BOLT: BREADTH-ORIENTED LEARNING VIA TRAINING-TIME EXPLORATION FOR UNSUPERVISED NEURAL COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised neural combinatorial optimization (NCO) avoids the need for costly high-quality solution labels, but it also removes the supervisory signals that facilitate learning. Existing unsupervised solvers often compensate for this missing guidance through iterative exploration at inference, requiring repeated neural network evaluations. Direct inference avoids such iterative computation, but also removes the opportunity for inference-time exploration. We introduce BOLT, a breadth-oriented learning framework that moves solution exploration from iterative inference to training time while retaining direct inference at test time. To enable this, we introduce self-improving population guidance to progressively construct improved targets relative to the current generator, and decoder basin matching to transfer these improvements back to the generator. Experiments on three graph optimization problems and two graph scales show that BOLT consistently improves solution quality over prior direct unsupervised NCO methods while retaining efficient inference.

1 INTRODUCTION

Combinatorial optimization (CO) underlies a wide range of problems in science and engineering, including scheduling, routing, resource allocation, and graph optimization (Lei et al., 2026; Wang et al., 2025a; Drakulic et al., 2025; Xu et al., 2026; Weissman et al., 2025; Garmendia et al., 2026). The prevalence and computational difficulty of these problems have motivated neural combinatorial optimization (NCO), which leverages neural networks to learn effective solution strategies (Gu et al., 2025; Reijnen et al., 2025; Soler & Raidl, 2025; Zhang et al., 2025; Wang et al., 2024; Jin et al., 2024; Feng et al., 2026a; Pan et al., 2025b). Many NCO methods rely on high-quality or near-optimal solution labels, which can be costly to obtain. This limitation has motivated growing interest in unsupervised NCO, where models learn directly from problem structures and computable objectives without external solution supervision (Hong et al., 2024; Sanokowski et al., 2024; 2025; Feng & Yang, 2025; Feng et al., 2026b).

Unsupervised NCO removes the dependence on high-quality solution labels, but it also removes the explicit guidance about where high-quality solutions lie (Bu et al., 2024; Min et al., 2023). The model therefore needs to obtain such guidance from the optimization process itself. Recent unsupervised solvers (Sanokowski et al., 2024; 2025; Feng & Yang, 2025; Feng et al., 2026b) address this challenge through iterative exploration and refinement, progressively improving candidate solutions during inference. These trajectories provide a mechanism for navigating the highly complex solution space and turning objective evaluations into useful optimization guidance. While effective, this strategy makes strong solution quality reliant on iterative generation trajectories and repeated neural network evaluations at inference time.

A complementary direction is direct inference, where a complete solution representation is produced in a single neural evaluation followed by decoding (Karalias & Loukas, 2020). By avoiding iterative neural computation, this paradigm provides an efficient inference path for generating complete solutions. Supervised NCO has shown that direct inference can generate strong solutions when high-quality solution targets are available (Li et al., 2024; Wang et al., 2025b). Achieving the same goal without solution supervision, however, exposes a bootstrapping dilemma. Specifically, a strong

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.

direct generator requires informative training targets, yet such targets are difficult to discover before the generator becomes sufficiently capable. Iterative unsupervised solvers alleviate this difficulty by repeatedly exploring and refining candidates during inference, using additional computation to discover promising regions of the solution space. Direct inference, however, removes this opportunity for inference-time solution exploration. This leaves unsupervised direct inference with a challenge: acquiring strong solution-exploration capability without high-quality external targets or iterative inference-time search.

We address this bootstrapping dilemma with BOLT, a breadth-oriented learning framework that moves solution exploration from iterative inference to training time. Rather than improving each candidate through a sequential inference process, BOLT explores multiple outputs of the current generator in parallel during training and progressively incorporates the discovered improvements into the direct generator. In this way, extensive solution exploration is carried out during training while keeping inference direct.

Realizing this idea is challenging since the generator is initially weak, and solutions discovered far from its current distribution may provide ineffective learning targets. We therefore introduce self-improving population guidance (SIPG). Starting from the current generator distribution, SIPG expands a population of candidate solutions and constructs an improved target distribution that favors higher-quality solutions while remaining close to what the generator can currently represent. The target is therefore improved relative to the current generator, keeping each update within a learnable range. Repeating this process allows target quality and generator capability to improve together, turning training-time breadth exploration into progressively stronger guidance for direct generation.

Discovering better solutions alone is not sufficient, since the decoder maps many generator outputs to the same solution. A high-quality solution therefore corresponds to a basin of valid generator outputs rather than a unique training target. We introduce decoder basin matching (DBM), which trains the generator toward regions of the output space that decode to the desired solutions, rather than forcing each solution to correspond to a single target representation. Together, SIPG and DBM close the self-improving loop: SIPG discovers progressively better target distributions, DBM transfers these improvements to the direct generator, and the improved generator provides a stronger starting distribution for subsequent solution exploration. Across three graph optimization problems, BOLT substantially improves solution quality over prior direct unsupervised NCO methods.

2 Related Work

Supervised direct neural CO solvers. Direct solution generation has been extensively explored under supervised learning. Early approaches learn parallel decision scores, edge affinities, or structured solution representations from labeled solutions (Li et al., 2018; Wang et al., 2020; Fu et al., 2021; Wang et al., 2021; Hudson et al., 2022), while later work distills sequential policies or accelerates iterative generative models into direct solvers (Woo et al., 2022; Xiao et al., 2024; Li et al., 2024; Wang et al., 2025a; Li et al., 2026a). These methods benefit from high-quality solution targets that provide explicit guidance for learning direct mappings, but obtaining such targets typically requires expensive optimization. Their supervision-dependent training and distillation strategies therefore do not directly transfer to unsupervised NCO, where high-quality solution labels are unavailable and useful training targets need to be discovered during learning.

Unsupervised neural combinatorial optimization. Unsupervised NCO learns directly from problem objectives and constraints without requiring high-quality solution labels (Schuetz et al., 2022; Wang et al., 2022a; Wang & Li, 2023; Ichikawa, 2024; Wang et al., 2022b; Karalias et al., 2026). One line of work strengthens solution exploration through iterative generation, refinement, or search (Sanokowski et al., 2023; Zhang et al., 2023; Sanokowski et al., 2024; 2025; Feng & Yang, 2025; Acikalin et al., 2025; Feng et al., 2026b; Heydaribeni et al., 2024; Huang et al., 2026). These methods devote additional inference computation to exploring the solution space and improving candidate solutions. In parallel, another line pursues direct inference, where a neural network predicts a complete solution representation in a single evaluation followed by decoding. Starting from EGN (Karalias & Loukas, 2020), this paradigm has continued to develop through richer objectives and architectures, as well as meta-learning, transferable, and multi-problem formulations (Karalias et al., 2022; Min et al., 2022; Wang & Li, 2023; Wenkel et al., 2024; Cantürk et al., 2026; Nguyen

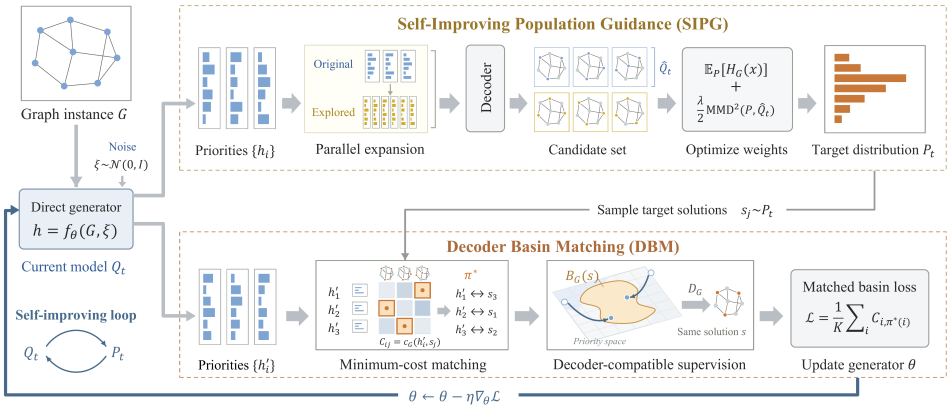

Figure 1: Overview of BOLT training. Starting from the current generator distribution Q_t, SIPG performs breadth-oriented population expansion and model-relative distributional optimization to construct an improved target distribution P_t. DBM then transfers this improvement back to the direct generator through decoder-compatible matching and supervision, yielding the updated distribution Q_{t+1}. Repeating these two stages forms a self-improving loop in which solution exploration is progressively incorporated into the generator.

& Safro, 2026). BOLT follows the direct-inference paradigm while moving stronger solution exploration into training. It performs breadth-oriented exploration during training and progressively transfers the discovered improvements into the direct generator.

Self-improvement provides one way to realize unsupervised NCO by deriving training signals from solutions generated or improved by the solver itself (Pirnay et al., 2023; Luo et al., 2023; Corsini et al., 2024; Chen et al., 2024; Luo et al., 2025; Xiao et al., 2025; Choi et al., 2025; Luttmann & Xie, 2026; Kim et al., 2024; Liao et al., 2025; Pan et al., 2025a; Li et al., 2026b). A common pattern in these methods is to treat improved solutions, partial solutions, or trajectories as stronger supervision for subsequent learning. BOLT differs in the object being optimized and transferred. Rather than selecting newly discovered solutions as replacement targets, BOLT explicitly constructs a target distribution relative to the current generator, jointly accounting for solution quality and deviation from the current model distribution. This distinction is particularly important for direct generation: an explored solution can have high quality while still providing a distant target for the current generator. By optimizing a model-relative target distribution instead of repeatedly replacing solution-level labels, BOLT turns self-improvement into distributional updates that the direct generator can progressively absorb.

3 BOLT: Proposed Approach

BOLT addresses the bootstrapping dilemma through two coupled stages: target improvement and target transfer. Self-improving population guidance (SIPG) improves the current solution distribution through training-time breadth exploration and distributional optimization, while decoder basin matching (DBM) transfers the resulting improvement back to the direct generator. Repeating these two stages forms a self-improving training loop in which solution exploration is progressively absorbed into the generator. Figure 1 illustrates this training procedure. At inference, the learned generator directly produces a complete solution representation followed by decoding, without the training-time exploration and target optimization.

3.1 Problem Formulation

Following prior unsupervised NCO solvers (Sanokowski et al., 2024), we formulate graph CO as binary energy minimization. Given a graph instance $G = (V, E)$ with $n = |V|$, a solution is represented by a binary configuration $\mathbf{x} \in \{0, 1\}^n$, and the optimization problem is written as

$$\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \{0,1\}^n} H_G(\mathbf{x}), \tag{1}$$

where $H_G(\mathbf{x})$ assigns an energy to each configuration. The detailed energy formulations are provided in Appendix A.1. Lower energy corresponds to better solutions, and global minima correspond to optimal solutions. For the constrained problems, constraint violations are incorporated into H_G through penalty terms chosen such that global minimizers correspond to feasible optima. This formulation facilitates unsupervised learning since H_G provides a direct measure of solution quality without requiring a reference solution.

Let ω denote a distribution over problem instances. The goal of unsupervised NCO is to learn a conditional solution distribution $Q_\theta(\cdot \mid G)$ without access to reference solutions, such that probability mass is concentrated on low-energy configurations. This goal can be expressed by minimizing the expected energy as follows:

$$\min_\theta \mathcal{J}(\theta) = \mathbb{E}_{G\sim\omega}\mathbb{E}_{\mathbf{x}\sim Q_\theta(\cdot|G)}\left[H_G(\mathbf{x})\right]. \tag{2}$$

In this work, $Q_\theta(\cdot \mid G)$ is induced by a stochastic direct generator. Given $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$, the generator outputs a priority vector $\mathbf{h} = f_\theta(G, \boldsymbol{\xi}) \in \mathbb{R}^n$, which is mapped by a problem-specific decoder D_G to a discrete solution $\mathbf{x} = D_G(\mathbf{h}) \in \{0,1\}^n$. Different latent vectors can produce different decoded solutions, thereby defining the conditional distribution $Q_\theta(\cdot \mid G)$. Each candidate requires only one evaluation of f_θ followed by decoding.

In this formulation, the conditional distribution $Q_\theta(\cdot \mid G)$ is implicit and accessed through samples without explicit evaluation of $Q_\theta(\mathbf{x} \mid G)$. This motivates the sample-based distributional learning mechanism introduced next.

3.2 Self-Improving Population Guidance

The first stage of the self-improving loop is target improvement. At training stage t, the current generator provides a solution distribution $Q_t(\cdot \mid G)$, but no external high-quality target is available to guide the next update. SIPG constructs this guidance from the current generator itself. It first expands the solutions available around the current distribution through parallel exploration, and then constructs an improved target distribution P_t that favors better solutions while remaining close to the current generator distribution. Each stage therefore asks the generator to absorb only an improvement over what it already represents, rather than learn a distant target from a weak starting point. Once this improvement is transferred to the generator, the updated solution distribution provides a stronger reference for the next stage. Repeating this process allows target quality and generator capability to advance together, turning the original bootstrapping dilemma into a sequence of model-relative improvements.

Model-relative target improvement. At training stage t, let $Q_t(\cdot \mid G)$ denote the solution distribution induced by the current generator for instance G. We draw K solutions and form the empirical distribution

$$\widehat{Q}_t = \frac{1}{K}\sum_{i=1}^{K} \delta_{\mathbf{x}_i}, \qquad \mathbf{x}_i \sim Q_t(\cdot \mid G), \tag{3}$$

where $\delta_{\mathbf{x}}$ denotes a unit point mass at solution $\mathbf{x}$.

An unrestricted target objective, $\min_P \mathbb{E}_{\mathbf{x}\sim P}[H_G(\mathbf{x})]$, is not suitable for constructing intermediate training targets. Since the objective is linear in P, its optimum is attained by distributions supported on globally optimal configurations, so solving it directly amounts to solving the original CO problem. Moreover, learning a direct mapping from random latent inputs to optimal solutions is inherently difficult. We therefore define target improvement relative to the current generator and seek a lower-energy distribution within a bounded neighborhood of $\widehat{Q}_t$,

$$\min_P \quad \mathbb{E}_{\mathbf{x}\sim P}\left[H_G(\mathbf{x})\right] \qquad \text{s.t.} \qquad D(P, \widehat{Q}_t) \le \rho, \tag{4}$$

where D measures discrepancy in solution distribution space and ρ controls the size of the neighborhood around the current generator distribution.

Since the generator distribution is implicit, the discrepancy should be estimable directly from samples without evaluating probability mass functions. We instantiate D with maximum mean discrepancy (Gretton et al., 2012). For two distributions P and Q,

$$\mathrm{MMD}^2_{k_G}(P, Q) = \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim P}\left[k_G(\mathbf{x}, \mathbf{x}')\right] + \mathbb{E}_{\mathbf{y},\mathbf{y}'\sim Q}\left[k_G(\mathbf{y}, \mathbf{y}')\right] - 2\mathbb{E}_{\mathbf{x}\sim P,\,\mathbf{y}\sim Q}\left[k_G(\mathbf{x}, \mathbf{y})\right], \tag{5}$$

where k_G is a positive semidefinite kernel defined on decoded combinatorial solutions, allowing distributional discrepancy to be measured directly in solution space.

With MMD as the discrepancy measure, we instantiate the model-relative target improvement as the following penalized objective,

$$P_t = \arg\min_P \left\{ \mathbb{E}_{\mathbf{x}\sim P}\left[H_G(\mathbf{x})\right] + \frac{\lambda}{2}\,\mathrm{MMD}^2_{k_G}\left(P, \widehat{Q}_t\right) \right\}, \tag{6}$$

where $\lambda > 0$ controls the strength of the proximity regularization. The energy term favors lower-energy solutions, while the MMD term anchors the target to the empirical distribution of the current generator. The resulting P_t therefore represents an improvement relative to the generator at the current training stage, rather than a fixed destination in solution space. After this improvement is transferred to the generator, this improvement provides the reference distribution for the next stage.

Parallel support expansion. The proximal objective specifies how the target distribution should improve, but its ability to improve is limited by the solutions available in its support. If target construction were restricted to samples from $\widehat{Q}_t$, it could only redistribute probability mass among solutions already produced by the current generator. Continued target improvement therefore requires exposing additional candidate solutions beyond the current samples.

To expand the candidate support without introducing sequential refinement, we adopt a differential-evolution-style population operator (Storn & Price, 1997). For each generator output $\mathbf{h}_i$, we independently generate L variants,

$$\widetilde{\mathbf{h}}_{i,\ell} = \mathbf{h}_i + F\left(\mathbf{h}_{a_{i,\ell}} - \mathbf{h}_{b_{i,\ell}}\right) + \sigma\boldsymbol{\epsilon}_{i,\ell}, \tag{7}$$

where $a_{i,\ell}$ and $b_{i,\ell}$ are sampled independently and uniformly from $\{1,\ldots,K\}$, and $\boldsymbol{\epsilon}_{i,\ell} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$ are independent Gaussian perturbations. The differential term exploits variation among current generator outputs to define exploration directions, while the Gaussian perturbation adds local stochasticity. All variants are generated independently from the original outputs, allowing support expansion to be performed fully in parallel.

Each explored priority vector is decoded as $\widetilde{\mathbf{x}}_{i,\ell} = D_G(\widetilde{\mathbf{h}}_{i,\ell})$. Together with the original generator samples, these solutions form the expanded candidate set $\mathcal{Y}_t = \{\mathbf{x}_i\}_{i=1}^K \cup \{\widetilde{\mathbf{x}}_{i,\ell}\}_{i,\ell}$. The original generator samples are retained so that $\widehat{Q}_t$ remains available during subsequent target optimization. Parallel exploration only expands the solutions available for improvement. Given the resulting support $\mathcal{Y}_t$, the next step determines how probability mass should be allocated over these candidates to construct the target distribution.

Target distribution construction. Given the expanded candidate set, we now realize the model-relative improvement objective over its finite support. Let $\mathcal{Y}_t = \{\mathbf{y}_1,\ldots,\mathbf{y}_M\}$. We consider distributions supported on $\mathcal{Y}_t$,

$$\mathcal{P}(\mathcal{Y}_t) = \left\{ P_{\mathbf{w}} = \sum_{j=1}^M w_j \delta_{\mathbf{y}_j} : \mathbf{w} \in \Delta_M \right\}, \tag{8}$$

where $\Delta_M = \{\mathbf{w} \in \mathbb{R}_+^M : \sum_{j=1}^M w_j = 1\}$. We now optimize equation 6 over distributions supported on $\mathcal{Y}_t$. Using the parameterization above, this gives

$$\mathbf{w}_t^* = \arg\min_{\mathbf{w}\in\Delta_M} \left\{ \mathbb{E}_{\mathbf{x}\sim P_{\mathbf{w}}}\left[H_G(\mathbf{x})\right] + \frac{\lambda}{2}\,\mathrm{MMD}^2_{k_G}\left(P_{\mathbf{w}}, \widehat{Q}_t\right) \right\}. \tag{9}$$

The optimized weights then define the target distribution

$$P_t = \sum_{j=1}^M w_{t,j}^* \delta_{\mathbf{y}_j}. \tag{10}$$

The energy term shifts probability toward lower-energy solutions, while the MMD term controls how far the target moves from the current generator distribution. Support expansion therefore determines which solutions are available for improvement, whereas target optimization determines how

probability mass is allocated among them. Since the energy term is linear in $\mathbf{w}$ and the kernel matrix induced by k_G is positive semidefinite, equation 9 is a convex quadratic program over the probability simplex and therefore admits a globally optimal solution.

The following proposition formalizes the improvement achieved by the optimized target. The proof is provided in Appendix A.4.

Proposition 1 (Target distribution improvement)**.** *Let $\mathcal{Y}_t$ contain all K samples defining $\widehat{Q}_t$, and let P_t be the target distribution defined by the optimal weights in equation 9. Then*

$$\mathbb{E}_{\mathbf{x}\sim\widehat{Q}_t}\left[H_G(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim P_t}\left[H_G(\mathbf{x})\right] \geq \frac{\lambda}{2}\,\mathrm{MMD}^2_{k_G}\left(P_t, \widehat{Q}_t\right) \geq 0. \tag{11}$$

Proposition 1 guarantees that the optimized target has no higher expected energy than the current empirical distribution. Moreover, moving farther from $\widehat{Q}_t$ under MMD requires a correspondingly larger reduction in expected energy. This guarantee only requires the original generator samples to remain in $\mathcal{Y}_t$ and does not depend on how the additional candidates are generated. Exploration can therefore focus on expanding the available support, while target optimization determines which discovered improvements should be transferred to the generator.

SIPG thus completes the target-improvement stage by producing an improved discrete distribution P_t relative to the current generator. The remaining task is to transfer this improvement into the direct generator, which we address next with decoder basin matching.

3.3 Decoder basin matching

SIPG produces an improved target distribution P_t in discrete solution space. Closing the self-improving loop requires transferring this improvement to the direct generator, whose outputs lie in the continuous priority space. Since the decoder D_G is many-to-one, a target solution $\mathbf{x}$ can be produced by many different priority vectors $\mathbf{h}$ and therefore does not specify a unique continuous training target. DBM resolves this ambiguity by supervising the generator through the regions of priority space that decode to the desired solutions.

Decoder basins. For a discrete solution $\mathbf{x}$, we define its decoder basin as

$$\mathcal{B}_G(\mathbf{x}) = \left\{\mathbf{h} \in \mathbb{R}^n : D_G(\mathbf{h}) = \mathbf{x}\right\}. \tag{12}$$

All priority vectors in $\mathcal{B}_G(\mathbf{x})$ produce the same decoded solution and are therefore equivalent for solution generation. The target $\mathbf{x}$ should thus supervise the generator toward its decoder basin rather than toward a single priority representation.

Directly enforcing membership in $\mathcal{B}_G(\mathbf{x})$ would require differentiating through the discrete decoder. We instead express the decoder decisions needed to reproduce $\mathbf{x}$ as constraints on the priority scores and replace violations of these constraints with continuous penalties. Their aggregate defines a decoder-compatible cost $c_G(\mathbf{h}, \mathbf{x})$, providing a differentiable measure of whether $\mathbf{h}$ lies in the desired decoder region. For greedy MIS decoding (Sun & Yang, 2023), for example, each node excluded from the target solution is associated with a selected neighbor that precedes it in the decoding order and blocks its selection. The corresponding cost penalizes violations of these required priority relations. By enforcing only the relations needed to reproduce $\mathbf{x}$, c_G guides the generator toward $\mathcal{B}_G(\mathbf{x})$ without forcing a particular priority representation. In particular, $c_G(\mathbf{h}, \mathbf{x}) = 0$ guarantees that $D_G(\mathbf{h}) = \mathbf{x}$. Settings for the other problems considered in this work and further details are provided in Appendix A.2.

Basin matching. The decoder-compatible cost specifies how a generator output should be supervised by a given target solution. Target transfer, however, also requires determining the correspondence between multiple stochastic generator outputs and samples from P_t. We draw $\mathbf{s}_1, \ldots, \mathbf{s}_K \sim P_t$ and independently generate $\mathbf{h}_i = f_\theta(G, \boldsymbol{\xi}_i)$ with $\boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$. Since neither collection has an intrinsic ordering, pairing samples by their indices would introduce a correspondence unrelated to decoder compatibility.

We therefore determine the correspondence by minimizing the total decoder-compatible cost,

$$\pi^* = \arg\min_{\pi\in\mathfrak{S}_K} \sum_{i=1}^{K} c_G\left(\mathbf{h}_i, \mathbf{s}_{\pi(i)}\right), \tag{13}$$

where $\mathfrak{S}_K$ denotes the set of permutations of K elements. Using the resulting assignment, the generator is trained with

$$\mathcal{L}(\theta) = \frac{1}{K} \sum_{i=1}^{K} c_G \left(f_\theta(G, \boldsymbol{\xi}_i), \mathbf{s}_{\pi^*(i)} \right). \tag{14}$$

This assignment removes dependence on sample ordering and matches generator outputs to target solutions according to decoder compatibility. The optimization in equation 13 is a minimum-cost bipartite matching problem and is solved exactly with the Hungarian algorithm (Kuhn, 1955). During each gradient update, the sampled target solutions and the assignment π^* are treated as fixed, so gradients are propagated only through f_θ.

DBM thus transfers the improvement represented by P_t into the direct generator. The updated generator induces the next solution distribution Q_{t+1}, which becomes the reference for the subsequent SIPG stage. Together, SIPG and DBM complete the self-improving loop from target improvement to target transfer. The complete training procedure is summarized in Appendix A.3 (Algorithm 1).

Inference. All exploration and target improvement are confined to training. At inference, each candidate is generated by $\mathbf{h} = f_\theta(G, \boldsymbol{\xi})$ followed by $\mathbf{x} = D_G(\mathbf{h})$, where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$. Thus, each candidate requires only one evaluation of the learned generator.

4 Experiments

Our experiments evaluate BOLT under direct inference. We first compare BOLT with existing unsupervised methods using a unified direct-inference protocol, reporting both solution quality and inference time. We then conduct controlled ablations of the target-improvement and target-transfer mechanisms to examine the contributions of SIPG and DBM.

4.1 Experimental Setup

Problems and datasets. We evaluate BOLT on three graph optimization problems following the benchmark protocol of prior unsupervised NCO solvers (Sanokowski et al., 2024; 2025): maximum independent set (MIS), maximum clique (MaxCl), and minimum dominating set (MDS). MIS and MaxCl are evaluated on RB-small and RB-large instances, while MDS is evaluated on BA-small and BA-large instances. Small instances contain 200–300 vertices, while large instances contain 800–1200 vertices. For all problems, we follow prior work (Sanokowski et al., 2024; 2025) in data generation and use the same training, validation, and test splits.

Metrics. We evaluate solution quality using the problem-specific objective and the relative gap with respect to reference solutions. For MIS and MaxCl, we report the average cardinality of the decoded independent set and clique, respectively, where larger values are better. For MDS, we report the average cardinality of the decoded dominating set, where smaller values are better. The relative gap is computed from test-set average objective values. Specifically, let $\bar{z}$ and $\bar{z}_{\mathrm{ref}}$ denote the average objective value obtained by a method and the average reference objective value over the test set, respectively. We compute the gap as $(\bar{z}_{\mathrm{ref}} - \bar{z})/\bar{z}_{\mathrm{ref}}$ for maximization problems and $(\bar{z} - \bar{z}_{\mathrm{ref}})/\bar{z}_{\mathrm{ref}}$ for minimization problems. All reported results use a single-sample inference protocol, with one stochastic latent sample and one generator evaluation per test instance followed by decoding, without repeated sampling or candidate selection. We additionally report the total end-to-end inference time over all test instances, including both neural network evaluation and decoding. Details of reference solution generation are provided in Appendix B.1.

Baselines. We compare BOLT with unsupervised neural CO methods that can be evaluated under direct inference without altering their core solver mechanisms. EGN (Karalias & Loukas, 2020), EGN-Anneal (Sun et al., 2022), ScatteringClique (Min et al., 2022), GCON (Wenkel et al., 2024; Cantürk et al., 2026), and UniHetCO (Nguyen & Safro, 2026) represent native direct solvers. Although UniHetCO was originally evaluated under different benchmark settings, its formulation directly applies to the graph subset-selection problems considered here. We therefore retrain it on our datasets and data splits for a controlled comparison without altering its core solver mechanism. We

Table 1: Direct-inference performance on MIS in terms of solution size, relative gap, and inference time. ↑ indicates higher is better and ↓ indicates lower is better.

	RB-small			RB-large		
Method	Size ↑	Gap (%) ↓	Time (s) ↓	Size ↑	Gap (%) ↓	Time (s) ↓
EGN	16.90	15.67	5.55	34.36	19.78	41.69
EGN-Anneal	16.93	15.52	8.96	34.41	19.66	58.97
GCON	18.12	9.58	–	–	–	–
UniHetCO	17.12	14.56	14.71	34.21	20.14	57.06
DiffUCO-Direct	17.34	13.47	8.48	31.91	25.50	22.10
SDDS-rKL-Direct	16.63	17.02	8.42	32.48	24.17	21.38
SDDS-fKL-Direct	16.53	17.51	8.42	33.07	22.79	21.50
BOLT (Ours)	**18.39**	**8.23**	**4.46**	**37.53**	**12.37**	**4.95**

Table 2: Direct-inference performance on MDS in terms of solution size, relative gap, and inference time. ↓ indicates lower is better.

	BA-small			BA-large		
Method	Size ↓	Gap (%) ↓	Time (s) ↓	Size ↓	Gap (%) ↓	Time (s) ↓
EGN	36.49	31.07	5.98	142.71	37.47	14.26
EGN-Anneal	36.99	32.87	14.54	136.21	31.21	51.28
ScatteringClique	31.07	11.60	–	121.26	16.81	–
GCON	30.26	8.69	–	113.47	9.31	–
UniHetCO	33.05	18.73	19.22	117.85	13.53	99.12
DiffUCO-Direct	29.86	7.26	14.98	112.69	8.55	45.79
SDDS-rKL-Direct	32.05	15.12	15.05	113.53	9.36	45.45
SDDS-fKL-Direct	33.27	19.50	15.08	116.23	11.96	45.75
BOLT (Ours)	**29.12**	**4.60**	**5.30**	**108.67**	**4.68**	**8.69**

additionally include DiffUCO (Sanokowski et al., 2024) and SDDS (Sanokowski et al., 2025). DiffUCO explicitly considers the single-step setting, while SDDS builds on the same discrete diffusion framework and introduces alternative reverse- and forward-KL training objectives. The released models and reported unsupervised NCO results for both methods focus on multi-step diffusion, while corresponding single-step checkpoints and benchmark results are not available. Directly truncating these models to one reverse step would introduce a mismatch between training and inference. We therefore train dedicated single-step models for direct inference. We denote these variants as DiffUCO-Direct, SDDS-rKL-Direct, and SDDS-fKL-Direct, respectively. For all baselines retrained by us, we use the same training, validation, and test splits as BOLT and retain the original learning objective, model architecture, and decoding procedure of each method. For DiffUCO and SDDS, we modify only the diffusion depth, setting it to one for direct inference. Hyperparameters and checkpoints are selected exclusively using the validation set, with no tuning on the test set. All retrained methods are evaluated under the same direct-inference protocol and runtime environment. Other methods discussed in Section 2 are not included in the numerical comparison when adapting them to our direct-inference protocol would require substantial changes to their core method design.

4.2 Main results

Tables 1, 2, and 3 compare the methods under direct inference. BOLT achieves the lowest relative gap across all three problems and both graph scales. Among methods evaluated in our runtime environment, BOLT also achieves the lowest inference time across all six settings. For GCON and ScatteringClique, we use previously reported results since corresponding pretrained checkpoints are unavailable. GCON results on MaxCl and MDS are taken from Wenkel et al. (2024), and its MIS RB-small result from Cantürk et al. (2026); ScatteringClique results are taken from the reproduction in Wenkel et al. (2024). Their runtimes are omitted since they were not measured in our environment. "–" denotes unavailable or non-comparable results.

Table 3: Direct-inference performance on MaxCl in terms of solution size, relative gap, and inference time. ↑ indicates higher is better and ↓ indicates lower is better.

	RB-small			RB-large		
Method	Size ↑	Gap (%) ↓	Time (s) ↓	Size ↑	Gap (%) ↓	Time (s) ↓
EGN	12.90	32.14	8.05	25.05	37.56	76.47
EGN-Anneal	12.70	33.19	9.23	25.31	36.91	68.75
ScatteringClique	15.80	16.89	–	29.36	26.82	–
GCON	15.87	16.52	–	29.46	26.57	–
UniHetCO	14.63	23.06	14.69	26.15	34.81	86.59
DiffUCO-Direct	14.95	21.36	8.84	23.74	40.83	31.52
SDDS-rKL-Direct	12.07	36.51	8.93	24.40	39.18	31.67
SDDS-fKL-Direct	13.98	26.46	9.13	25.23	37.11	31.73
BOLT (Ours)	**17.47**	**8.10**	**4.66**	**36.45**	**9.15**	**5.01**

Table 4: Ablation of the main design choices in BOLT. ↑ indicates higher is better and ↓ indicates lower is better.

Variant	MIS RB-small ↑	MaxCl RB-small ↑	MDS BA-small ↓
Self-labeling	17.46	15.69	30.36
Best-candidate target	18.32	17.28	29.25
Pointwise regression	18.35	17.14	29.47
Random pairing	18.16	17.31	29.50
BOLT (Ours)	**18.39**	**17.47**	**29.12**

4.3 Ablation Studies

We design the ablations around the two transitions in the self-improving loop of BOLT: converting discovered solutions into a learnable target distribution and transferring this distribution back to the direct generator. Table 4 reports the results under the same training and direct-inference protocol.

For target improvement, we first replace SIPG with self-labeling, where improved solutions discovered during training are directly reused as supervision. The resulting degradation across all three problems highlights the distinction between discovering a better solution and constructing a suitable learning target. Self-labeling does not account for how far the new targets lie from the current generator distribution. We further retain the explored candidate pool but replace the model-relative target distribution with the single best discovered candidate. This variant also performs consistently worse, despite explicitly selecting the highest-quality available solution. Together, these controls indicate that SIPG benefits not only from discovering better candidates, but also from providing a gradual, model-relative path through which improvements can be absorbed by the generator.

For target transfer, pointwise regression replaces decoder-compatible supervision with regression toward a particular priority representation of each target solution. Its degradation is consistent with the many-to-one decoder structure: a discrete solution corresponds to a region of valid priority representations, so forcing a single representation imposes unnecessary constraints. Random pairing retains decoder-compatible supervision but removes the minimum-cost correspondence between stochastic generator outputs and target solutions. Its degradation further shows that basin-level supervision alone is insufficient without resolving the correspondence between the two unordered sample sets. Additional ablations are provided in Appendix C.

5 Conclusion

We introduced BOLT, a direct unsupervised NCO framework that moves solution exploration from iterative inference to training-time breadth exploration. Across three graph optimization problems, BOLT consistently improves solution quality over prior direct unsupervised NCO methods while preserving efficient direct inference.

AI USE STATEMENT

Generative AI tools, including large language models (LLMs), were used during this work to assist with surveying and summarizing related literature, and drafting and editing parts of the manuscript for clarity and readability. All AI-assisted literature-related information and manuscript edits were independently reviewed and verified by the authors. Experimental results reported in the paper were obtained from our own implementations and experiments, and claims about prior work were checked against the corresponding original sources. The authors made all final decisions regarding the content and presentation of the paper and take full responsibility for this work.

ETHICS STATEMENT

This work studies unsupervised neural methods for combinatorial optimization using synthetic and publicly available benchmark datasets. It does not involve human participants, personally identifiable information, or sensitive data. We are not aware of any specific ethical concerns arising from this work beyond standard considerations associated with computational research.

REPRODUCIBILITY STATEMENT

We provide implementation details, experimental settings, and evaluation protocols in the main text and appendix to support reproduction of the reported results.

REFERENCES

Utku Umur Acikalin, Aaron Ferber, and Carla Gomes. Learning to explore and exploit with gnns for unsupervised combinatorial optimization. In *ICLR*, 2025.

Fanchen Bu, Hyeonsoo Jo, Soo Yong Lee, Sungsoo Ahn, and Kijung Shin. Tackling prevalent conditions in unsupervised combinatorial optimization: Cardinality, minimum, covering, and more. In *ICML*, 2024.

Semih Cantürk, Thomas Sabourin, Frederik Wenkel, Michael Perlmutter, and Guy Wolf. Can computational reducibility lead to transferable models for graph combinatorial optimization? In *ICML*, 2026.

Ming Chen, Jie Chun, Shang Xiang, Luona Wei, Yonghao Du, Qian Wan, Yuning Chen, and Yingwu Chen. Learning to solve quadratic unconstrained binary optimization in a classification way. In *NeurIPS*, 2024.

Inguk Choi, Woo-Jin Shin, Sang-Hyun Cho, and Hyun-Jung Kim. Towards generalizable multi-policy optimization with self-evolution for job scheduling. In *NeurIPS*, 2025.

Andrea Corsini, Angelo Porrello, Simone Calderara, and Mauro Dell'Amico. Self-labeling the job shop scheduling problem. In *NeurIPS*, 2024.

Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. GOAL: A generalist combinatorial optimization agent learner. In *ICLR*, 2025.

Shengyu Feng and Yiming Yang. Regularized langevin dynamics for combinatorial optimization. In *ICML*, 2025.

Shengyu Feng, Weiwei Sun, Shanda Li, Ameet Talwalkar, and Yiming Yang. FrontierCO: Real-world and large-scale evaluation of machine learning solvers for combinatorial optimization. In *ICLR*, 2026a.

Shengyu Feng, Tarun Suresh, and Yiming Yang. Unsupervised diffusion solver for combinatorial optimization via combinatorial adjoint matching. In *ICML*, 2026b.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. In *AAAI*, 2021.

Andoni I Garmendia, Josu Ceberio, and Alexander Mendiburu. Enabling population-based architectures for neural combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 2026.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13:723–773, 2012.

Shengda Gu, Kai Li, Junliang Xing, Yifan Zhang, and Jian Cheng. Synergizing reinforcement learning and genetic algorithms for neural combinatorial optimization. *arXiv preprint arXiv:2506.09404*, 2025.

Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2023. URL `https://www.gurobi.com`.

Nasimeh Heydaribeni, Xinrui Zhan, Ruisi Zhang, Tina Eliassi-Rad, and Farinaz Koushanfar. Distributed constrained combinatorial optimization leveraging hypergraph neural networks. *Nature Machine Intelligence*, 2024.

Seong-Hyun Hong, Hyun-Sung Kim, Zian Jang, Deunsol Yoon, Hyungseok Song, and Byung-Jun Lee. Unsupervised training of diffusion models for feasible solution generation in neural combinatorial optimization. *arXiv preprint arXiv:2411.00003*, 2024.

Yixin Huang, Shengyu Feng, and Yiming Yang. Unsupervised neural langevin sampler for mixed integer linear programming. In *ICML*, 2026.

Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. In *ICLR*, 2022.

Yuma Ichikawa. Controlling continuous relaxation for combinatorial optimization. In *NeurIPS*, 2024.

Yaochu Jin, Xueming Yan, Shiqing Liu, and Xiangyu Wang. A unified framework for combinatorial optimization based on graph neural networks. *arXiv preprint arXiv:2406.13125*, 2024.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *NeurIPS*, 2020.

Nikolaos Karalias, Joshua Robinson, Andreas Loukas, and Stefanie Jegelka. Neural set function extensions: Learning with discrete functions in high dimensions. In *NeurIPS*, 2022.

Nikolaos Karalias, Akbar Rafiey, Yifei Xu, Zhishang Luo, Behrooz Tahmasebi, Connie Jiang, and Stefanie Jegelka. Geometric algorithms for neural combinatorial optimization with constraints. In *NeurIPS*, 2026.

Hyeonah Kim, Minsu Kim, Sungsoo Ahn, and Jinkyoo Park. Symmetric replay training: Enhancing sample efficiency in deep reinforcement learning for combinatorial optimization. In *ICML*, 2024.

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

Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *NeurIPS*, 2015.

Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F Werneck. Finding near-optimal independent sets at scale. In *ALENEX*, 2016.

Haoyu Lei, Kaiwen Zhou, Yinchuan Li, Zhitang Chen, and Farzan Farnia. Boosting cross-problem generalization in diffusion-based neural combinatorial solver via inference time adaptation. In *AAAI*, 2026.

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 *NeurIPS*, 2024.

Yuanshu Li, Di Wang, Wei Du, Xuan Wu, Peng Zhao, Yubin Xiao, and You Zhou. Efficient few-step solution generation via discrete flow matching for combinatorial optimization. In *AAAI*, 2026a.

Yuanyu Li, Jintao Xu, Zijiang Liu, Yongzhi Qi, Ningxuan Kang, Jianshen Zhang, Wei Qi, Chen Xie, and Zuo-Jun Max Shen. Sspo: Structure-aware similarity-weighted preference optimization for neural combinatorial optimization. *arXiv preprint arXiv:2608.12443*, 2026b.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *NeurIPS*, 2018.

Zijun Liao, Jinbiao Chen, Debing Wang, Zizhen Zhang, and Jiahai Wang. Bopo: Neural combinatorial optimization via best-anchored and objective-guided preference optimization. In *ICML*, 2025.

Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *NeurIPS*, 2023.

Fu Luo, Xi Lin, Yaoxin Wu, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Qingfu Zhang. Boosting neural combinatorial optimization for large-scale vehicle routing problems. In *ICLR*, 2025.

Laurin Luttmann and Lin Xie. Multi-action self-improvement for neural combinatorial optimization. In *ICLR*, 2026.

Yimeng Min, Frederik Wenkel, Michael Perlmutter, and Guy Wolf. Can hybrid geometric scattering networks help solve the maximum clique problem? In *NeurIPS*, 2022.

Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. In *NeurIPS*, 2023.

Kien X Nguyen and Ilya Safro. Unihetco: A unified heterogeneous representation for multi-problem learning in unsupervised neural combinatorial optimization. *arXiv preprint arXiv:2603.11456*, 2026.

Mingjun Pan, Guanquan Lin, You-Wei Luo, Bin Zhu, Zhien Dai, Lijun Sun, and Chun Yuan. Preference optimization for combinatorial optimization problems. In *ICML*, 2025a.

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 *ICLR*, 2025b.

Jonathan Pirnay, Quirin Göttl, Jakob Burger, and Dominik Gerhard Grimm. Policy-based self-competition for planning problems. In *ICLR*, 2023.

Robbert Reijnen, Yaoxin Wu, Zaharah Bukhsh, and Yingqian Zhang. Graph-supported dynamic algorithm configuration for multi-objective combinatorial optimization. In *ICML*, 2025.

Sebastian Sanokowski, Wilhelm Berghammer, Sepp Hochreiter, and Sebastian Lehner. Variational annealing on graphs for combinatorial optimization. In *NeurIPS*, 2023.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *ICML*, 2024.

Sebastian Sanokowski, Wilhelm Franz Berghammer, Haoyu Peter Wang, Martin Ennemoser, Sepp Hochreiter, and Sebastian Lehner. Scalable discrete diffusion samplers: Combinatorial optimization and statistical physics. In *ICLR*, 2025.

Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

Joan Salvà Soler and Günther R Raidl. A denoising diffusion-based evolutionary algorithm framework: Application to the maximum independent set problem. *arXiv preprint arXiv:2510.08627*, 2025.

Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

Haoran Sun, Etash K Guha, and Hanjun Dai. Annealed training for combinatorial optimization on graphs. *arXiv preprint arXiv:2207.11542*, 2022.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *NeurIPS*, 2023.

Chenguang Wang, Zhouliang Yu, Stephen McAleer, Tianshu Yu, and Yaodong Yang. ASP: Learn a universal neural solver! *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6): 4102–4114, 2024.

Haoyu Wang and Pan Li. Unsupervised learning for combinatorial optimization needs meta-learning. In *ICLR*, 2023.

Haoyu Peter Wang, Nan Wu, Hang Yang, Cong Hao, and Pan Li. Unsupervised learning for combinatorial optimization with principled objective relaxation. In *NeurIPS*, 2022a.

Mingzhao Wang, You Zhou, Zhiguang Cao, Yubin Xiao, Xuan Wu, Wei Pang, Yuan Jiang, Hui Yang, Peng Zhao, and Yuanshu Li. An efficient diffusion-based non-autoregressive solver for traveling salesman problem. In *SIGKDD*, 2025a.

Runzhong Wang, Junchi Yan, and Xiaokang Yang. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5261–5279, 2021.

Runzhong Wang, Li Shen, Yiting Chen, Xiaokang Yang, Dacheng Tao, and Junchi Yan. Towards one-shot neural combinatorial solvers: Theoretical and empirical notes on the cardinality-constrained case. In *ICLR*, 2022b.

Tao Wang, He Liu, Yidong Li, Yi Jin, Xiaohui Hou, and Haibin Ling. Learning combinatorial solver for graph matching. In *CVPR*. IEEE, 2020.

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 *NeurIPS*, 2025b.

Lowell Weissman, Michael Krumdick, and A Lynn Abbott. Complexity scaling laws for neural models using combinatorial optimization. In *NeurIPS*, 2025.

Frederik Wenkel, Semih Cantürk, Stefan Horoi, Michael Perlmutter, and Guy Wolf. Towards a general recipe for combinatorial optimization with multi-filter gnns. In *LoG*, 2024.

Honguk Woo, Hyunsung Lee, and Sangwoo Cho. An efficient combinatorial optimization model using learning-to-rank distillation. In *AAAI*, 2022.

Yubin Xiao, Di Wang, Boyang Li, Mingzhao Wang, Xuan Wu, Changliang Zhou, and You Zhou. Distilling autoregressive models to obtain high-performance non-autoregressive solvers for vehicle routing problems with faster inference speed. In *AAAI*, 2024.

Yubin Xiao, Yuesong Wu, Rui Cao, Di Wang, Zhiguang Cao, Xuan Wu, Peng Zhao, Yuanshu Li, You Zhou, and Yuan Jiang. Dgl: dynamic global-local information aggregation for scalable vrp generalization with self-improvement learning. In *IJCAI*, 2025.

Zhenxing Xu, Zeyuan Ma, Weidong Bao, Hui Yan, Yan Zheng, and Ji Wang. Rethink efficiency side of neural combinatorial solver: An offline and self-play paradigm. *arXiv preprint arXiv:2602.20730*, 2026.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with GFlowNets. In *NeurIPS*, 2023.

Zhiqin Zhang, Yining Ma, Zhiguang Cao, and Hoong Chuin Lau. Probing neural combinatorial optimization models. In *NeurIPS*, 2025.

A Additional Method Details

A.1 Problem Definitions

We consider three graph combinatorial optimization problems: maximum independent set (MIS), minimum dominating set (MDS), and maximum clique (MaxCl). Let $G = (V, E)$ be an undirected graph with $n = |V|$, and let $\mathbf{x} \in \{0, 1\}^n$ denote a binary node-selection vector, where $x_i = 1$ indicates that vertex i is selected. Following prior unsupervised NCO formulations (Sanokowski et al., 2024; 2025), each problem is represented by an energy function $H_G(\mathbf{x})$ whose global minimizers correspond to feasible optimal solutions.

Maximum independent set. An independent set is a subset of vertices containing no pair of adjacent vertices. MIS seeks an independent set of maximum cardinality and can be written as

$$\max_{\mathbf{x}\in\{0,1\}^n} \quad \sum_{i\in V} x_i \qquad \text{s.t.} \qquad x_i + x_j \leq 1, \quad \forall (i,j) \in E. \tag{15}$$

We use the penalty-based energy

$$H_G^{\mathrm{MIS}}(\mathbf{x}) = -A \sum_{i\in V} x_i + B \sum_{(i,j)\in E} x_i x_j, \tag{16}$$

where $A > 0$ rewards selected vertices and $B > A$ penalizes violations of the independence constraint. If a configuration contains an edge $(i, j) \in E$ with $x_i = x_j = 1$, removing either endpoint decreases the cardinality reward by A while eliminating at least one penalty of magnitude B. Such a configuration therefore cannot be a global minimizer. For every feasible independent set, the penalty term vanishes, so minimizing H_G^{MIS} is equivalent to maximizing independent-set cardinality.

Minimum dominating set. A dominating set is a subset of vertices such that every vertex is either selected or adjacent to at least one selected vertex. MDS seeks a dominating set of minimum cardinality:

$$\min_{\mathbf{x}\in\{0,1\}^n} \quad \sum_{i\in V} x_i \qquad \text{s.t.} \qquad x_i + \sum_{j\in\mathcal{N}(i)} x_j \geq 1, \quad \forall i \in V, \tag{17}$$

where $\mathcal{N}(i)$ denotes the neighborhood of vertex i. A vertex i is undominated exactly when neither i nor any of its neighbors is selected, which is indicated by

$$u_i(\mathbf{x}) = (1 - x_i) \prod_{j\in\mathcal{N}(i)} (1 - x_j). \tag{18}$$

The corresponding energy is

$$H_G^{\mathrm{MDS}}(\mathbf{x}) = A \sum_{i\in V} x_i + B \sum_{i\in V} (1 - x_i) \prod_{j\in\mathcal{N}(i)} (1 - x_j), \tag{19}$$

where the first term measures the size of the selected set and the second penalizes undominated vertices. For $B > A$, a configuration containing an undominated vertex cannot be globally optimal, since selecting that vertex incurs a cost of A while removing at least one penalty of magnitude B. For feasible dominating sets, the penalty term is zero, and minimizing H_G^{MDS} reduces to minimizing dominating-set cardinality.

Maximum clique. A clique is a subset of vertices in which every pair of selected vertices is connected by an edge. MaxCl seeks a clique of maximum cardinality and can be written as

$$\max_{\mathbf{x}\in\{0,1\}^n} \quad \sum_{i\in V} x_i \qquad \text{s.t.} \qquad x_i + x_j \leq 1, \quad \forall (i,j) \notin E. \tag{20}$$

We define

$$H_G^{\mathrm{MaxCl}}(\mathbf{x}) = -A \sum_{i\in V} x_i + B \sum_{(i,j)\notin E} x_i x_j, \tag{21}$$

with $B > A$. The penalty term counts pairs of selected vertices that are not connected in G. Any configuration containing such a pair can be improved by removing one endpoint, so every global

minimizer is a feasible clique. Among feasible configurations, minimizing equation 21 is equivalent to maximizing clique cardinality.

These energy formulations instantiate the objective H_G introduced in Section 3.1 and provide the solution-quality signal used throughout unsupervised training.

A.2 Decoder-Compatible Costs

This section specifies the problem-specific decoders and the corresponding decoder-compatible costs used by DBM. Let $G = (V, E)$ be an undirected graph with $n = |V|$, and let $\mathbf{h} \in \mathbb{R}^n$ denote the priority vector produced by the direct generator. A deterministic decoder D_G maps the priorities to a binary solution $\mathbf{x} = D_G(\mathbf{h}) \in \{0, 1\}^n$. Ties in priority are resolved by a fixed ordering of node indices.

As discussed in Section 3.3, a discrete target $\mathbf{x}$ does not determine a unique priority representation. We therefore characterize sufficient priority relations for reproducing $\mathbf{x}$ under the corresponding decoder and penalize violations of these relations. For the decoder-generated targets considered below, the resulting cost satisfies

$$c_G(\mathbf{h}, \mathbf{x}) \geq 0, \qquad c_G(\mathbf{h}, \mathbf{x}) = 0 \Longrightarrow D_G(\mathbf{h}) = \mathbf{x}. \tag{22}$$

Thus, the supervision constrains the priority relations needed to reproduce the target solution without requiring regression toward a particular continuous representation.

Priority normalization. Before evaluating the decoder-compatible costs, we standardize the priority scores to zero mean and unit variance and denote the resulting priorities by $\mathbf{z}$. A small lower bound is applied to the standard deviation for numerical stability. This transformation preserves the priority ordering used by the decoder while making the margin constraints insensitive to the overall scale of the generator output. The costs below use squared hinge penalties $[a]_+^2 = \max(a, 0)^2$ with a positive margin γ and are normalized by the number of vertices n. Exact hyperparameter values are provided in Appendix B.2.

Maximum independent set. For MIS, vertices are processed in descending order of priority. A vertex is selected if none of its previously selected neighbors has already blocked it. This procedure returns a maximal independent set.

For a target solution $\mathbf{x}$, let

$$S(\mathbf{x}) = \{v \in V : x_v = 1\}. \tag{23}$$

Since targets are produced by the same greedy decoder, $S(\mathbf{x})$ is maximal. Hence, every excluded vertex $v \notin S(\mathbf{x})$ has at least one selected neighbor. Reproducing the target is ensured if some such neighbor precedes v in the priority order:

$$\max_{u \in S(\mathbf{x}) \cap \mathcal{N}(v)} z_u > z_v, \qquad \forall v \notin S(\mathbf{x}), \tag{24}$$

where $\mathcal{N}(v)$ denotes the neighborhood of v.

We therefore define

$$c_G^{\mathrm{MIS}}(\mathbf{h}, \mathbf{x}) = \frac{1}{n} \sum_{v \notin S(\mathbf{x})} \left[\gamma + z_v - \max_{u \in S(\mathbf{x}) \cap \mathcal{N}(v)} z_u \right]_+^2. \tag{25}$$

If $S(\mathbf{x}) = V$, the sum is empty and the cost is defined as zero.

To verify equation 22, suppose $c_G^{\mathrm{MIS}}(\mathbf{h}, \mathbf{x}) = 0$. Then every excluded vertex v has a selected neighbor u satisfying $z_u \geq z_v + \gamma$, so u is processed strictly before v. Assume that an excluded vertex is nevertheless selected, and consider the first such vertex in the decoding order. Its blocking neighbor has already been processed. No earlier excluded vertex has been selected by construction, and vertices in $S(\mathbf{x})$ are mutually nonadjacent, so the blocking neighbor has already been selected. This contradicts the selection of v. Therefore all excluded vertices are rejected and all vertices in $S(\mathbf{x})$ are selected, giving

$$c_G^{\mathrm{MIS}}(\mathbf{h}, \mathbf{x}) = 0 \Longrightarrow D_G^{\mathrm{MIS}}(\mathbf{h}) = \mathbf{x}. \tag{26}$$

Maximum clique. For MaxCl, vertices are again processed in descending order of priority. A vertex is selected only if it is adjacent to every previously selected vertex. The decoder therefore returns a maximal clique.

For a vertex v, define its conflict set

$$\overline{\mathcal{N}}(v) = V \setminus \big(\mathcal{N}(v) \cup \{v\}\big), \tag{27}$$

containing all vertices that are not adjacent to v. For a target clique $S(\mathbf{x})$, maximality implies that every excluded vertex $v \notin S(\mathbf{x})$ conflicts with at least one selected vertex. It is therefore sufficient that one such conflicting target vertex precedes v:

$$\max_{u \in S(\mathbf{x}) \cap \overline{\mathcal{N}}(v)} z_u > z_v, \qquad \forall v \notin S(\mathbf{x}). \tag{28}$$

The decoder-compatible cost is

$$c_G^{\mathrm{MaxCl}}(\mathbf{h}, \mathbf{x}) = \frac{1}{n} \sum_{v \notin S(\mathbf{x})} \left[\gamma + z_v - \max_{u \in S(\mathbf{x}) \cap \overline{\mathcal{N}}(v)} z_u \right]_+^2. \tag{29}$$

If $S(\mathbf{x}) = V$, the cost is again defined as zero.

The consistency argument mirrors that of MIS under the corresponding conflict relation. If the cost is zero, every excluded vertex is preceded by a selected vertex with which it conflicts. Such a vertex is therefore rejected when it is processed, whereas the mutually adjacent target vertices never block one another. Hence

$$c_G^{\mathrm{MaxCl}}(\mathbf{h}, \mathbf{x}) = 0 \Longrightarrow D_G^{\mathrm{MaxCl}}(\mathbf{h}) = \mathbf{x}. \tag{30}$$

Minimum dominating set. For MDS, decoding proceeds by reverse deletion. The decoder starts from $A = V$ and processes vertices in ascending order of priority. A vertex $v \in A$ is removed whenever $A \setminus \{v\}$ remains a dominating set. Writing

$$\mathcal{N}[u] = \mathcal{N}(u) \cup \{u\} \tag{31}$$

for the closed neighborhood of u, vertex v can be removed exactly when

$$|A \cap \mathcal{N}[u]| \geq 2, \qquad \forall u \in \mathcal{N}[v]. \tag{32}$$

The decoder maintains feasibility throughout the procedure. Moreover, once a vertex cannot be removed, subsequent deletions cannot make it removable. The final solution is therefore an inclusion-wise minimal dominating set.

Let $S = S(\mathbf{x})$ denote a target dominating set. For each $s \in S$, define its private-witness set

$$W_S(s) = \{u \in V : \mathcal{N}[u] \cap S = \{s\}\}. \tag{33}$$

Since S is an inclusion-wise minimal dominating set, every selected vertex $s \in S$ has at least one private witness. Such a witness u would become undominated if s were removed from S.

To ensure that the decoder retains s, it is sufficient that, for at least one $u \in W_S(s)$, all other vertices capable of dominating u are processed before s:

$$z_s > \max_{v \in \mathcal{N}[u] \setminus \{s\}} z_v. \tag{34}$$

For a candidate witness $u \in W_S(s)$, define

$$\ell_{s,u}(\mathbf{z}) = \begin{cases} \left[\gamma + \max_{v \in \mathcal{N}[u] \setminus \{s\}} z_v - z_s \right]_+^2, & \mathcal{N}[u] \setminus \{s\} \neq \varnothing, \\ 0, & \mathcal{N}[u] \setminus \{s\} = \varnothing. \end{cases} \tag{35}$$

The MDS decoder-compatible cost is then

$$c_G^{\mathrm{MDS}}(\mathbf{h}, \mathbf{x}) = \frac{1}{n} \sum_{s \in S(\mathbf{x})} \min_{u \in W_{S(\mathbf{x})}(s)} \ell_{s,u}(\mathbf{z}). \tag{36}$$

The minimum over private witnesses reflects that retaining s requires only one witness whose domination depends uniquely on s, rather than satisfying the ordering condition for every possible witness.

We again verify the zero-cost property. Assume $c_G^{\mathrm{MDS}}(\mathbf{h}, \mathbf{x}) = 0$ and consider the vertices in the decoder order. Inductively suppose that all previously processed vertices outside S have been removed and all vertices in S remain selected. If the current vertex $v \notin S$, removing it preserves the entire target dominating set S, so the decoder removes v. If the current vertex is $s \in S$, zero cost guarantees a private witness $u \in W_S(s)$ for which every vertex in $\mathcal{N}[u] \setminus \{s\}$ has strictly lower priority than s. These vertices lie outside S and have therefore already been removed. At this point, u is dominated only by s, so deleting s would violate domination and the decoder retains it. By induction, the final decoded set is exactly S, and hence

$$c_G^{\mathrm{MDS}}(\mathbf{h}, \mathbf{x}) = 0 \Longrightarrow D_G^{\mathrm{MDS}}(\mathbf{h}) = \mathbf{x}. \tag{37}$$

Across all three problems, the decoder-compatible costs constrain only the priority relations required to reproduce the target decoding decisions. They therefore provide region-level supervision in priority space rather than regression toward a fixed continuous representation. These costs constitute the pairwise entries used by decoder basin matching in Section 3.3.

A.3 Complete Training Procedure

Algorithm 1 summarizes the complete BOLT training procedure. Each iteration consists of two stages. SIPG first performs target improvement by expanding the support of the current generator and optimizing a model-relative target distribution over the resulting candidates. DBM then performs target transfer by matching samples from this distribution to stochastic generator outputs and updating the generator through the decoder-compatible cost. The procedure is applied independently to each problem instance in a mini-batch.

Here, `Target-Optimize` solves the finite-support convex quadratic program in equation 9. It is initialized at the empirical generator distribution $\widehat{Q}_b$, which is feasible since the original generator samples are retained in the expanded candidate set. We optimize the candidate weights using away-step Frank–Wolfe (Lacoste-Julien & Jaggi, 2015). At each Frank–Wolfe iteration, probability mass is moved from an active candidate with a large gradient component toward one with a smaller component, with the step size determined by exact line search. This preserves the probability-simplex constraints and returns the weights $\mathbf{w}_b^*$ defining P_b.

The operation `Hungarian` denotes the minimum-cost assignment in equation 13. The explored priority vectors used by SIPG are constructed directly from the current generator outputs and therefore require no additional generator evaluations. During target transfer, the optimized target distribution, sampled target solutions, and resulting assignment are treated as fixed, so gradients are propagated only through the generator outputs.

After the parameter update, the generator induces the next solution distribution, which serves as the reference for the subsequent SIPG stage. Repeating target improvement with SIPG and target transfer with DBM realizes the self-improving cycle described in Section 3.3.

A.4 Proof of Target Distribution Improvement

We provide the proof of Proposition 1. For a distribution P supported on $\mathcal{Y}_t$, define

$$\mathcal{F}_t(P) = \mathbb{E}_{\mathbf{x}\sim P}\left[H_G(\mathbf{x})\right] + \frac{\lambda}{2}\,\mathrm{MMD}^2_{k_G}\left(P, \widehat{Q}_t\right). \tag{38}$$

By construction, $\mathcal{Y}_t$ contains all samples used to form $\widehat{Q}_t$. Hence, $\widehat{Q}_t$ is a feasible distribution in $\mathcal{P}(\mathcal{Y}_t)$. Since P_t minimizes equation 38 over this family,

$$\mathcal{F}_t(P_t) \le \mathcal{F}_t(\widehat{Q}_t). \tag{39}$$

Using $\mathrm{MMD}^2_{k_G}(\widehat{Q}_t, \widehat{Q}_t) = 0$, equation 39 gives

$$\mathbb{E}_{\mathbf{x}\sim P_t}\left[H_G(\mathbf{x})\right] + \frac{\lambda}{2}\,\mathrm{MMD}^2_{k_G}\left(P_t, \widehat{Q}_t\right) \le \mathbb{E}_{\mathbf{x}\sim \widehat{Q}_t}\left[H_G(\mathbf{x})\right]. \tag{40}$$

Algorithm 1 Training of BOLT

Require: Generator f_θ, problem distribution ω, batch size B, population size K, expansion multiplicity L, exploration parameters F, σ, proximity weight λ, learning rate η, total iterations T_{train}

for $t = 1, \dots, T_{\text{train}}$ **do**
  Sample a mini-batch $\{G_b\}_{b=1}^{B} \sim \omega$
  for $b = 1, \dots, B$ **do**
  Target improvement with SIPG
    for $i = 1, \dots, K$ **do**
      Sample $\boldsymbol{\xi}_b^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
      $\mathbf{h}_b^{(i)} \leftarrow f_\theta(G_b, \boldsymbol{\xi}_b^{(i)})$
      $\mathbf{x}_b^{(i)} \leftarrow D_{G_b}(\mathbf{h}_b^{(i)})$
    end for
    $\widehat{Q}_b \leftarrow \frac{1}{K} \sum_{i=1}^{K} \delta_{\mathbf{x}_b^{(i)}}$
    for $i = 1, \dots, K$ **do**
      for $\ell = 1, \dots, L$ **do**
        Sample $a_{i,\ell}, b_{i,\ell} \sim \mathrm{Unif}(\{1, \dots, K\})$ and $\boldsymbol{\epsilon}_{i,\ell} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
        $\widetilde{\mathbf{h}}_b^{(i,\ell)} \leftarrow \mathbf{h}_b^{(i)} + F\left(\mathbf{h}_b^{(a_{i,\ell})} - \mathbf{h}_b^{(b_{i,\ell})}\right) + \sigma \boldsymbol{\epsilon}_{i,\ell}$
        $\widetilde{\mathbf{x}}_b^{(i,\ell)} \leftarrow D_{G_b}(\widetilde{\mathbf{h}}_b^{(i,\ell)})$
      end for
    end for
    $\mathcal{Y}_b \leftarrow \{\mathbf{x}_b^{(i)}\}_{i=1}^{K} \cup \{\widetilde{\mathbf{x}}_b^{(i,\ell)}\}_{i,\ell}$
    $\mathbf{w}_b^* \leftarrow \texttt{Target-Optimize}(\mathcal{Y}_b, \widehat{Q}_b, H_{G_b}, \lambda)$
    $P_b \leftarrow \sum_{j=1}^{|\mathcal{Y}_b|} w_{b,j}^* \delta_{\mathbf{y}_{b,j}}$
  Target transfer with DBM
    Sample $\{\mathbf{s}_b^{(i)}\}_{i=1}^{K} \overset{\text{i.i.d.}}{\sim} P_b$
    for $i = 1, \dots, K$ **do**
      Sample $\boldsymbol{\xi}_b'^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
      $\mathbf{h}_b'^{(i)} \leftarrow f_\theta(G_b, \boldsymbol{\xi}_b'^{(i)})$
    end for
    $C_{b,ij} \leftarrow c_{G_b}(\mathbf{h}_b'^{(i)}, \mathbf{s}_b^{(j)}) \quad \forall i, j$
    $\pi_b^* \leftarrow \texttt{Hungarian}(C_b)$
    $\mathcal{L}_b \leftarrow \frac{1}{K} \sum_{i=1}^{K} c_{G_b}\left(\mathbf{h}_b'^{(i)}, \mathbf{s}_b^{(\pi_b^*(i))}\right)$
  end for
  $\mathcal{L}(\theta) \leftarrow \frac{1}{B} \sum_{b=1}^{B} \mathcal{L}_b$
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$
end for
return θ

Rearranging yields

$$\mathbb{E}_{\mathbf{x} \sim \widehat{Q}_t}\left[H_G(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x} \sim P_t}\left[H_G(\mathbf{x})\right] \geq \frac{\lambda}{2} \mathrm{MMD}_{k_G}^2\left(P_t, \widehat{Q}_t\right). \tag{41}$$

Finally, $\lambda > 0$ and squared MMD is nonnegative, so

$$\frac{\lambda}{2} \mathrm{MMD}_{k_G}^2\left(P_t, \widehat{Q}_t\right) \geq 0. \tag{42}$$

Combining the two inequalities proves Proposition 1. □

Table 5: Average reference objective values over the test sets used for computing relative gaps.

Problem	Dataset	Solver	Reference objective
MIS	RB-small	KaMIS	20.04
MIS	RB-large	KaMIS	42.83
MaxCl	RB-small	KaMIS	19.01
MaxCl	RB-large	KaMIS	40.12
MDS	BA-small	Gurobi	27.84
MDS	BA-large	Gurobi	103.81

B Additional Experimental Details

B.1 Reference Solution Generation

The relative gaps reported in the main experiments are computed from test-set average objective values with respect to reference solutions obtained using established combinatorial optimization solvers. Each test instance is solved independently under a fixed time limit. Since these solvers are terminated after a finite computation budget, the resulting solutions are treated as reference solutions rather than certified global optima.

For MIS, we use KaMIS (Lamm et al., 2016) to obtain a high-quality independent set for each test graph. For MaxCl, we apply KaMIS to the complement graph and interpret the resulting independent set as a clique in the original graph. For MDS, we use Gurobi (Gurobi Optimization, LLC, 2023) to solve the corresponding integer optimization problem. We allocate 300 seconds per instance on the small datasets and 600 seconds per instance on the large datasets for all three problems.

Table 5 summarizes the resulting average reference objective values over the test sets.

Let z_i and $z_{\mathrm{ref},i}$ denote the objective value obtained by a method and the corresponding reference objective value for test instance i, respectively. We define the test-set average objective values as

$$\bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i, \qquad \bar{z}_{\mathrm{ref}} = \frac{1}{N}\sum_{i=1}^{N} z_{\mathrm{ref},i}, \tag{43}$$

where N denotes the number of test instances.

For MIS and MaxCl, where larger objective values are better, the reported relative gap is defined as

$$\mathrm{Gap}(\bar{z}, \bar{z}_{\mathrm{ref}}) = \frac{\bar{z}_{\mathrm{ref}} - \bar{z}}{\bar{z}_{\mathrm{ref}}} \times 100\%. \tag{44}$$

For MDS, where smaller objective values are better, we use

$$\mathrm{Gap}(\bar{z}, \bar{z}_{\mathrm{ref}}) = \frac{\bar{z} - \bar{z}_{\mathrm{ref}}}{\bar{z}_{\mathrm{ref}}} \times 100\%. \tag{45}$$

B.2 Implementation Details

Generator architecture. BOLT uses a graph neural network to map node-wise Gaussian noise and the problem graph to unconstrained scalar priority scores. The generator consists of an input MLP, eight message-passing blocks with hidden dimension 64, and a two-layer node-wise MLP followed by a linear prediction head. Each message-passing block combines neighborhood aggregation, GraphNorm, residual node features, and LayerNorm. The same generator architecture is used across the three optimization problems, and the final prediction head produces one real-valued priority for each vertex.

Training configuration. We train BOLT with AdamW using a learning rate of 10^{-4} and weight decay 10^{-4}, together with a cosine learning-rate schedule. For each graph, the generator produces $K = 32$ initial samples, and SIPG generates $L = 8$ explored variants from each sample, resulting in a candidate set of $K(1 + L) = 288$ solutions. We use differential coefficient $F = 0.5$, exploration noise scale $\sigma = 0.03$, and proximity coefficient $\lambda = 10$.

Table 6: Ablation of the population exploration mechanism in SIPG. All variants use the same candidate budget. ↑ indicates higher is better and ↓ indicates lower is better.

Variant	MIS RB-small ↑	MaxCl RB-small ↑	MDS BA-small ↓
Generator samples only	18.27	17.44	29.21
Gaussian exploration	18.30	17.46	29.27
BOLT (Ours)	**18.39**	**17.47**	**29.12**

SIPG implementation. The MMD term is computed on decoded binary solutions. We standardize each solution across the node dimension and use an equally weighted mixture of four RBF kernels,

$$k_G(\mathbf{x}, \mathbf{y}) = \frac{1}{4} \sum_{b \in \{0.25, 0.5, 1, 2\}} \exp\left(-\frac{\|\phi_G(\mathbf{x}) - \phi_G(\mathbf{y})\|_2^2}{2nb^2} \right), \tag{46}$$

where $\phi_G(\mathbf{x})$ denotes the standardized representation of $\mathbf{x}$.

DBM implementation. Before evaluating the decoder-compatible costs, we standardize the priority scores independently for each generator output. All task-specific costs use squared hinge penalties with margin $\gamma = 0.25$ and are normalized by the number of vertices. For each graph, we sample $K = 32$ target solutions with replacement from the optimized target distribution and construct a $K \times K$ pairwise cost matrix against the generator outputs.

Hardware and runtime. All runtime measurements are conducted on a single NVIDIA RTX 4090 GPU. We sequentially process all 500 test instances with a batch size of one and report the total runtime over the complete test set. Runtime includes both the neural network evaluation and the problem-specific decoding procedure. All runtime measurements reported for methods evaluated by us are obtained under the same hardware environment.

C Additional Ablation Study

Population exploration. The main ablation study focuses on target construction and target transfer. Here we further examine the population-expansion mechanism used by SIPG. To isolate the effect of the differential exploration operator, we consider two alternatives while keeping the number of candidate solutions unchanged. *Generator samples only* replaces all explored variants with additional independent samples from the current generator. *Gaussian exploration* retains local perturbation but removes the differential term between generator outputs. The full BOLT variant uses both the differential direction and Gaussian perturbation as defined in equation 7.

As shown in Table 6, BOLT achieves the best performance across all three problems. Additional generator samples already provide a competitive candidate set, indicating that increasing the breadth of the available support is itself useful. Gaussian exploration further improves some settings, while the full differential exploration gives the most consistent results overall. These observations suggest that SIPG benefits primarily from exposing the target optimization to a broader set of solutions, with population differences providing additional guidance for support expansion.