Relaxation-Aligned State Control for Test-Time Scaling in Generative Combinatorial Optimization

Bohao Li[1,*] **Pei He**[1,*] **Yangming Guo**[1] **Ying Li**[1,2†]
[1]Northwestern Polytechnical University
[2]Ningbo Institute of Northwestern Polytechnical University
{bh_li, hepei002, liying_npu}@mail.nwpu.edu.cn
yangming_g@nwpu.edu.cn

Abstract

Generative solvers for combinatorial optimization benefit from additional test-time compute when their refinement states remain aligned with task relaxations. We introduce RAGenCO, a relaxation-aligned test-time scaling framework for generative combinatorial optimization. The central idea is to treat projection onto a task relaxation as a state control primitive: refinement first maintains a projected control state, and only then forms the task-dependent decoder input or expands candidates. For TSP and ATSP, Sinkhorn projection aligns matrix states with the cycle-cover relaxation during refinement and decoding. For MIS and MVC, Bernoulli projection controls refinement on the independent-set side, while raw logits are retained for terminal greedy decoding because their ordering carries the decoder signal. Best-of-B branching then reallocates compute by expanding candidates around aligned control states rather than sampling from unconstrained logits. Because these operations are used only at inference, the diffusion training objective remains unchanged. Under equal runtime budgets, this alignment of control state, decoder input, and branching yields a strong quality-runtime trade-off on TSP and ATSP and competitive results on MIS and MVC. The codes are publicly available at https://github.com/53mins/RAGenCO.

1 Introduction

Combinatorial optimization problems (COPs) arise across logistics planning [1], resource allocation [2, 3], and chip design [4], where the goal is to select a discrete structure that satisfies hard constraints while optimizing an objective. Classical solvers [5, 6] can be highly effective, but they often depend on problem-specific modeling choices and carefully engineered heuristics. Machine learning for combinatorial optimization (ML4CO) seeks to amortize this expertise by learning instance-conditioned heuristics from data. In this setting, generative models have become attractive as solvers that can be used across tasks because they implement an iterative inference process whose solution quality can improve with additional inference-time compute. Diffusion models are a prominent example: iterative denoising offers a direct quality-runtime trade-off, and recent work on acceleration, such as consistency training and adaptive computation [7, 8], reshapes this trade-off by reducing or reallocating refinement steps.

Applying diffusion-based iterative generation to COPs poses a distinctive challenge: the refinement process evolves in a continuous space, while good discrete solutions are recovered only after decoding and repair. In this setting, the main difficulty is often not exact feasibility at every intermediate iterate, but whether intermediate states stay close to task-relevant relaxations that keep the end-to-end inference procedure structurally aligned with the target task. If the refinement trajectory drifts away

†Corresponding author. * Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

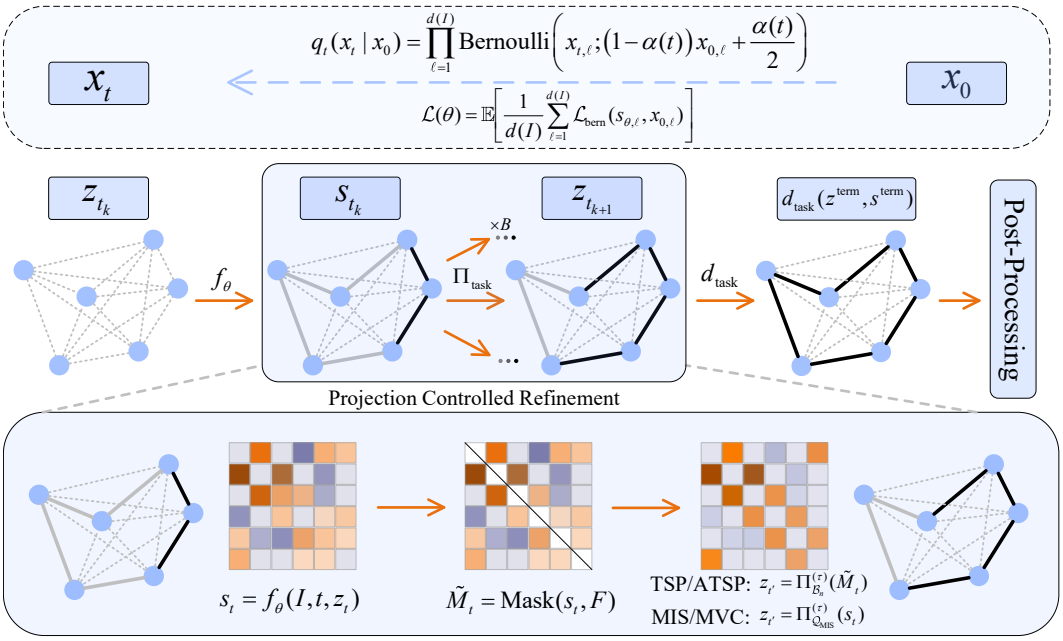


Figure 1: Overview of RAGenCO. **Training:** the denoiser f_θ predicts the clean solution representation x_0 from a corrupted sample x_t. **Inference:** RAGenCO first aligns the refinement state with a task relaxation, feeds the projected control state back into the next denoising step, and forms the terminal decoder input only at the end. For TSP and ATSP, masked edge logits are projected onto the cycle-cover relaxation and the final projected matrix is decoded. For MIS and MVC, Bernoulli projection controls refinement on the local stable-set relaxation, while terminal decoding uses raw vertex logits.

from these relaxations, the final decoder must recover feasible solutions from states that are less compatible with the desired combinatorial structure, and additional inference-time compute becomes harder to use effectively. This observation suggests that the bottleneck is not only how much extra compute is used at test time, but also whether each additional refinement or branch operates on states that remain compatible with the downstream decoder. This issue is especially visible in permutation-structured tasks such as the traveling salesman problem (TSP) and asymmetric TSP (ATSP), where degree structure should be respected before subtours are repaired, and in vertex-subset problems such as maximum independent set (MIS) and minimum vertex cover (MVC), where refinement should suppress conflicts without erasing the rank information used by the final decoder.

Recent diffusion-based ML4CO work studies inference-time compute allocation and solution quality. DIFUSCO [9] established a discrete diffusion baseline, T2T and Fast T2T [10, 7] integrated objective-guided refinement, GenSCO [11] emphasized search-style generation, StruDiCO [12] redesigned structured diffusion, and COExpander [8] improved adaptive decision fixing. These methods improve refinement or search, but leave open how to shape intermediate states so that additional branch-and-refine compute is used more effectively.

We propose RAGenCO, a framework for relaxation-aligned test-time scaling in generative combinatorial optimization. RAGenCO leaves the diffusion training objective unchanged and acts purely at inference time. Its thesis is simple: test-time compute is most useful when the candidate state is first aligned with a task relaxation and only then refined, decoded, or expanded. As summarized in Fig. 1, each refinement step maps the current control state to raw logits, applies task-specific entropic projection to obtain the next projected control state, and forms the terminal decoder input only after the final step.

This design has three implications: (1) projection is a primitive for state control, not post-processing: the projected state is fed back into refinement so that later denoising steps operate near a tractable relaxation; (2) vertex-subset decoding is deliberately decode-aware: MIS and MVC use projected control states to suppress conflicts during refinement, but keep raw logits for the final greedy ordering; and (3) best-of-B is not ordinary sampling from an unconstrained model; it expands candidates around states that have already been pulled toward the task relaxation. We instantiate this principle

with Sinkhorn projection for TSP/ATSP and Bernoulli projection for MIS/MVC, and we evaluate the resulting quality-runtime trade-off under equal runtime budgets.

2 Related Work

Learning-Based CO Solvers and Inference-Time Scaling. Learning-based CO solvers span constructive, improvement-based, divide-and-conquer, and global-prediction paradigms; ML4CO-Bench-101 [13] systematizes these settings. Diffusion-based solvers such as DIFUSCO [9] predict dense relaxed structures and rely on decoding and repair to recover feasibility. Recent work studies more effective inference-time compute allocation: T2T and Fast T2T [10, 7] combine denoising with objective-guided refinement, GenSCO [11] treats generation as a search operator, StruDiCO [12] redesigns structured diffusion, and COExpander [8] adaptively expands partial decisions. RAGenCO addresses the same problem from a state control perspective: before additional refinement or branching is spent, the candidate state is aligned with a task relaxation.

Constraint Handling and State Control. Neural CO solvers differ both in how they score candidates and in when they impose task structure. Many generative solvers defer hard constraints to decoding and repair [9], so intermediate states may remain only weakly aligned with the target combinatorial object. RAGenCO instead uses projection inside the refinement loop as a control operation. Sinkhorn projection onto the Birkhoff relaxation and Bernoulli projection onto a local stable-set relaxation are two task instantiations of the same principle. Relative to progressive decision fixing and other inference-time search wrappers [8, 11], RAGenCO emphasizes keeping the states used for compute scaling close to task-relevant relaxations.

Generalist ML4CO. Recent ML4CO work has also studied transfer and reuse across tasks, either through shared representations, common modeling principles, or more general solver designs. UniCO [14] reduces diverse problems to a general TSP representation encoded as a matrix, Unify ML4TSP [15] distills transferable design principles from TSP-style solvers, and GOAL [16] studies a more general solver for combinatorial optimization. These works emphasize shared modeling or problem reduction. RAGenCO instead keeps the same overall inference recipe across tasks while letting each problem retain its own relaxation, projection operator, and decoder.

3 Preliminaries

Combinatorial Optimization. We study both permutation-structured problems and vertex-subset problems. For TSP and ATSP on complete directed graphs with node set V and instance-dependent costs $C(I) \in \mathbb{R}_+^{|V|\times|V|}$, a solution is an edge-indicator matrix $x \in \{0,1\}^{|V|\times|V|}$ obtained by

$$\begin{aligned} \min_{x\in\mathcal{X}(I)} \quad & J(x;I) \;:=\; \sum_{i\in V}\sum_{j\in V} C_{ij}(I)\, x_{ij}, \\ \text{s.t.} \quad & x\mathbf{1} = b, \qquad x^\top \mathbf{1} = b, \qquad x_{ii} = 0, \end{aligned} \tag{1}$$

where $b \in \mathbb{N}^{|V|}$ specifies required in/out-degrees. For TSP and ATSP, $b = \mathbf{1}$ and $\mathcal{X}(I)$ additionally enforces a single Hamiltonian cycle. For vertex-subset problems on a graph $G = (V, E)$, we use an independent-set indicator $y \in \{0,1\}^{|V|}$ with objective

$$\max_{y\in\{0,1\}^{|V|}} \sum_{i\in V} y_i \qquad \text{s.t.} \quad y_i + y_j \le 1,\ \forall (i,j) \in E. \tag{2}$$

MVC is the complementary cover formulation on the same graph: if y is an independent-set indicator, then $c = \mathbf{1} - y$ is a vertex-cover indicator, yielding the equivalent objective $\min_{c\in\{0,1\}^{|V|}} \sum_{i\in V} c_i$ with constraints $c_i + c_j \ge 1$ for all $(i,j) \in E$.

Preliminaries for RAGenCO. RAGenCO maintains a *projected control state* z obtained from raw logits s by entropic projection; the final decoder receives a task-specific terminal input $d_{\text{task}}(z, s)$. At a generic level, the state control map is $\Pi_{\mathcal{Q}(I)}^{(\tau,\Phi_I)}(s) := \arg\max_{z\in\mathcal{Q}(I)} \big[\langle z, s\rangle - \tau\Phi_I(z)\big]$, where $\mathcal{Q}(I)$ is a tractable task-relevant relaxation and Φ_I is the corresponding negative-entropy geometry.

For permutation-structured problems, the feasible set is the set of permutation matrices $\mathcal{P}_n := \{P \in \{0,1\}^{n\times n} : P\mathbf{1} = \mathbf{1},\ P^\top\mathbf{1} = \mathbf{1}\}$, and its convex hull

$$\mathcal{B}_n := \{X \in \mathbb{R}_+^{n\times n} : X\mathbf{1} = \mathbf{1},\ X^\top\mathbf{1} = \mathbf{1}\} \tag{3}$$

serves as the cycle-cover relaxation. Let f_θ map (I, t, z_t) to logits $s_t = f_\theta(I, t, z_t) \in \mathbb{R}^{n\times n}$. For a score matrix M and temperature $\tau > 0$, define the entropic projection

$$\Pi_{\mathcal{B}_n}^{(\tau)}(M) \ := \ \arg\max_{X\in\mathcal{B}_n} \langle X, M\rangle + \tau H(X), \tag{4}$$

where $H(X) = -\sum_{i,j} X_{ij}\log X_{ij}$ is Shannon entropy. With a set F of masked matrix positions and masking in score space $\mathrm{Mask}(\cdot, F)$ defined in Eq. (8), one refinement step is $z_{t'} = \Pi_{\mathcal{B}_n}^{(\tau)}\Big(\mathrm{Mask}\big(f_\theta(I, t, z_t), F\big)\Big)$ for the next time $t' < t$. In practice we use fixed-budget Sinkhorn scaling on $K = \exp(\mathrm{Mask}(f_\theta(I, t, z_t), F)/\tau)$ [17, 18]; under standard positivity conditions, its iterates converge to the entropic projection onto $\mathcal{B}_n$ [19–21].

For vertex-subset problems, the control state is a vertex probability vector $z \in [0,1]^n$ on the independent-set side, with local stable-set relaxation

$$\mathcal{Q}_{\mathrm{MIS}} := \{z \in [0,1]^n :\ z_i + z_j \le 1,\ \forall (i,j) \in E\}, \tag{5}$$

and Bernoulli entropic projection

$$\Pi_{\mathcal{Q}_{\mathrm{MIS}}}^{(\tau)}(s) \ := \ \arg\max_{z\in\mathcal{Q}_{\mathrm{MIS}}} \left[\langle z, s\rangle + \tau\sum_{i=1}^{n} h(z_i)\right], \quad h(u) := -u\log u - (1-u)\log(1-u). \tag{6}$$

This relaxation is local but tractable, and it controls edge conflicts during refinement. Appendix A.1 places Eqs. (4) and (6) under the same KL/Bregman projection view.

4 Methodology

RAGenCO implements relaxation-aligned test-time scaling by coupling deterministic denoising with task-specific state control and anchor-based branching. Raw logits are projected to the next control state before the next denoising step, and the terminal decoder input is formed only after the final refinement step. Permutation-structured tasks apply masking before Sinkhorn projection, whereas vertex-subset tasks project raw logits for refinement control but preserve raw-logit ordering for terminal decoding. The two inference-time compute knobs are serial refinement depth T_s and parallel candidate budget B.

4.1 Inference Procedure

At each refinement step, the denoiser produces raw logits s, projection maps them to the next control state z, and the final decoder uses a task-specific terminal input extracted from $(z^{\mathrm{term}}, s^{\mathrm{term}})$. Thus projection shapes the state seen by later refinement steps rather than merely repairing the last decoder input.

We use a decreasing schedule $1 = t_0 > t_1 > \cdots > t_{T_s} = 0$ with uniform discretization $t_k = 1 - \frac{k}{T_s}$. For $k = 0, \ldots, T_s - 1$, RAGenCO predicts $s_{t_k} = f_\theta(I, t_k, z_{t_k})$ and updates the control state by $z_{t_{k+1}} = \Pi_{\mathrm{task}}(s_{t_k})$. After the last refinement step, let $s^{\mathrm{term}} := s_{t_{T_s-1}}$ and $z^{\mathrm{term}} := z_{t_{T_s}}$. RAGenCO uses $d_{\mathrm{task}}(z^{\mathrm{term}}, s^{\mathrm{term}}) = z^{\mathrm{term}}$ for TSP/ATSP and $d_{\mathrm{task}}(z^{\mathrm{term}}, s^{\mathrm{term}}) = s^{\mathrm{term}}$ for MIS/MVC, then returns $\hat{y} = D_{\mathrm{task}}(d_{\mathrm{task}}(z^{\mathrm{term}}, s^{\mathrm{term}}), I)$. The operator Π_{task} differs between permutation-structured and vertex-subset tasks. Theorem A.6 shows that, under local Lipschitz assumptions on the denoiser and bounded projection error, this projected refinement remains stable across the serial chain. The next two subsections instantiate these settings.

4.2 Sinkhorn Entropic Projection for Permutation-Structured Problems

For TSP and ATSP, the control state is a relaxed adjacency matrix and the terminal decoder input is the final projected matrix.

Let $\mathbf{1} \in \mathbb{R}^n$ denote the all ones vector. For TSP and ATSP, degree constraints correspond to the cycle-cover relaxation given by the Birkhoff polytope $\mathcal{B}_n = \{X \in \mathbb{R}_+^{n\times n} : X\mathbf{1} = \mathbf{1},\ X^\top\mathbf{1} = \mathbf{1}\}$

Definition 4.1 (Entropic projection onto $\mathcal{B}_n$)**.** Given a score matrix $M \in \mathbb{R}^{n\times n}$ and a temperature $\tau > 0$, the entropic projection onto $\mathcal{B}_n$ is

$$\begin{aligned} \Pi^{(\tau)}_{\mathcal{B}_n}(M) &= \arg\max_{X\in\mathcal{B}_n} \langle X, M\rangle + \tau H(X), \\ H(X) &= -\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}\log X_{ij}. \end{aligned} \tag{7}$$

For finite score matrices M, Eq. (7) is an entropic/KL projection computable by Sinkhorn scaling [17, 18, 22]; related differentiable permutation relaxations use the same operator [23]. In RAGenCO, Sinkhorn is the permutation-task instantiation of state control: it updates the refinement state while keeping iterates close to the cycle-cover relaxation.

Masking. Let $[n] = \{1, \dots, n\}$ and let $F \subseteq [n] \times [n]$ denote forbidden edges. We assume F is feasible; in particular, the diagonal mask $F = \{(i, i)\}$ is feasible for $n \geq 2$. We apply masking in score space using a large finite penalty $\Gamma > 0$:

$$\mathrm{Mask}(M, F)_{ij} = M_{ij} - \Gamma\, \mathbb{I}\{(i, j) \in F\}. \tag{8}$$

Given M and $\tau > 0$, the corresponding Sinkhorn kernel $K = \exp\big(\mathrm{Mask}(M, F)/\tau\big)$ is strictly positive. Forbidden positions are down-weighted by $\exp(-\Gamma/\tau)$; Lemma A.7 bounds the total projected mass assigned to them.

Fixed projection budget. We use a fixed Sinkhorn budget T_p for predictable latency and monitor the marginal residual

$$\mathrm{res}(X) = \|X\mathbf{1} - \mathbf{1}\|_1 + \|X^\top\mathbf{1} - \mathbf{1}\|_1. \tag{9}$$

Controlled residual bounds under finite T_p appear in Appendix A.6.

Task-specific decoding. For TSP, $d_{\mathrm{TSP}}(z, s) = z$ is decoded by greedy edge selection, subtour merging, and 2Opt. For ATSP, $d_{\mathrm{ATSP}}(z, s) = z$ is decoded by Hungarian assignment followed by patch-merge repair and local improvement [24, 25]; details appear in Algorithm 2. Here the aligned state supplies both the refinement state and the decoder input.

4.3 Decode-Aware Entropic Projection for Vertex-Subset Problems

For MIS and MVC, refinement should suppress local conflicts on the independent-set side, whereas terminal greedy decoding depends only on the ordering of the raw logits. RAGenCO therefore uses a projected control vector over vertices during refinement while setting $d_{\mathrm{MIS}}(z, s) = d_{\mathrm{MVC}}(z, s) = s$ at decoding time.

Let $G = (V, E)$ be the input graph and let $s \in \mathbb{R}^{|V|}$ denote raw vertex logits. We use Bernoulli entropic projection onto the local stable-set relaxation in Eq. (5); MIS decodes from the terminal raw logits, and MVC returns the complementary cover after the same decoding step.

Definition 4.2 (Entropic projection onto $\mathcal{Q}_{\mathrm{MIS}}$)**.** Given vertex logits $s \in \mathbb{R}^{|V|}$ and temperature $\tau > 0$, the MIS entropic projection is

$$\Pi^{(\tau)}_{\mathcal{Q}_{\mathrm{MIS}}}(s) \;=\; \arg\max_{z\in\mathcal{Q}_{\mathrm{MIS}}} \left[\langle z, s\rangle + \tau\sum_{i=1}^{|V|} h(z_i)\right]. \tag{10}$$

In the equivalent minimization form, Eq. (10) is strictly convex and therefore admits a unique optimum; Appendix A.3 gives the Bernoulli-KL view, dual update, and uniqueness argument. The resulting projected control state suppresses local conflicts during refinement.

Decode-aware decoding. Greedy MIS decoding depends only on the ordering of the raw logits and is invariant under strictly increasing pointwise transforms. In contrast, the projected control state $z = \Pi^{(\tau)}_{\mathcal{Q}_{\mathrm{MIS}}}(s)$ is coupled across adjacent vertices by the stable-set constraints and therefore need not preserve the ranking induced by the raw logits. It is therefore useful for refinement control but not as a terminal decoder input; MVC inherits the same logic. Theorems A.1 and A.2 formalize this separation.

4.4 Anchor Best-of-B Branching

With control and decoding fixed, RAGenCO allocates additional inference-time compute through a shared branching operator. To scale beyond sequential refinement (T_s), we introduce parallel branching around an anchor state z_{anc}, defined as the final serial output ($T_s > 1$) or a single prediction from a neutral initialization. We generate B perturbed candidates, project them back to the task-specific relaxation, refine once, decode, and keep the best. Algorithm 1 gives the unified anchor best-of-B branching operator. The branches are expanded around an aligned control state rather than drawn as unconstrained samples. Permutation-structured tasks perturb matrix-valued anchor logits, apply masked Sinkhorn projection, and decode from the final projected matrix; vertex-subset tasks perturb vertex logits on the independent-set side, apply Bernoulli entropic projection onto the local stable-set relaxation, and decode from terminal raw logits.

Algorithm 1: Anchor best-of-B branching operator.

Input: instance I, denoiser f_θ, anchor control state z_{anc} at time t_{anc}, branch count B, clipping constant $\varepsilon_{\mathrm{clip}}$, perturbation scale σ_{cand} (Gaussian) or β_{cand} (Gumbel), branching projection $\widetilde{\Pi}_{\mathrm{task}}$ (masked for permutation-structured problems; direct for vertex-subset problems), decoder D_{task}, objective selector $\mathrm{opt}_{\mathrm{task}} \in \{\min, \max\}$
Initialize: $s_{\mathrm{anc}} \leftarrow \mathrm{logit}\big(\mathrm{clip}(z_{\mathrm{anc}}; \varepsilon_{\mathrm{clip}})\big)$
for all $b \in \{1, \dots, B\}$ **in parallel do**
  sample $\xi^{(b)} \sim \mathcal{N}(0, I)$ **or** $g^{(b)} \sim \mathrm{Gumbel}(0, 1)$
  $s^{(b)} \leftarrow s_{\mathrm{anc}} + \sigma_{\mathrm{cand}}\, \xi^{(b)}$
  or $s^{(b)} \leftarrow s_{\mathrm{anc}} + \beta_{\mathrm{cand}}\, g^{(b)}$
  $\bar{z}^{(b)} \leftarrow \widetilde{\Pi}_{\mathrm{task}}(s^{(b)})$
  $s_0^{(b)} \leftarrow f_\theta(I, t_{\mathrm{anc}}, \bar{z}^{(b)})$
  $z_0^{(b)} \leftarrow \widetilde{\Pi}_{\mathrm{task}}(s_0^{(b)})$
  $d^{(b)} \leftarrow d_{\mathrm{task}}(z_0^{(b)}, s_0^{(b)})$
  $(\hat{y}^{(b)}, J^{(b)}) \leftarrow D_{\mathrm{task}}(d^{(b)}, I)$
end for
$b^\star \in \arg\mathrm{opt}_{\mathrm{task},\, b \in \{1,\dots,B\}}\ J^{(b)}$
$\hat{y} \leftarrow \hat{y}^{(b^\star)}$
Output: best feasible solution $\hat{y}$

For $\varepsilon_{\mathrm{clip}} \in (0, \frac{1}{2})$, we use $\mathrm{logit}(p) = \log \frac{p}{1-p}$ and

$$\mathrm{clip}(p; \varepsilon_{\mathrm{clip}}) = \min\{1 - \varepsilon_{\mathrm{clip}}, \max\{p, \varepsilon_{\mathrm{clip}}\}\}. \tag{11}$$

Under a simple conditional i.i.d. branch model, increasing B improves the quality of a single branch. A formal statement and bound are given in Theorem A.8.

4.5 Training Objective and Model Architecture

Let I denote a CO instance and x_0 a ground-truth solution representation. We train a time-conditioned denoiser f_θ to predict x_0 in logit space. The output logits match the task representation, and the objective is normalized by its dimension.

Backbone. We instantiate f_θ as a lightweight transformer that embeds node features, timestep t, and relaxed state x_t, then applies transformer blocks with additive structural biases; it outputs edge logits for TSP/ATSP and vertex logits for MIS/MVC.

Forward corruption. We use an entrywise Bernoulli corruption process. For permutation-structured tasks, the matrix form is

$$q_t(x_t \mid x_0) = \prod_{i=1}^{n} \prod_{j=1}^{n} \mathrm{Bernoulli}\big(x_{t,ij};\, \pi_{t,ij}\big), \qquad \pi_{t,ij} = (1 - \alpha(t))x_{0,ij} + \tfrac{1}{2}\alpha(t). \tag{12}$$

where $\alpha(t) = t$ for $t \in [0, 1]$. For training-time sampling we draw $u \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu = -0.8$ and $\sigma = 0.8$, then set $t = \mathrm{sigmoid}(u)$.

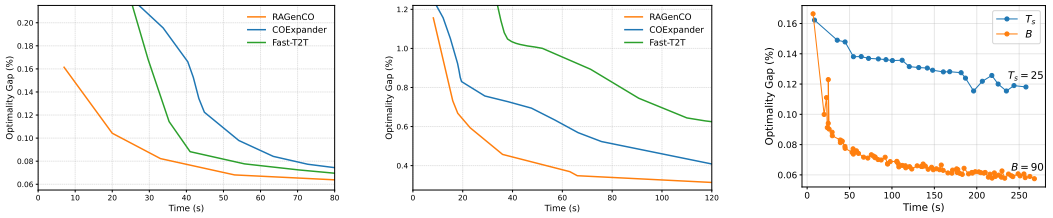
Training loss. We train with a Bernoulli negative log likelihood in logit space. For $x \in \{0, 1\}$ and $a \in \mathbb{R}$ define $\mathcal{L}_{\mathrm{bern}}(a, x) = \log(1 + e^a) - xa$. Let $d(I)$ denote the representation dimension. Writing $x_{0,\ell}$ and $s_{\theta,\ell}$ for task coordinates, the objective is

$$\mathcal{L}(\theta) = \mathbb{E}\left[\frac{1}{d(I)} \sum_{\ell=1}^{d(I)} \mathcal{L}_{\mathrm{bern}}(s_{\theta,\ell}, x_{0,\ell})\right]. \tag{13}$$

The expectation is over (I, x_0), $t \sim p(t)$, and $x_t \sim q_t(\cdot \mid x_0)$, where $s_\theta = f_\theta(I, t, x_t)$. Projection and branching are inference-only, so the training objective remains unchanged.

Table 1: Results on TSP. AS: Active Search, G: Greedy Decoding, S: Sampling Decoding, BS: Beam Search.

Method	**TSP-100** (1280 inst.)			**TSP-500** (128 inst.)			**TSP-1000** (128 inst.)		
	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
Mathematical Solvers or Heuristics									
Concorde [26]	7.76	0.00%	4.1m	16.55	0.00%	33.2m	23.12	0.00%	2.5h
LKH-3(512) [6]	7.76	0.00%	2.1m	16.55	0.00%	2.1m	23.12	0.01%	5.3m
Non-Generative Neural Solvers									
AM [27] BS	7.95	2.48%	20.8m	19.53	18.03%	4.3m	29.90	29.24%	21.1m
POMO [28] ×8 augment	7.77	0.13%	13s	20.22	22.19%	1.5m	32.55	40.57%	13.4m
GCN [29] BS	8.41	8.38%	9.7m	30.37	83.55%	58.9m	51.26	121.73%	1.4h
DIMES [30] AS+S+2Opt	7.93	2.18%	5.2h	17.29	4.48%	3.3h	24.32	5.17%	7.5h
LEHD [31] PRC 100	7.76	0.01%	2.9m	16.61	0.34%	12.4m	23.44	1.22%	1.2h
BQ-NCO [32] BS	7.76	0.01%	6.6m	16.64	0.55%	23.2m	23.47	1.38%	1.1h
GLOP [33]	–	–	–	16.91	1.99%	2.3m	23.84	3.11%	5.0m
COExpander (T_s=5,D_s=1) +2Opt [8]	7.76	0.08%	1.2m	16.69	0.85%	28s	23.43	1.33%	1.6m
COExpander (T_s=5,D_s=3) +2Opt [8]	7.76	0.04%	3.5m	16.63	0.52%	1.3m	23.34	0.98%	4.2m
Generative Neural Solvers									
DIFUSCO (T_s=50) G+2Opt [9]	7.78	0.28%	13.2m	16.80	1.49%	4.6m	23.55	1.89%	15.8m
DIFUSCO (T_s=100) S+2Opt [9]	7.76	0.06%	1.4h	16.69	0.87%	30.4m	23.42	1.31%	1.7h
T2T (T_s=50,T_g=30) G+2Opt [10]	7.76	0.07%	31.3m	16.68	0.82%	9.8m	23.44	1.40%	28.9m
T2T (T_s=50,T_g=30) S+2Opt [10]	7.76	0.02%	1.6h	16.63	0.48%	33.2m	23.37	1.07%	1.6h
Fast T2T (T_s=5) G+2Opt [7]	7.76	0.07%	1.4m	16.71	1.00%	52s	23.38	1.13%	3.1m
RAGenCO (T_s=1,B=3) G+2Opt	7.76	0.10%	20s	16.67	0.76%	21s	23.43	1.35%	1.5m
RAGenCO (T_s=3,B=3) G+2Opt	7.76	0.08%	50s	16.64	0.59%	23s	23.42	1.31%	1.6m
RAGenCO (T_s=5,B=10) G+2Opt	7.76	0.06%	1.3m	16.61	0.37%	1.1m	23.37	1.09%	5.1m
RAGenCO (T_s=15,B=40) G+2Opt	7.76	0.04%	4.2m	16.59	0.23%	4.0m	23.32	0.88%	20.6m



(a) TSP-100. (b) TSP-500. (c) Scaling B vs. T_s.

Figure 2: Compute allocation under equal runtime budgets on TSP. (a–b) Optimality gap (%) versus runtime on TSP-100 and TSP-500. (c) On TSP-100, scaling B is more effective than scaling T_s; T_s sweeps fix $B = 1$, and B sweeps fix $T_s = 1$.

5 Experiments

We evaluate RAGenCO on standard TSP, ATSP, MIS, and MVC benchmarks using the official ML4CO-Bench-101 [13] splits and reference solutions, enabling direct comparison with recent diffusion-based baselines.

5.1 Experimental setup

Problems and datasets. We use the official ML4CO-Bench-101 [13] splits: TSP/ATSP on complete graphs, MIS on RB-SMALL, RB-LARGE, ER-700-800, and SATLIB, and MVC trained on RB-SMALL/RB-LARGE and evaluated on RB-SMALL, RB-LARGE, Twitter, and COLLAB. We report official test set results at each scale; TSP cross-scale generalization is deferred to Appendix C.7. Hyperparameters appear in Tab. 7.

RAGenCO implementation and inference settings. RAGenCO follows Sec. 4, with serial depth T_s and branch budget B as its inference-time compute knobs after relaxation alignment. Permutation tasks use masked fixed-budget Sinkhorn projection with temperature τ and budget T_p; MIS/MVC use the decode-aware vertex-subset setting of Sec. 4.3. Unless noted otherwise, all results use batch size 1. Training is performed on a single NVIDIA A100 40GB GPU, while all reported RAGenCO inference runtimes are measured on a single NVIDIA RTX 3080 GPU. Following the benchmark

Table 2: Results on ATSP.

Method	**ATSP-50** (2500 inst.)			**ATSP-100** (2500 inst.)			**ATSP-200** (100 inst.)		
	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
LKH (100) [6]	1.55	0.00%	2.7m	1.57	0.00%	5.0m	1.56	0.00%	24s
LKH (1000) [6]	1.55	0.00%	4.2m	1.57	0.00%	9.9m	1.56	0.00%	1.2m
MatNet [34]	1.58	1.32%	1.7m	1.62	3.26%	2.5m	3.83	145.08%	11s
MatNet [34] (×16)	1.56	0.30%	1.7m	1.59	1.58%	2.9m	3.73	138.40%	16s
GOAL [16]	1.65	5.85%	12.9m	1.64	4.61%	26.7m	1.62	3.35%	2.3m
GOAL [16] + Beam-16	1.63	4.74%	14.6m	1.62	3.53%	37.9m	1.61	2.86%	7.8m
COExpander (T_s=1,D_s=1) +2Opt	1.61	3.81%	22s	1.63	4.11%	50s	1.65	5.64%	7s
COExpander (T_s=5,D_s=3) +2Opt	1.57	1.13%	4.1m	1.60	2.26%	11.1m	1.60	2.57%	90s
COExpander (S=4,T_s=5,D_s=3) +2Opt	1.56	0.17%	11.2m	1.58	0.95%	37.8m	1.59	1.50%	5.8m
RAGenCO (T_s=1,B=1) +2Opt	1.56	0.60%	19s	1.59	1.49%	25s	1.60	2.17%	2s
RAGenCO (T_s=1,B=10) +2Opt	1.56	0.41%	54s	1.58	0.84%	1.7m	1.59	1.55%	13s
RAGenCO (T_s=5,B=100) +2Opt	1.56	0.33%	6.3m	1.58	0.62%	14.2m	1.58	1.20%	1.5m

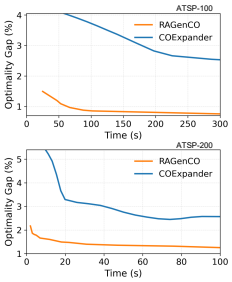


Figure 3: Optimality gap (%) versus runtime on ATSP.

Table 3: Vertex-subset results: MIS (top) and MVC (bottom); * marks results from the original paper.

Maximum Independent Set (MIS)	**RB-SMALL**			**RB-LARGE**			**ER-700-800**			**SATLIB**		
	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time	Obj.↑	Gap↓	Time
KaMIS [35]	20.09	0.00%	16.2h	43.00	0.00%	20.2h	44.97	0.00%	5.5h	425.95	0.00%	8.6h
Gurobi [36]	20.09	0.00%	11.5m	42.19	1.83%	12.0h	38.78	13.75%	5.5h	425.92	0.01%	1.4h
DIFUSCO (T_s=50) [9]	–	–	–	–	–	–	38.64	14.08%	16.7m	424.74	0.29%	41.7m
DIFUSCO (S=4,T_s=50) [9]	–	–	–	–	–	–	40.97	8.89%	35.0m	425.11	0.20%	51.3m
Fast T2T (T_s=5,T_g=5) [7]	19.50	2.89%	6.3m	–	–	–	40.69	9.51%	6.1m	424.44	0.36%	25.9m
Fast T2T (S=4,T_s=5,T_g=5) [7]	19.74	1.70%	17.2m	–	–	–	41.74	7.17%	12.6m	425.00	0.25%	46.3m
DiffUCO: CE [37]	19.20	4.37%	7.3m	38.49	10.43%	1.8h	–	–	–	–	–	–
DiffUCO: CE (S=4) [37]	19.38	3.46%	24.6m	39.55	7.94%	9.1h	–	–	–	–	–	–
COExpander (D_s=20) [8]	19.66	2.10%	1.7m	41.25	4.04%	14.4m	42.38	5.75%	2.8m	425.05	0.21%	3.5m
COExpander (S=4,D_s=20) [8]	19.71	1.88%	2.4m	41.44	3.58%	27.8m	42.56	5.34%	4.5m	424.78	0.28%	4.7m
COExpander (S=4,T_s=20,D_s=5) [8]	19.79	1.48%	13.4m	40.69	5.31%	1.9h	42.17	6.22%	27.1m	425.28	0.16%	22.4m
RAGenCO (T_s=1,B=3)	19.52	2.76%	1.0m	40.95	4.71%	3.1m	42.50	5.52%	34s	424.84	0.26%	6.2m
RAGenCO (T_s=1,B=10)	19.69	1.94%	2.9m	41.42	3.61%	8.0m	43.02	4.35%	1.5m	425.04	0.22%	19.5m
RAGenCO (T_s=1,B=20)	19.76	1.61%	5.0m	41.65	3.09%	15.1m	43.27	3.78%	2.7m	425.14	0.19%	41.6m
RAGenCO (T_s=1,B=50)	19.85	1.15%	11.5m	41.85	2.61%	39.4m	43.63	3.00%	7.4m	425.21	0.18%	2.0h

Minimum Vertex Cover (MVC)	**RB-SMALL**			**RB-LARGE**			**Twitter**			**COLLAB**		
	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time	Obj.↓	Gap↓	Time
Gurobi* [36]	205.76	0.00%	27.8m	968.23	0.00%	40.3h	85.25	0.00%	26s	65.09	0.00%	58s
Meta-EGN* [38]	208.97	1.56%	2.5m	1010.69	4.40%	8.6m	92.73	8.67%	37s	65.69	1.25%	2.5m
COExpander (D_s=20) [8]	206.55	0.38%	1.5m	969.90	0.17%	11.7m	85.50	0.30%	30s	65.16	0.12%	1.1m
COExpander (S=4,D_s=20) [8]	206.36	0.29%	1.6m	969.81	0.16%	24.5m	85.49	0.31%	21s	65.14	0.09%	1.1m
RAGenCO (T_s=1,B=3)	206.35	0.28%	47s	971.87	0.38%	2.2m	85.99	0.87%	25s	65.47	0.59%	1.1m
RAGenCO (T_s=1,B=5)	206.28	0.25%	1.1m	971.52	0.34%	3.4m	85.64	0.46%	30s	65.31	0.35%	1.4m
RAGenCO (T_s=1,B=10)	206.18	0.20%	1.9m	971.12	0.30%	6.4m	85.50	0.29%	49s	65.15	0.10%	2.3m
RAGenCO (T_s=1,B=20)	206.12	0.17%	3.3m	970.82	0.27%	12.3m	85.44	0.22%	1.4m	65.13	0.07%	4.0m

protocol, runtime is reported as end-to-end wall-clock time aggregated over the full official test split and includes inference, projection, decoding, and any listed post-processing. Projection sweeps and vertex-subset diagnostics are deferred to Appendices C.1 and C.6.

Metrics. For TSP and ATSP, we report objective and optimality gap (%) relative to $J^\star$: $\mathrm{Gap}(x) = \frac{J(x)-J^\star}{J^\star} \cdot 100\%$. For MIS and MVC, we analogously report task objective together with $\mathrm{Gap}(y) = \frac{J^\star-J(y)}{J^\star} \cdot 100\%$ or $\mathrm{Gap}(c) = \frac{J(c)-J^\star}{J^\star} \cdot 100\%$. Runtime is end-to-end wall-clock time over the full official test split, including inference, projection, decoding, and any listed post-processing, and is *not* normalized per instance.

5.2 Results on Permutation-Structured Problems

TSP Results. Tab. 1 shows that RAGenCO is competitive with strong adaptive expansion and diffusion baselines and improves as T_s and B increase. The runtime curves in Figs. 2a and 2b show favorable optimality gap versus runtime relative to recent baselines.

Compute allocation behavior. Fig. 2c shows that scaling B around aligned anchor states yields larger gains per unit time than scaling T_s on TSP-100 under matched runtime budgets; analogous plots for ATSP-100, RB-SMALL (MIS), and RB-SMALL (MVC) appear in Appendix C.3 and Fig. 6.

ATSP Results. Tab. 2 shows that RAGenCO is competitive with standard adaptive expansion and stronger multi-start baselines, while Fig. 3 suggests lower gaps under comparable end-to-end budgets.

Because ATSP is sensitive to feasibility restoration from the soft directed matrix, decoding is not a minor implementation detail: Tab. 11 shows that Hungarian $\rightarrow$ patch-merge $\rightarrow$ 2Opt clearly outperforms greedy decoding at similar runtimes.

5.3 Results on Vertex-Subset Problems

MIS Results. Tab. 3 shows that RAGenCO remains competitive across all four MIS suites. On the harder RB and ER benchmarks, increasing branch budget steadily improves independent-set size, and on RB-LARGE the best setting reaches a 2.61% gap.

Decoding behavior. In MIS, greedy decoding depends only on the ordering of the raw logits. This matches the decode-aware separation in Sec. 4.3: projection is useful because it suppresses local conflicts during refinement, while final greedy decoding remains most stable when driven by the raw-logit ordering. The compact ablation in Appendix C.1.1 supports this interpretation directly: using projection during refinement while still decoding from raw logits outperforms both removing projection altogether and decoding directly from the projected control state.

MVC Results. The lower panel of Tab. 3 shows that the same decode-aware recipe transfers to MVC with only the terminal map changed to the complementary cover, and Appendix C.1.2 confirms the same projected-refinement advantage.

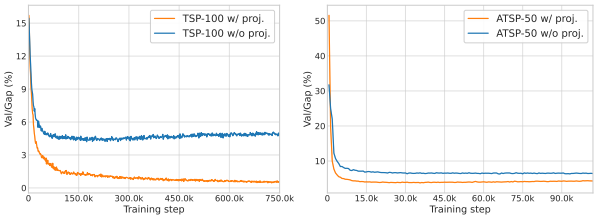
5.4 Ablation Study

Relaxation alignment and state control. Tab. 4 compares the same checkpoints with and without projection. Across tasks, aligning the refinement state with the task relaxation improves or stabilizes end-to-end performance under matched inference settings; Fig. 5 gives the validation-time view, and decode-aware vertex-subset controls and trajectory diagnostics are deferred to Appendices C.1 and C.9.

Additional ablations. Extended ablations on projection hyperparameters appear in Tabs. 12 to 17; anchor perturbation and ATSP decoding ablations appear in Tabs. 11 and 18.

Table 4: Ablation study of task-specific entropic projection.

Method	w/ proj.		w/o proj.	
	Gap	Time	Gap	Time
TSP-100 (T_s=5,B=10)	0.06%	1.3m	0.24%	1.5m
TSP-500 (T_s=5,B=10)	0.37%	1.1m	0.72%	1.3m
TSP-1000 (T_s=5,B=10)	1.09%	5.1m	1.38%	5.0m
ATSP-50 (T_s=1,B=10)	0.41%	54s	0.44%	56s
ATSP-100 (T_s=1,B=10)	0.84%	1.7m	0.97%	1.7m
ATSP-200 (T_s=1,B=10)	1.55%	13s	1.66%	13s
MIS RB-SMALL (T_s=1,B=20)	1.61%	5.0m	2.24%	3.6m
MIS RB-LARGE (T_s=1,B=20)	3.09%	15.1m	4.99%	12.0m
MVC RB-SMALL (T_s=1,B=20)	0.17%	3.3m	0.32%	3.4m
MVC RB-LARGE (T_s=1,B=20)	0.27%	12.3m	0.51%	11.9m



(a) TSP-100 (b) ATSP-50

Figure 5: Validation gap (%) versus training steps.

6 Conclusion

RAGenCO shows that test-time compute in generative CO is more effective when candidate states are first aligned with task relaxations and only then refined, decoded, or branched. Projection acts as a state control primitive during inference rather than as a final repair step. The same principle supports decode-aware vertex-subset inference, where projection controls refinement while terminal decisions are made from raw logits, and it supports anchor best-of-B branching around aligned states. Together these choices yield strong end-to-end performance under equal runtime budgets on the tasks studied here.

Limitations: for TSP and ATSP, RAGenCO controls only the cycle-cover relaxation and still relies on the decoder for subtour restoration; for MIS and MVC, projection improves conflict suppression during refinement but does not directly optimize the final rank ordering used by node decoding; and best-of-B increases parallel compute and memory. Future work includes stronger relaxations, adaptive projection schedules, and training procedures that better align denoising with controlled inference states.

Acknowledgments and Disclosure of Funding

Funding in direct support of this work: This work was partly supported by the Young Talent Fund of Xi'an Association for Science and Technology (Grant 0959202513148), the Zhejiang Provincial Excellent Funding for Postdoctoral Researchers (Grant ZJ2025156), and the China Postdoctoral Science Foundation (Grant 2025M784428). Bohao Li was supported by the Doctoral Student Academic Scholarship of Northwestern Polytechnical University.

Competing interests: The authors declare no competing interests.

References

[1] Qi Wang and Chunlei Tang. Deep reinforcement learning for transportation network combinatorial optimization: A survey. *Knowledge-Based Systems*, 233:107526, 2021.

[2] Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *29th IEEE Conference on Decision and Control*, pages 2130–2132. IEEE, 1990.

[3] Guman Singh and Mohammad Rizwanullah. Combinatorial optimization of supply chain networks: A retrospective & literature review. *Materials today: proceedings*, 62:1636–1642, 2022.

[4] Andrew B Kahng, Chul-Hong Park, Xu Xu, and Hailong Yao. Layout decomposition for double patterning lithography. In *2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 465–472. IEEE, 2008.

[5] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

[6] 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.

[7] 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. In *Advances in Neural Information Processing Systems*, 2024.

[8] Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. Coexpander: Adaptive solution expansion for combinatorial optimization. In *International Conference on Machine Learning*, 2025.

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

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

[11] Yang Li, Lvda Chen, Haonan Wang, Runzhong Wang, and Junchi Yan. Generation as search operator for test-time scaling of diffusion-based combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2025.

[12] 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. In *Advances in Neural Information Processing Systems*, 2025.

[13] Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. Ml4co-bench-101: Benchmark machine learning for classic combinatorial problems on graphs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[14] Wenzheng Pan, Hao Xiong, Jiale Ma, Wentao Zhao, Yang Li, and Junchi Yan. Unico: On unified combinatorial optimization via problem reduction to matrix-encoded general tsp. In *International Conference on Learning Representations*, 2025.

[15] Yang Li, Jiale Ma, Wenzheng Pan, Runzhong Wang, Haoyu Geng, Nianzu Yang, and Junchi Yan. Unify ml4tsp: Drawing methodological principles for tsp and beyond from streamlined design space of learning and search. In *International Conference on Learning Representations*, 2025.

[16] Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner. *arXiv preprint arXiv:2406.15079*, 2024.

[17] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[18] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[19] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[20] Richard L Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

[21] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018.

[22] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[23] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*, 2018.

[24] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[25] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[26] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. Concorde TSP solver, 2006.

[27] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:77416–77429, 2023.

[33] Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. Glop: Learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 20284–20292, 2024.

[34] 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.

[35] Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F Werneck. Finding near-optimal independent sets at scale. In *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)*, pages 138–150. SIAM, 2016.

[36] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2023.

[37] Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *International Conference on Machine Learning*, pages 43346–43367. PMLR, 2024.

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

[39] Garrett Birkhoff. Tres observaciones sobre el algebra lineal. *Universidad Nacional de Tucumán. Revista A*, 5:147–151, 1946.

[40] John von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, 2:5–12, 1953.

[41] Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pages 146–158, 1975.

[42] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[43] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

A Theory, Proofs, and Additional Analysis

A.1 Negative-Entropy Projection View

The four tasks studied in RAGenCO share the same abstract control primitive: align model scores with a task-specific tractable relaxation under a negative-entropy geometry, then feed the aligned state back into inference. Let I denote an instance, let $\mathcal{Z}(I) \subseteq \mathbb{R}^{d(I)}$ denote the ambient relaxed state space, and let $\mathcal{Q}(I) \subseteq \mathcal{Z}(I)$ denote the tractable relaxation used for state control. Let $s \in \mathbb{R}^{d(I)}$ denote model scores and let Φ_I be a convex negative-entropy potential on $\mathcal{Z}(I)$. We define the generic entropic control operator by

$$\Pi_{\mathcal{Q}(I)}^{(\tau,\Phi_I)}(s) \;:=\; \arg\max_{z\in\mathcal{Q}(I)} \left[\langle z, s\rangle - \tau\Phi_I(z)\right], \qquad \tau > 0. \tag{14}$$

Equivalently, if $q(s)$ denotes the unconstrained Gibbs/Bernoulli state induced by s, Eq. (14) is the Bregman projection of $q(s)$ onto $\mathcal{Q}(I)$ under the divergence generated by Φ_I.

We use this formulation in two settings.

- **TSP / ATSP.** Here $\mathcal{Z}(I) = \mathbb{R}_+^{n\times n}$, $\mathcal{Q}(I) = \mathcal{B}_n$, and $\Phi_I(X) = \sum_{i,j} X_{ij}\log X_{ij}$, which yields the KL projection onto the Birkhoff relaxation that is computable by Sinkhorn scaling.
- **MIS / MVC.** Here $\mathcal{Z}(I) = [0,1]^n$, $\mathcal{Q}(I) = \mathcal{Q}_{\mathrm{MIS}}$, and $\Phi_I(z) = \sum_i \big[z_i \log z_i + (1 - z_i)\log(1-z_i)\big]$, which yields the Bernoulli-KL projection onto the local stable-set relaxation on the independent-set side. MVC uses the same projected control state during refinement and recovers the output cover by complementing the set decoded from the terminal raw logits.

In both settings, the projected control state is fed back into refinement, while terminal decoding is specified by $d_{\mathrm{TSP}}(z,s) = d_{\mathrm{ATSP}}(z,s) = z$ and $d_{\mathrm{MIS}}(z,s) = d_{\mathrm{MVC}}(z,s) = s$.

Table 5 instantiates this abstract interface across the four tasks studied in RAGenCO.

Table 5: Task-specific relaxation alignment and decoder choices used by RAGenCO.

task	control state z	terminal decoder input $d_{\mathrm{task}}(z,s)$	relaxation $\mathcal{Q}(I)$	projection solver	decoder/repair
TSP	projected soft adjacency	terminal projected matrix	cycle-cover polytope $\mathcal{B}_n$	masked Sinkhorn	greedy + merge + 2Opt
ATSP	projected directed adjacency	terminal projected matrix	cycle-cover polytope $\mathcal{B}_n$	masked Sinkhorn	Hungarian + patch-merge + 2Opt
MIS	projected vertex state	raw vertex logits	local stable-set relaxation $\mathcal{Q}_{\mathrm{MIS}}$	Bernoulli entropic projection	greedy MIS
MVC	projected vertex state (IS side)	raw vertex logits (IS side)	local stable-set relaxation $\mathcal{Q}_{\mathrm{MIS}}$	Bernoulli entropic projection	complement of greedy MIS

A.2 Permutation matrices and the Birkhoff relaxation

For permutation-structured COPs, we represent a discrete solution by a permutation matrix. Let $\mathbf{1} \in \mathbb{R}^n$ denote the all-ones vector and define

$$\begin{aligned} \mathcal{P}_n &:= \{P \in \{0,1\}^{n\times n} : P\mathbf{1} = \mathbf{1},\ P^\top\mathbf{1} = \mathbf{1}\}, \\ \mathcal{B}_n &:= \{X \in \mathbb{R}_+^{n\times n} : X\mathbf{1} = \mathbf{1},\ X^\top\mathbf{1} = \mathbf{1}\}. \end{aligned} \tag{15}$$

Matrices in $\mathcal{P}_n$ encode cycle covers, and Hamiltonian cycles form a strict subset. The set $\mathcal{B}_n$ is the cycle-cover relaxation that enforces degree marginals but does not impose subtour elimination. By the classical Birkhoff–von Neumann theorem, $\mathcal{B}_n = \mathrm{conv}(\mathcal{P}_n)$, so linear objectives over $\mathcal{B}_n$ attain optima at permutation matrices [39, 40].

A.3 Decode-Aware Entropic Projection on the Local Stable-Set Relaxation

For vertex-subset problems on a graph $G = (V, E)$ with vertex logits $s \in \mathbb{R}^{|V|}$, RAGenCO uses a decode-aware entropic projection design on the independent-set side. Its projected control operator is the Bernoulli entropic projection onto the local stable-set relaxation

$$\mathcal{Q}_{\mathrm{MIS}} := \{z \in [0,1]^{|V|} : z_i + z_j \le 1,\ \forall (i,j) \in E\}. \tag{16}$$

given by

$$\Pi_{\mathcal{Q}_{\mathrm{MIS}}}^{(\tau)}(s) = \arg\max_{z\in\mathcal{Q}_{\mathrm{MIS}}} \left[\langle z, s\rangle + \tau\sum_i h(z_i)\right], \qquad h(u) = -u\log u - (1-u)\log(1-u). \tag{17}$$

Equivalently, if $q_i = \sigma(s_i/\tau)$, Eq. (17) is the Bernoulli-KL projection of q onto $\mathcal{Q}_{\mathrm{MIS}}$. Since the feasible set is defined by linear inequalities and the Bernoulli-KL objective is strictly convex on $(0,1)^{|V|}$, the optimum is unique.

Introducing dual variables $\lambda_{ij} \geq 0$ for the edge constraints gives the stationarity condition

$$z_i(\lambda) = \sigma\left(\frac{s_i}{\tau} - \sum_{j:(i,j)\in E} \lambda_{ij}\right), \tag{18}$$

which motivates the projected dual-ascent update used by our implementation:

$$\lambda_{ij}^{k+1} = \left[\lambda_{ij}^{k} + \eta\left(z_i^k + z_j^k - 1\right)\right]_+. \tag{19}$$

We monitor the residual

$$\mathrm{res}_{\mathrm{MIS}}(z) := \max_{(i,j)\in E} (z_i + z_j - 1)_+, \tag{20}$$

which measures how well the projected control state respects the local stable-set relaxation. MIS decodes the selected set directly from the raw logits, while MVC complements the decoded set into a cover. The decode-aware entropic projection layer is therefore shared by both vertex-subset objectives. For MIS and MVC, this residual serves as a feasibility diagnostic for the projected state on the independent-set side and need not vary monotonically with the final decoded gap, which is mainly determined by the ranking of the raw logits under greedy decoding.

A.4 Why Decode-Aware Terminal Decoding Uses the Raw Logits

The decode-aware entropic projection design for vertex-subset problems follows from two simple facts: greedy MIS decoding depends only on the ordering of the raw logits, while entropic projection couples adjacent vertices through the edge constraints and therefore need not preserve that ordering. MVC inherits the same argument because its output cover is obtained by complementing the decoded independent set.

Proposition A.1. *Let* $\mathrm{GreedyMIS}(s)$ *denote greedy MIS decoding with a deterministic tie-breaking rule applied to vertex scores* $s \in \mathbb{R}^{|V|}$*. If* $\varphi : \mathbb{R} \to \mathbb{R}$ *is strictly increasing and applied pointwise, then*

$$\mathrm{GreedyMIS}(\varphi(s)) = \mathrm{GreedyMIS}(s). \tag{21}$$

Proof. Greedy MIS decoding depends only on the induced node ordering, and a strictly increasing pointwise transform preserves that ordering exactly. □

Proposition A.2. *The MIS entropic projection* $\Pi^{(\tau)}_{\mathcal{Q}_{\mathrm{MIS}}}$ *is not, in general, an order-preserving pointwise transform. In particular, there exists a path graph* G *on nodes* 1–2–3*, a temperature* $\tau = 1$*, and logits* $s = (3, 2, 1)$ *such that for the unique optimum* $z^\star = \Pi^{(1)}_{\mathcal{Q}_{\mathrm{MIS}}}(s)$ *we have* $s_2 > s_3$ *but*

$$\left[z^\star\right]_2 < \left[z^\star\right]_3. \tag{22}$$

Proof. Consider the path graph 1–2–3 with edges $(1,2)$ and $(2,3)$, temperature $\tau = 1$, and logits $s = (3,2,1)$. Define

$$z^\star := \left(\sigma(2/3), \sigma(-2/3), \sigma(2/3)\right) \tag{23}$$

and dual multipliers

$$\lambda^\star_{12} := 7/3, \qquad \lambda^\star_{23} := 1/3. \tag{24}$$

We verify the KKT conditions for Eq. (17).

Primal feasibility. Since $\sigma(a) + \sigma(-a) = 1$ for all a, we have

$$z^\star_1 + z^\star_2 = 1, \qquad z^\star_2 + z^\star_3 = 1, \tag{25}$$

so $z^\star \in \mathcal{Q}_{\mathrm{MIS}}$.

Dual feasibility. Clearly $\lambda^\star_{12}, \lambda^\star_{23} \geq 0$.

Stationarity. Using Eq. (18) with $\tau = 1$,

$$\begin{aligned} z_1^\star &= \sigma(3 - \lambda_{12}^\star) = \sigma\big(3 - \tfrac{7}{3}\big) = \sigma(2/3), \\ z_2^\star &= \sigma(2 - \lambda_{12}^\star - \lambda_{23}^\star) = \sigma\big(2 - \tfrac{7}{3} - \tfrac{1}{3}\big) = \sigma(-2/3), \\ z_3^\star &= \sigma(1 - \lambda_{23}^\star) = \sigma\big(1 - \tfrac{1}{3}\big) = \sigma(2/3), \end{aligned} \tag{26}$$

which matches Eq. (23).

Complementary slackness. Both edge constraints are active by Eq. (25), so

$$\lambda_{12}^\star(z_1^\star + z_2^\star - 1) = 0, \qquad \lambda_{23}^\star(z_2^\star + z_3^\star - 1) = 0. \tag{27}$$

The feasible set $\mathcal{Q}_{\mathrm{MIS}}$ is convex and satisfies Slater's condition. Moreover, $z^\star \in (0,1)^3$, so the box constraints are inactive at $z^\star$. Since the objective in Eq. (17) is strictly concave, the KKT conditions are sufficient and certify that $z^\star$ is the unique optimizer. Finally, $s_2 = 2 > 1 = s_3$ while

$$z_2^\star = \sigma(-2/3) < \sigma(2/3) = z_3^\star, \tag{28}$$

which proves that the MIS entropic projection can reverse score order and is therefore not an order-preserving pointwise transform. □

Theorems A.1 and A.2 jointly justify the decode-aware separation used by RAGenCO: projected control states are appropriate control states for refinement, while raw logits should be retained as the terminal decoder input for MIS and, after complementation, for MVC.

A.5 Entropic projection onto the Birkhoff polytope

RAGenCO inserts a KL projection onto $\mathcal{B}_n$ inside refinement. For $\tau > 0$, define Shannon entropy

$$H(X) := -\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij} \log X_{ij}, \tag{29}$$

with the convention $0 \log 0 = 0$, and define the entropic projection

$$\Pi_{\mathcal{B}_n}^{(\tau)}(M) := \arg\max_{X \in \mathcal{B}_n} \big\{ \langle X, M \rangle + \tau H(X) \big\}. \tag{30}$$

We also use the generalized KL divergence for nonnegative matrices,

$$D_{\mathrm{KL}}(X \| Y) := \sum_{i=1}^{n}\sum_{j=1}^{n} \Big(X_{ij} \log \frac{X_{ij}}{Y_{ij}} - X_{ij} + Y_{ij} \Big), \tag{31}$$

with the convention that $Y_{ij} > 0$ whenever $X_{ij} > 0$.

Proposition A.3. *For any $M \in \mathbb{R}^{n \times n}$ and $\tau > 0$, define the positive kernel*

$$K(M, \tau) := \exp(M/\tau). \tag{32}$$

Then the entropic projection admits the KL characterization

$$\Pi_{\mathcal{B}_n}^{(\tau)}(M) = \arg\min_{X \in \mathcal{B}_n} D_{\mathrm{KL}}\big(X \| K(M, \tau)\big), \tag{33}$$

and there exist scaling vectors $u, v \in \mathbb{R}_{++}^n$ such that the optimizer has the form

$$\Pi_{\mathcal{B}_n}^{(\tau)}(M) = \mathrm{diag}(u)\, K(M, \tau)\, \mathrm{diag}(v). \tag{34}$$

This is the standard entropic optimal-transport characterization of matrix scaling and follows directly from classical duality for entropic projections [18, 22].

A.6 Sinkhorn scaling, convergence, and feasibility after a finite number of iterations

Let

$$\mathcal{R} := \{X \in \mathbb{R}_+^{n\times n} : X\mathbf{1} = \mathbf{1}\}, \qquad \mathcal{C} := \{X \in \mathbb{R}_+^{n\times n} : X^\top \mathbf{1} = \mathbf{1}\}. \tag{35}$$

Then $\mathcal{B}_n = \mathcal{R} \cap \mathcal{C}$. We measure degree feasibility by the marginal residual defined in Eq. (9).

Theorem A.4 (Sinkhorn–Knopp)**.** *Let $K \in \mathbb{R}_{++}^{n\times n}$ and define the alternating row and column normalizations*

$$\begin{aligned} X^{(0)} &:= K, \\ X^{(2\ell+1)} &:= \Pi_{\mathcal{R}}\big(X^{(2\ell)}\big), \\ X^{(2\ell+2)} &:= \Pi_{\mathcal{C}}\big(X^{(2\ell+1)}\big), \end{aligned} \tag{36}$$

where $\Pi_{\mathcal{R}}$ and $\Pi_{\mathcal{C}}$ denote the KL projections onto $\mathcal{R}$ and $\mathcal{C}$. Then $X^{(k)} \to X^\star$, where

$$X^\star = \arg\min_{X\in\mathcal{B}_n} D_{\mathrm{KL}}(X\|K), \tag{37}$$

and $\mathrm{res}(X^{(k)}) \to 0$ as $k \to \infty$.

Theorem A.4 is the classical convergence statement for matrix scaling and alternating KL projections [41, 17, 18, 22].

Remark (strict positivity vs. hard zeros). The strict-positivity assumption $K \in \mathbb{R}_{++}^{n\times n}$ avoids additional "total support" / "full indecomposability" conditions needed when K contains hard zeros [17]. In our masked setting we therefore use a finite-penalty mask so the effective kernel remains strictly positive. In practice, for small τ and large Γ we use log-domain stabilized Sinkhorn updates to avoid numerical underflow which can otherwise create effective zeros.

Corollary A.5 (Residual control after a finite number of iterations)**.** *Let $K \in \mathbb{R}_{++}^{n\times n}$, let $X^\star$ be defined by Eq.* (37)*, and let $\{X^{(k)}\}_{k\ge 0}$ be the Sinkhorn iterates in Eq.* (36)*. Then for every $k \ge 0$,*

$$\mathrm{res}(X^{(k)}) \le 2\sqrt{2n\, D_{\mathrm{KL}}(X^\star\|X^{(k)})}. \tag{38}$$

Remark. Theorem A.5 gives a theoretical relation between KL suboptimality and marginal feasibility. In practice, $D_{\mathrm{KL}}(X^\star\|X^{(k)})$ depends on the unknown optimum $X^\star$, so we monitor $\mathrm{res}(X^{(k)})$ directly as a computable stopping criterion, as is standard in Sinkhorn implementations [17, 22].

Proof. Using $X^\star\mathbf{1} = \mathbf{1}$ and $(X^\star)^\top\mathbf{1} = \mathbf{1}$, we obtain $\mathrm{res}(X) \le 2\|X - X^\star\|_1$. Pinsker's inequality for nonnegative matrices with total mass n gives $\|X - X^\star\|_1 \le \sqrt{2n\, D_{\mathrm{KL}}(X^\star\|X)}$, which yields Eq. (38). □

A.7 Stability of projected refinement under bounded projection error

The training-time denoiser is conditioned on corrupted discrete states, while inference applies it to continuous projected states. The next proposition formalizes a modest but useful guarantee: if the denoiser is locally Lipschitz on the visited region and each projection step is sufficiently stable, then the mismatch introduced by projected refinement is recursively controlled rather than exploding without structure across T_s.

Proposition A.6. *Fix an instance I and a schedule $t_0 > \cdots > t_{T_s}$. Let the exact task-specific projected refinement operator at step k be*

$$F_k(z) := \Pi_k\big(f_\theta(I, t_k, z)\big), \tag{39}$$

where Π_k denotes the exact step-k projection operator. Suppose:

1. *the denoiser is locally $L_f^{(k)}$-Lipschitz on the relevant region,*

$$\|f_\theta(I, t_k, u) - f_\theta(I, t_k, v)\| \le L_f^{(k)}\|u - v\|;$$

2. *the exact projection is locally $L_\Pi^{(k)}$-Lipschitz,*

$$\|\Pi_k(a) - \Pi_k(b)\| \le L_\Pi^{(k)}\|a - b\|;$$

3. the implemented projection incurs additive error at most ε_k, i.e.,

$$\|\widehat{\Pi}_k(a) - \Pi_k(a)\| \le \varepsilon_k \qquad \textit{for all relevant } a.$$

Let $\{z_k\}$ be the exact projected refinement trajectory defined by $z_{k+1} = F_k(z_k)$, and let $\{\hat{z}_k\}$ be the implemented trajectory defined by

$$\hat{z}_{k+1} = \widehat{\Pi}_k\big(f_\theta(I, t_k, \hat{z}_k)\big). \tag{40}$$

Then the stepwise mismatch $\delta_k := \|\hat{z}_k - z_k\|$ satisfies

$$\delta_{k+1} \le \alpha_k\, \delta_k + \varepsilon_k, \qquad \alpha_k := L_\Pi^{(k)} L_f^{(k)}. \tag{41}$$

Consequently,

$$\delta_K \le \left(\prod_{k=0}^{K-1} \alpha_k\right) \delta_0 \;+\; \sum_{j=0}^{K-1} \left(\prod_{k=j+1}^{K-1} \alpha_k\right) \varepsilon_j. \tag{42}$$

In particular, if $\alpha_k \le \bar{\alpha} < 1$ and $\varepsilon_k \le \bar{\varepsilon}$ for all k, then

$$\delta_K \le \bar{\alpha}^K \delta_0 + \frac{1 - \bar{\alpha}^K}{1 - \bar{\alpha}} \bar{\varepsilon}. \tag{43}$$

Proof. Using the triangle inequality and the additive projection error bound,

$$\begin{aligned}
\delta_{k+1} &= \Big\| \widehat{\Pi}_k\big(f_\theta(I, t_k, \hat{z}_k)\big) - \Pi_k\big(f_\theta(I, t_k, z_k)\big) \Big\| \\
&\le \Big\| \widehat{\Pi}_k\big(f_\theta(I, t_k, \hat{z}_k)\big) - \Pi_k\big(f_\theta(I, t_k, \hat{z}_k)\big) \Big\| \\
&\quad + \big\| \Pi_k\big(f_\theta(I, t_k, \hat{z}_k)\big) - \Pi_k\big(f_\theta(I, t_k, z_k)\big) \big\| \\
&\le \varepsilon_k + L_\Pi^{(k)} \, \|f_\theta(I, t_k, \hat{z}_k) - f_\theta(I, t_k, z_k)\| \\
&\le \varepsilon_k + L_\Pi^{(k)} L_f^{(k)} \|\hat{z}_k - z_k\| \\
&= \varepsilon_k + \alpha_k \delta_k,
\end{aligned} \tag{44}$$

which proves Eq. (41). Unrolling the recursion yields Eq. (42). If $\alpha_k \le \bar{\alpha} < 1$ and $\varepsilon_k \le \bar{\varepsilon}$, then

$$\delta_K \le \bar{\alpha}^K \delta_0 + \bar{\varepsilon} \sum_{j=0}^{K-1} \bar{\alpha}^{K-1-j} = \bar{\alpha}^K \delta_0 + \frac{1 - \bar{\alpha}^K}{1 - \bar{\alpha}} \bar{\varepsilon}, \tag{45}$$

which proves Eq. (43). □

Interpretation. Theorem A.6 shows that projected inference behaves as a controlled perturbation of the ideal refinement dynamics when the denoiser is smooth on the visited region and projection error remains bounded. In practice, Sinkhorn residuals after a finite number of iterations and the MIS/MVC projection residual provide computable proxies for the additive errors ε_k.

A.8 Suppressing forbidden entries with finite masking

RAGenCO masks forbidden edges by subtracting a large penalty before entropic projection. The next lemma quantifies the induced mass on forbidden entries.

Lemma A.7. *Let $M \in \mathbb{R}^{n \times n}$ and let $F \subseteq \{1, \dots, n\} \times \{1, \dots, n\}$ be a set of forbidden indices. For $\Gamma > 0$, define masked scores $\tilde{M}_{ij} = M_{ij} - \Gamma \mathbb{1}\{(i, j) \in F\}$. Fix $\tau > 0$ and let*

$$X^\star \in \arg\max_{X \in \mathcal{B}_n} \big\{ \langle X, \tilde{M} \rangle + \tau H(X) \big\}. \tag{46}$$

Assume there exists $\bar{X} \in \mathcal{B}_n$ such that $\bar{X}_{ij} = 0$ for all $(i, j) \in F$. Let $M_{\max} = \max_{i,j} M_{ij}$ and $M_{\min} = \min_{i,j} M_{ij}$. Then the forbidden mass $\delta := \sum_{(i,j) \in F} X^\star_{ij}$ satisfies

$$\delta \le \frac{n\big(M_{\max} - M_{\min}\big) + \tau n \log n}{\Gamma}. \tag{47}$$

Proof. Define the masked objective $\mathcal{J}(X) = \langle X, \tilde{M} \rangle + \tau H(X)$ over $\mathcal{B}_n$. Since $X^\star$ maximizes $\mathcal{J}$,

$$\mathcal{J}(X^\star) \geq \mathcal{J}(\bar{X}). \tag{48}$$

By definition of $\tilde{M}$,

$$\langle X^\star, \tilde{M} \rangle = \langle X^\star, M \rangle - \Gamma\delta. \tag{49}$$

Since $\bar{X}$ places zero mass on F, $\langle \bar{X}, \tilde{M} \rangle = \langle \bar{X}, M \rangle$, hence Eq. (48) becomes

$$\langle X^\star, M \rangle - \Gamma\delta + \tau H(X^\star) \geq \langle \bar{X}, M \rangle + \tau H(\bar{X}). \tag{50}$$

Rearranging yields

$$\Gamma\delta \leq \langle X^\star, M \rangle - \langle \bar{X}, M \rangle + \tau\big(H(X^\star) - H(\bar{X})\big). \tag{51}$$

We bound the two terms. Since $X^\star \in \mathcal{B}_n$, $\sum_{i,j} X^\star_{ij} = n$, hence

$$\langle X^\star, M \rangle = \sum_{i,j} X^\star_{ij} M_{ij} \leq M_{\max} \sum_{i,j} X^\star_{ij} = n M_{\max}. \tag{52}$$

Similarly, $\sum_{i,j} \bar{X}_{ij} = n$ implies

$$\langle \bar{X}, M \rangle = \sum_{i,j} \bar{X}_{ij} M_{ij} \geq M_{\min} \sum_{i,j} \bar{X}_{ij} = n M_{\min}. \tag{53}$$

Combining yields

$$\langle X^\star, M \rangle - \langle \bar{X}, M \rangle \leq n\big(M_{\max} - M_{\min}\big). \tag{54}$$

For the entropy term, $H(X) \geq 0$ for all $X \geq 0$ and each row of $X \in \mathcal{B}_n$ has entropy at most $\log n$, hence

$$H(X) \leq n \log n \qquad \text{for all } X \in \mathcal{B}_n. \tag{55}$$

Therefore $H(X^\star) - H(\bar{X}) \leq n \log n$. Substituting this and Eq. (54) into Eq. (51) yields $\Gamma\delta \leq n(M_{\max} - M_{\min}) + \tau n \log n$, which implies Eq. (47). □

Remark. Lemma A.7 assumes feasibility of the hard-mask constraints, i.e., $\exists\, \bar{X} \in \mathcal{B}_n$ with $\bar{X}_{ij} = 0$ for all $(i, j) \in F$; if this set is empty, the statement does not apply. The bound depends on the score dynamic range $M_{\max} - M_{\min}$; if scores are unbounded, the inequality can be loose, so in practice we control this range through normalization or clipping [22].

A.9 The effect of best-of-B branching

This subsection gives the formal version of the intuition stated in Sec. 4.4 by characterizing the effect of best-of-B branching under a simple conditional i.i.d. branch model.

Proposition A.8. *Fix an anchor branch and let p_Δ denote the probability that a single random branch improves the anchor by at least $\Delta > 0$. If the B random branches are conditionally i.i.d. given the anchor, then the probability that best-of-B improves the anchor by at least Δ is*

$$1 - (1 - p_\Delta)^B. \tag{56}$$

If $\mathcal{I}_B$ denotes the achieved best-of-B improvement over the anchor, then

$$\mathbb{E}[\mathcal{I}_B] \geq \Delta\big[1 - (1 - p_\Delta)^B\big]. \tag{57}$$

The proof is immediate from conditional independence of the random branches and the identity $\mathcal{I}_B \geq \Delta\, \mathbf{1}\{\mathcal{I}_B \geq \Delta\}$.

B Additional Experimental Details

B.1 Benchmark data provenance and protocol

Datasets. All results in the main paper use the open-source datasets from ML4CO-Bench-101 [13] with their official train/validation/test splits and reference solutions. We do not construct any

Table 6: Datasets details: *Type* denotes the instance distribution, *Model* the corresponding RAGenCO checkpoint, *Training* reports the number of training instances and storage footprint, and *Testing* reports the number of test instances and reference cost $J^\star$ used for gap computation.

Problem	Type	Model	Training		Testing	
			Data Size	Storage	Data Size	Obj.
TSP	Uniform-100	Uniform-100	1,280,000	4.95GB	1280	7.756
	Uniform-500	Uniform-500	64,000	1.26GB	128	16.546
	Uniform-1K	Uniform-1K	64,000	2.53GB	128	23.118
ATSP	Uniform-50	Uniform-50	640,000	14.72GB	2500	1.5545
	Uniform-100	Uniform-100	128,000	11.78GB	2500	1.5660
	Uniform-200	Uniform-200	32,000	11.76GB	100	1.5647
MIS	RB-SMALL	RB-SMALL	64,000	3.52GB	500	20.090
	RB-LARGE	RB-LARGE	6,400	4.74GB	500	43.004
	ER-700-800	ER-700-800	12,800	7.83GB	128	44.969
	SATLIB	SATLIB	39,500	3.75GB	500	425.954
MVC	RB-SMALL	RB-SMALL	128,000	7.01GB	500	205.764
	Twitter	RB-SMALL	–	–	195	85.251
	COLLAB	RB-SMALL	–	–	1000	65.086
	RB-LARGE	RB-LARGE	6,400	4.74GB	500	968.228

additional data splits. Throughout, we follow the benchmark conventions adopted by recent generative ML4CO baselines evaluated on the same suites [8, 11].

Problem scales. The benchmark settings and scales emphasized in the main paper are TSP-100/500/1000 (Tab. 1), ATSP-50/100/200 (Tab. 2), and the vertex-subset benchmarks MIS on RB-SMALL, RB-LARGE, ER-700-800, and SATLIB together with MVC on RB-SMALL, RB-LARGE, Twitter, and COLLAB (Tab. 3). Cross-scale transfer results for TSP are reported separately in Appendix C.7.

B.2 Hardware and training configuration

Hardware. All training is performed on exactly a single NVIDIA A100 40GB GPU. All testing/inference runtimes and other reported experimental results are measured on exactly a single NVIDIA RTX 3080 GPU, except for the starred Gurobi and Meta-EGN rows in the MVC panel of Tab. 3, which are directly obtained from the original paper.

Training objective and corruption sampling. RAGenCO is trained under the Bernoulli forward corruption process in Eq. (12) with the negative log-likelihood objective in Eq. (13). As stated below Eq. (12), diffusion time t is sampled from a fixed distribution $p(t)$, implemented via a logit-normal sampler.

Hyperparameters and architecture. All per-task hyperparameters (data size, epochs, batch size, learning rate, warmup steps) and model architecture parameters (layers, attention heads, hidden dimension) are reported in Tab. 7. Across the reported experiments, we use a constant base learning rate of 2×10^{-4} with 1000 warmup steps; batch sizes and training epochs vary with problem scale as in Tab. 7. For scale generalization experiments, the "Pretrain" column of Tab. 7 specifies the initialization chain used for larger-scale models.

Training cost accounting. Tab. 8 reports the observed per-epoch and total wall-clock time and peak GPU memory for each trained scale, computed on the single A100 GPU.

B.3 Inference, feasibility, and decoding

Two inference-time compute knobs. RAGenCO exposes a serial refinement depth T_s and a parallel candidate budget B. Candidates are generated by perturbing the anchor state, aligning the perturbed state with the task relaxation, and then applying one additional refinement step. They are selected by

Table 7: Hyperparameters and datasets for training RAGenCO on different COPs.

Problem	Model	Pretrain	Training Parameters					Model Parameters		
			Data Size	Epoch	Batch Size	LR	Warmup	Layers	Heads	Hidden Dim.
TSP	Uniform-100	–	1,280,000	800	1024	2e-4	1000	16	8	256
	Uniform-500	Uniform-100	64,000	800	64	2e-4	1000	16	8	256
	Uniform-1K	Uniform-500	64,000	600	16	2e-4	1000	16	8	256
ATSP	Uniform-50	–	640,000	600	1024	2e-4	1000	16	8	256
	Uniform-100	–	128,000	400	512	2e-4	1000	16	8	256
	Uniform-200	–	32,000	400	256	2e-4	1000	16	8	256
MIS	RB-SMALL	–	64,000	400	256	2e-4	1000	16	4	128
	RB-LARGE	RB-SMALL	6,400	400	32	2e-4	1000	16	4	128
	ER-700-800	–	12,800	400	32	2e-4	1000	16	4	128
	SATLIB	–	39,500	400	32	2e-4	1000	16	4	128
MVC	RB-SMALL	–	128,000	400	256	2e-4	1000	16	4	128
	RB-LARGE	RB-SMALL	6,400	400	32	2e-4	1000	16	4	128

Table 8: Details about the training resource requirement of RAGenCO. The results are calculated on a single A100 GPU.

Problem	Model	Epochs	Batch Size	Time		GPU Mem
				1 Epoch	Total	
TSP	Uniform-100	800	1024	10m02s	133h47m	39.3 GB
	Uniform-500	800	64	12m34s	167h33m	33.1 GB
	Uniform-1K	600	16	24m15s	242h30m	30.0 GB
ATSP	Uniform-50	600	1024	4m31s	45h10m	16.4 GB
	Uniform-100	400	512	4m40s	31h07m	20.5 GB
	Uniform-200	400	256	2m12s	14h40m	28.8 GB
MIS	RB-SMALL	400	256	3m52s	25h47m	12.2 GB
	RB-LARGE	400	32	5m05s	33h53m	28.8 GB
	ER-700-800	400	32	3m53s	25h53m	14.1 GB
	SATLIB	400	32	11m44s	78h13m	20.7 GB
MVC	RB-SMALL	400	256	5m09s	34h20m	12.2 GB
	RB-LARGE	400	32	1m35s	10h33m	28.8 GB

the decoded objective via the unified anchor best-of-B operator in Algorithm 1. Perturbations are applied in logit space using the numerically stable clipping/logit map in Eq. (11); the perturbation distribution and scale are ablated in Tab. 18.

Relaxation alignment and self-loop masking for permutation-structured COPs. For TSP and ATSP, intermediate iterates are kept near the Birkhoff relaxation through Sinkhorn entropic projection (Sec. 4.2). Operationally, we forbid self-loops by masking diagonal entries and apply a fixed budget projection with temperature τ and iteration budget T_p (Eq. (4)). We report parameter sweeps in Tabs. 12 to 17. Entropic projection is closely related to KL projections onto marginal constraints [41, 17] and modern entropic optimal transport [18, 22]; for differentiable assignment relaxations in similar permutation-structured settings, see [23].

Task-specific decoding and post-processing. The terminal projected control state is converted into a feasible discrete solution by task-specific decoding. For TSP, as described in Sec. 4.2, greedy edge selection produces a cycle cover which is then merged into a single tour, followed by 2Opt local improvement. For ATSP, decoding is assignment-based: we solve a linear assignment problem via Hungarian decoding [24, 25], then patch-merge subtours and apply 2Opt, as specified in Algorithm 2. For MIS and MVC, decode-aware entropic projection is used during refinement, raw logits are retained as the terminal decoder input for greedy decoding, and MVC returns the complementary cover at the end. All post-processing steps explicitly named in a method configuration are included in the reported end-to-end runtime.

Algorithm 2 ATSP decoding (Hungarian $\rightarrow$ Patch-Merge $\rightarrow$ 2Opt).

Input: instance I with directed costs $C(I) \in \mathbb{R}_+^{n\times n}$, terminal projected matrix $z^{\mathrm{term}} \in [0,1]^{n\times n}$, clip $\varepsilon_{\mathrm{clip}}$, assignment temperature τ_{LA}, 2Opt budget $T_{2\mathrm{opt}}$
Output: feasible tour $\hat{y} = (\hat{y}_0, \ldots, \hat{y}_n = \hat{y}_0)$ and objective $J(\hat{y}; I)$
$X \leftarrow \mathrm{clip}(z^{\mathrm{term}}; \varepsilon_{\mathrm{clip}})$; $\quad X_{ii} \leftarrow 0 \; \forall i \in V$ *// enforce $x_{ii} = 0$*
for all $i, j \in V$ **do**
  if $i = j$ **then**
    $L_{ij} \leftarrow +\infty$
  else
    $L_{ij} \leftarrow -\log(X_{ij})/\tau_{\mathrm{LA}}$
  end if
end for
$\pi \leftarrow Hungarian(L)$ *// permutation $\pi : i \mapsto \pi(i)$ gives cycle cover*
define cycle-cover adjacency $x_{\mathrm{cc}} \in \{0,1\}^{n\times n}$ by $x_{\mathrm{cc},i,\pi(i)} = 1$ *// $x_{\mathrm{cc}} \in \mathcal{P}_n$*
$\mathcal{C}_{\mathrm{cyc}} \leftarrow DecomposeCycles(\pi)$ *// subtours of the cycle cover*
while $|\mathcal{C}_{\mathrm{cyc}}| > 1$ **do**
  pick two cycles $c_a = (a_0, \ldots, a_{m-1})$ and $c_b = (b_0, \ldots, b_{k-1})$ in $\mathcal{C}_{\mathrm{cyc}}$
  pick breakpoints $i \in [0, m-1]$, $j \in [0, k-1]$ minimizing patch cost increase:
    $\Delta(i,j) = C_{a_i, b_{(j+1) \bmod k}}(I) + C_{b_j, a_{(i+1) \bmod m}}(I)$
      $- C_{a_i, a_{(i+1) \bmod m}}(I) - C_{b_j, b_{(j+1) \bmod k}}(I)$
  $c_{ab} \leftarrow (a_0, \ldots, a_i, b_{(j+1) \bmod k}, \ldots, b_{k-1}, b_0, \ldots, b_j,$
    $a_{(i+1) \bmod m}, \ldots, a_{m-1})$
  $\mathcal{C}_{\mathrm{cyc}} \leftarrow (\mathcal{C}_{\mathrm{cyc}} \setminus \{c_a, c_b\}) \cup \{c_{ab}\}$ *// patch-merge cycles*
end while
let the only cycle in $\mathcal{C}_{\mathrm{cyc}}$ be $c = (v_0, \ldots, v_{n-1})$
$\hat{y} \leftarrow (v_0, \ldots, v_{n-1}, v_0)$ *// close into a Hamiltonian tour*
$\hat{y} \leftarrow 2Opt(\hat{y}, C(I), T_{2\mathrm{opt}})$ *// local improvement*
$J(\hat{y}; I) \leftarrow \sum_{t=0}^{n-1} C_{\hat{y}_t, \hat{y}_{t+1}}(I)$
return $(\hat{y}, J(\hat{y}; I))$

B.4 Metrics and runtime measurement

Objective and optimality gap. For minimization COPs (TSP/ATSP/MVC), we report the objective value $J(\hat{y})$ and the optimality gap $\mathrm{Gap}(\hat{y}) = \frac{J(\hat{y}) - J^\star}{J^\star} \cdot 100\%$, where $J^\star$ is the benchmark provided reference cost. For the maximization problem MIS, we report $\mathrm{Gap}(\hat{y}) = \frac{J^\star - J(\hat{y})}{J^\star} \cdot 100\%$ with benchmark references as in Tab. 3.

Runtime. All reported runtimes are total end-to-end wall-clock time aggregated over the full official test split, including RAGenCO inference, entropic projection, decoding, and any post-processing explicitly named in the method. Unless otherwise noted, inference uses batch size 1, and runtime is therefore not reported as a per-instance normalized metric. For the curves under equal runtime budgets in Figs. 2 and 3, we compare methods under matched total test split time budgets by sweeping each approach along its natural inference-time compute knob, while keeping the decoding/post-processing stack fixed per method configuration.

C Supplementary Experiments and Discussions

C.1 Results on Vertex-Subset Problems

The main vertex-subset benchmark table is reported in Tab. 3. This appendix complements it with additional discussion of the vertex-subset results and compact decode-aware ablations.

C.1.1 Results on MIS

For MIS, we observe that a single high-quality denoiser prediction ($T_s = 1$) followed by parallel candidate expansion is highly effective. Scaling the parallel budget B from 3 to 50 yields a monotonic

reduction in optimality gap across all datasets, from 5.52% to 3.00% on ER-700-800, confirming the effectiveness of the anchor best-of-B operator described in Sec. 4.4. This contrasts with baselines such as DiffUCO [37] or DIFUSCO [9], which often rely on longer serial chains to reach comparable performance, typically at higher runtime. The result indicates that for vertex-subset problems, expanding candidates around a strong aligned anchor is often a more effective use of inference-time compute than deepening the serial chain.

The MIS setting is benchmarked against strong task-specific references such as KaMIS [35] and Gurobi [36]. This comparison is useful because RAGenCO uses the same refinement and branching procedure across objectives while changing only the output map at decode time. The following table isolates the role of refinement with projection, refinement without projection, and direct decoding from the projected control state within the MIS branch.

Table 9: Decode-aware MIS ablation layout. The three modes isolate why RAGenCO uses a projected control state for refinement while keeping terminal decoding from raw logits on the maximization side of the vertex-subset branch.

Mode	RB-SMALL Gap (%)	RB-LARGE Gap (%)
w/ proj. during refinement + decoding from raw logits	1.61	3.09
w/o proj. during refinement + decoding from raw logits	2.24	4.99
w/ proj. during refinement + decoding from the projected control state	2.21	4.97

C.1.2 Results on MVC

MVC reuses the same refinement design on the independent-set side and the same terminal decoding rule based on raw logits as MIS, and changes only the final output map from selected set to complementary cover. It is therefore the MVC counterpart of the MIS setting. The following table mirrors Tab. 9 and reports the aligned decode-aware controls for the cover objective.

Table 10: Decode-aware MVC ablation layout. The three modes isolate the same projected state versus terminal decoding trade-off under the cover objective.

Mode	RB-SMALL Gap (%)	RB-LARGE Gap (%)
w/ proj. during refinement + complement decoding	0.17	0.27
w/o proj. during refinement + complement decoding	0.32	0.51
w/ proj. during refinement + decoding from the projected control state	0.29	0.46

C.2 Compute allocation under equal runtime budgets on TSP

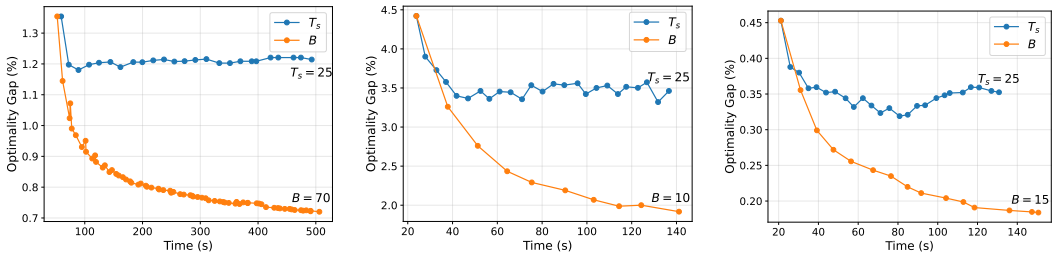
Fig. 2 characterizes the quality-runtime trade-off of RAGenCO on TSP under increasing inference-time computation. Panels Figs. 2a and 2b plot the optimality gap versus end-to-end runtime for TSP-100 and TSP-500, respectively, comparing RAGenCO against diffusion-based scaling baselines evaluated on the same benchmark suites (notably [8, 7]). Two empirical observations follow.

First, RAGenCO exhibits a consistently favorable optimality gap versus runtime profile: increasing inference compute steadily improves solution quality without requiring the long sampling chains typical of classic diffusion inference [42, 43, 9]. Related recent work also studies inference-time compute allocation in diffusion-based COP solvers, for example by restructuring the inference trajectory through adaptive expansion in COExpander [8] or by accelerating the training-to-testing pipeline in Fast T2T/T2T [7, 10].

Second, the separation between curves in Figs. 2a and 2b indicates that RAGenCO remains competitive as instance size increases, using the same decoding/post-processing primitives (Greedy + 2Opt) and the same benchmark evaluation protocol for all methods. Importantly, the curves emphasize that improvements are obtained through inference-time computation only, without modifying trained model parameters.

C.3 Parallel candidate budget is more effective than deeper serial refinement

Fig. 2c isolates RAGenCO's two inference-time compute knobs on TSP-100 under matched end-to-end wall-clock budgets: serial refinement depth T_s and parallel candidate budget B (when sweeping



(a) ATSP-100. (b) RB-SMALL (MIS). (c) RB-SMALL (MVC).

Figure 6: **Supplementary scaling under equal runtime budgets beyond TSP.** Additional plots corresponding to Fig. 2c on ATSP-100, RB-SMALL (MIS), and RB-SMALL (MVC). Each panel compares scaling the parallel candidate budget B against scaling the serial refinement depth T_s under matched end-to-end runtime budgets; when varying T_s we fix $B = 1$, and when varying B we fix $T_s = 1$.

T_s we fix $B = 1$; when varying B we fix $T_s = 1$). The plot shows diminishing returns as T_s grows, while increasing B yields a larger gap reduction per unit time via one-shot anchor best-of-B selection around aligned states. This addresses the same question of how to use additional inference-time compute, but with a more specific mechanism: unlike GenSCO [11], which uses Generation as Search Operator with disruption operators and rectified-flow training, or StruDiCO [12], which redesigns the structured diffusion process, here we keep the trained denoiser and refinement update fixed and reallocate compute only after task relaxation alignment. To show that the same comparison under equal runtime budgets is not unique to TSP-100, we include the corresponding plots for ATSP-100, RB-SMALL (MIS), and RB-SMALL (MVC) in Fig. 6.

C.4 Optimality gap versus runtime on ATSP

Fig. 3 plots optimality gap versus runtime on ATSP-100 and ATSP-200. Compared with TSP, ATSP introduces asymmetric costs and a directed permutation structure, making solution quality more sensitive to how feasibility is restored from the soft matrix. Consistent with this, Tab. 11 shows that assignment-based decoding (Hungarian $\rightarrow$ patch-merge $\rightarrow$ 2Opt; Algorithm 2) substantially improves solution quality over a purely greedy decoding strategy at comparable runtimes. This highlights a practical point when comparing diffusion-based COP solvers on ATSP: differences in feasibility restoration can dominate the marginal gains from additional denoising depth, so runtime comparisons should include decoding and repair as part of the end-to-end budget (as done in Fig. 3).

Table 11: Ablation study of ATSP decoding versus greedy decoding on ATSP.

Method	Hungarian			Greedy		
	Obj.	Gap	Time	Obj.	Gap	Time
ATSP-50 (T_s=1,B=10)	1.561	0.411%	54s	1.611	3.621%	45s
ATSP-100 (T_s=1,B=10)	1.579	0.844%	1.7m	1.648	5.246%	1.5m
ATSP-200 (T_s=1,B=10)	1.589	1.550%	13s	1.680	7.380%	12s

C.5 Relaxation alignment during training progress on TSP/ATSP

Fig. 5 plots validation gap trajectories across training steps for TSP-100 and ATSP-50, evaluating each checkpoint with vs. without Sinkhorn entropic projection at inference. **Figure protocol.** We use $T_s = 1, B = 3$ for both tasks, with Sinkhorn (τ, T_p) set to the tuned values in Tabs. 12 and 15 (TSP-100: $(0.10, 2)$; ATSP-50: $(0.05, 1)$); we decode TSP with greedy cycle-cover extraction + subtour merge repair and ATSP with assignment-based cycle-cover extraction + patch-merge repair, and we compute Gap on the final repaired tour $\hat{y}$ (after subtour merging/patching), relative to $J^\star$, with 2Opt disabled for this diagnostic (both tasks).

Across both tasks, projection yields lower validation gaps and more stable convergence, whereas omitting projection leads to worse plateaus. The direction of improvement is consistent with Tab. 4, though absolute gaps are not directly comparable because Fig. 5 uses the above diagnostic pipeline (no local search; assignment-based decoding for ATSP), while Tab. 4 reports final test gaps under the full evaluation pipeline used throughout Sec. 5.

From a modeling perspective, Sinkhorn projection is the cycle-cover instance of relaxation-aligned state control, reducing the burden on the downstream decoder to repair gross degree violations. This behavior is compatible with classical results on iterative scaling and KL projections [17, 41] and modern entropic optimal transport formulations that interpret Sinkhorn iterations as KL-regularized projections onto marginal constraints [18, 22]. Projection is not used in the forward pass of training; it is applied only at validation/inference time, where it controls the refinement trajectory and, for TSP/ATSP, also supplies the terminal decoder input. In practice, Fig. 5 indicates that relaxation alignment at evaluation time provides a tighter and more stable refinement/decoding trajectory for many training snapshots, without changing the underlying training procedure.

C.6 Sensitivity to projection and perturbation hyperparameters

Projection temperature and iteration budget. Tabs. 12 to 17 sweeps the Sinkhorn temperature τ and projection iteration budget T_p across multiple scales. These sweeps illustrate a typical trade-off: overly weak projection (high τ or very small T_p) can leave degree violations to the decoder and degrade final gaps, while overly sharp projection can reduce diversity and harm candidate exploration. Accordingly, we select (τ, T_p) per scale on the official validation split.

Table 12: Ablation studies of Sinkhorn temperature τ and iteration budget T_p on TSP-100.

T_p	$\tau = 0.05$		$\tau = 0.10$		$\tau = 0.15$		$\tau = 0.20$		$\tau = 0.25$		$\tau = 0.30$		$\tau = 0.35$	
	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓
1	7.760	0.058%	7.759	0.046%	7.761	0.063%	7.762	0.081%	7.763	0.094%	7.764	0.106%	7.765	0.123%
2	7.760	0.058%	**7.759**	**0.044%**	7.761	0.062%	7.762	0.081%	7.763	0.091%	7.764	0.108%	7.765	0.117%
3	7.760	0.058%	7.759	0.046%	7.761	0.064%	7.762	0.082%	7.763	0.097%	7.764	0.105%	7.765	0.119%
4	7.760	0.059%	7.759	0.046%	7.761	0.065%	7.762	0.081%	7.763	0.097%	7.764	0.110%	7.765	0.118%
5	7.760	0.059%	7.759	0.047%	7.761	0.064%	7.762	0.080%	7.763	0.095%	7.764	0.106%	7.765	0.121%

Table 13: Ablation studies of Sinkhorn temperature τ and iteration budget T_p on TSP-500.

T_p	$\tau = 0.05$		$\tau = 0.10$		$\tau = 0.15$		$\tau = 0.20$		$\tau = 0.25$		$\tau = 0.30$		$\tau = 0.35$	
	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓
1	16.587	0.252%	16.629	0.503%	16.642	0.582%	16.648	0.620%	16.650	0.630%	16.658	0.679%	16.660	0.693%
2	16.585	0.235%	16.626	0.482%	16.642	0.579%	16.645	0.599%	16.651	0.633%	16.657	0.672%	16.659	0.685%
3	**16.585**	**0.234%**	16.627	0.488%	16.639	0.566%	16.647	0.614%	16.655	0.659%	16.654	0.654%	16.660	0.689%
4	16.588	0.258%	16.627	0.493%	16.639	0.561%	16.650	0.630%	16.655	0.660%	16.655	0.657%	16.660	0.688%
5	16.585	0.234%	16.624	0.470%	16.639	0.566%	16.650	0.629%	16.650	0.627%	16.655	0.659%	16.660	0.688%

Table 14: Ablation studies of Sinkhorn temperature τ and iteration budget T_p on TSP-1000.

T_p	$\tau = 0.05$		$\tau = 0.10$		$\tau = 0.15$		$\tau = 0.20$		$\tau = 0.25$		$\tau = 0.30$		$\tau = 0.35$	
	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓
1	**23.321**	**0.876%**	23.361	1.055%	23.374	1.109%	23.384	1.149%	23.395	1.199%	23.404	1.235%	23.411	1.269%
2	23.328	0.910%	23.365	1.066%	23.377	1.120%	23.387	1.165%	23.392	1.186%	23.401	1.223%	23.412	1.273%
3	23.327	0.906%	23.360	1.048%	23.373	1.104%	23.385	1.156%	23.390	1.175%	23.398	1.213%	23.414	1.280%
4	23.328	0.906%	23.363	1.060%	23.376	1.114%	23.389	1.174%	23.392	1.183%	23.401	1.222%	23.408	1.254%
5	23.328	0.908%	23.360	1.045%	23.375	1.110%	23.385	1.155%	23.389	1.174%	23.400	1.222%	23.417	1.292%

Candidate perturbation distributions. Tab. 18 compares Gaussian versus Gumbel perturbations for the best-of-B candidate operator. Across both TSP and ATSP, both perturbation choices can be effective, with performance depending on the scale and inference setting. This supports treating the perturbation distribution as a modular component in the inference pipeline rather than a fixed part of the solver design.

Table 15: Ablation studies of Sinkhorn temperature τ and iteration budget T_p on ATSP-50.

T_p	$\tau = 0.05$		$\tau = 0.10$		$\tau = 0.15$		$\tau = 0.20$		$\tau = 0.25$		$\tau = 0.30$		$\tau = 0.35$	
	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓
1	**1.560**	**0.334%**	1.560	0.372%	1.561	0.390%	1.561	0.413%	1.561	0.422%	1.561	0.418%	1.561	0.424%
2	1.560	0.343%	1.560	0.378%	1.561	0.393%	1.561	0.414%	1.561	0.422%	1.561	0.425%	1.561	0.424%
3	1.560	0.358%	1.561	0.384%	1.561	0.397%	1.561	0.416%	1.561	0.424%	1.561	0.423%	1.561	0.422%
4	1.560	0.366%	1.561	0.388%	1.561	0.399%	1.561	0.415%	1.561	0.425%	1.561	0.425%	1.561	0.425%
5	1.560	0.371%	1.561	0.388%	1.561	0.401%	1.561	0.417%	1.561	0.422%	1.561	0.425%	1.561	0.426%

Table 16: Ablation studies of Sinkhorn temperature τ and iteration budget T_p on ATSP-100.

T_p	$\tau = 0.05$		$\tau = 0.10$		$\tau = 0.15$		$\tau = 0.20$		$\tau = 0.25$		$\tau = 0.30$		$\tau = 0.35$	
	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓
1	**1.576**	**0.623%**	1.577	0.699%	1.578	0.779%	1.580	0.864%	1.580	0.909%	1.580	0.920%	1.581	0.923%
2	1.576	0.644%	1.577	0.714%	1.579	0.794%	1.580	0.885%	1.580	0.919%	1.581	0.928%	1.581	0.920%
3	1.577	0.671%	1.578	0.731%	1.579	0.802%	1.580	0.890%	1.581	0.924%	1.581	0.931%	1.581	0.923%
4	1.577	0.692%	1.578	0.751%	1.579	0.825%	1.580	0.893%	1.581	0.931%	1.581	0.932%	1.580	0.916%
5	1.577	0.709%	1.578	0.770%	1.579	0.831%	1.580	0.896%	1.581	0.929%	1.581	0.931%	1.581	0.921%

Table 17: Ablation studies of Sinkhorn temperature τ and iteration budget T_p on ATSP-200.

T_p	$\tau = 0.05$		$\tau = 0.10$		$\tau = 0.15$		$\tau = 0.20$		$\tau = 0.25$		$\tau = 0.30$		$\tau = 0.35$	
	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓	OBJ.↓	GAP↓
1	1.584	1.238%	1.584	1.237%	1.585	1.306%	1.586	1.352%	1.588	1.452%	1.588	1.475%	1.588	1.463%
2	1.585	1.276%	**1.584**	**1.204%**	1.584	1.253%	1.586	1.354%	1.587	1.426%	1.588	1.457%	1.587	1.432%
3	1.584	1.222%	1.585	1.261%	1.585	1.304%	1.586	1.383%	1.587	1.433%	1.587	1.435%	1.587	1.436%
4	1.584	1.256%	1.584	1.248%	1.585	1.310%	1.587	1.398%	1.587	1.424%	1.588	1.449%	1.588	1.457%
5	1.584	1.227%	1.585	1.260%	1.585	1.293%	1.586	1.352%	1.587	1.443%	1.588	1.448%	1.587	1.441%

Table 18: Ablation study of perturbation scale σ_{cand} (Gaussian) versus β_{cand} (Gumbel) on TSP and ATSP.

Method	Gumbel		Gaussian	
	Obj.	Gap	Obj.	Gap
TSP-100 ($T_s = 5, B = 10$)	7.760	0.058%	7.761	0.064%
TSP-500 ($T_s = 5, B = 10$)	16.607	0.369%	16.612	0.402%
TSP-1000 ($T_s = 5, B = 10$)	23.371	1.093%	23.361	1.052%
ATSP-50 ($T_s = 1, B = 10$)	1.561	0.411%	1.561	0.431%
ATSP-100 ($T_s = 1, B = 10$)	1.579	0.844%	1.580	0.892%
ATSP-200 ($T_s = 1, B = 10$)	1.589	1.550%	1.591	1.691%

C.7 Generalization experiments

Finally, we report supplementary generalization studies that probe transfer beyond the in-distribution training scale.

Cross-scale generalization on synthetic instances. We follow the cross-scale generalization protocol of [10, 7] and evaluate trained TSP models under this cross-scale generalization setting without fine-tuning. Tab. 19 shows that RAGenCO transfers well when trained on larger scales and evaluated on smaller ones, while remaining competitive in the opposite direction, indicating that inference-time compute allocation and decoding can partially compensate for distribution shift while remaining within the benchmark protocol.

Table 19: Cross-scale generalization results on TSP. Entries report objective and gap.

Testing \ Training		TSP-100	TSP-500	TSP-1000
TSP50	DIFUSCO ($T_s = 50$)	5.70, 0.25%	5.83, 2.55%	5.84, 2.71%
	T2T ($T_s = 50, T_g = 30$)	**5.70, 0.11%**	5.78, 1.60%	5.75, 1.10%
	RAGenCO (T_s=15,B=40)	5.70, 0.15%	**5.71, 0.44%**	**5.73, 0.77%**
TSP100	DIFUSCO ($T_s = 50$)	7.78, 0.23%	8.03, 3.44%	8.02, 3.31%
	T2T ($T_s = 50, T_g = 30$)	7.77, 0.08%	7.95, 2.47%	7.91, 1.96%
	RAGenCO (T_s=15,B=40)	**7.76, 0.04%**	**7.85, 1.16%**	**7.87, 1.42%**
TSP500	DIFUSCO ($T_s = 50$)	17.05, 3.04%	16.78, 1.40%	16.86, 1.85%
	T2T ($T_s = 50, T_g = 30$)	**16.92, 2.25%**	16.68, 0.81%	16.72, 1.00%
	RAGenCO (T_s=15,B=40)	17.03, 2.94%	**16.59, 0.23%**	**16.71, 1.00%**
TSP1K	DIFUSCO ($T_s = 50$)	24.04, 3.98%	23.65, 2.30%	23.63, 2.21%
	T2T ($T_s = 50, T_g = 30$)	**23.85, 3.16%**	**23.47, 1.51%**	23.41, 1.23%
	RAGenCO (T_s=15,B=40)	23.87, 3.23%	23.52, 1.77%	**23.32, 0.88%**

C.8 Comparison of Three Projection Modes and Restoration Burden

To separate the effect of in-loop control from final-step post-processing, we compare three modes: no projection, projection applied only at the final step, and in-loop projection. We report pre-restoration quality, final quality, proxies for restoration burden, and runtime components in separate columns so that attribution is explicit.

Table 20: Projection attribution across three modes on TSP. Columns report pre-local-search gap, post-2Opt gap, a proxy for restoration burden, and pre/post runtime separately.

Mode	TSP-100					TSP-500					TSP-1000				
	Pre gap	Post gap	Burden	Pre time	Post time	Pre gap	Post gap	Burden	Pre time	Post time	Pre gap	Post gap	Burden	Pre time	Post time
None	4.77	0.24	2.5e3	73s	91s	8.35	0.72	1.3e5	66s	78s	9.62	1.38	5.1e5	264s	299s
Final-step Sinkhorn	0.10	0.07	234	69s	71s	3.56	0.46	2.9e4	59s	66s	8.88	1.53	5.5e4	262s	304s
In-loop Sinkhorn	0.10	0.06	220	75s	78s	3.19	0.37	1.8e4	57s	64s	7.37	1.09	4.8e4	298s	306s

Table 21: Projection attribution across three modes on ATSP. Columns report pre-restoration gap, post-2Opt gap, patch-merge burden, and runtime components separately.

Mode	ATSP-50					ATSP-100					ATSP-200				
	Pre gap	Post gap	Burden	Pre time	Post time	Pre gap	Post gap	Burden	Pre time	Post time	Pre gap	Post gap	Burden	Pre time	Post time
None	0.49	0.44	1.78	54s	56s	0.99	0.97	2.72	103s	104s	1.72	1.66	3.89	13s	13s
Final-step Sinkhorn	0.45	0.44	1.42	54s	55s	1.04	1.04	2.60	103s	103s	1.79	1.78	3.46	13s	13s
In-loop Sinkhorn	0.44	0.41	1.40	54s	54s	1.02	0.84	2.61	103s	103s	1.61	1.55	3.11	13s	13s

These controls show that in-loop projection improves the quality of intermediate proposal states, lowers downstream restoration burden, and preserves the ranking of method variants even after local search.

C.9 Trajectory diagnostics for projected refinement

The next table isolates how in-loop projection changes the refinement path before any final restoration. We evaluate a common lightweight cycle-cover extractor with repair and 2Opt disabled.

C.10 Paired consistency analysis

Point estimates alone understate how consistently RAGenCO improves over baselines evaluated under equal runtime budgets. For each test instance we compute $\Delta = \mathrm{Gap}_{\text{baseline}} - \mathrm{Gap}_{\text{RAGenCO}}$, so positive Δ favors RAGenCO. We then bootstrap the mean paired gap improvement and report per-instance win rates.

Table 22: Trajectory diagnostics on TSP-100 with $T_s = 5$ and $B = 1$. Projection primarily improves marginal feasibility and reduces downstream restoration burden, while improvements in global tour structure are smaller but still visible.

step	$\mathrm{res}(X)$ proj	$\mathrm{res}(X)$ no-proj	subtours proj	subtours no-proj	merges proj	merges no-proj
1	0.37	9342.40	49.24	49.24	48.24	48.24
2	0.31	9334.62	48.38	49.52	47.38	48.52
3	0.25	9326.99	47.48	49.60	46.48	48.60
4	0.14	9315.76	46.76	49.76	45.76	48.76
5	0.06	9308.88	45.87	49.89	44.87	48.89

Table 23: Paired consistency analysis. Positive Δ favors RAGenCO, and all reported confidence intervals remain above zero.

Setting	Baseline	95% CI for mean Δ	Win rate	Median Δ	P75 Δ
TSP-100	COExpander	0.017 [0.005, 0.029]	67.2%	0.008	0.015
TSP-100	no projection	0.207 [0.171, 0.243]	72.5%	0.012	0.037
TSP-500	COExpander	0.280 [0.204, 0.358]	75.9%	0.208	0.528
TSP-500	no projection	0.386 [0.309, 0.464]	83.6%	0.270	0.598
ATSP-50	COExpander	3.394 [3.278, 3.513]	88.6%	3.157	5.213
ATSP-50	no projection	0.026 [0.010, 0.042]	68.4%	0.022	0.022
ATSP-100	COExpander	3.259 [3.174, 3.348]	94.2%	3.075	4.617
ATSP-100	no projection	0.247 [0.219, 0.277]	60.6%	0.013	0.402

C.11 Runtime breakdown and controls with the same decoder

Because ATSP performance depends materially on restoration quality, we complement the end-to-end comparisons with a runtime decomposition and a control using the same decoder.

Table 24: ATSP-100 runtime breakdown for RAGenCO with $T_s = 1$ and $B = 10$. Sinkhorn is visible but not the runtime bottleneck.

Stage	Time (s)	% Total
Denoiser forward	33.910	40.0%
Sinkhorn	1.610	1.9%
Branching / candidate select	0.523	0.6%
Hungarian	4.945	5.8%
Patch / subtour merge	35.903	42.3%
2Opt / local search	3.032	3.6%
Total	84.856	100.0%

Table 25: ATSP comparison with a shared lightweight decoder. RAGenCO retains an advantage when the downstream restoration pipeline is matched.

Dataset	COExpander		RAGenCO	
	Gap (%)	Time (s)	Gap (%)	Time (s)
ATSP-50	0.7	246.1	0.3	219.5
ATSP-100	1.1	692.1	0.6	610.1
ATSP-200	1.9	94.9	1.2	84.3

C.12 Non-uniform TSP shift

We additionally evaluate non-uniform TSP distributions to test whether the effect on candidate quality persists away from the uniform training distribution. Following the same evaluation protocol as

the main text, we report Cluster and Gaussian shifts at multiple scales and compare RAGenCO to COExpander [8] under matched solver settings.

Table 26: Non-uniform TSP shift results.

Scale	Shift	Method	Obj.	Gap (%)	Time (s)
TSP-100	Cluster	COExpander	5.544	0.3	128.9
		RAGenCO w/o proj.	5.560	0.6	98.2
		RAGenCO w/ proj.	5.547	0.4	82.2
	Gaussian	COExpander	94.486	175.7	1321.5
		RAGenCO w/o proj.	35.351	3.9	117.9
		RAGenCO w/ proj.	35.298	3.7	119.6
TSP-200	Cluster	COExpander	7.117	3.0	43.3
		RAGenCO w/o proj.	7.048	2.0	23.5
		RAGenCO w/ proj.	7.034	1.8	23.2
	Gaussian	COExpander	209.688	333.4	531.3
		RAGenCO w/o proj.	50.122	4.1	23.1
		RAGenCO w/ proj.	50.004	3.9	21.9
TSP-500	Cluster	COExpander	11.008	2.7	79.8
		RAGenCO w/o proj.	10.997	2.6	85.6
		RAGenCO w/ proj.	10.942	2.0	92.2
	Gaussian	COExpander	84.167	8.6	105.5
		RAGenCO w/o proj.	80.061	3.3	94.5
		RAGenCO w/ proj.	80.502	3.8	99.4

The Cluster shift remains comparatively mild, while the Gaussian shift is substantially harsher for all methods. Even in the latter regime, RAGenCO remains more stable than the matched COExpander [8] control on the reported settings, which supports the interpretation that in-loop state control can improve the candidate states themselves rather than merely compensating through the downstream decoder.

C.13 Visualization of projected refinement trajectories.

Figures 7–8 visualize RAGenCO inference on TSP and MIS instances. Each figure shows five displayed states from left to right: the neutral control state, three projected refinement states, and the decoded feasible solution. For TSP, intermediate panels show high-confidence edges from the Sinkhorn-projected control matrix, and the final panel shows the decoded tour. For MIS, panels show a local induced subgraph with node color proportional to the projected independent-set probability, and the final panel marks the decoded independent set.

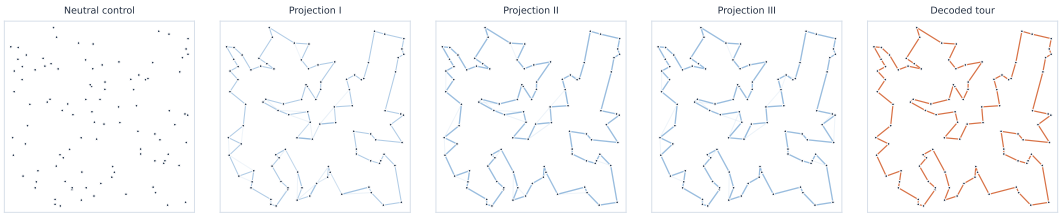


Figure 7: Projected refinement trajectory for RAGenCO on a TSP instance.

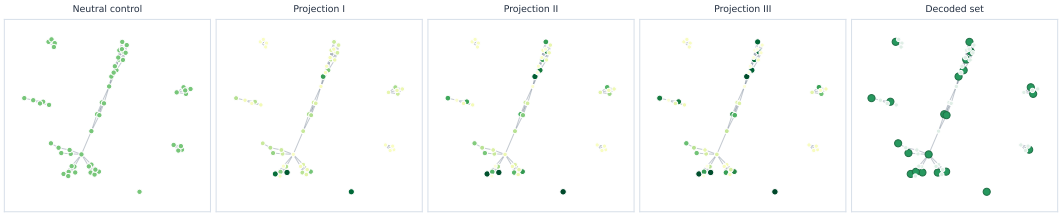


Figure 8: Projected refinement trajectory for RAGenCO on a MIS instance.

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 state the paper's main contributions, scope, and task coverage. These claims are supported by the results in Sec. 5 and the supplementary analyses in Appendices B and C.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 limitations explicitly in the Conclusion, including relaxation mismatch, decoder dependence, and the compute and memory cost of best-of-B branching.

   Guidelines:

   - The answer [N/A] 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-specific theoretical results, assumptions, and proofs are collected in Appendix A. Standard results such as Sinkhorn–Knopp are explicitly cited when used.

Guidelines:

- The answer [N/A] 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: Sec. 5 specifies the task setting, evaluation protocol, and inference configuration. Appendix B provides dataset provenance, hyperparameters, hardware, decoding details, and runtime measurement conventions.

Guidelines:

- The answer [N/A] 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: An anonymized supplemental package accompanying the submission provides the RAGenCO code together with reproduction scripts and usage instructions for the reported experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 5 summarizes the datasets, metrics, runtime protocol, and main inference settings. Appendix B reports the per-task hyperparameters, hardware and training configuration, and decoding details needed to interpret the results.

   Guidelines:

   - The answer [N/A] 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: Appendix C.10 reports bootstrap 95% confidence intervals and per-instance win rates for paired gap improvements. These statistics provide significance evidence for the main comparative claims.

   Guidelines:

- The answer [N/A] 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: Sec. 5 states that training uses a single NVIDIA A100 40GB GPU and inference runtimes are measured on a single NVIDIA RTX 3080 GPU. Appendix B reports peak memory, per-epoch and total training time, and the runtime measurement protocol.

   Guidelines:

   - The answer [N/A] 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: We reviewed the NeurIPS Code of Ethics, and the research described in the paper conforms to it.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: This paper studies a foundational inference-time method for benchmark combinatorial optimization and does not target a specific deployment setting with direct societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk generative models, scraped datasets, or other assets that would require special misuse safeguards.

Guidelines:

- The answer [N/A] 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 original papers that introduce models and datasets used in the paper are cited in Sec. 5 and Appendix B.

Guidelines:

- The answer [N/A] 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: [Yes]

Justification: An anonymized supplemental package accompanying the submission provides the RAGenCO code together with documentation and scripts for reproducing the reported experiments.

Guidelines:

- The answer [N/A] 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: [N/A]

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

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The paper does not involve human subjects research and therefore does not require IRB or equivalent review.

Guidelines:

- The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not an important, original, or non-standard component of the core methodology in this research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.