PARALLEL TEMPERING FOR DIFFUSION-BASED COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete diffusion models have emerged as a powerful paradigm for solving combinatorial optimization (CO) problems on graphs by learning to sample high-quality solutions. A common inference-time approach is to generate multiple candidate solutions independently and return the best-performing sample, improving solution quality at the expense of an increase in computational cost. In this work, we introduce PT-Denoise, an inference-time procedure that allows these concurrent denoising trajectories to interact through parallel tempering, without requiring retraining or fine-tuning of the underlying denoiser. Our method assigns a temperature to each diffusion process and allows processes to swap temperatures based on their relative performance. This dynamically reallocates promising, low-energy trajectories to colder, more concentrated sampling regimes while allowing higher-energy states to escape local minima through randomized exploration. Experiments on canonical graph-structured CO problems show that our approach consistently improves the quality of the best solution found, while only adding minimal computational overhead.

1 INTRODUCTION

Combinatorial optimization (CO) problems over graphs, such as the maximum independent set (MIS), minimum dominating set (MDS), and maximum cut problems, are fundamental challenges in computer science and engineering. Many of these problems are NP-hard and are commonly addressed using approximation algorithms or problem-specific heuristics. Deep learning has emerged as an alternative paradigm, leveraging neural networks to learn problem-specific heuristics directly from data (Bello et al., 2016; Kool et al., 2019; Karalias & Loukas, 2020). Within this domain, generative models, and specifically diffusion models, have proven highly effective (Sun & Yang, 2023). They frame CO as sampling from a distribution over solutions and learn to iteratively denoise a uniform prior into high-quality CO solutions.

At inference time, a common strategy for improving solution quality is to generate multiple candidates independently. Here, a pretrained denoiser generates N candidate trajectories, and a non-learned decoder maps these candidates to feasible solutions. The final output is the best of the decoded solutions. Increasing N can improve the best solution found, but requires proportionally more denoiser evaluations. The corresponding increase in wall-clock time depends on batching and available parallel hardware. However, independent sampling at a fixed temperature does not use information about the relative progress of each trajectory in the ensemble to adapt each trajectory's sampling behavior. We therefore ask whether coordinating these trajectories can improve the best final solution under a fixed budget of denoiser evaluations.

To address this limitation, we introduce PT-Denoise, an inference-time procedure that coordinates concurrent denoising trajectories, which we call replicas, through parallel tempering. Our approach requires no retraining, fine-tuning, or architectural modifications to the underlying pretrained denoiser. We assign a temperature parameter to each replica slot, thereby controlling the level of randomness in its denoising steps. Our approach periodically evaluates intermediate states and proposes state exchanges between slots at neighboring temperatures. This mechanism favors assigning lower-energy states to colder, more concentrated sampling regimes and higher-energy states to hotter, more exploratory regimes, using intermediate energy as a surrogate for final decoded solution quality. Fig. 1 shows an overview of our approach.

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.

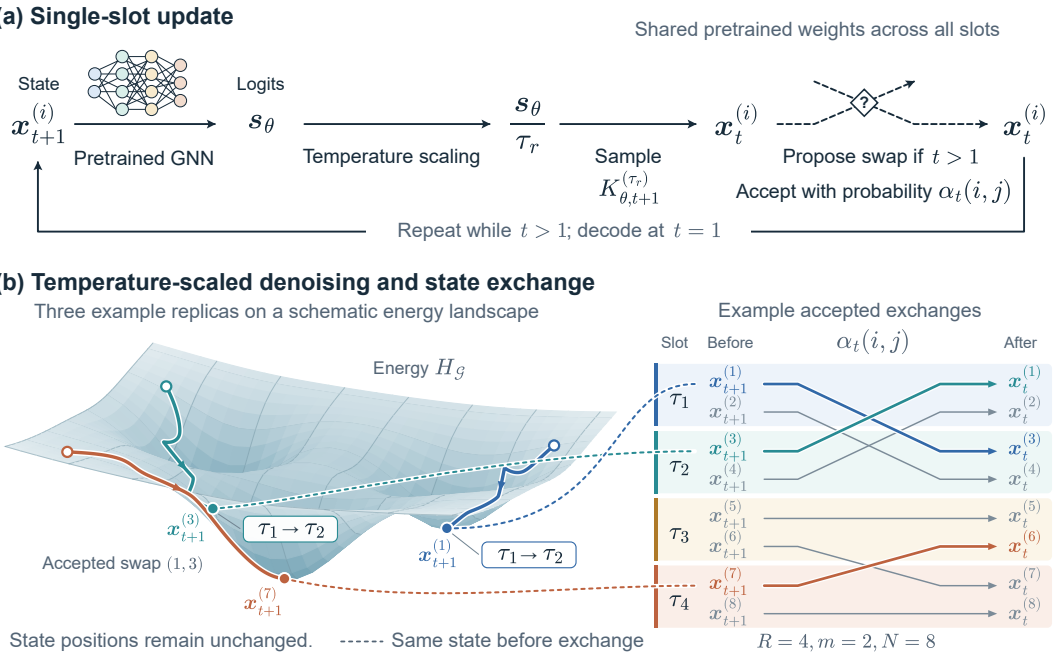


Figure 1: Overview of our approach. Top: One diffusion step of a single replica within an ensemble. We alternate between diffusion steps and replica swaps. Bottom: Swapping replicas between rungs on the temperature ladder. Neighboring temperature rungs are paired; in this example, the pairs are $(\tau_1, \tau_2), (\tau_3, \tau_4)$. A uniform one-to-one matching is sampled between the slots of paired temperatures. Swaps are proposed between matched slots and accepted with probability $\alpha_t(i, j)$.

Experiments on four graph-structured CO problems (MIS, MDS, maximum clique, and maximum cut) show that our approach improves mean best-of-N solution quality for both DiffUCO and SDDS, with small absolute runtime increases under the evaluated settings. We make the following contributions:

- *A coordinated inference protocol*: We propose PT-Denoise, a plug-and-play inference-time procedure that coordinates concurrent diffusion trajectories through state exchanges across temperatures without retraining the base model.
- *Decoupled population and temperature ladder*: Since trajectories have only a limited number of denoising steps in which to move across the temperature ladder, we maintain multiple trajectories per temperature rung. This allows increasing the population size without increasing the number of neighboring exchanges needed to traverse the ladder.
- *Empirical evaluation under a fixed denoiser budget*: We evaluate our approach across four graph-structured CO problems using two pretrained diffusion backbones, demonstrating improvements in mean solution quality and measuring the accompanying inference-time overhead.

2 Related Work

Neural combinatorial optimization. Early neural approaches learn constructive heuristics through reinforcement learning, using recurrent networks (Bello et al., 2016), graph embeddings (Khalil et al., 2017), attention-based architectures (Kool et al., 2019), or progressive decision-deferring schemes (Ahn et al., 2020). Beyond reinforcement learning, Erdős Goes Neural (Karalias & Loukas, 2020; Sun et al., 2022) combines unsupervised optimization of probabilistic relaxations with derandomized decoding, while Let the Flows Tell (Zhang et al., 2023) trains conditional GFlowNets to generate high-quality graph solutions. Li et al. (2018) use a learned model to guide a parallel tree search, a paradigm later critically evaluated by Böther et al. (2022). These approaches motivate learned solution generation; our focus is on coordinating trajectories from an existing diffusion solver at inference time.

Diffusion for combinatorial optimization. DIFUSCO (Sun & Yang, 2023) introduced graph-based diffusion solvers, while DiffUCO (Sanokowski et al., 2024) and SDDS (Sanokowski et al., 2025) developed methods for training discrete diffusion samplers against energy-based objectives without solution labels. DISCO (Zhao et al., 2025) improves solution quality and inference efficiency through a residual-based diffusion formulation. Closely related, GenSCO (Li et al., 2025b) treats generation as a search operator, alternating solution disruption and learned generation; its search loop is supported by a dedicated solution-enhancement training procedure. Our approach instead couples concurrent denoising trajectories through temperature exchanges, using an existing pretrained denoiser without retraining or fine-tuning.

Inference-time steering and interacting particles. Particle Guidance (Corso et al., 2024) couples diffusion trajectories through a joint potential to encourage diversity. Feynman–Kac steering (Singhal et al., 2025) resamples particles according to intermediate reward potentials, while soft value-based decoding (Li et al., 2025a) guides generation using estimates of future reward. Related sequential Monte Carlo methods derive importance-sampling corrections and proposal mechanisms to control discrete diffusion models (Lee et al., 2025; Ou et al., 2026). Our method uses energy-dependent exchanges between temperature groups: each exchange preserves the current collection of states while changing the sampling temperatures governing their subsequent evolution.

Parallel tempering and generative sampling. Parallel tempering coordinates replicas through exchanges across a temperature ladder (Swendsen & Wang, 1986; Geyer, 1991; Hukushima & Nemoto, 1996). Ensemble-based implementations and adaptive ladder selection were studied by Vousden et al. (2016). Accelerated Parallel Tempering (Zhang et al., 2026) incorporates neural transports, including flows and diffusion processes, into parallel tempering while preserving asymptotic consistency under its stated assumptions. Source Parallel Tempering (Wang et al., 2026) uses parallel tempering to guide flow-based models, but operates on continuous latent priors in the source space before generation. In contrast, our method couples discrete denoising trajectories across diffusion timesteps. CREPE (He et al., 2026) applies parallel tempering to inference-time control of pretrained diffusion models, including discrete masked diffusion, using replicas at different diffusion times. Our approach exchanges states at the same denoising time across different logit temperatures and assigns multiple replicas to each rung, allowing population size to increase without extending the ladder. We use this mechanism for finite-budget optimization. Temperature-scaled denoising generally does not preserve Boltzmann distributions, so the overall procedure does not inherit the classical parallel-tempering sampling guarantees.

3 BACKGROUND

Combinatorial optimization. A combinatorial optimization (CO) problem asks us to minimize an objective function $f : \mathcal{F} \to \mathbb{R}$ on a discrete set $\mathcal{F}$. Without loss of generality, maximization problems can be converted to minimization problems by changing the sign. Many CO problems admit natural graph representations. In particular, much of the machine learning literature for CO focuses on node subset problems (Karalias & Loukas, 2020; Zhang et al., 2023). Given a graph $\mathcal{G} = (V, E)$, the goal is to minimize an objective function $f_{\mathcal{G}} : \mathcal{F}_{\mathcal{G}} \to \mathbb{R}$ over subsets of nodes, where $\mathcal{F}_{\mathcal{G}} \subseteq \{0, 1\}^{|V|}$. A relaxed energy function $H_{\mathcal{G}} : [0, 1]^{|V|} \to \mathbb{R}$ is often used to evaluate approximate continuous solutions or solutions that do not meet the constraints. It usually consists of a term that relaxes the discrete objective function and one that softly enforces the CO problem's constraints, and is designed to match the objective function $f_{\mathcal{G}}$ on $\mathcal{F}_{\mathcal{G}}$ (Karalias & Loukas, 2020).

Examples for frequently studied CO problems are as follows. The *maximum independent set (MIS)* problem asks us to find the largest subset of nodes $S \subseteq V$ in which no pair of nodes are neighbors. In the *minimum dominating set (MDS)* problem, the goal is to find the smallest subset $S \subseteq V$ such that each node in V is either in S or has a neighbor in S. The *maximum cut* problem asks us to find a subset $S \subseteq V$ that maximizes the number of edges between S and $V \setminus S$. Finally, there is the *maximum clique* problem, where the goal is to find the largest clique, i.e. a subset of nodes where each node is connected to each other node in the set.

Diffusion for CO. Diffusion models for CO frame solving CO problems over graphs as generative modeling over the solution space $\mathcal{X} = \{0, \dots, k-1\}^{|V|}$ disregarding constraints, where $k = 2$ for node subset problems (Sun & Yang, 2023). The diffusion model can be trained to approximately

sample from a Boltzmann distribution based on the CO problem's energy function $H_{\mathcal{G}}$:

$$p^*_{\mathcal{G}}(\boldsymbol{x}) = \frac{1}{Z(\mathcal{G})} \exp\big(-\beta_B H_{\mathcal{G}}(\boldsymbol{x})\big), \qquad Z(\mathcal{G}) = \sum_{\boldsymbol{x}\in\mathcal{X}} \exp\big(-\beta_B H_{\mathcal{G}}(\boldsymbol{x})\big),$$

where $\beta_B > 0$ is an inverse temperature (Sanokowski et al., 2024; 2025). This assigns the highest probabilities to solutions $\boldsymbol{x}$ with low energy.

Discrete diffusion connects the target distribution $p^*_{\mathcal{G}}$ to a simple reference distribution through a fixed corruption process and a learned denoising process. The *forward process* $q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)$ is a non-trainable Markov chain that progressively corrupts an initial assignment $\boldsymbol{x}_0$. For discrete variables, each variable is resampled from a categorical transition distribution determined by its current value and the timestep. A sampled value may coincide with the current value or replace it with another of the k possible categories. The noise schedule progressively removes information about $\boldsymbol{x}_0$, making the terminal distribution approximately a factorized uniform prior,

$$p(\boldsymbol{x}_T) = \prod_{v\in V} \mathrm{Cat}\left(x_{T,v}; \frac{1}{k}\mathbf{1}\right).$$

The *reverse process* learns to undo this corruption. Starting from an assignment sampled from $p(\boldsymbol{x}_T)$, it applies learned categorical transitions conditioned on the graph $\mathcal{G}$:

$$p_\theta(\boldsymbol{x}_{0:T} \mid \mathcal{G}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \mathcal{G}). \tag{1}$$

The parameters θ are trained so that the resulting marginal distribution $p_\theta(\boldsymbol{x}_0 \mid \mathcal{G})$ approximates the target distribution $p^*_{\mathcal{G}}$ over high-quality CO solutions.

At inference time, the forward process is discarded entirely. Generation proceeds by sampling initial states from the prior $p(\boldsymbol{x}_T)$ and sequentially evaluating the transitions $p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ down to $t = 0$. The reverse transition can be parameterized via a graph neural network (GNN) $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t, \mathcal{G}) \in \mathbb{R}^{|V|\times k}$, which predicts node-level logits based on the intermediate state $\boldsymbol{x}_t$, timestep t, and problem graph $\mathcal{G}$:

$$K_{\theta,t}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \mathcal{G}) = \prod_{v\in V} \mathrm{Cat}\Big(\boldsymbol{x}_{t-1,v};\ \mathrm{softmax}\big(\boldsymbol{s}_{\theta,v}(\boldsymbol{x}_t, t, \mathcal{G})\big)\Big).$$

While the output of the reverse process $\boldsymbol{x}_0 \in \mathcal{X}$ is discrete, it is not guaranteed to meet the CO problem's constraints. To address this, the logits at the final denoising step $\boldsymbol{s}_\theta(\boldsymbol{x}_1, 1, \mathcal{G})$ are fed into a *decoder* $\boldsymbol{d}_{\mathcal{G}} : \mathbb{R}^{|V|\times k} \to \mathcal{F}_{\mathcal{G}}$, an algorithm that constructs a feasible solution to the CO problem.

Parallel tempering. Parallel tempering, also known as replica exchange Markov chain Monte Carlo (MCMC) sampling (Swendsen & Wang, 1986; Geyer, 1991), is an algorithm used in statistical physics and computational chemistry to sample from multimodal distributions $p^*(\boldsymbol{x}) \propto \exp(-H(\boldsymbol{x})/\tau)$, where $\tau \in \mathbb{R}$ is a temperature and H is an energy function.

Parallel tempering maintains a set of N replicas operating at distinct temperatures along a temperature ladder $\mathcal{T} = \{\tau_1 < \tau_2 < \cdots < \tau_N\}$, where typically $\tau_1 = 1$. Each replica $i \in \{1, \ldots, N\}$ evolves its sample $\boldsymbol{x}^{(i)}$ independently, e.g. according to a standard MCMC algorithm. Replicas only interact periodically at predetermined intervals by swapping temperatures: For each pair of neighboring temperatures τ_i, τ_{i+1}, replicas $i,\ j = i+1$ swap according to the Metropolis–Hastings criterion with probability

$$\alpha(i,j) = \min\Big\{1,\ \exp\big((\beta_i - \beta_j)(H(\boldsymbol{x}^{(i)}) - H(\boldsymbol{x}^{(j)}))\big)\Big\}, \tag{2}$$

where $\beta_i = 1/\tau_i$. High-temperature replicas explore broad regions of state space and can cross high-energy barriers easily, while low-temperature replicas exploit local minima.

Conflicts can arise if swaps between replica pairs $(i, i+1)$ and $(i+1, i+2)$ are proposed in the same iteration. To address this, a common approach is to group replica pairs $(i, i+1)$ into pairs with even i and pairs with odd i, and to alternate between proposing swaps using even and odd pairs (Hukushima & Nemoto, 1996).

4 METHOD

We consider a CO problem on a graph $\mathcal{G}$, with feasible solutions $\mathcal{F}_\mathcal{G} \subseteq \mathcal{X}$ and objective $f_\mathcal{G} : \mathcal{F}_\mathcal{G} \to \mathbb{R}$ to be minimized. A pretrained discrete denoiser and a non-learned decoder together define a stochastic procedure for generating feasible solutions. A common inference strategy (Sun & Yang, 2023; Sanokowski et al., 2024; 2025) is to generate N candidate solutions $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)} \in \mathcal{F}_\mathcal{G}$ independently and return

$$\boldsymbol{x}^* \in \operatorname*{arg\,min}_{\boldsymbol{x} \in \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}\}} f_\mathcal{G}(\boldsymbol{x}).$$

Increasing N can improve the best solution found, but also increases the computational cost. We therefore ask whether coordinating a fixed number of denoising trajectories can improve their best final solution.

We propose PT-Denoise, an inference-time procedure that couples concurrent denoising trajectories through temperature exchanges. Different sampling temperatures control the randomness of the denoising transitions, while energy-dependent exchanges reassign intermediate states to these temperatures. The procedure uses the same pretrained denoiser throughout and requires no retraining or fine-tuning. Fig. 1 shows an overview of our approach.

Our objective is to improve

$$\mathbb{E}\left[\min_{i \in \{1,\dots,N\}} f_\mathcal{G}\big(\boldsymbol{x}^{(i)}\big)\right]$$

under a fixed inference budget. We use parallel tempering as an optimization mechanism. Importantly, we do not assume that the resulting denoising trajectories are equilibrium samples from tempered Boltzmann distributions, as this is not required for the problem of finding high-quality solutions.

Temperature-scaled denoising. Let $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t, \mathcal{G}) \in \mathbb{R}^{|V| \times k}$ denote the logits of a pretrained reverse transition. For a sampling temperature $\tau > 0$, define

$$K^{(\tau)}_{\theta,t}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \mathcal{G}) = \prod_{v \in V} \mathrm{Cat}\left(\boldsymbol{x}_{t-1,v};\ \mathrm{softmax}\left(\frac{1}{\tau}\boldsymbol{s}_{\theta,v}(\boldsymbol{x}_t, t, \mathcal{G})\right)\right). \tag{3}$$

At $\tau = 1$, this recovers the original reverse transition. Larger temperatures flatten the categorical distributions, whereas smaller temperatures concentrate probability on the denoiser's preferred assignments. Note that the sampling temperature τ used here is separate from the inverse temperature β_B in the Boltzmann distribution used to train the model.

Temperature scaling acts on the transition distribution:

$$K^{(\tau)}_{\theta,t}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \mathcal{G}) \propto \left[K^{(1)}_{\theta,t}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \mathcal{G})\right]^{1/\tau}, \tag{4}$$

where the normalizing constant depends on $\boldsymbol{x}$, t, and τ. Consequently, scaling the transition logits does not generally imply that intermediate or terminal states follow a Boltzmann distribution with temperature τ.

Concurrent denoising with parallel tempering. We maintain N replica slots. Each slot i has an assigned temperature τ_i and contains an intermediate state $\boldsymbol{x}_t^{(i)}$. Here, temperature subscripts index slots, and multiple slots may share the same temperature. All replicas follow the same denoising schedule. Starting from independent samples of the original prior, each slot generates its next state using

$$\boldsymbol{x}_t^{(i)} \sim K^{(\tau_i)}_{\theta,t+1}\big(\cdot \mid \boldsymbol{x}_{t+1}^{(i)}, \mathcal{G}\big). \tag{5}$$

After a denoising transition, we evaluate each state using the energy function $H_\mathcal{G}$. This acts as a surrogate for the final solution quality at $t = 0$ after decoding, which we cannot measure directly.

For two slots i and j at neighboring temperatures, we propose exchanging their states and accept with probability

$$\alpha_t(i,j) = \min\left\{1,\ \exp\left[(\beta_i - \beta_j)\left(H_\mathcal{G}(\boldsymbol{x}_t^{(i)}) - H_\mathcal{G}(\boldsymbol{x}_t^{(j)})\right)\right]\right\}, \qquad \beta_i = \frac{1}{\tau_i}. \tag{6}$$

Algorithm 1 Concurrent denoising with temperature exchange.

Inputs: Pretrained denoiser $\boldsymbol{s}_\theta$, graph $\mathcal{G}$, decoder $\boldsymbol{d}_\mathcal{G}$, prior p_T, temperature ladder $0 < \tau_1 < \cdots < \tau_R$, replicas per rung m, diffusion steps $T \geq 2$, energy function $H_\mathcal{G}$, objective function $f_\mathcal{G}$.
Setup: $N \leftarrow Rm$; partition replica slots $\{1, \ldots, N\}$ into disjoint $\mathcal{I}_1, \ldots, \mathcal{I}_R$ with $|\mathcal{I}_r| = m$. Set $\beta_i \leftarrow \tau_r^{-1}$ for each $r \in \{1, \ldots, R\}$ and $i \in \mathcal{I}_r$.

(a) Denoising and selection

$\boldsymbol{x}_T^{(i)} \overset{\text{iid}}{\sim} p_T$, $i = 1, \ldots, N$; $q \leftarrow 0$
for $t = T-1, \ldots, 1$ **do**
  for $r = 1, \ldots, R$, $i \in \mathcal{I}_r$ **do**
    $\boldsymbol{x}_t^{(i)} \sim K_{\theta,t+1}^{(\tau_r)}(\cdot \mid \boldsymbol{x}_{t+1}^{(i)}, \mathcal{G})$
  if $t > 1$ **then**
    $\boldsymbol{X}_t \leftarrow (\boldsymbol{x}_t^{(i)})_{i=1}^N$
    $\boldsymbol{X}_t \leftarrow \textsc{Exchange}(\boldsymbol{X}_t, q)$
    $q \leftarrow q + 1$
for $i = 1, \ldots, N$ **do**
  $\boldsymbol{x}^{(i)} \leftarrow \boldsymbol{d}_\mathcal{G}\Big(\boldsymbol{s}_\theta(\boldsymbol{x}_1^{(i)}, 1, \mathcal{G})\Big)$
$i^\star \in \arg\min_{i \in \{1,\ldots,N\}} f_\mathcal{G}(\boldsymbol{x}^{(i)})$
return $\boldsymbol{x}^{(i^\star)}$

(b) $\textsc{Exchange}(\boldsymbol{X}_t, q)$

$\mathcal{P} \leftarrow \mathcal{P}^{\text{odd}}$ if q is even, otherwise $\mathcal{P} \leftarrow \mathcal{P}^{\text{even}}$
for all $(r, s) \in \mathcal{P}$ **do**
  $\mathcal{M} \leftarrow \textsc{UniformMatch}(\mathcal{I}_r, \mathcal{I}_s)$
  for all $(i, j) \in \mathcal{M}$ **do**
    $\Delta \leftarrow (\beta_i - \beta_j)\big[H_\mathcal{G}(\boldsymbol{x}_t^{(i)}) - H_\mathcal{G}(\boldsymbol{x}_t^{(j)})\big]$
    $u \sim \text{Uniform}(0, 1)$
    if $\log u < \min\{0, \Delta\}$ **then**
      $(\boldsymbol{x}_t^{(i)}, \boldsymbol{x}_t^{(j)}) \leftarrow (\boldsymbol{x}_t^{(j)}, \boldsymbol{x}_t^{(i)})$
$\boldsymbol{X}_t = (\boldsymbol{x}_t^{(i)})_{i=1}^N$
return $\boldsymbol{X}_t$

UNIFORMMATCH draws a uniformly random one-to-one matching. $\mathcal{P}^{\text{odd}}$ and $\mathcal{P}^{\text{even}}$ are the rung pairings defined in Eq. (7). Sampling: Eqs. (3), (5); exchange: Eq. (6); decoding: Eq. (8).

Temperatures remain attached to slots, so an accepted exchange changes the temperature used for the subsequent evolution of each exchanged state. If $\tau_i < \tau_j$, an exchange that moves the lower-scoring, i.e. more promising, state into slot i is always accepted. The reverse exchange may also be accepted, allowing states to move between more concentrated and more randomized denoising transitions.

An exchange preserves the current collection of states, including its best score. Its effect is therefore on subsequent denoising: It changes the sampling temperature each state encounters. This differs from resampling, which can duplicate some states and discard others (Singhal et al., 2025).

Decoupling the temperature ladder from population size. We use a ladder of R distinct temperatures, $\mathcal{T} = \{\tau_1 < \cdots < \tau_R\}$, and assign m replica slots to each temperature, giving $N = Rm$ replicas in total. Let $\mathcal{I}_r$ denote the slots assigned to rung r. In this grouped notation, temperature subscripts index ladder rungs rather than slots; for each $i \in \mathcal{I}_r$, the slot's inverse temperature is $\beta_i = 1/\tau_r$. This construction follows the ensemble-tempering principle, maintaining multiple states at each temperature (Vousden et al., 2016).

This choice is motivated by the fact that using one replica per rung ties population size to the number of temperatures, potentially causing multiple problems. Since a state can move at most one rung per exchange step, a longer ladder can limit movement between its endpoints within a short denoising horizon. Allowing R and N to be adjusted separately allows us to increase the number of candidates without increasing the number of rungs they must traverse for the same temperature change. Moreover, increasing the number of temperatures means either increasing the maximum temperature, or decreasing the distance between temperatures. In the first case, larger parts at the top end of the temperature ladder turn to pure noise, ignoring the diffusion model's learned distribution. In the second case, if the difference between neighboring temperatures becomes too small, the acceptance probability according to the Metropolis–Hastings criterion approaches 1, meaning that neighboring replicas will almost always swap regardless of energy. These problems are resolved by assigning multiple replica slots to each temperature.

To avoid conflicting exchanges, we alternate between the disjoint rung pairings

$$\mathcal{P}^{\text{odd}} = \{(1,2), (3,4), \ldots\}, \qquad \mathcal{P}^{\text{even}} = \{(2,3), (4,5), \ldots\}, \tag{7}$$

retaining only pairs with indices in $\{1, \ldots, R\}$. For each active pair (r, s), we sample a uniform one-to-one matching between $\mathcal{I}_r$ and $\mathcal{I}_s$ and apply Eq. (6) to each matched pair. Because pairs are disjoint, their exchanges can be evaluated in parallel.

Table 1: Results on MIS. Top: diffusion methods. Center: learned non-diffusion methods. Bottom: non-learned heuristics.

	RB Small			RB Large		
Method	IS Size ↑	Relative Gap ↓	Time	IS Size ↑	Relative Gap ↓	Time
DiffUCO	19.74±0.03	1.79%±0.15	12ms	41.65±0.03	3.48%±0.08	303ms
DiffUCO + Temp. Ladder	19.93±0.04	0.83%±0.18	12ms	41.66±0.02	3.45%±0.05	303ms
DiffUCO + PT-Denoise (ours)	20.01±0.01	0.47%±0.03	13ms	41.68±0.02	3.40%±0.06	303ms
SDDS	19.95±0.02	0.72%±0.12	12ms	41.97±0.07	2.73%±0.16	304ms
SDDS + Temp. Ladder	20.01±0.02	0.44%±0.12	12ms	**41.99±0.02**	**2.69%±0.09**	303ms
SDDS + PT-Denoise (ours)	**20.05±0.02**	**0.26%±0.08**	13ms	**42.01±0.03**	**2.64%±0.08**	304ms
LwtD	19.01	5.42%	154ms	32.32	25.10%	906ms
INTEL	18.47	8.11%	1.57s	34.47	20.12%	2.43s
DGL	17.36	13.61%	1.53s	34.50	20.05%	2.87s
LTFT	19.18	4.57%	64ms	37.48	13.14%	524ms
KaMIS	20.10	—	10.10s	43.15	—	14.83s
Gurobi	19.98	0.60%	5.71s	40.90	5.21%	15.65s

Table 2: Results on MDS. Top: diffusion methods. Center: learned non-diffusion methods. Bottom: non-learned heuristics.

	BA Small			BA Large		
Method	DS Size ↓	Relative Gap ↓	Time	DS Size ↓	Relative Gap ↓	Time
DiffUCO	28.01±0.05	0.43%±0.18	10ms	104.01±0.02	0.00%±0.02	56ms
DiffUCO + PT-Denoise (ours)	27.91±0.01	0.08%±0.04	11ms	103.92±0.01	-0.08%±0.01	56ms
SDDS	27.93±0.01	0.14%±0.04	10ms	103.90±0.02	-0.11%±0.02	56ms
SDDS + PT-Denoise (ours)	**27.90±0.00**	**0.02%±0.01**	11ms	**103.84±0.03**	**-0.16%±0.03**	57ms
EGN	30.68	10.00%	120ms	116.76	12.26%	472ms
EGN-Anneal	29.24	4.84%	122ms	111.50	7.20%	470ms
LTFT	28.61	2.58%	280ms	110.28	6.03%	3.86s
Gurobi	27.89	—	214ms	104.01	—	1.66s

Final decoding and computational cost. After reaching $t = 1$, each replica is decoded using

$$\boldsymbol{x}^{(i)} = \boldsymbol{d}_{\mathcal{G}}\left(\boldsymbol{s}_\theta(\boldsymbol{x}_1^{(i)}, 1, \mathcal{G})\right) \in \mathcal{F}_{\mathcal{G}}, \tag{8}$$

where $\boldsymbol{d}_{\mathcal{G}}$ is the same feasibility-ensuring decoder used by the base diffusion model. We return the candidate with the smallest $f_{\mathcal{G}}$. With $T-1$ sampled transitions and one final denoiser evaluation for decoding, the procedure uses NT per-replica denoiser evaluations, matching independent generation with the same N and T. The additional cost consists of intermediate score evaluations, random matching, and state exchanges. Each exchange sweep proposes at most $N/2$ swaps. An exchange immediately before a common deterministic decoder only permutes the final collection of candidates and cannot change the returned objective value. It can therefore be omitted.

Exchanges assign intermediate states to different temperatures for subsequent denoising. The acceptance rule favors moving lower-energy states to colder temperatures, where sampling is more concentrated. Since lower energy corresponds to better objective values, this aims to refine promising states while allowing others to explore through more randomized transitions. An exchange itself leaves the current candidates unchanged and its benefit depends on whether intermediate energy predicts final decoded quality and whether the temperature reassignment improves subsequent denoising. Because temperature-scaled denoising does not generally preserve Boltzmann distributions, we treat parallel tempering as an optimization heuristic without claiming the sampling guarantees of classical parallel tempering. We summarize the complete denoising procedure in Alg. 1.

5 Experiments

We evaluate whether coordinating diffusion trajectories through temperature exchange improves solution quality under a fixed budget of denoiser evaluations. Our experiments cover four graph CO problems in MIS, MDS, maximum cut, and maximum clique and compare PT-Denoise with

Table 3: Results on maximum cut. Top: diffusion methods. Center: learned non-diffusion methods. Bottom: non-learned heuristics.

	BA Small			BA Large		
Method	Cut Size ↑	Relative Gap ↓	Time	Cut Size ↑	Relative Gap ↓	Time
DiffUCO	734.18±0.08	-0.45%±0.01	9ms	2966.36±0.17	-0.75%±0.01	37ms
DiffUCO + PT-Denoise (ours)	**734.55±0.01**	**-0.50%±0.00**	10ms	**2966.64±0.18**	**-0.76%±0.00**	38ms
SDDS	734.29±0.10	-0.47%±0.01	9ms	2966.29±0.26	-0.74%±0.01	38ms
SDDS + PT-Denoise (ours)	**734.56±0.05**	**-0.50%±0.01**	10ms	**2966.89±0.30**	**-0.76%±0.01**	38ms
EGN	693.45	5.12%	92ms	2870.34	2.51%	338ms
EGN-Anneal	696.73	4.67%	90ms	2863.23	2.76%	336ms
LTFT	704.30	3.64%	354ms	2864.61	2.71%	2.56s
Gurobi	730.87	—	1.57s	2944.38	—	7.86s

independent sampling using the same diffusion backbones, alongside established CO baselines. We report both solution quality and inference time to assess the gains from coordination and its computational overhead. We build upon DiffUCO (Sanokowski et al., 2024) and SDDS (Sanokowski et al., 2025), using the same model architectures and trained weights. For PT-Denoise, we use $N = 100$ replicas and 18 diffusion steps in our experiments. To allow for a fair comparison, all diffusion baselines also sample 100 solutions using the same number of diffusion steps, and the best result is reported. We use conditional expectation (Sanokowski et al., 2024) as the decoder for PT-Denoise and all diffusion baselines.

CO problems and datasets. Our evaluation closely follows the protocol established by Zhang et al. (2023); Sanokowski et al. (2024; 2025). We evaluate on four CO problems: MIS, MDS, maximum cut, and maximum clique. We use RB graphs (Xu et al., 2005) for MIS and max clique, and Barabási–Albert (BA) graphs (Barabási & Albert, 1999) for MDS and max cut. For each dataset, we evaluate on a small variant with 200-300 nodes and a large variant with 800-1200 nodes. For max clique, we only evaluate on RB small, following Sanokowski et al. (2024; 2025). All test sets contain 1,000 instances. The energy functions used for each CO problem are detailed in Appendix A.

Table 4: Results on maximum clique. Top: diffusion methods. Center: learned non-diffusion methods. Bottom: non-learned heuristics.

	RB Small		
Method	Clique Size ↑	Relative Gap ↓	Time
DiffUCO	18.24±0.14	4.27%±0.74	23ms
DiffUCO + PT-Denoise (ours)	18.91±0.13	0.75%±0.67	24ms
SDDS	18.98±0.00	0.37%±0.02	23ms
SDDS + PT-Denoise (ours)	**18.99±0.00**	**0.33%±0.03**	24ms
EGN	12.02	36.90%	82ms
EGN-Anneal	14.10	25.98%	82ms
LTFT	16.24	14.75%	84ms
Gurobi	19.05	—	230ms

Baselines. Our primary diffusion baselines are SDDS (Sanokowski et al., 2025) and DiffUCO (Sanokowski et al., 2024). On MIS, we compare against learned unsupervised baselines *let the flows tell* (LTFT) (Zhang et al., 2023) and *learning what to defer* (LwtD) (Ahn et al., 2020), supervised baselines DGL (Böther et al., 2022) and INTEL (Li et al., 2018), as well as the non-learned mixed-integer program solver Gurobi (Gurobi Optimization, LLC, 2026) and the non-learned heuristic solver KaMIS (Lamm et al., 2017). On MDS, max cut, and max clique, we compare with learned unsupervised baselines LTFT, *Erdős goes neural* (EGN) (Karalias & Loukas, 2020), and an annealed variant EGN-Anneal (Sun et al., 2022), as well as Gurobi. In our tables, results for all non-diffusion baselines are reported as in (Zhang et al., 2023). We note that Gurobi and KaMIS often have substantially longer runtimes than the other methods and should therefore not be compared directly in those cases.

Metrics. We report the value of the objective function $f_{\mathcal{G}}$, i.e. the size of the solution set (for MIS, MDS, and max clique) or resulting cut (for max cut). Mean and standard deviation are calculated over three published reference models[1] trained with different seeds. We also measure the execution time, averaged per CO problem instance. Finally, we report the relative gap to the best non-learned baseline. The relative gap (in %) is calculated as $(f_{\mathcal{G}}(\boldsymbol{x}) - f_{\mathcal{G}}(\boldsymbol{x}^*))/f_{\mathcal{G}}(\boldsymbol{x}^*) \cdot 100$ for minimization problems and $(f_{\mathcal{G}}(\boldsymbol{x}^*) - f_{\mathcal{G}}(\boldsymbol{x}))/f_{\mathcal{G}}(\boldsymbol{x}^*) \cdot 100$ for maximization problems. Here, $\boldsymbol{x}$ is the learned

[1] See Appendix B for details

method's predicted solution and $\boldsymbol{x}^*$ is the solution from the best non-learned baseline. Note that $\boldsymbol{x}^*$ is not necessarily the optimal solution, and the gap becomes negative if $\boldsymbol{x}$ is better than $\boldsymbol{x}^*$.

Results. We summarize our main empirical results in Tabs. 1 to 4. The best result among learned methods as well as each result for which the respective mean lies within one standard deviation of the best method are marked in bold. Across all four CO problems and on both small and large graphs, incorporating PT-Denoise consistently improves the solution quality of the base diffusion models, DiffUCO and SDDS. SDDS + PT-Denoise shows the best mean solution quality on all CO problems and datasets among evaluated learned methods.

On MIS (Tab. 1), PT-Denoise substantially reduces the relative gap to the best non-learned baseline. For instance, applying our method to DiffUCO reduces the relative gap from 1.79% to 0.47% on RB small, while SDDS + PT-Denoise achieves the best overall diffusion performance with a relative gap of 0.26%. The improvement persists on RB large, though to a much smaller extent. In addition to this, the augmented diffusion models substantially outperform all learned non-diffusion baselines and outperform Gurobi, approaching the performance of KaMIS while being orders of magnitude faster.

As an ablation, we also ran DiffUCO and SDDS on MIS using only a ladder of different temperatures, but without temperature swaps. For both diffusion models and both problem sizes, this performs better than a single temperature but worse than PT-Denoise. For MDS (Tab. 2), augmenting the base models with PT-Denoise strictly improves the dominating set sizes. Similarly, on max cut (Tab. 3), PT-Denoise increases the cut size across the board, but to a much lesser extent. On max clique (Tab. 4), where we only evaluate on small graphs, applying our method to DiffUCO reduces the relative gap from 4.27% to 0.75%, and applying it to SDDS results in the best performance overall by a small margin. In all of our experiments, using PT-Denoise adds at most 1ms per CO problem instance to the inference time. This means that any solution quality gained by using our method comes at very little additional computational cost.

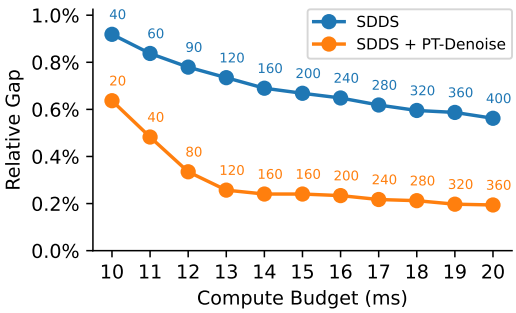


Figure 2: The best relative gap achievable on MIS, RB small within each given per-instance compute budget. The numbers next to each data point indicate how many replicas can be processed within the compute budget.

Fig. 2 visualizes the solution quality achievable within a given per-instance time budget. This only uses replica counts divisible by 10, since we use 10 temperatures for PT-Denoise. Relative gaps w.r.t. KaMIS and inference times are averaged over three trained models. At every compute budget, using PT-Denoise reduces the relative gap.

6 Conclusion

We introduced PT-Denoise, a novel inference-time procedure that enhances discrete diffusion models for graph-structured CO by coupling concurrent denoising trajectories through parallel tempering. By assigning a temperature to each diffusion process and allowing them to swap temperatures, our approach enables both exploration and local refinement without requiring retraining or fine-tuning of the base model. Our empirical evaluations across four canonical CO problems demonstrate that our approach consistently improves the solution quality of state-of-the-art diffusion models, while incurring minimal computational overhead.

Limitations and future work. While this work focuses explicitly on CO, the parallel tempering approach is general and could be applied to other diffusion domains. Consequently, adapting PT-Denoise to other application areas remains an open direction for future research. Our approach relies on a relaxed energy function $H_{\mathcal{G}}$ to evaluate particles and guide temperature swaps. While $H_{\mathcal{G}}$ serves as an effective heuristic surrogate, it may not perfectly correlate with the final decoded solution quality at early denoising timesteps. A potential solution could lie in training a lightweight neural network to predict the final solution quality based on intermediate states.

AI USE STATEMENT

In this work, we used large language models (LLMs) to help implement our method, write the scripts used to generate plots visualizing experimental results, and to suggest and improve formulations in the manuscript. LLMs were not used to develop theoretical models, conceptual frameworks or mathematical claims, to propose or refine hypotheses, to design or provide feedback on research methodology or experiments, to support qualitative and thematic data analysis, or to interpret results. All synthetic datasets were created using non-learned algorithms, with no LLM-assisted cleaning or reformatting. The remaining required disclosure tasks—assisting in the writing of proofs, providing critical ingredients for proving mathematical claims, and assisting with translation—are not applicable to this work. All LLM-generated code and text has been manually reviewed by the authors. We take responsibility for the final content of this work, including the code and text produced with the aid of LLMs.

REPRODUCIBILITY STATEMENT

The code required to reproduce the experiments was submitted as supplementary material, and will be made public once the paper is accepted. This includes scripts for re-generating the datasets. We list hyperparameters and checkpoints used in the experiments in Appendix B, and the hardware we used in Appendix C.

REFERENCES

Sungsoo Ahn, Younggyo Seo, and Jinwoo Shin. Learning what to defer for maximum independent sets. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 134–144. PMLR, Jul 2020.

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

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

Maximilian Böther, Otto Kißig, Martin Taraz, Sarel Cohen, Karen Seidel, and Tobias Friedrich. What's wrong with deep learning in tree search for combinatorial optimization. In *International Conference on Learning Representations*, 2022.

Gabriele Corso, Yilun Xu, Valentin De Bortoli, Regina Barzilay, and Tommi Jaakkola. Particle guidance: non-i.i.d. diverse sampling with diffusion models. In *International Conference on Learning Representations*, pp. 22480–22507, 2024.

David J Earl and Michael W Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

Charles J Geyer. Markov chain monte carlo maximum likelihood. In *Computing science and statistics: Proceedings of the 23rd Symposium on the Interface*, pp. 156–163. Interface Foundation of North America, 1991.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2026.

Jiajun He, Paul Jeha, Peter Potaptchik, Leo Zhang, José Miguel Hernández Lobato, Yuanqi Du, Saifuddin Syed, and Francisco Vargas. Crepe: Controlling diffusion with replica exchange. In *International Conference on Learning Representations*, pp. 99464–99494, 2026.

Koji Hukushima and Koji Nemoto. Exchange monte carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608, 1996.

Nikolaos Karalias and Andreas Loukas. Erdős goes neural: An unsupervised learning framework for combinatorial optimization on graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6659–6672. Curran Associates, Inc., 2020.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

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

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

Cheuk Kit Lee, Paul Jeha, Jes Frellsen, Pietro Lio, Michael Samuel Albergo, and Francisco Vargas. Debiasing guidance for discrete diffusion with sequential Monte Carlo. *arXiv preprint arXiv:2502.06079*, 2025.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 95507–95545. Curran Associates, Inc., 2025a.

Yang Li, Lvda Chen, Haonan Wang, Runzhong Wang, and Junchi Yan. Generation as search operator for test-time scaling of diffusion-based combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 127168–127196. Curran Associates, Inc., 2025b.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Zijing Ou, Chinmay Pani, and Yingzhen Li. Inference-time scaling of discrete diffusion models via importance weighting and optimal proposal design. In *International Conference on Learning Representations*, pp. 36740–36775, 2026.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43346–43367. PMLR, Jul 2024.

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

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen Mckeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55810–55827. PMLR, Jul 2025.

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 *Advances in Neural Information Processing Systems*, volume 36, pp. 3706–3731. Curran Associates, Inc., 2023.

Robert Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical Review Letters*, 57:2607–2609, Nov 1986.

Will D Vousden, Will M Farr, and Ilya Mandel. Dynamic temperature selection for parallel tempering in markov chain monte carlo simulations. *Monthly Notices of the Royal Astronomical Society*, 455 (2):1919–1937, 2016.

Shih-Hsin Wang, Joel A. Keller, Taos Transue, Drake Benjamin Brown, Thomas Strohmer, and Bao Wang. Test-time guidance for flow-based generative models via parallel tempering on source distributions. In *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2026.

Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. A simple model to generate hard satisfiable instances. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 337–342. Morgan Kaufmann Publishers Inc., 2005.

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 *Advances in Neural Information Processing Systems*, volume 36, pp. 11952–11969. Curran Associates, Inc., 2023.

Leo Zhang, Peter Potaptchik, Jiajun He, Yuanqi Du, Arnaud Doucet, Francisco Vargas, Hai-Dang Dau, and Saifuddin Syed. Accelerated parallel tempering via neural transports. In *International Conference on Learning Representations*, pp. 90874–90907, 2026. Note: This work has previously been published under the title "Generalised parallel tempering: flexible replica exchange via flows and diffusions".

Hang Zhao, Kexiong Yu, Yuhang Huang, Renjiao Yi, Chenyang Zhu, and Kai Xu. DISCO: Efficient diffusion solver for large-scale combinatorial optimization problems. *Graphical Models*, 141, 2025.

A Energy Functions

Following Sanokowski et al. (2024; 2025), we used these energy functions in our experiments:

$$\text{MIS:} \quad H_{\mathcal{G}}(\boldsymbol{x}) = -A\sum_{i=1}^{|V|} \boldsymbol{x}_i + B \sum_{(i,j)\in E} \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$

$$\text{MDS:} \quad H_{\mathcal{G}}(\boldsymbol{x}) = A\sum_{i=1}^{|V|} \boldsymbol{x}_i + B \sum_{i=1}^{|V|}(1-\boldsymbol{x}_i) \prod_{j\in\mathcal{N}_{\mathcal{G}}(i)} (1-\boldsymbol{x}_j)$$

$$\text{Max cut:} \quad H_{\mathcal{G}}(\boldsymbol{x}) = -\sum_{(i,j)\in E} \frac{1-\sigma_i\sigma_j}{2}, \quad \text{where } \sigma_i = 2\boldsymbol{x}_i - 1$$

$$\text{Max clique:} \quad H_{\mathcal{G}}(\boldsymbol{x}) = -A\sum_{i=1}^{|V|} \boldsymbol{x}_i + B \sum_{(i,j)\notin E} \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$

Here, subscripts refer to indices of the vector, not diffusion time steps. $\mathcal{N}_{\mathcal{G}}(i)$ refers to the neighbors of node i in graph $\mathcal{G} = (V, E)$. $A, B \in \mathbb{R}$ are hyperparameters; all experiments use $A = 1$ and $B = 1.1$.

B Hyperparameters

A common choice for parallel tempering is to space temperatures geometrically between a minimum and a maximum temperature, meaning that $\tau_{i+1}/\tau_i = \text{const}$ for all $i \in \{1, \dots, R-1\}$ (Earl & Deem, 2005). The minimum temperature is usually set to $\tau_1 = 1$. We follow these choices here.

For diffusion-related hyperparameters, we chose the same values used by Sanokowski et al. (2025). For comparability to DiffUCO and SDDS, our experiments use the trained models available here: `https://github.com/ml-jku/DIffUCO/tree/main/Checkpoints`. For SDDS, models trained using rKL with RL are used, as they perform better on most CO problems and datasets.

Tab. 5 lists the values chosen for each hyperparameter.

Table 5: Hyperparameters used in our experiments.

Hyperparameter	Small Datasets	Large Datasets
Minimum temperature τ_1	1.0	
Maximum temperature τ_R	5.0	2.5
Number of temperatures R	10	
Number of replicas N	100	
Number of diffusion steps T	18	

C Hardware

Experiments were performed using an NVIDIA H100 GPU with 80GB of vRAM and an AMD EPYC 9654 96-Core processor.