WHEN RANDOM EXPLORATION FAILS: DIRECTED EXPLORATION FOR COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse-reward reinforcement learning (RL) poses a fundamental challenge for combinatorial optimization problems (COPs): when rewarding solutions are rarely sampled, the policy receives little useful gradient signal and training can stall. A natural remedy is to perturb the learning process to encourage exploration, but it remains unclear whether simply increasing stochasticity is sufficient. We systematically study undirected perturbations, including Gaussian noise in the weight, reward, and gradient spaces and NoisyNet, and find a consistent pattern: they can accelerate learning when rewarding trajectories are already reachable, but do not reliably enable learning once the vanilla policy is trapped in a zero-reward regime. This motivates *directed* exploration. We introduce Construction-Sequence Exploration (CSE), which maintains visitation counts over consecutive decisions in the autoregressive construction sequence and rewards trajectories containing under-explored transitions. Unlike random perturbations, this count-based novelty signal remains informative when the extrinsic reward vanishes and explicitly directs the policy toward less explored solution structures. Across eight routing, scheduling, and subset-selection COPs, CSE enables learning in challenging sparse-reward settings where vanilla RL and undirected exploration often fail, while preserving performance under dense rewards.

1 INTRODUCTION

Combinatorial optimization problems (COPs) arise in many real-world applications, including logistics, manufacturing, and chip design. Since many COPs are NP-hard, exact methods often become computationally prohibitive at scale, while handcrafted heuristics require substantial domain expertise. Deep reinforcement learning (DRL) offers a promising alternative by learning constructive policies directly from problem instances (Kool et al., 2018; Kwon et al., 2020; Mazyavkina et al., 2021). Once trained, these policies can efficiently construct high-quality solutions. However, their scalability is often limited by sparse rewards. Neural solvers typically construct solutions autoregressively but receive quality feedback only after construction is complete (Bello et al., 2016); under strict feasibility or quality criteria, only a small fraction of completed solutions may receive non-zero rewards. A training batch may therefore contain few or no rewarding trajectories, yielding little or no task-directed policy-gradient signal. Consequently, the policy may remain near its initialization, converge extremely slowly, or prematurely concentrate on suboptimal solutions. This problem becomes increasingly severe as the solution space grows, hindering the application of DRL to challenging large-scale COPs.

A natural way to address this problem is to perturb the learning process so that the policy continues to explore even when the extrinsic reward provides little useful guidance (Diaz-Bone et al., 2026). However, it remains unclear what kind of perturbation is effective. Generic mechanisms, such as entropy regularization or Gaussian noise injected into the policy parameters, rewards, or gradients, can increase stochasticity and prevent the policy from becoming completely static (Burda et al., 2018; Pathak et al., 2017; Plappert et al., 2017). Nevertheless, these mechanisms are structure-agnostic, that they do not record which parts of the solution space have already been explored, nor do they indicate where the policy should explore next. Consequently, they may accelerate learning when rewarding trajectories are already reachable, but they cannot reliably escape regimes in which the vanilla policy receives no useful learning signal in a massive and highly structured combinatorial space (Ecoffet et al., 2021; Osband et al., 2016).

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

In this work, we study how perturbation affects DRL for COPs and identify a critical distinction between *undirected* and *construction-aware* exploration. We first investigate several undirected perturbation mechanisms by injecting Gaussian noise into the weight, reward, and gradient spaces. We further introduce an adaptive perturbation schedule whose magnitude is controlled online by the observed reward sparsity and policy entropy. Our results show that these perturbations can accelerate convergence in relatively tractable regimes. However, because the injected noise contains no information about previously explored solutions, it behaves essentially as an undirected random search and fails to reliably enable convergence when the original policy is completely stalled.

To address the limitation, we propose **Construction-Sequence Exploration (CSE)**, a lightweight, structure-aware perturbation mechanism for autoregressive combinatorial optimization. CSE maintains decayed visitation counts for transitions between consecutive construction decisions and assigns a larger novelty bonus to trajectories containing rarely visited transitions. The resulting intrinsic signal encourages the policy to explore under-visited construction patterns rather than perturbing it in arbitrary directions. Importantly, this signal remains informative even when the extrinsic reward is zero, allowing the policy to continue updating until rewarding regions of the solution space are discovered. As training progresses, the novelty bonus is gradually reduced so that the policy can shift from exploration to exploitation. In this way, CSE serves two complementary purposes: it accelerates convergence when the original policy can already learn, and also enables convergence in sparse-reward regimes where vanilla RL and undirected perturbations fail.

Our main contributions are summarized as follows:

- We systematically investigate perturbation-based exploration for sparse-reward combinatorial optimization, including Gaussian noise in the weight, reward, and gradient spaces, as well as NoisyNet. We show that these undirected perturbations can accelerate an already learnable policy but cannot reliably enable convergence when the vanilla policy is stalled.
- We propose CSE, which uses decayed visitation counts of consecutive construction decisions to provide a structure-aware novelty signal. CSE directs exploration toward under-visited regions of the construction space and supplies a useful learning signal even in the absence of extrinsic reward.
- We evaluate CSE on eight representative routing, scheduling, and subset-selection problems. The results show that CSE both enables convergence in challenging sparse-reward regimes and accelerates convergence in regimes where existing methods can already learn, without degrading performance under dense feedback.

2 Related Work

2.1 Deep Reinforcement Learning for Combinatorial Optimization

Driven by their ability to learn fast and scalable heuristics, neural network-based approaches for COPs have garnered significant attention and achieved widespread success in recent years. Typically, there are two main ways to use DRL to solve combinatorial optimization problems: end-to-end training and hybridization with classical heuristics. End-to-end approaches train DRL agents to output complete solutions directly, mapping problem instances to solutions without relying on external solvers. Conversely, hybrid methods synergize neural networks with classical operations research algorithms, utilizing learned models to make routing decisions within meta-heuristics or to generate informative heatmaps that prune the search space for exact solvers.

End-to-end methods primarily focus on designing advanced neural architectures (Xin et al., 2021; Kim et al., 2021; Remmerden et al., 2025), formulating effective training strategies (Corsini et al., 2024; Lei et al., 2022), optimizing solution pipelines (Bonetta et al., 2023), and expanding application domains (Zong et al., 2022). For instance, Xin et al. (2021) propose a Multi-Decoder Attention Model (MDAM) to train multiple diverse policies simultaneously. Kim et al. (2021) present a hierarchical problem-solving strategy, Learning Collaborative Policies (LCP), which trains two distinct DRL policies to alternately explore and refine candidate solutions. Similarly, Zong et al. (2022) propose a multi-agent reinforcement learning framework to solve cooperative pickup and delivery problems. Within the hybrid paradigm, agents typically learn to make decisions inside algorith-

mic loops (Rokbani et al., 2021; Ye et al., 2023). Specifically, Ye et al. (2023) propose a generic framework that leverages DRL to automate and enhance heuristic designs.

Despite their structural differences, a common thread among these methods is that they optimize a graded objective directly, *i.e.*, every candidate solution, no matter how poor, yields an informative, non-zero learning signal. Recently, Generative Flow Networks (GFlowNets) (Bengio et al., 2023) have also emerged as an alternative paradigm for sampling diverse candidates in large discrete spaces; however, they still generally rely on sufficiently dense reward landscapes to shape their flow matching losses (Zhang et al., 2023). Our work instead targets the far sparser, highly constrained regime in which the solver receives only a weak, thresholded, or gated signal that vanishes for the overwhelming majority of candidate solutions encountered during training.

2.2 Sparse Rewards in Reinforcement Learning

Sparse reward has been studied along several complementary axes. Classical techniques include reward shaping (RS), which relies on handcrafted potential functions, and Hindsight Experience Replay (HER), which relabels trajectories against achieved goals but presupposes an explicit goal space that single-instance COPs typically do not possess. Return-decomposition methods (Arjona-Medina et al., 2019; Devidze et al., 2022) redistribute the delayed terminal reward across intermediate steps, yet still require a graded terminal signal to redistribute.

A second line is goal selection and curriculum learning. Diaz-Bone et al. (2026) (DISCOVER) selects exploratory goals in the direction of the target task, achieving state-of-the-art results in long-horizon goal-conditioned control, and Badia et al. (2020) learn directed exploratory policies for hard-exploration games. These methods, however, operate over a space of goals or tasks — a space that does not exist in single-instance COPs, where the challenge is not which task to attempt but *where to search within one solution space*.

A third, and most relevant, line of work studies intrinsic motivation and count-based exploration in sparse-reward reinforcement learning. Bellemare et al. (2016) derive pseudo-counts from density models, unifying count-based and intrinsic-motivation exploration; Pathak et al. (2017) reward prediction error as an intrinsic exploration signal; and Burda et al. (2018) use random network distillation to encourage novel behaviors. Orthogonally, parameter-space noise (Fortunato et al., 2018; Plappert et al., 2017) and gradient noise (Neelakantan et al., 2015) inject undirected perturbations into the model weights or optimization gradients.

Despite these advances, standard exploration mechanisms remain insufficient for COPs. The solution space is massive, discrete, and highly non-convex: undirected noise is misdirected in such a space, while generic count-based novelty defined over raw states ignores the structured, autoregressive construction of a combinatorial solution. Our contribution is a count-based novelty signal defined directly over the edges of the construction sequence, coupled with an adaptive undirected gradient perturbation, yielding exploration that is directed and tailored to COPs.

3 Methodology

3.1 Preliminaries

Define a sparse-reward Markov Decision Process (MDP) for COPs as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the finite state and action spaces, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition kernel with $\sum_{s' \in \mathcal{S}} P(s'|s, a) = 1$ for all $s \in \mathcal{S}, a \in \mathcal{A}$, and γ is the discount factor. $R : \mathcal{S} \times \mathcal{A} \to \{0, r^*\}$ is an *extremely sparse* reward: $R(s, a) = r^* > 0$ only when a complete, globally-optimal solution has been constructed, and $R(s, a) = 0$ otherwise.

Let π_θ denote the policy network with parameters θ, and let $\theta_L \subset \theta$ denote the parameters of its final (output) layer. A trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ is a complete decision sequence, with discounted return

$$G(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t). \tag{1}$$

The objective is to maximize the expected return

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\big[G(\tau)\big], \tag{2}$$

whose policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\cdot G(\tau)\right]. \tag{3}$$

A plain fact is the under the extremely sparse reward, $G(\tau) = 0$ for the vast majority of trajectories, hence $\nabla_\theta J(\theta) \approx \mathbf{0}$ and the network receives no parameter update — the policy stalls at its initialized (near-random) behavior.

3.2 How Does the Perturbation Affect Training

Considering the plain fact that adding perturbation during training can enhance the agent's exploration (Devidze et al., 2022; Fortunato et al., 2018). The stall arises because the objective is locally flat: when every sampled trajectory has zero return, the gradient vanishes identically. Our method injects an artificial signal that keeps the parameters moving until a non-zero reward is reached. We consider two complementary injections.

Gradient-space perturbation. We inject noise Δ directly into the final-layer gradient, replacing the standard gradient with $\tilde{g}_L = g_L + \Delta_L$. In the zero-reward regime where $g_L = 0$, the update reduces to a bounded random walk:

$$\theta_L \leftarrow \theta_L - \eta\Delta_L, \quad \Delta_L = \text{Clip}(\sigma\xi, -\epsilon, \epsilon), \quad \xi \sim \mathcal{N}(0, I). \tag{4}$$

Weight-space perturbation: Similar to NoisyNets (Fortunato et al., 2018), we perturb the network parameters directly before the forward pass, such that $\tilde{\theta} = \theta + \sigma\xi$. This temporarily alters the agent's action distribution, forcing it to sample novel trajectories.

Reward-space perturbation: We augment the zero-return with a fictitious noise scalar, $\tilde{G}(\tau) = G(\tau) + \sigma\xi$. When substituted into the policy gradient equation, this assigns random non-zero advantages to sampled trajectories, artificially forcing parameter updates.

While these undirected perturbations successfully prevent the output distribution from collapsing to a fixed point, they share a critical flaw: ξ carries no structural information about the COPs. Furthermore, if the noise amplitude σ is kept constant, it will severely disrupt the delicate representations the network learns once a valid reward signal is finally found.

Construction-sequence perturbation (directed). In contrast, we can shape the reward with a signal that is large precisely where the policy has not yet explored. Augmenting the return with an intrinsic novelty bonus $\tilde{G}(\tau) = G(\tau) + \beta\, N(\tau)$ yields a non-zero gradient even when $G(\tau) = 0$,

$$\nabla_\theta \tilde{J}(\theta) = \beta\, \mathbb{E}_\tau\left[\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)\cdot N(\tau)\right] \neq \mathbf{0}, \tag{5}$$

which *directly* rewards moving into under-explored regions. This is the key distinction of our approach, formalized in Sec. 3.4.

3.3 Adaptive Undirected Perturbation Injection

To resolve the disruption caused by static noise, we propose an adaptive mechanism to dynamically control the perturbation amplitude σ. This mechanism is governed by a core principle: the injected noise should be aggressively scaled up when the reward vanishes (forcing exploration) and decayed to near-zero when the reward is informative (allowing stable exploitation).

We compute the adaptive amplitude $\sigma(S, H)$ using two scalar statistics of the current policy—the observed reward sparsity $S(\tau)$ and the policy entropy $H(\pi_\theta)$:

$$\sigma(S, H) = \sigma_0 \exp\big(-\lambda_H H(\pi_\theta) + \lambda_S S(\tau)\big), \tag{6}$$

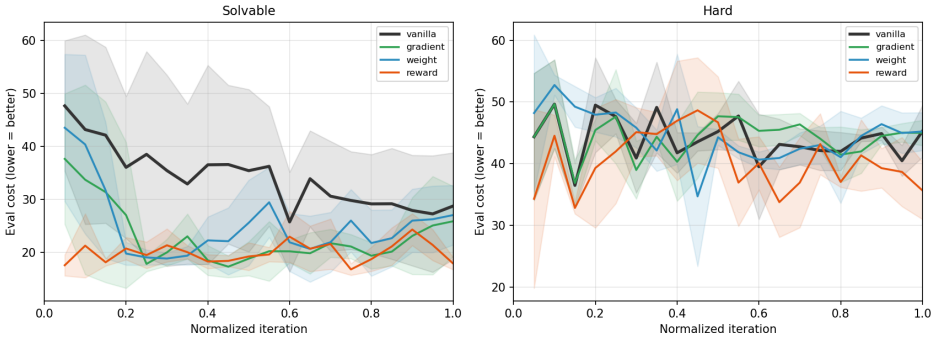


Figure 1: Convergence of undirected perturbation methods on solvable and hard instances. We train vanilla RL and three undirected perturbations (gradient, weight, and reward noise) for 500 iterations on TSP-100, with a loose temporal-sparse reward in the solvable regime (left) and a tight gated reward in the hard regime (right), and plot the evaluation cost against the normalized iteration. We clearly observe that the perturbations give at best marginal and inconsistent acceleration in the solvable regime and, like vanilla, fail to enable convergence in the hard regime, indicating that undirected noise explores blindly.

where σ_0 is the base noise scale, and $\lambda_H, \lambda_S > 0$ are temperature hyperparameters. The sparsity metric tracks the fraction of zero-reward trajectories within a recent sliding window of size N:

$$S(\tau) = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\big(G(\tau_i) \neq 0\big). \tag{7}$$

Simultaneously, the policy entropy measures the inherent stochasticity of the agent's current behavioral distribution:

$$H(\pi_\theta) = -\mathbb{E}_{a\sim\pi_\theta}\big[\pi_\theta(a|s)\log\pi_\theta(a|s)\big]. \tag{8}$$

Eq. 6 mathematically realizes our desired behavior. When the policy begins to act deterministically but still receives no reward (small H, large S), the exponent becomes heavily positive, driving σ up to artificially dislodge the policy from the stalled regime. Conversely, if the policy is already exploring broadly (large H) or is receiving dense, informative reward signals (small S), the exponential term heavily penalizes the amplitude, shrinking σ toward zero to facilitate precise gradient descent.

As is shown in Fig. 1, we empirically demonstrate this limitation. We observe that while this adaptive bounded walk efficiently manages *when* to explore, it still explores blindly. Consequently, it fails to enable convergence in highly sparse regimes, leaving the original problem unsolved. To truly navigate the massive combinatorial space, the exploration signal should be *directed*.

3.4 Construction-sequence Exploration

The undirected gradient perturbation of Sec. 3.3 prevents collapse but explores the parameter space blindly. We now introduce a *directed* exploration signal based on the novelty of the constructed sequence, which rewards the policy for visiting solutions it has not seen.

Edge-count novelty. A trajectory $\tau = (a_0, \ldots, a_T)$ is a sequence of decisions; we call each consecutive pair $e_t = (a_{t-1}, a_t)$ an *edge*. We maintain a running count table $c[i, j]$ over all edges, updated after every step with a decay factor $\gamma_c \in [0, 1)$ to keep novelty *recent*:

$$c[i,j] \;\leftarrow\; \gamma_c\, c[i,j], \qquad c[a_{t-1}, a_t] \;\leftarrow\; c[a_{t-1}, a_t] + 1 \quad \forall t. \tag{9}$$

The novelty of a sequence is the average inverse-square-root count of its edges,

$$N(\tau) = \frac{1}{T}\sum_{t=1}^{T} \frac{1}{\sqrt{c[a_{t-1}, a_t] + 1}} \;\in (0, 1], \tag{10}$$

which is maximal ($\to 1$) when every edge is unseen and decays toward 0 as the policy repeats the same solutions.

Table 1: Final evaluation costs (lower is better; mean ± standard deviation over 3 seeds). While undirected perturbations occasionally help, Construction-Sequence Exploration (CSE) uniquely enables convergence in highly sparse and constrained tasks (e.g., OP, PCTSP, SMTWTP) without degrading performance in dense regimes (e.g., MKP, RCPSP).

	vanilla	gradient	weight	reward	NoisyNet	DISCOVER	**CSE**
TSP	$44.5_{\pm 16.4}$	$47.1_{\pm 13.8}$	$38.8_{\pm 24.9}$	$\mathbf{10.5}_{\pm 0.1}$	$22.2_{\pm 9.3}$	$11.0_{\pm 0.9}$	$11.4_{\pm 12.5}$
CVRP	$19.0_{\pm 6.1}$	$18.9_{\pm 5.8}$	$23.4_{\pm 2.9}$	$13.2_{\pm 0.7}$	$24.5_{\pm 1.2}$	$16.3_{\pm 3.8}$	$\mathbf{13.4}_{\pm 1.0}$
OP	-9.7 ± 0.1	-9.7 ± 0.0	-9.6 ± 0.0	-9.8 ± 0.0	-9.1 ± 0.2	-9.8 ± 0.2	-9.9 ± 0.1
PCTSP	$24.7_{\pm 0.1}$	$24.7_{\pm 0.1}$	$24.7_{\pm 0.0}$	$23.7_{\pm 1.2}$	$25.0_{\pm 1.0}$	$23.8_{\pm 0.1}$	$\mathbf{16.4}_{\pm \mathbf{1.8}}$
SOP	$21.2_{\pm 0.1}$	$21.2_{\pm 0.1}$	$21.2_{\pm 0.1}$	$20.9_{\pm 0.6}$	$\mathbf{20.7}_{\pm \mathbf{0.2}}$	$21.1_{\pm 0.2}$	$21.2_{\pm 0.8}$
SMTWTP	62.2 ± 50.4	58.6 ± 48.1	62.7 ± 50.0	$\mathbf{9.8 \pm 0.8}$	15.7 ± 1.4	28.7 ± 14.2	10.5 ± 1.1
MKP	-11.9 ± 0.4	-11.9 ± 0.3	-11.9 ± 0.4	-11.9 ± 0.3	-11.7 ± 0.5	-11.8 ± 0.4	-11.8 ± 0.2
RCPSP	16.2 ± 0.8	16.2 ± 0.5	16.2 ± 01.2	16.2 ± 1.3	16.3 ± 2.5	16.2 ± 1.4	16.3 ± 0.1

Intrinsic reward. We augment the return of the sampled trajectory with a novelty bonus whose weight anneals over training (large early, small late):

$$\tilde{G}(\tau) = G(\tau) + \beta_t \, N(\tau), \qquad \beta_t = \beta_0 \cdot \text{anneal}(t). \tag{11}$$

Substituting into Eq. equation 3 gives the exploration gradient

$$\nabla_\theta \tilde{J}(\theta) = \mathbb{E}_\tau \left[\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \big(G(\tau) + \beta_t \, N(\tau) \big) \right], \tag{12}$$

which is non-zero even when $G(\tau) = 0$. Unlike $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the bonus $N(\tau)$ is *count-based and directed*: it is largest exactly on edges the policy has not yet used, so the gradient pushes the policy toward unexplored parts of the solution space — the mechanism that lets it escape the zero-reward flat region and eventually reach the sparse reward.

4 Experiment

4.1 Experimental Setup

Test environment We perform our method and baselines on eight classic combinatorial optimization problems: Traveling Salesman Problem (TSP), Capacitated Vehicle Routing Problem (CVRP), Orienteering Problem (OP), Prize Collecting Traveling Salesman Problem (PCTSP), Sequential Ordering Problem (SOP), Single Machine Total Weighted Tardiness Problem (SMTWTP), Resource-Constrained Project Scheduling Problem (RCPSP), and Multiple Knapsack Problem (MKP). These problems cover routing, assignment, scheduling, and subset COP types. More detailed settings for reproduction can be found in the Appendix.

Baselines & Hardware We use self-critical REINFORCE (Rennie et al., 2017) as Vanilla RL, NoisyNet (Fortunato et al., 2018), weight noise, reward noise, and gradient noise (as described in Sec. 3.3). We report the mean over three seeds, *i.e.*, 0, 1, 2. All experiments were conducted on a 128-core AMD EPYC 9754 Processor and four NVIDIA GeForce RTX 4090 GPUs.

4.2 Main Results

Table 1 summarizes the final evaluation costs across all eight combinatorial optimization problems. As expected, Vanilla RL suffers from severe performance degradation on highly constrained tasks (yielding costs of 24.7 on PCTSP, and 62.2 on SMTWTP) because the heavily gated sparse reward causes the policy gradient to vanish. Furthermore, the results reveal a critical limitation of existing exploration methods: while undirected perturbations (e.g., gradient, weight, and reward noise) occasionally accelerate learning on simpler landscapes, they fail to consistently rescue the policy in truly sparse regimes. For instance, on PCTSP, undirected noise methods perform virtually no better than Vanilla RL, demonstrating that blindly forcing parameters to change is insufficient when the combinatorial space is massive and locally flat.

Table 2: Final evaluation costs in the dense reward regime (lower is better; mean ± std). While Vanilla RL natively succeeds in these informative environments, aggressive undirected perturbations (such as NoisyNet or weight noise) introduce detrimental variance that disrupts exploitation. In contrast, CSE reliably matches or slightly improves upon the vanilla baseline, safely annealing without harming final performance.

	vanilla	gradient	weight	reward	NoisyNet	ϵ-gr.	Disc.	**CSE**
TSP	9.51±.08	9.52±.04	9.54±.06	9.67±.09	12.68±.15	11.51±.07	**9.50±.06**	9.55±.02
CVRP	11.64±.13	11.64±.14	11.47±.02	11.40±.13	15.61±1.42	12.15±.16	11.70±.27	**11.45±.04**
OP	**−10.63±.02**	−10.41±.10	−10.37±.17	−10.30±.08	−9.49±.08	−10.18±.10	−10.36±.00	−10.38±.04
PCTSP	7.09±.06	**6.98±.08**	7.16±.10	7.15±.12	9.08±.05	7.99±.15	7.03±.03	7.05±.14
SOP	13.26±.03	13.22±.03	13.20±.01	13.25±.01	14.32±.23	13.83±.02	13.35±.08	**13.17±.01**
SMTWTP	11.41±.33	**9.35±.02**	57.68±43.1	15.89±.68	18.87±6.76	12.41±3.06	28.31±7.57	10.53±.43
MKP	−11.84±.03	−11.85±.01	−11.76±.09	−11.80±.02	−11.80±.02	−11.76±.05	−11.80±.05	−11.82±.01
RCPSP	16.21±.05	16.23±.07	16.22±.06	16.24±.04	16.40±.08	16.31±.03	16.22±.04	**16.20±.05**

Table 3: Convergence speed in the dense regime, measured as training steps required to reach within 5% of the best final cost (lower is faster; "—" indicates the threshold was never reached). CSE not only maintains final performance but often accelerates convergence (e.g., on TSP and PCTSP) by actively guiding the policy through the dense landscape.

	vanilla	gradient	weight	reward	NoisyNet	ϵ-gr.	Disc.	**CSE**
TSP	162	150	162	188	—	—	138	**112**
CVRP	150	150	188	**138**	—	225	162	150
OP	—	—	—	—	—	—	—	—
PCTSP	200	188	212	225	—	—	**175**	188
SOP	75	75	**50**	75	—	250	75	75
SMTWTP	100	88	112	100	—	250	**88**	125
MKP	—	—	—	—	—	—	—	—
RCPSP	25	25	25	25	38	25	25	25

In contrast, CSE robustly enables convergence across the majority of the benchmark suite by providing a directed exploration signal. By tracking structural edge-visitation, CSE explicitly guides the policy toward under-explored combinations, achieving state-of-the-art final costs on the hardest sparse-reward tasks (e.g., reducing PCTSP cost to 16.4 and SMTWTP to 10.5). However, we also observe that CSE is not a panacea; in certain exceptionally difficult cases, even directed exploration struggles to achieve stable convergence. For instance, CSE exhibits high variance on the TSP (11.4 ± 12.5) indicating failure to converge on some seeds, and on the OP and SOP, all methods including CSE fail to meaningfully improve upon the vanilla baseline. Nevertheless, where convergence is possible, these findings empirically validate that directed edge-level novelty provides a vastly superior and computationally cheap mechanism compared to undirected random walks.

4.3 Performance under Dense Situations

While the primary contribution of CSE is enabling convergence in sparse-reward regimes, a practical exploration strategy must not degrade performance when standard, dense feedback is available. To evaluate this, we test our method against baselines under standard dense-reward formulations, where Vanilla RL is already known to learn efficiently.

Table 2 presents the final evaluation costs under these dense configurations. As expected, Vanilla RL achieves strong, stable performance across the board. Notably, several undirected exploration methods actively harm the policy in this regime. For example, NoisyNet and ϵ-greedy consistently yield worse final costs (e.g., scoring 12.68 and 11.51 on TSP compared to Vanilla's 9.51), and weight noise triggers a massive performance collapse on SMTWTP (57.68 ± 43.1). Because these undirected methods blindly inject stochasticity regardless of the learning progress, they introduce detrimental variance that prevents the policy from stably exploiting the dense reward signal. Conversely, CSE reliably matches or marginally improves upon Vanilla RL (e.g., yielding 13.17 on SOP and 10.53 on SMTWTP). Because the CSE novelty bonus is count-based, it naturally diminishes as the policy hones in on high-quality construction edges, ensuring safe exploitation.

Furthermore, directed exploration yields benefits beyond mere stability. Table 3 reports the convergence speed, defined as the number of training steps required to reach within 5% of the best final

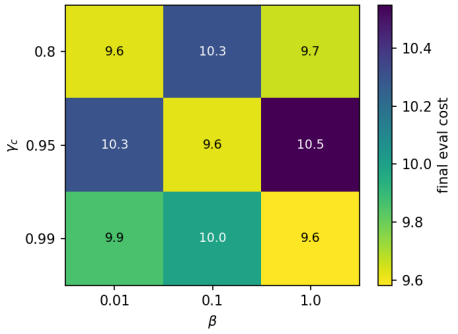


Figure 2: Sensitivity of CSE to the intrinsic-reward scale β and the count-decay factor γ_c on TSP-100 (gated, 250 steps). Each cell is the final greedy cost; vanilla collapses to 43.2. The surface is flat and uniformly near-optimal, showing the headline setting $(\beta, \gamma_c) = (0.1, 0.95)$ is not knife-edge.

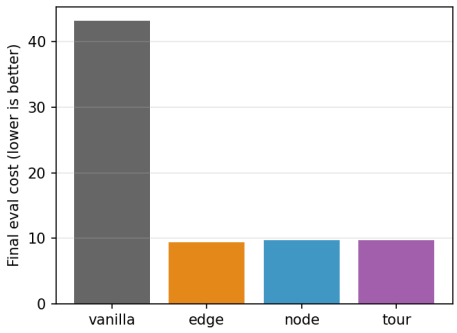


Figure 3: Feature ablation of the CSE components. Tracking structural edge transitions drastically outperforms naive node-level visitation counts, and annealing the novelty bonus (β) is crucial to prevent destabilization during late-stage exploitation.

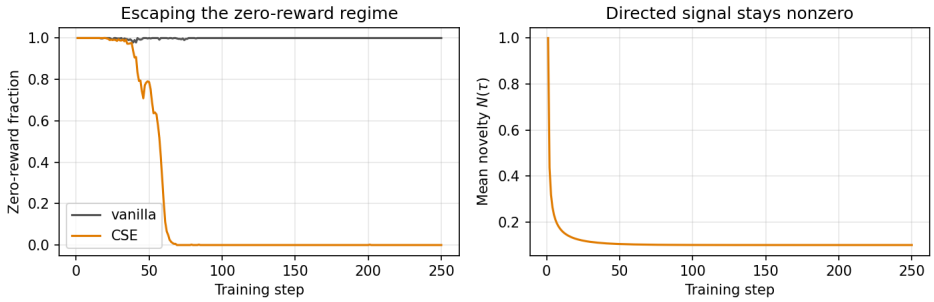


Figure 4: Cumulative number of distinct valid tours sampled during training. By explicitly rewarding novel edge transitions, CSE forces the policy to systematically traverse uncharted regions of the combinatorial space, discovering orders of magnitude more unique structures than undirected noise.

cost. While undirected methods like NoisyNet often fail to ever reach this threshold, CSE frequently accelerates learning. For instance, on the TSP, CSE reaches the target in just 112 steps compared to Vanilla's 162; similar accelerations are observed on PCTSP. These results demonstrate that CSE is not merely a specialized tool for sparse environments, but a robust, safe augmentation that can actively accelerate learning in dense scenarios without risking late-stage performance degradation.

4.4 Ablation Study

We ablate the CSE signal on the hard TSP-100 gated regime where vanilla RL collapses (final cost 43.2, vs. the nearest-neighbor reference of ≈ 9.3). All ablations use a single training budget of 2500 steps and greedy evaluation.

Sensitity of β and decay The intrinsic reward scale β_0 and the count decay factor γ_c dictate the strength and memory of the exploration signal. Fig. 2 reports the final cost of CSE across a 3×3 grid of the intrinsic-reward scale $\beta \in \{0.01, 0.1, 1.0\}$ and the count-decay factor $\gamma_c \in \{0.8, 0.95, 0.99\}$. Every cell converges to 9.6–10.6, uniformly far below the vanilla baseline (43.2), and no cell collapses during the entire training process. The headline setting $(\beta = 0.1, \gamma_c = 0.95)$ is therefore not a knife-edge tuning: the method is insensitive to both hyperparameters over a wide range, and the decay factor γ_c has no qualitative effect.

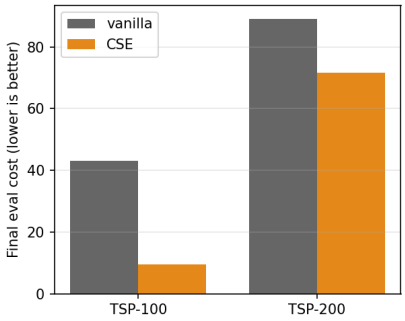


Figure 5: Scalability of exploration mechanisms across varying problem sizes ($N = 100, 200$). As the combinatorial space grows factorially, undirected perturbations degenerate into blind random walks, whereas CSE's localized edge-based novelty continues to successfully guide convergence.

Feature ablation To test whether the *edge* is the right granularity for the count table, we compare it against counting *node* visitations and against hashing the *full tour* as a single unit Fig. 3). All three count-based signals enable convergence where vanilla collapses (43.2), but the edge-level signal is the best (9.45), followed by node (9.68) and full-tour (9.75). The monotone ordering is consistent with our claim: edges are the finest unit that still captures *structure* (which transitions were taken), while a node count discards adjacency and a tour hash is too coarse to distinguish nearby solutions, giving an almost undirected bonus.

Impact of the number of distinct tours Fig. 3 explains how CSE escapes the flat region. The left panel plots the fraction of training episodes that receive zero reward: vanilla is pinned at 1.0 for the entire run (its gradient vanishes identically, so it never improves), whereas CSE drives the zero-reward fraction to 0 within the first $\sim$ 100 steps, after which the extrinsic reward takes over. The right panel shows the mean novelty signal $N(\tau)$, which stays bounded away from zero throughout training, supplying exactly the non-zero exploration gradient of Eq. equation 12 that vanilla lacks.

Performance on different scales Fig. 5 verifies that the benefit of CSE, and its $\mathcal{O}(N^2)$ count table, do not deteriorate at larger scales. At TSP-100, the CSE advantage is dramatic (9.62 vs. 43.21). When scaling up to TSP-200, the edge-count table grows to only $200^2 = 4 \times 10^4$ entries (requiring a negligible 0.16 MB of memory). CSE continues to successfully learn and improve (71.69), whereas vanilla RL remains completely trapped at its random initialization (89.07).

5 Conclusion

In this work, we study sparse-reward reinforcement learning for combinatorial optimization through the lens of exploration. We show that undirected perturbations can accelerate learning when rewarding trajectories are reachable, but do not reliably recover policies stalled in highly sparse regimes. We therefore introduce Construction-Sequence Exploration (CSE), which rewards under-explored decision edges and provides a directed learning signal even when extrinsic reward vanishes. Across eight COPs, CSE enables learning in challenging sparse-reward settings while preserving performance under dense feedback. These results highlight the importance of directing exploration based on what has already been visited, and motivate future extensions to dynamic environments, search-based inference, and cross-instance exploration.

Potential limitation and future directions. Several directions remain open. Extending edge-level novelty to dynamic environments with changing state or action spaces would require more flexible memory mechanisms. CSE could also be incorporated into search-based inference procedures such as MCTS or beam search, where novelty may help allocate search toward less explored constructions. Finally, sharing or transferring visitation statistics across problem instances may provide a path toward more efficient exploration on heterogeneous problem distributions. Together, these directions point to construction-aware exploration as a simple but promising principle for learning under extremely sparse combinatorial feedback.

AI USE STATEMENT

In this work, we used generative AI tools for drafting and refining sections of the manuscript. We have not used generative AI tools for generating the core research idea, deriving the mathematical formulations, or writing the primary experimental code, and generating artificial experimental datasets is not applicable to this work. Additionally, we used generative AI tools for grammar checking, stylistic text polishing, and providing structural suggestions for the ablation study. We have reviewed all AI-assisted work. Specifically, all AI-polished text was meticulously reviewed by the authors to ensure scientific accuracy and the preservation of our original intent. Furthermore, while the LLM provided suggestions for potential ablation setups, these suggestions were manually evaluated for scientific relevance, and all corresponding experiments were implemented, executed, and analyzed exclusively by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

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

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Giovanni Bonetta, Davide Zago, Rossella Cancelliere, and Andrea Grosso. Job shop scheduling via deep reinforcement learning: a sequence to sequence approach. In *International Conference on Learning and Intelligent Optimization*, pp. 475–490. Springer, 2023.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Andrea Corsini, Angelo Porrello, Simone Calderara, and Mauro Dell'Amico. Self-labeling the job shop scheduling problem. *Advances in Neural Information Processing Systems*, 37:105528–105551, 2024.

Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. Exploration-guided reward shaping for reinforcement learning under sparse rewards. *Advances in neural information processing systems*, 35:5829–5842, 2022.

Leander Diaz-Bone, Marco Bagatella, Jonas Hübotter, and Andreas Krause. Discover: Automated curricula for sparse-reward reinforcement learning. *Advances in Neural Information Processing Systems*, 38:21863–21896, 2026.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration (2019). *arXiv preprint arXiv:1706.10295*, 2018.

Minsu Kim, Jinkyoo Park, et al. Learning collaborative policies to solve np-hard routing problems. *Advances in Neural Information Processing Systems*, 34:10418–10430, 2021.

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

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.

Kun Lei, Peng Guo, Wenchao Zhao, Yi Wang, Linmao Qian, Xiangyin Meng, and Liansheng Tang. A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem. *Expert Systems with Applications*, 205:117796, 2022.

Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Jesse van Remmerden, Zaharah Bukhsh, and Yingqian Zhang. Offline reinforcement learning for learning to dispatch for job shop scheduling. *Machine Learning*, 114(8):191, 2025.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1179–1195. IEEE, 2017.

Nizar Rokbani, Raghvendra Kumar, Ajith Abraham, Adel M Alimi, Hoang Viet Long, Ishaani Priyadarshini, and Le Hoang Son. Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. *Soft Computing*, 25(5):3775–3794, 2021.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 12042–12049, 2021.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. Deepaco: Neural-enhanced ant systems for combinatorial optimization. *Advances in neural information processing systems*, 36: 43706–43728, 2023.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with gflownets. *Advances in neural information processing systems*, 36:11952–11969, 2023.

Zefang Zong, Meng Zheng, Yong Li, and Depeng Jin. Mapdp: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 9980–9988, 2022.

Table 4: Eight combinatorial optimization problems: node features, action space, and objective.

Problem	Category	Features	Action space	Objective
TSP	routing	(x, y)	permutation of N cities	closed-tour length
CVRP	routing	(x, y, depot, d)	node sequence	tour length (capacity-constrained)
OP	routing	(x, y, depot, p)	subset tour	collected prize
PCTSP	routing	(x, y, p, π)	subset tour	tour length + skipped penalties
SOP	routing	(x, y, ℓ)	precedence-respecting permutation	closed-tour length
SMTWTP	scheduling	(p, d, w)	job permutation	total weighted tardiness
RCPSP	scheduling	(p, r, ℓ)	activity priority order	makespan
MKP	subset	(w, v)	item packing order	total packed value

A Environment Details

All problems are formulated as autoregressive *construction* tasks: the policy picks one node/activity at a time, and a feasibility mask prevents infeasible choices. Table 4 summarizes the eight COPs.

A.1 Shared Setup

All eight problems use the attention model (Kool et al., 2018) with embedding dimension 128, 8 heads, 3 encoder layers, and feed-forward dimension 512, trained by self-critical RL (Rennie et al., 2017) (Adam, learning rate 3×10^{-4}, gradient clip 1.0, batch size 256). All problems implement four reward modes: **dense** (the raw objective, negated for minimization problems), **threshold** (binary $0/1$ vs a per-instance bar), **gated** (graded margin above/below the bar), and **sparse** (graded margin delivered on a random feedback subset). The bar is $\lambda\times$ a per-instance *reference heuristic* $\hat{c}$, reported per problem below.

A.2 TSP

N points sampled uniformly in $[0, 1]^2$; the tour is a permutation of all points and the objective is the closed-tour length. Reference heuristic $\hat{c}$ is the nearest-neighbor (NN) tour length. Bar multiplier $\lambda = 3.0$.

A.3 CVRP

One depot at $(0.5, 0.5)$ plus $N-1$ customers in $[0, 1]^2$ with demands $d \in \{1, \ldots, 9\}$; the depot has demand 0. A single vehicle of capacity 40 serves all customers, returning to the depot when remaining capacity cannot fit any unserved demand. The mask forbids revisited customers and demand-overflowing customers, and forces the depot only when no customer is feasible. Objective is the total tour length. Reference $\hat{c}$: NN length. Bar multiplier $\lambda = 3.0$.

A.4 OP

One depot at $(0.5, 0.5)$ plus $N-1$ prize nodes with prize $p \sim U(0, 1)$ (the depot has prize 0). The agent collects prizes along a path that starts and ends at the depot, within a length budget of $1.5\sqrt{2}$; returning to the depot ends the episode. The mask forbids revisited nodes and nodes unreachable within the remaining budget (the depot stays reachable). Objective is the total collected prize (reported as negative cost). Reference $\hat{c}$: prize of a greedy nearest-unvisited route under the same budget. Bar multiplier $\lambda = 0.5$.

A.5 PCTSP

One depot at $(0.5, 0.5)$ plus $N-1$ customers in $[0, 1]^2$, each with a prize $p \sim U(0, 1)$ and a penalty $\pi \sim U(0, 1)$ (the depot has both 0). The agent builds a closed tour that may *skip* customers; each skipped customer incurs its penalty. Objective is the tour length plus the sum of penalties of unvisited customers. Reference $\hat{c}$: NN length over all nodes. Bar multiplier $\lambda = 1.2$.

A.6 SOP

N points in $[0,1]^2$; each node has an integer level $\ell \sim U\{0,\dots,4\}$. Precedence constraint: node i must precede node j whenever $\ell_i < \ell_j$. The mask exposes only unvisited nodes of the minimum unvisited level. Objective is the closed-tour length. Reference $\hat{c}$: NN length (ignoring precedence). Bar multiplier $\lambda = 1.5$.

A.7 SMTWTP

N jobs, each with processing time $p \sim U(0.01, 1)$, due date $d \sim U(0,\, 0.5N)$, and weight $w \sim U(0,1)$. The agent orders the jobs on one machine; job j has completion time C_j equal to the cumulative processing time up to and including j, and tardiness $T_j = \max(0,\, C_j - d_j)$. Objective is the total weighted tardiness $\sum_j w_j T_j$. Reference $\hat{c}$: the weighted-shortest-processing-time (WSPT) heuristic. Bar multiplier $\lambda = 0.8$.

A.8 RCPSP

N activities, each with integer duration $p \sim U\{1,2\}$, resource demand $r \sim U\{1,2\}$, and level $\ell \sim U\{0,\dots,3\}$. A single renewable resource of capacity 3 must not be exceeded by concurrently running activities; precedence is level-based as in SOP. The agent learns a priority order, and a deterministic list scheduler assigns each activity the earliest start time respecting both precedence and resource capacity. Objective is the makespan. Reference $\hat{c}$: the shortest-processing-time-first priority under the same list scheduler. Bar multiplier $\lambda = 1.1$.

A.9 MKP

N items with weight $w \sim U(0.01, 1)$ and value $v \sim U(10^{-3}, 1)$, and $M = 3$ bins each of capacity $0.5 \cdot \sum_i w_i / 3$. The agent learns a packing order (a permutation), and a deterministic greedy decoder places each item into the first bin with enough remaining capacity, or drops it. Objective is the total packed value (reported as negative cost). Reference $\hat{c}$: the value/weight-ratio order under the same greedy assignment. Bar multiplier $\lambda = 0.95$.