Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.

George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1): 80–91, 1959.

Darko Drakulić, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner. In *International Conference on Learning Representations*, volume 2025, pp. 52465–52488, 2025.

Iain Dunning, Swati Gupta, and John Silberholz. What works best when? a systematic evaluation of heuristics for max-cut and qubo. *INFORMS Journal on Computing*, 30(3):608–624, 2018.

Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. Invit: A generalizable routing problem solver with invariant nested view transformer. In *International Conference on Machine Learning*, pp. 12973–12992. PMLR, 2024.

Shengyu Feng and Yiming Yang. Regularized langevin dynamics for combinatorial optimization. In *Proceedings of the 42nd International Conference on Machine Learning*. PMLR, 2025.

Shengyu Feng, Tarun Suresh, and Yiming Yang. Unsupervised diffusion solver for combinatorial optimization via combinatorial adjoint matching. In *Proceedings of the 43rd International Conference on Machine Learning*. PMLR, 2026.

Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. *arXiv preprint arXiv:2308.14104*, 2023.

Wei Guo, Jaemoo Choi, Yuchen Zhu, Molei Tao, and Yongxin Chen. Proximal diffusion neural sampler. In *International Conference on Learning Representations*, volume 2026, pp. 34373–34408, 2026a.

Yingqing Guo, Yukang Yang, Hui Yuan, and Mengdi Wang. Training-free guidance beyond differentiability: Scalable path steering with tree search in diffusion and flow models. *Advances in Neural Information Processing Systems*, 38:73343–73384, 2026b.

Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12:966–980, 2017.

Vera C Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & operations research*, 39(12):3215–3228, 2012.

Peter Holderrieth, Michael Samuel Albergo, and Tommi Jaakkola. LEAPS: A discrete neural sampler via locally equivariant networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 23397–23416. PMLR, 13–19 Jul 2025.

André Hottung, Mridul Mahajan, and Kevin Tierney. Polynet: Learning diverse solution strategies for neural combinatorial optimization. In *International Conference on Learning Representations*, volume 2025, pp. 80417–80435, 2025.

Cristian Jensen, Luca Schaufelberger, Riccardo De Santi, Kjell Jorner, and Andreas Krause. Value matching: Scalable and gradient-free reward-guided flow adaptation. In *International Conference on Learning Representations*, volume 2026, pp. 144926–144958, 2026.

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.

Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. Fine-tuning masked diffusion for provable self-correction. *arXiv preprint arXiv:2510.01384*, 2025.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:1936–1949, 2022.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in neural information processing systems*, 33:21188–21198, 2020.

Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021.

Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F Werneck. Finding near-optimal independent sets at scale. *Journal of Heuristics*, 23(4):207–229, 2017.

Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

Hongpei Li, Hui Yuan, Han Zhang, Jianghao Lin, Dongdong Ge, Mengdi Wang, and Yinyu Ye. Fmip: Joint continuous-integer flow for mixed-integer linear programming. In *International Conference on Learning Representations*, volume 2026, pp. 151594–151614, 2026a.

Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:50020–50040, 2023.

Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. *Advances in Neural Information Processing Systems*, 37:30179–30206, 2024.

Yang Li, Lvda Chen, Haonan Wang, Runzhong Wang, and Junchi Yan. Generation as search operator for test-time scaling of diffusion-based combinatorial optimization. *Advances in Neural Information Processing Systems*, 38:127168–127196, 2026b.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems*, 31, 2018.

Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36:8845–8864, 2023.

Fu Luo, Xi Lin, Yaoxin Wu, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Qingfu Zhang. Boosting neural combinatorial optimization for large-scale vehicle routing problems. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Fu Luo, Xi Lin, Mengyuan Zhong, Fei Liu, Zhenkun Wang, Jianyong Sun, and Qingfu Zhang. Learning to insert for constructive neural vehicle routing solver. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. COExpander: Adaptive solution expansion for combinatorial optimization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 42130–42164. PMLR, 13–19 Jul 2025.

Zijing Ou, Ruixiang Zhang, and Yingzhen Li. Discrete neural flow samplers with locally equivariant transformer. *Advances in Neural Information Processing Systems*, 38:127197–127237, 2026.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 35:25531–25546, 2022.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 43346–43367. PMLR, 2024.

Sebastian Sanokowski, Wilhelm Berghammer, Haoyu Wang, Martin Ennemoser, Sepp Hochreiter, and Sebastian Lehner. Scalable discrete diffusion samplers: Combinatorial optimization and statistical physics. In *International Conference on Learning Representations*, volume 2025, pp. 87053–87082, 2025.

Sebastian Sanokowski, Lukas Gruber, Christoph Bartmann, Sepp Hochreiter, and Sebastian Lehner. Rethinking losses for diffusion bridge samplers. *Advances in Neural Information Processing Systems*, 38:108635–108664, 2026.

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in neural information processing systems*, 36:3706–3731, 2023.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Yu Wang, Yang Li, Jiale Ma, Junchi Yan, and Yi Chang. Nexco: Native solution expansion for diffusion-based combinatorial optimization. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Yu Wang, Yang Li, Junchi Yan, and Yi Chang. Strudico: Structured denoising diffusion with gradient-free inference-stage boosting for memory and time efficient combinatorial optimization. *Advances in Neural Information Processing Systems*, 38:71201–71228, 2026b.

Yuchen Zhu, Wei Guo, Jaemoo Choi, Guan-Horng Liu, Yongxin Chen, and Molei Tao. Mdns: Masked diffusion neural sampler via stochastic optimal control. *Advances in Neural Information Processing Systems*, 38:35260–35308, 2026.

APPENDIX

A RELATED WORK

A.1 LEARNING-BASED COMBINATORIAL OPTIMIZATION

Learning-based methods have emerged as an alternative to hand-crafted heuristics for combinatorial optimization, using neural networks to learn construction or improvement policies directly from problem instances (Bengio et al., 2021; Bogyrbayeva et al., 2024; Ba et al., 2026). Early approaches primarily rely on supervised learning from high-quality solutions (Vinyals et al., 2015; Luo et al., 2023; 2025a; Drakulić et al., 2025; Luo et al., 2025b; Li et al., 2023; 2024; 2026b; Wang et al., 2026a;b; Ma et al., 2025) or reinforcement learning with task-specific rewards (Kool et al., 2019; Kwon et al., 2020; 2021; Kim et al., 2022; Gao et al., 2023; Fang et al., 2024; Hottung et al., 2025; Qiu et al., 2022). More recently, generative models have provided an alternative paradigm that models distributions over candidate solutions and enables iterative refinement. Diffusion-based approaches, in particular, have achieved strong performance across graph and routing problems (Sun & Yang, 2023; Li et al., 2023; 2024; 2026b; Wang et al., 2026a;b). Despite their effectiveness, many of these methods depend on high-quality solution labels or task-specific supervision. This dependence has motivated unsupervised diffusion methods that instead learn directly from the combinatorial objective.

A.2 UNSUPERVISED DIFFUSION FOR COMBINATORIAL OPTIMIZATION

Recent unsupervised diffusion methods differ primarily in how optimization feedback is constructed and propagated through the generative process. DiffUCO (Sanokowski et al., 2024) formulates diffusion-based CO through entropy-regularized reinforcement learning, allowing terminal optimization rewards to supervise the diffusion trajectory. Building on this formulation, SDDS (Sanokowski et al., 2025) introduces an importance-sampling-based objective to improve scalability while retaining unsupervised optimization feedback. A complementary line of work constructs more direct learning signals from the optimization objective. RLNN (Feng & Yang, 2025) derives a local objective from regularized Langevin dynamics, providing stable optimization signals without labeled solutions, although its local formulation offers limited mechanisms for assigning long-range credit across a sampled trajectory. CAM (Feng et al., 2026) addresses trajectory-level credit assignment through discrete adjoint dynamics, propagating terminal optimization information through the diffusion trajectory to construct structured supervision for individual transitions.

Collectively, these methods have developed increasingly effective mechanisms for estimating and propagating optimization feedback without solution labels. However, improved credit assignment does not necessarily imply that the resulting transition-level supervision is equally discriminative across different transition directions. In particular, the distinction between target-favored and target-suppressed transitions within constructive and corrective directions has received comparatively little attention. Our work investigates this complementary aspect of unsupervised diffusion training. Rather than introducing another mechanism for propagating terminal feedback, we analyze the discriminative structure of the resulting transition-level supervision, identify a constructive-corrective transition asymmetry, and develop CCTC to calibrate this supervision according to signal magnitude and transition direction.

A.3 Discrete Diffusion Training and Guidance

Beyond combinatorial optimization, recent work has investigated improved architectures, objectives, and optimization principles for discrete neural samplers. LEAPS (Holderrieth et al., 2025) and DNFS (Ou et al., 2026) develop locally equivariant architectures for discrete sampling, while MDNS (Zhu et al., 2026) formulates masked diffusion sampling from a stochastic-control perspective. Other approaches, including rKL-LD (Sanokowski et al., 2026) and PDNS (Guo et al., 2026a), study alternative training objectives and optimization principles for learning discrete generative dynamics. These works primarily improve the formulation or optimization of the underlying discrete sampler. A related direction adapts or steers pretrained diffusion and flow models using reward guidance, test-time adaptation, or search-based mechanisms (Kim et al., 2025; Jensen et al., 2026; Guo et al., 2026b). Such methods focus primarily on modifying the generation or adaptation process using external objectives. CCTC instead operates directly on the transition-level supervision used during training and introduces no additional inference-time guidance mechanism.

B Gradient Analysis of CCTC

This section provides the gradient analysis of the calibrated matching objectives introduced in Section 4. We analyze only the matching terms modified by CCTC, since the remaining regularization terms are unchanged from Eq. 5.

B.1 Node-Selection Objective

For node-selection problems, CCTC calibrates the signed transition signal as

$$\bar{y}_{t,i} = \tanh\left(\frac{y_{t,i}}{\gamma}\right), \tag{19}$$

and applies the direction-conditioned weight

$$\rho_{t,i} = \begin{cases} \alpha_{\mathrm{con}}, & X_{t,i} = 0 \text{ and } y_{t,i} > 0, \\ 1, & \text{otherwise}, \end{cases} \qquad \alpha_{\mathrm{con}} \geq 1, \tag{20}$$

The resulting calibrated signal is $\widetilde{y}_{t,i} = \rho_{t,i}\bar{y}_{t,i}$. From Eq. 15, the matching contribution associated with coordinate i is

$$\mathcal{L}_{\mathrm{match},t,i} = -p_{t,i}\widetilde{y}_{t,i} = -\rho_{t,i}\bar{y}_{t,i}p_{t,i}. \tag{21}$$

During optimization, the calibrated transition signal is treated as detached from the policy parameters. Let $z_{t,i}$ denote the corresponding transition logit and $p_{t,i} = \sigma(z_{t,i})$. Since

$$\frac{\partial p_{t,i}}{\partial z_{t,i}} = p_{t,i}(1 - p_{t,i}), \tag{22}$$

the matching-term gradient with respect to the logit is

$$\frac{\partial \mathcal{L}_{\mathrm{match},t,i}}{\partial z_{t,i}} = -\rho_{t,i}\bar{y}_{t,i}p_{t,i}(1 - p_{t,i}). \tag{23}$$

This expression makes the roles of the two CCTC components explicit. Bounded calibration restricts the magnitude of the signed signal through $|\bar{y}_{t,i}| \leq 1$, while direction-conditioned weighting selectively scales target-favored constructive contributions through $\rho_{t,i}$. In particular, increasing α_{con} increases the matching-gradient magnitude only when $X_{t,i} = 0$ and $y_{t,i} > 0$, leaving target-suppressed constructive transitions unamplified.

Since $\rho_{t,i} \leq \alpha_{\mathrm{con}}$, $|\bar{y}_{t,i}| \leq 1$, and $p_{t,i}(1 - p_{t,i}) \leq 1/4$, Eq. 23 satisfies

$$\left|\frac{\partial \mathcal{L}_{\mathrm{match},t,i}}{\partial z_{t,i}}\right| \leq \frac{\alpha_{con}}{4} \tag{24}$$

This bound applies to the matching contribution of an individual transition logit rather than the gradient of the complete training objective. Moreover, Eq. 23 retains the sigmoid factor $p_{t,i}(1-p_{t,i})$. Therefore, CCTC bounds the influence of large signed transition signals but does not eliminate gradient attenuation when the corresponding transition probability approaches zero or one.

B.2 Edge-Selection Objective

For edge-selection problems, the matching target $q_{t,e} \in [0,1]$ has a probabilistic interpretation. CCTC therefore calibrates the target around 0.5 as

$$\bar{q}_{t,e} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{2q_{t,e}-1}{\gamma}\right), \tag{25}$$

and defines the direction-conditioned weight as

$$\rho_{t,e} = \begin{cases} \alpha_{\mathrm{con}}, & X_{t,e}=0 \text{ and } q_{t,e} > 0.5, \\ 1, & \text{otherwise}, \end{cases} \qquad \alpha_{\mathrm{con}} \geq 1, \tag{26}$$

Unlike the signed node-selection signal, the calibrated probabilistic target is not multiplied directly by $\rho_{t,e}$. Instead, the direction-conditioned weight scales the binary cross-entropy matching term. For an individual edge,

$$\mathcal{L}_{\mathrm{match},t,e} = \rho_{t,e}\mathrm{BCE}(p_{t,e}, \bar{q}_{t,e}), \tag{27}$$

where

$$\mathrm{BCE}(p,q) = -q\log p - (1-q)\log(1-p). \tag{28}$$

Let $p_{t,e} = \sigma(z_{t,e})$. Treating $\bar{q}_{t,e}$ and $\rho_{t,e}$ as detached from the policy parameters, the standard binary cross-entropy derivative gives

$$\frac{\partial \mathcal{L}_{\mathrm{match},t,e}}{\partial z_{t,e}} = \rho_{t,e}\left(p_{t,e} - \bar{q}_{t,e}\right). \tag{29}$$

Eq. 29 shows that bounded calibration controls the probabilistic target through $\bar{q}_{t,e}$, while direction-conditioned weighting directly scales the corresponding matching-gradient contribution. In particular, α_{con} selectively increases the contribution of target-favored constructive edge transitions without increasing the weight of target-suppressed constructive transitions.

B.3 Comparison of the Two Instantiations

The node- and edge-selection formulations implement the same calibration principle through different matching objectives. In both cases, bounded calibration controls the scale of the supervision while preserving its preference, and direction-conditioned weighting selectively increases the contribution of target-favored constructive transitions. Their gradient mechanisms nevertheless differ because the underlying target semantics are different. The node-selection objective directly couples the transition probability with a signed optimization signal, yielding the multiplicative factor $p_{t,i}(1-p_{t,i})$ in Eq. 23. The edge-selection objective instead uses a probabilistic target through binary cross-entropy, yielding the residual form $p_{t,e} - \bar{q}_{t,e}$ in Eq. 29. CCTC therefore preserves the appropriate matching mechanism for each problem class rather than imposing an identical loss across node- and edge-selection problems.

C Detailed Experimental Setup

This section provides the complete experimental setup, including benchmark specifications, baseline configurations, decoding and post-processing procedures, and evaluation metrics.

C.1 Problems and Benchmarks

Following (Sun & Yang, 2023; Sanokowski et al., 2024; Ma et al., 2025; Feng et al., 2026), we evaluate CCTC on four representative binary combinatorial optimization problems, including maximum independent set (MIS), Max-Cut, the traveling salesman problem (TSP), and the capacitated vehicle routing problem (CVRP). These benchmarks cover both node- and edge-selection decision spaces and include different graph distributions and problem scales.

For MIS, we consider RB and Erdős–Rényi (ER) graphs at two scales. RB instances contain 200–300 or 800–1,200 nodes, while ER instances contain 700–800 or 9,000–11,000 nodes. We use the suffix "-[m–M]" to denote the corresponding node range. The ER-[9000–11000] benchmark

additionally evaluates size generalization, where the model trained on ER-[700–800] is directly evaluated on graphs containing 9,000–11,000 nodes without retraining.

For Max-Cut, we use Barabási–Albert (BA) graphs containing 800–1,200 nodes (Barabási & Albert, 1999). For routing problems, TSP instances contain either 500 or 1,000 cities sampled uniformly from the unit square (Sun & Yang, 2023), while CVRP instances contain 100 customers (Ma et al., 2025; Feng et al., 2026).

C.2 Baselines

We compare CCTC against classical optimization methods, supervised neural solvers, and unsupervised or reinforcement-learning-based approaches.

For MIS, we compare against the classical solver Gurobi and the specialized MIS solver KaMIS (Lamm et al., 2017), the supervised neural methods INTEL (Li et al., 2018), DGL (Böther et al., 2022), and DIFUSCO (Sun & Yang, 2023), and the unsupervised or reinforcement-learning-based methods DIMES (Qiu et al., 2022), DiffUCO (Sanokowski et al., 2024), SDDS (Sanokowski et al., 2025), RLNN (Feng & Yang, 2025), and CAM (Feng et al., 2026).

For TSP, the baselines include the classical solvers Concorde and LKH-3 (Helsgaun, 2017), the supervised neural methods GCN (Joshi et al., 2019), DIFUSCO (Sun & Yang, 2023), and FMIP (Li et al., 2026a), and the unsupervised or reinforcement-learning-based methods AM (Kool et al., 2019), POMO (Kwon et al., 2020), DIMES (Qiu et al., 2022), SDDS (Sanokowski et al., 2025), and CAM (Feng et al., 2026).

For Max-Cut, we compare against MQLib (Dunning et al., 2018), the supervised neural methods DIFUSCO (Sun & Yang, 2023) and FMIP (Li et al., 2026a), and the unsupervised methods DiffUCO (Sanokowski et al., 2024), SDDS (Sanokowski et al., 2025), RLNN (Feng & Yang, 2025), and CAM (Feng et al., 2026).

For CVRP, we compare against HGS (Vidal et al., 2013), the supervised neural methods DIFUSCO (Sun & Yang, 2023) and FMIP (Li et al., 2026a), and the unsupervised methods SDDS (Sanokowski et al., 2025) and CAM (Feng et al., 2026).

Following (Feng et al., 2026), results for non-diffusion baselines are taken from their reported evaluations, while diffusion-based methods are evaluated under the same task-specific experimental protocol. For SDDS, we use the forward-KL formulation and select between the Bernoulli and annealing denoising distributions according to validation performance. CAM serves as the primary baseline for isolating the effect of transition calibration because CCTC retains its model architecture and sampling procedure while modifying only the transition-level training supervision.

C.3 Decoding and Post-Processing

To isolate differences in the learned diffusion policies, all diffusion-based methods use the same generative-step budget, number of independent samples, decoder, and post-processing procedure for each task. Post-processing is applied only to the terminal state and does not access intermediate rollout states.

For MIS, the generated state is converted into a feasible independent set using greedy decoding. For Max-Cut, the generated binary state is rounded and subsequently refined through local flip optimization.

For TSP, the generated edge representation is converted into a feasible tour through greedy edge insertion, followed by 2-OPT local search. For CVRP, score-guided feasible route merging is used to construct feasible routes, followed by task-specific CVRP local search. These decoding and post-processing procedures are kept fixed when comparing diffusion-based methods on the same task (Sun & Yang, 2023; Sanokowski et al., 2024; Feng et al., 2026).

For each test instance, we generate multiple independent candidate solutions and report the best solution obtained among these sampling runs.

Table 5: Default CCTC-specific hyperparameters used in the main experiments. These parameters modify only the training objective and introduce no additional inference-time operation.

Symbol	Description	Default
γ	Target-calibration scale	1.0
α_{con}	Target-favored constructive weight	3.0

Table 6: Task-specific hyperparameters. S is the number of training instances sampled per epoch, B is the graph batch size, and B_μ is the mini-batch size used to optimize the generated rollout transitions. H and L denote the hidden dimension and number of message-passing layers, respectively.

Task	H	L	S	B	B_μ	T_{tr}	K_{tr}	T_{te}	K_{te}	Learning rate
MIS RB-[200–300]	64	8	4000	4	256	50	10	50	20	5×10^{-4}
MIS RB-[800–1200]	64	6	1000	1	128	50	20	50	20	5×10^{-4}
MIS ER-[700–800]	64	8	1000	1	512	50	20	200	20	5×10^{-4}
MIS ER-[9000–11000]	64	8	–	1	–	–	–	200	20	–
Max-Cut BA-[800–1200]	64	6	1000	1	256	50	20	50	20	5×10^{-4}
TSP-500	256	12	1024	4	1	10	8	10	16	2×10^{-4}
TSP-1000	256	12	1024	2	1	10	4	10	16	2×10^{-4}
CVRP-100	128	6	1024	1	8	10	4	10	4	2×10^{-4}

C.4 Evaluation Metrics

We report independent-set size for MIS and cut value for Max-Cut, where higher values indicate better solutions. For TSP and CVRP, we report tour or route length, where lower values are better.

We additionally report the relative optimality gap

$$\text{Gap}(c, c^\star) = \frac{|c - c^\star|}{|c^\star|}, \tag{30}$$

where c denotes the objective value obtained by the evaluated method and $c^\star$ denotes the reference value specified in the corresponding result table.

Runtime is measured as the wall-clock time required to process the complete test set sequentially. CCTC modifies only the training objective and introduces no changes to CAM's model architecture, diffusion sampling process, decoder, or post-processing procedure at inference time. Therefore, under the same hardware configuration and evaluation protocol, CCTC has the same inference-time computational cost as CAM, up to normal runtime variation. Unless otherwise specified, all reported solution-quality comparisons use the same task-specific evaluation protocol across diffusion-based methods.

C.5 Hyperparameter Configuration

Tables 5 and 6 summarize the default CCTC parameters and the task-specific training and inference configurations. Here, T_{tr} and K_{tr} denote the number of rollout steps and parallel trajectories used to generate training data, while T_{te} and K_{te} denote the corresponding inference-time values. Unless otherwise stated, CCTC uses the same architecture, optimizer, sampling budget, decoder, and post-processing procedure as CAM.

D Multi-Seed Construction-Collapse Dynamics

We further examine the construction-collapse dynamics of CAM across five training seeds. Consistent with Section 3.2, we operationally detect the onset of a collapse episode as the first epoch at which the constructive transition flux satisfies $\Phi_{01} < 10^{-3}$ for at least two consecutive epochs. This detection rule depends only on the constructive transition flux, while the conditional constructive flip probability, constructive discrimination, and validation solution quality are treated as associated diagnostics.

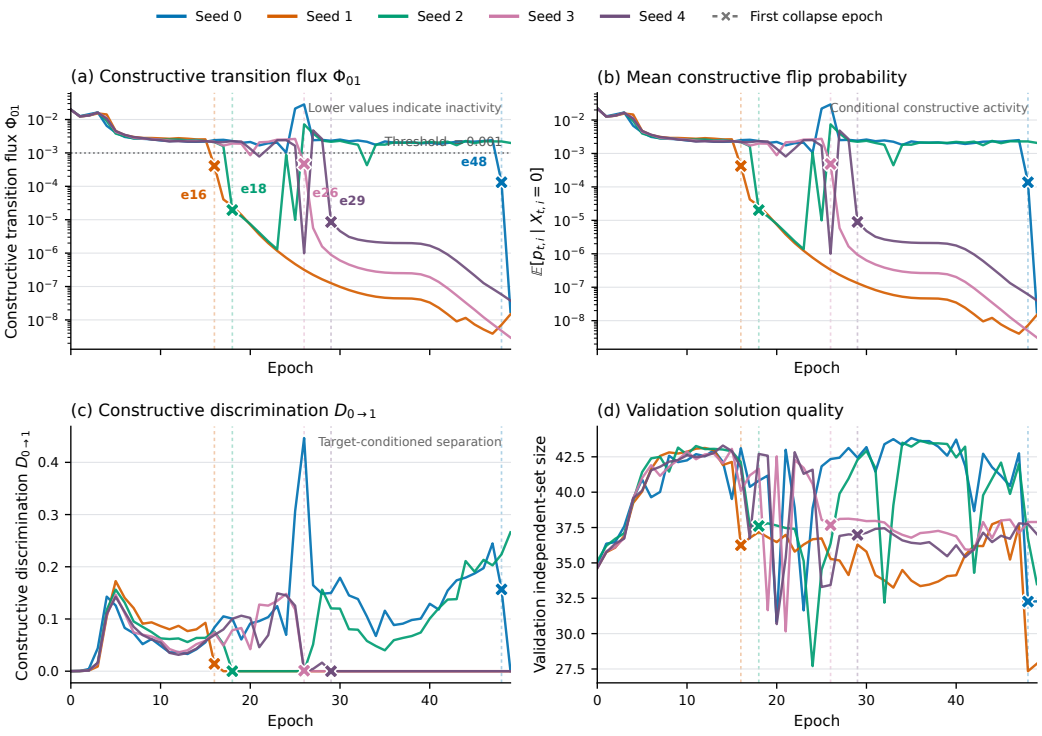


Figure 7: **Multi-seed construction-collapse dynamics of CAM on MIS ER-[700–800].** Curves show five unsmoothed training trajectories. (a) Constructive transition flux Φ_{01}, with the horizontal dashed line indicating the operational threshold $\Phi_{01} = 10^{-3}$. (b) Mean flip probability conditioned on constructive transitions. (c) Constructive transition discrimination $D_{0\to1}$. (d) Validation solution quality. Cross markers and vertical dashed lines indicate the first detected collapse epoch for each seed, defined as the first epoch for which Φ_{01} remains below the threshold for two consecutive epochs. Collapse onset and persistence vary substantially across seeds, and some trajectories subsequently recover constructive activity, illustrating both the reproducibility and trajectory-dependent variability of the failure mode.

Fig. 7 shows the unsmoothed training trajectories for all five seeds. As shown in Fig. 7 (a), every run exhibits at least one low-construction episode, although the onset and persistence vary substantially across seeds. The corresponding conditional flip probabilities in Fig. 7 (b) closely track the constructive flux, confirming that the low-flux regimes are associated with strongly suppressed $0 \to 1$ transition probabilities. Fig. 7 (c) further shows that persistent low-construction episodes are accompanied by a loss of constructive discrimination in several runs, whereas the recovered runs regain nonzero discrimination. Finally, Fig. 7 (d) shows that persistent low-construction regimes are generally associated with degraded validation solution quality, although the timing and magnitude of these changes vary across runs.

Overall, the multi-seed results show that the construction-collapse behavior observed in Fig. 2 is reproducible across CAM training runs but remains dependent on the individual training trajectory. In particular, some runs recover from an initially detected low-construction episode, indicating that construction collapse is not necessarily irreversible. These results support construction collapse as an extreme dynamic manifestation of weak constructive learning without implying a unique causal ordering among transition activity, discrimination, and solution quality.