Revisiting Sampling for Combinatorial Optimization

Haoran Sun [*1] Katayoon Goshvadi [2] Azade Nova [2] Dale Schuurmans [2] Hanjun Dai [2]

Abstract

Sampling approaches like Markov chain Monte Carlo were once popular for combinatorial optimization, but the inefficiency of classical methods and the need for problem-specific designs curtailed ongoing develpment. Recent work has favored data-driven approaches that mitigate the need for hand-craft heuristics, but these are often not usable as out-of-the-box solvers due to dependence on in-distribution training and limited scalability to large instances. In this paper, we revisit the idea of using sampling for combinatorial optimization, motivated by the significant recent advances of gradient-based discrete MCMC and new techniques for parallel neighborhood exploration on accelerators. Remarkably, we find that modern sampling strategies can leverage landscape information to provide general-purpose solvers that require no training and yet are competitive with state of the art combinatorial solvers. In particular, experiments on cover vertex selection, graph partition and routing demonstrate better speed-quality trade-offs over current learning based approaches, and sometimes even superior performance to commercial solvers and specialized algorithms.

1. Introduction

Combinatorial optimization (CO) is a core challenge in domains like logistics, supply chain management and hardware design, and has been a fundamental problem of study in computer science for decades. Since CO problems are typically NP-hard, developing approximate algorithms that can efficiently find high-quality sub-optimal solutions has been a critical concern. For problems of this kind, sampling based approaches offer an appealing property by providing a simple trade-off between runtime and solution quality.

A long and fruitful literature has investigated the connections between CO and sampling methods. A particularly famous approach is simulated annealing (SA) (Kirkpatrick et al., 1983), which leverages local thermal fluctuations enforced by Metropolis-Hastings updates (Metropolis et al., 1953; Hastings, 1970). SA and its variants, such as tempered transitions (Neal, 1996) and parallel tempering (Iba, 2001), have demonstrated good performance in many applications (Johnson et al., 1989; 1991; Earl & Deem, 2005). However, except for a few algorithms, such as Swedesen-Wang (Swendsen & Wang, 1987) and Hamze-Freitas-Selbey (Hamze & de Freitas, 2012) that exploit special structure of the underlying problem, sampling in general discrete spaces has primarily relied on Gibbs sampling, which exhibits notoriously poor efficiency in high dimensional spaces. This disadvantage has prevented sampling from being a primary method for combinatorial optimization for quite some time.

A recent trend in CO has been to leverage learning for optimization in a data-driven way, which has alleviated the reliance on hand-crafted heuristics while being able to leverage modern accelerators like GPUs and TPUs to improve the efficiency. Though some learning methods require supervised information (Li et al., 2018; Gasse et al., 2019; Gupta et al., 2020) which can be hard to obtain, many approaches have leveraged reinforcement learning (Khalil et al., 2017; Kool et al., 2018; Chen & Tian, 2019) or other unsupervised learning techniques (Karalias & Loukas, 2020; Sun et al., 2022c; Wang et al., 2022) to broaden applicability. However, despite obtaining better speed-quality trade-offs for modest sized problems, the requirement of learning on in-distribution instances typically makes it hard to use these approaches as an out-of-the-box solver on problems drawn from arbitrary distributions. Additionally, scaling a learned optimizer to large problems is difficult in general without the assistance of search algorithms (Nair et al., 2020) or problem specific decomposition (Manchanda et al., 2020).

In this paper, we revisit sampling as a viable approach to develop an out-of-the-box CO solver. Together with motivation from recent theoretical findings (Ma et al., 2019; Dong & Tong, 2021) on the advantages of sampling for non-convex problems, we are also motivated by the recent advances in Markov chain Monte Carlo (MCMC) for discrete spaces (Sun et al., 2022a). By treating sampling methods as a simulation of Langevin dynamics in a discrete space (Sun

[*]Work done during an internship at Google [1]Georgia Tech [2]Google Deepmind. Correspondence to: Haoran Sun <hsun349@gatech.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

et al., 2022a), the efficiency of Gibbs sampling, which is analogous to coordinate descent, can be greatly accelerated by far more effective simulations (Grathwohl et al., 2021; Sun et al., 2021; Zhang et al., 2022). The core of these advances relies on estimating an objective ratio among the neighborhood of a current point (Zanella, 2020). We show that these objective ratios can be effectively obtained (or approximated) by gradients with low bias in many CO problems. Moreover, these ratios can be obtained in parallel for large neighborhoods, allowing us to leverage the power of Machine Learning (ML) compilers like XLA or frameworks like JAX with autograd on modern accelerators, in much the same way as learning based CO solvers. Our main goal is to show that, with these improvements, sampling for CO provides a very strong and simple alternative to data-driven approaches, and their potential advantages should be carefully recalibrated.

We empirically justify our claims on five common CO problems in the areas of vertex selection, graph partitioning and routing. In many cases, improved sampling for CO produces a better speed-quality trade-off than data-driven approaches, and in some cases improves upon commercial solvers and the best available specialized algorithms.

2. Background

Our approach will build upon the classical foundations of energy based models for flexible distribution representation, Metropolis-Hastings for designing MCMC chains that achieve desired stationary distributions, and simulated annealing to drive an MCMC chain to a low energy state.

Energy Based Model (EBM). Let $\mathcal{S}$ be the state space. An EBM defines an energy function $f : \mathcal{S} \to \mathbb{R}$ with the target distribution $\pi(x) = e^{-f(x)/\tau}/Z$, where τ is a temperature parameter used to control the smoothness of the system, and $Z = \sum_{z \in \mathcal{S}} e^{-f(z)/\tau}$ is the partition function (van Hemmen, 1986; LeCun et al., 2006). The energy function provides a great deal of flexibility in characterizing a complex distribution. However, in general it is difficult to obtain samples from such a distribution.

Metropolis-Hastings (M-H) Algorithm. M-H is a generic framework for sampling from a target distribution. Let π denote the distribution we want to draw sample from. Given a current state $x^{(t)}$, an M-H sampler (Metropolis et al., 1953; Hastings, 1970) proposes a candidate state y from a proposal distribution $q(x^{(t)}, y)$. Then, with probability

$$\min\left\{1, \frac{\pi(y)q(y, x^{(t)})}{\pi(x^{(t)})q(x^{(t)}, y)}\right\}, \tag{1}$$

the proposed state is accepted and $x^{(t+1)} = y$; otherwise, $x^{(t+1)} = x^{(t)}$. In this way, the detailed balance condition is satisfied and the M-H sampler generates a Markov chain $x^{(0)}, x^{(1)}, \ldots$ that has π as its stationary distribution.

Simulated Annealing. Simulated annealing (SA) for combinatorial optimization was introduced by Kirkpatrick et al. (1983) and independently by Černỳ (1985). SA is a probabilistic relaxation of local search (Dowsland & Thompson, 2012). At each iteration, one site in the current configuration x is subjected to a small displacement, the energy difference $\delta E = f(x') - f(x)$ calculated, and the new state x' accepted with probability $\exp(-\delta E/\tau)$. By gradually decreasing the temperature τ to 0, the configuration x will stop at a low energy state with high probability.

3. Method

We first formalize the optimization problem in Section 3.1 and revisit a generic framework for sampling from a sequence of distributions that seeks higher quality solutions. Then we show how the generic framework can be instantiated with improved efficiency and parallelism in Section 3.2, by leveraging the latest advances in discrete space MCMC and ML compilers for accelerators. We will denote the final improved sampling algorithm for combinatorial optimization as iSCO.

3.1. Optimization via sampling

Problem formulation: Without loss of generality, we consider combinatorial optimization problems that admit the following integer program form:

$$\min_{x \in \mathcal{S} = \{0,1,\ldots,n-1\}^d} \quad a(x), \quad \text{s.t.} \quad b(x) = 0 \tag{2}$$

where the solution $x \in \mathcal{S}$ is a d dimensional vector such that each dimension takes a discrete value from $\{0, 1, \ldots, n-1\}$. We will primarily focus on discrete variables as these are usually the most challenging to deal with, although it will be possible to extend the approach to the mixed-integer case. For ease of exposition, we also assume $b(x) \geqslant 0, \forall x \in \mathcal{S}$, but otherwise do not limit the form of a and b, except for some mild conditions required by the sampling algorithms.

Sampling framework: To convert the optimization problem to a sampling problem, we first rewrite the constrained optimization into a penalty form via a penalty coefficient λ, then treat this as an energy function for an EBM. In particular, the energy function takes the form:

$$f(x) = a(x) + \lambda \cdot b(x) \tag{3}$$

Furthermore, we introduce a parameter τ to serve as a temperature that balances the exploration and exploitation trade-off in much the same way as SA algorithms. That is, we define the probability of x at temperature τ by:

$$p_\tau(x) \propto \exp(-f(x)/\tau) \tag{4}$$

The original optimization problem essentially seeks samples from $\lim_{\tau\to 0} p_\tau(x)$. A naive approach to this problem would be to attempt to directly sample from $p_{\tau\to 0}(x)$, but such a distribution is highly nonsmooth and unsuitable for MCMC methods. Instead, following classical SA and recent advances in distribution matching (Song & Ermon, 2020), we define a sequence of distributions parameterized by a sequence of decaying temperatures:

$$\mathscr{P} = [p_{\tau_0}(x), p_{\tau_1}(x), \dots, p_{\tau_T}(x)] \tag{5}$$

where the sequence $\tau_0 > \tau_1 > \dots \tau_T \to 0$ converges to zero as T increases.

3.2. Improved implementation

In generic SA for discrete spaces, Gibbs sampling is the default algorithm for obtaining samples from $p_\tau(x)$. Even though the acceptance rate for Gibbs sampling is always 1, this sampling strategy is extremely inefficient since only one dimension can be updated at a time, and the loop over dimensions cannot generally be parallelized. Thus it is prohibitive to tackle CO problems with Gibbs sampling whenever the number of decision variables is large.

However, recent advances in sampling for discrete spaces have revealed important new opportunities for improving efficiency to achieve practical scalability in many CO problems. To simplify our notation, we will drop the temperature parameter τ and write $p(x)$ for $p_\tau(x)$ when appropriate.

Sun et al. (2022a) showed how a principled gradient flow toward $p(x)$ can be defined and characterized by a continuous time Markov chain, whose discrete time simulation $x_{t_1}, x_{t_2}, ..., x_{t_n}$ is still a Markov chain that converges to $p(x)$. Denote ρ^t as the probability distribution on $\mathcal{S}$. Then the continuous time Markov chain can be defined by the Langevin dynamics

$$\frac{d}{dt}\rho^t = \rho^t R, \quad R \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}, \tag{6}$$

where the rate matrix R satisfies

$$R(x,y) = \begin{cases} c(x,y) g\left(\frac{p(y)}{p(x)}\right) 1_{\{y\in\mathcal{N}(x)\}}, & y \neq x \\ -\sum_{z\in\mathcal{N}(x)} R(x,z), & y = x \end{cases}. \tag{7}$$

Here, $g(\cdot): \mathbb{R}_+ \to \mathbb{R}_+$ is a locally balanced weight function that satisfies $g(z) = zg(\frac{1}{z})$, for example $g(z) = \sqrt{z}$ or $g(z) = \frac{z}{z+1}$ (Zanella, 2020). The $\mathcal{N}(x)$ is the neighborhood of x, for example a 1-Hamming Ball. The weight $c(x,y)$ is an edge weight that serves as a preconditioning of the underlying graph, which we will just set to $c(x,y) \equiv 1$ in this paper.

Note that if the ratio $p(y)/p(x)$ can be obtained easily for all $y \in \mathcal{N}(x)$, then the Langevin dynamics can be simulated efficiently (Sun et al., 2022a). We will demonstrate how this can be done for some CO problems in Section 3.2.1, and present a variant of a recently proposed sampler for CO to realize the simulation in Section 3.2.2.

3.2.1. Efficient neighborhood ratio estimation

Let $N_x := |\mathcal{N}(x)|$ denote the size of the neighborhood around x. Note that since $p(y)/p(x) \propto \exp(f(x) - f(y))$, we are interested in calculating the set of energy differences, $f(y) - f(x)$, between x and its neighboring states $y \in \mathcal{N}(x)$. If we enumerate the set of neighborhood states $y^{(1)}, y^{(2)}, ..., y^{(N_x)} \in \mathcal{N}(x)$ we can recover the vector of energy differences $\Delta(x) \in \mathbb{R}^{N_x}$:

$$\Delta(x) := [f(y^{(1)}) - f(x), f(y^{(2)}) - f(x), ..., f(y^{(N_x)}) - f(x)]$$

For the special case when $\mathcal{N}(x)$ is the 1-Hamming Ball of x, we can write $\Delta(x)_{(i,j)} = f([x_1, x_2, ..., y_i = j, ...]) - f(x)$ where the neighbor y indexed by (i,j) is obtained by copying $y := x$ and changing the i-th dimension of y to value j. In the binary case, we omit j as one can only flip x_i from 1 to 0 or vice versa.

Evaluating $\Delta(x)$ directly is embarrassingly parallelizable but still expensive. However, for many CO problems we can identify efficient exact methods for calculating $\Delta(x)$, or obtain an approximation to the probability ratio in a cheaper way. In a general case, the probability ratio $p(y)/p(x)$ can be approximated by a first-order Taylor expansion $\frac{p(y)}{p(x)} \approx \exp(\langle -\nabla f(x), y - x\rangle)$ (Grathwohl et al., 2021), which only requires the gradient $\nabla f(x)$ to be evaluated once for the entire neighborhood $y \in \mathcal{N}(x)$. We will find that this general technique is sufficiently efficient and accurate to be effective in the CO problems we consider below.

The following two exemplar CO problems are expressed over graphs, so let $G = (V, E)$ denote a graph with nodes $V = \{1, ..., d\}$, node weights c, and edges $E = \{e(v_i, v_j) | v_i, v_j \in V\}$.

Maximum Independent Set. For a graph G, we use $x_i = 1$ to indicate that node i is selected and $x_i = 0$ to indicate that node i is not selected. Then the maximum independent set can be formulated as an optimization problem:

$$\min_{x\in\{0,1\}^d} -\sum_{i=1}^d c_i x_i, \quad \text{s.t. } x_i x_j = 0,\ \forall (i,j) \in E \tag{8}$$

The corresponding energy function is a quadratic function

$$f(x) := -c^T x + \lambda \frac{x^T A x}{2} \tag{9}$$

where A is the adjacency matrix of the graph G. Since A has a zero diagonal, the value change of flipping one dimension can be exactly calculated by:

$$\Delta(x) = (1-2x)\odot(-c + \lambda Ax) = (1-2c)\odot\nabla f(x) \tag{10}$$

where $\odot$ denotes the element-wise product. Equation (10) shows that evaluating $\Delta(x)$ can be as cheap as evaluating $f(x)$ once, and the gradient approximation corresponds to the exact energy difference calculation in this problem.

Balanced Graph Partition. A graph G can be partitioned into K disjoint sets $S_1, ..., S_K$, such that $\bigcup_{k=1}^K S_k = V$ and $S_i \cap S_j = \emptyset$ for any $S_i \neq S_j$. We denote $\text{cut}(P, Q) = \sum_{i \in P, j \in Q} e(i, j)$ and $\bar{Q} = V \backslash Q$. The objective for balanced graph partition is expressed by:

$$\text{Ncut}(S_1, ..., S_K) = \sum_{k=1}^K \frac{\text{cut}(S_k, \bar{S}_k)}{\text{vol}(S_k, V)} \tag{11}$$

where $\text{vol}(S_k, V) = \sum_{v_i \in S_k, v_j \in V} e(v_i, v_j)$. To write this in a matrix form, we denote $x \in \{0, 1\}^{d \times k}$, where each row $x^i \in \{0, 1\}^k$ is a one hot vector that represents which cluster node i is assigned to. Here, $\Delta(x) \in \mathbb{R}^{d \times k}$ gives the energy difference of switching a node from a current cluster to an alternative cluster. Also, we overload the notation a bit to let $S_j \in \{0, 1\}^d$ denote the j-th column of x, which is the binary indicator of the cluster S_j in (11). Let A denote the adjacency matrix of G and assign $\alpha = A\mathbf{1} \in \mathbb{R}^n$ to be the weighted degree for each node. Then, the energy function can be written as:

$$f(x) = \sum_{i=1}^K \frac{S_j^T A(1 - S_j)}{S_j^T \alpha} \tag{12}$$

Note that the denominator in (12) implicitly restricts S_j^T to be nonzero, so in practice we can add a sufficiently small ϵ to the denominator for numerical stability.

When we flip a node x^i from cluster k to cluster l, the exact change of the objective function is:

$$\begin{aligned}\Delta(x) =& \frac{S_k^T A(1 - S_k) + S_k^T A e_i - e_i^T A(1 - S_k)}{S_k^T \alpha - \alpha_i} + \\ & \frac{S_l^T A(1 - S_l) + e_i^T A(1 - S_l) - S_l^T A e_i}{S_l^T \alpha + \alpha_i} - \\ & \frac{S_k^T A(1 - S_k)}{S_k^T \alpha} - \frac{S_l^T A(1 - S_l)}{S_l^T \alpha} \qquad (13) \\ =& \frac{\alpha_i S_k^T A(1 - S_k)}{(S_k^T \alpha - \alpha_i) S_k^T \alpha} + \frac{S_k^T A e_i - e_i^T A(1 - S_k)}{S_k^T \alpha - \alpha_i} - \\ & \frac{\alpha_i S_l^T A(1 - S_l)}{(S_l^T \alpha + \alpha_i) S_l^T \alpha} - \frac{S_l^T A e_i - e_i^T A(1 - S_l)}{S_l^T \alpha + \alpha_i}.\end{aligned} \tag{14}$$

Now, if we instead consider the gradient approximation $\tilde{\Delta}(x) = \langle \nabla f(x), y - x \rangle$ where y is the matrix after the flip, we obtain:

$$\begin{aligned}\tilde{\Delta}(x) =& \frac{\alpha_i S_k^T A(1 - S_k)}{S_k^T \alpha S_k^T \alpha} + \frac{S_k^T A e_i - e_i^T A(1 - S_k)}{S_k^T \alpha} - \\ & \frac{\alpha_i S_l^T A(1 - S_l)}{S_l^T \alpha S_l^T \alpha} - \frac{S_l^T A e_i - e_i^T A(1 - S_l)}{S_l^T \alpha}.\end{aligned} \tag{15}$$

Comparing (14) and (15), one can see that the only difference is the $S_l^T \alpha + \alpha_i$ versus $S_l^T \alpha$ and $S_k^T \alpha - \alpha_i$ versus $S_k^T \alpha$ in the denominators. When each cluster has a sufficient number of nodes, this difference is negligible.

In summary, for many CO problems, one can leverage the ML infrastructure for gradient backpropagation to efficiently approximate the probability ratio in a generic way without too much manual engineering.

3.2.2. Parallelizable sampling

Eq. (6) shows that it is possible to modify multiple dimensions of x at each step (Zanella, 2020; Sun et al., 2021; Zhang et al., 2022; Sun et al., 2022a), depending on the resolution of simulation time. This can greatly improve the efficiency of sampling for CO. In this paper, we employ the Path Auxiliary Sampler (PAS) (Sun et al., 2021) to simulate the Langevin dynamics as it is numerically more stable in the low temperature regime. However, the original PAS only applies to binary variables, so we need to generalize the approach to categorical variables to meet the needs of many CO problems.

Given a current state x and $\Delta(x)$ as computed (or approximated) in the previous section, a new state y is proposed via the following steps:

1. Sample indices $\mathcal{J} \subseteq \{1..d\}$ from the categorical distribution $\{w_j\}_{j=1}^d$ without replacement, where

$$w_j \propto \sum_{s \neq x_j} g\Big(\exp\left(-\Delta(x)_{(j,s)}\right) \Big). \tag{16}$$

2. For $j \notin \mathcal{J}$, set $y_j = x_j$. For $j \in \mathcal{J}$, sample $y_j \sim q_x^j(s)$, for $s \neq x_j$ where

$$q_x^j(s) = \frac{g\left(\exp(-\Delta(x)_{(j,s)})\right)}{\sum_{s' \neq x_j} g\left(\exp(-\Delta(x)_{(j,s')})\right)}. \tag{17}$$

3. Finally, apply M-H to correct the proposal.

The hyperparamter $L = |\mathcal{J}|$ determines the scale of the neighborhood. We follow Sun et al. (2022b) to tune L so that the average M-H acceptance rate is 0.574. In particular, we sample $L \sim \text{Poisson}(\mu)$, which is adaptively updated:

$$\mu_{t+1} \leftarrow \text{clip}(\mu_t + 0.001 * (\bar{A}_t - 0.574), \min = 1, \max = d),$$

where $\bar{A}_t$ is the empirical average acceptance rate in each step t. We summarize the algorithm in Algorithm 1. More details are given in Appendix C.

Algorithm 1 Sampling for Combinatorial Optimization

```
1: Input: initial state $x_0$, initial temperature $\tau = \tau_0$, outer
   loop size $m$, inner loop size $n$.
2: for i = 0, ..., m-1 do
3:     for j = 0, ..., n-1 do
4:         $x_{in+j+1}, \bar{A} \leftarrow$ PAS-MH-Step($x_{in+j}, \pi_\tau, \mu$)
5:         $\mu \leftarrow$ Update-$\mu$-Step($\mu, \bar{A}$)
6:     end for
7:     $z_{i+1} \leftarrow$ Post-Processing($x_{(i+1)n}$) in Section A
8:     Update temperature $\tau = \tau_0(1 - \frac{i+1}{m})$
9: end for
```

Connection to Gibbs: The relationship between PAS and Gibbs is analogous to that between gradient descent and coordinate descent. In scenarios where computing the full gradient is expensive, coordinate descent is preferable. When efficient parallel computation of the gradient is available (e.g., in typical CO problems), gradient descent will be more efficient. In our iSCO method, the parallel computation of the ratio $\frac{\pi(y)}{\pi(x)}$ in Section 3.2.1 allows us to efficiently perform PAS, a full coordinate update, in every MH step.

4. Related work

Sampling based methods (Metropolis et al., 1953; Hastings, 1970; Neal, 1996; Iba, 2001) have been widely used for CO problems, including, for example, TSP (Kirkpatrick et al., 1983; Černỳ, 1985; Wang et al., 2009), VLSI design (Sechen et al., 1988; Chandy & Banerjee, 1996; Wong et al., 2012), planning (Chen & Ke, 2004; Jwo et al., 1995), scheduling (Seçkiner & Kurt, 2007; Thompson & Dowsland, 1998), and routing (Tavakkoli-Moghaddam et al., 2007; Van Breedam, 1995). However, these previous methods rely on Gibbs sampling, which is too slow for high dimensional CO problems.

Learning based methods for combinatorial optimization also has a long history. In initial attempts, Hopfield & Tank (1985) and Ramanujam & Sadayappan (1995) transformed CO problems into neural network optimization problems with differentible objectives. Due to hardware limitations however, these approaches are not competitive against carefully designed message passing algorithms, such as mean-field annealing (Bilbro et al., 1988), perturbed belief propagation (Ravanbakhsh & Greiner, 2015), and survey propagation (Braunstein et al., 2005). Recently, with the rise of the modern accelerators like GPUs and TPUs, the reconsideration of learning based methods has been significantly stimulated (see our introduction section). However, by contrast, sampling based methods, which also significantly benefit from accelerators, have not been as vigorously explored.

5. Experiment

In this section we experimentally verify the efficiency and effectiveness of iSCO through five combinatorial optimization problems: max independent set (MIS) and max clique in Section 5.1, maxcut and graph balanced partition in Section 5.2, and the Traveling Salesman Problem (tsp) in Section 5.3. For each set of the problems we run on both synthetic and real-world benchmark datasets commonly used in the literature, and compare against existing generic or specialized solvers, heuristics, and learning based methods.

Setup: By default we approximate the evaluation of the energy functions through the first-order Taylor expansion, use the variant of the PAFS sampler in Section 3.2 to simulate the Langevin dynamics, and report the runtime of iSCO on a machine with a single Nvidia 1080Ti GPU (unless otherwise noted), in alignment with existing data-driven approaches. We find the runtime can be easily improved with more powerful GPUs that have more cores, or simply by adding more GPUs and running multiple Markov chains in parallel. We run iSCO with different initial temperatures, and use an exponential temperature decay schedule by default. Since more steps or more Markov chains for iSCO would always result in better solution quality, we halt the sampling procedure when the solution quality is sufficient or the gain plateaus. In Section 5.4 we provide an ablation study to isolate the impact of different hyper-parameters, and to also show the efficiency gains over classical samplers.

We follow standard convention and use the approximation ratio α—the ratio between the found solution and the optimal solution—to measure the solution quality. This means that for a maximization problem, $\alpha \leqslant 1$ and the solution is optimal when $\alpha = 1$. Note that for hard problems there is no optimality guarantee for a solution, so we either compare objective values directly, or report the ratio against the solution found by a commercial solver with the best effort (which means α can be larger than 1).

Please refer to Appendix A for more details on the implementation, including the specific energy functions used and practical implementation details for accelerators. Also see Appendix B for more experimental details and results.

5.1. Max independent set and max clique

MIS: We use the MIS benchmark from the recent work (Qiu et al., 2022), which consists of graphs from SATLIB (Hoos & Stützle, 2000) and also Erdős–Rényi (ER) random graphs (Erdős et al., 1960) of different sizes. Since iSCO does not require training, we directly report the results on the test dataset provided at [1]. In the end, we have 500 test graphs from SATLIB with 403 to 449 clauses each (which

[1]https://github.com/dimesteam/dimes

Table 1. Results of MIS on three benchmarks provided by DIMES (Qiu et al., 2022). The runtime and solution quality of baselines are from DIMES, since exactly the same test graphs are used on the same type of GPU. Baselines involve solvers from the Operation Research (OR) community, and data-driven approaches using Reinforcement Learning (RL), Supervised Learning (SL) equipped with Tree Search (TS), Greedy decoding (G) or sampling (S). Methods that cannot produce results in 10x time limit of DIMES are labeled as N/A.

Method	Type	SATLIB			ER-[700-800]			ER-[9000-11000]		
		Size↑	Drop↓	Time↓	Size↑	Drop↓	Time↓	Size↑	Drop↓	Time↓
KaMIS	OR	425.96*	–	37.58m	44.87*	–	52.13m	381.31*	–	7.6h
Gurobi	OR	425.95	0.00%	26.00m	41.38	7.78%	50.00m	N/A	N/A	N/A
Intel (Li et al., 2018)	SL+TS	N/A	N/A	N/A	38.80	13.43%	20.00m	N/A	N/A	N/A
	SL+G	420.66	1.48%	23.05m	34.86	22.31%	6.06m	284.63	25.35%	5.02m
DGL (Böther et al., 2022)	SL+TS	N/A	N/A	N/A	37.26	16.96%	22.71m	N/A	N/A	N/A
LwD (Ahn et al., 2020)	RL+S	422.22	0.88%	18.83m	41.17	8.25%	6.33m	345.88	9.29%	7.56m
DIMES (Qiu et al., 2022)	RL+G	421.24	1.11%	24.17m	38.24	14.78%	6.12m	320.50	15.95%	5.21m
	RL+S	423.28	0.63%	20.26m	42.06	6.26%	12.01m	332.80	12.72%	12.51m
iSCO (Ours)	fewer steps	423.66	0.54%	5.85m	44.77	0.2%	1.38m	377.5	1.00%	9.38m
	more steps	**424.16**	**0.42%**	15.72m	**45.15**	**-0.6%**	5.56m	**384.20**	**-0.7%**	1.25h

* indicates the baseline for computing the performance drop.

translates to at most 1,347 nodes and 5,978 edges), 128 test graphs for ER graphs with 700 to 800 nodes each, and 16 test graphs for ER graphs with 9000 to 11000 nodes each. Following Qiu et al. (2022) we report runtime on a single A100 GPU. For each dataset we run iSCO with two settings: a smaller number of steps T that can be more efficient than most learning based methods while achieving better results, and a larger number of steps that can attain solution quality comparable to or even better than the best solvers. See Appendix B.1 for more details on the setup.

We report solution quality and runtime in Table 1. Here we can see that iSCO is very effective, producing the best solution compared to existing learning based methods (including specially designed ones like LwD for MIS) within a comparable or even shorter runtime. Notably, iSCO outperforms KaMIS (Lamm et al., 2017; Hespe et al., 2019) – the winner of PACE 2019 and probably the best solver for MIS right now, on both ER graphs with a much smaller runtime.

We also conduct experiments to measure how iSCO performs on graphs with different densities. We consider densities $\{0.05, 0.10, 0.15, 0.20, 0.25\}$. For each density, we generate 128 ER graphs with 700 to 800 nodes each. We run iSCO for 1 minute and compare it with Gurobi and Gurobi-clique running for 1 hour. The Gurobi-clique defines each constraint on a clique, which gives a stronger linear relaxation, hence a better performance. See Appendix B.1 for more details. In Table 2, one can observe that, over all densities, iSCO has set size around 8% larger than Gurobi with a much shorter running time.

Max clique: Theoretically the MIS and max clique problems can easily be reduced to one another, though some existing algorithms like RUN-CSP (Toenshoff et al., 2021) can only handle one formulation and not the other (Karalias & Loukas, 2020). We include the results on the max

Table 2. Results of MIS on ER-[800-800] with different densities

Density	0.05	0.10	0.15	0.20	0.25
iSCO (1m)	105.00	62.50	44.77	34.81	28.38
Gurobi-edge(1h)	97.78	57.28	41.38	31.12	26.15
Gurobi-clique(1h)	98.59	57.40	41.68	31.56	26.25

clique benchmarks mainly for completeness, with an additional goal to show the flexibility of iSCO on handling both problem formulations directly. We follow the setting in Karalias & Loukas (2020) and Wang et al. (2022) and report the approximation ratio on synthetic graphs generated with RB model (Xu et al., 2007) and a real-world Twitter graph (Leskovec & Krevl, 2014).

Table 3. Approximation ratio ↑ comparison on max clique tasks.

Method	Twitter	RBtest
EPM (Karalias & Loukas, 2020)	0.924 ± 0.133 (0.17s/g)	0.788 ± 0.065 (0.23s/g)
AFF (Wang et al., 2022)	0.926 ± 0.113 (0.17s/g)	0.787 ± 0.065 (0.33s/g)
RUN-CSP (Toenshoff et al., 2021)	0.987 ± 0.063 (0.39s/g)	0.789 ± 0.053 (0.47s/g)
iSCO (ours)	$\mathbf{1.000 \pm 0.000}$ (1.67s/g)	$\mathbf{0.857 \pm 0.062}$ (1.67s/g)

Table 3 shows the results of iSCO run with 1k steps on each test graph instance from scratch, compared to other learning based methods which are trained on the same distribution of graphs. iSCO achieves significantly better quality while taking a negligible amount of additional time, considering that iSCO requires no prior training.

5.2. Maxcut and balanced graph partition

Maxcut: We follow the same maxcut experiment setup as in Dai et al. (2020), where the benchmark contains random graphs and corresponding solutions obtained by running Gurobi for 1 hour. We run on both Erdős–Rényi (ER) graphs and Barabási–Albert (BA) graphs on all graph sizes ranging from 16 to 1,100 nodes and up to 91,239

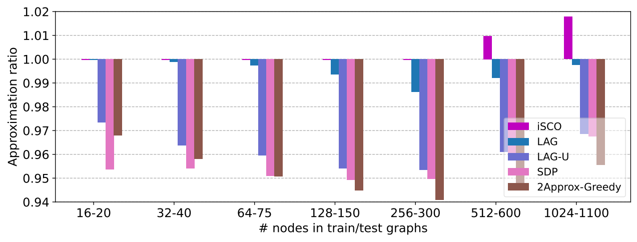

Figure 1. Maxcut results on BA graphs. The ratio is calculated against a reference solution obtained by running Gurobi for 1 hour, and the larger the better.

edges. We report the ratio against the solutions provided by Gurobi, and compare against the LAG (Dai et al., 2020) with either supervised learning via Li et al. (2018) (denoted as LAG) or unsupervised learning through Karalias & Loukas (2020)(denoted as LAG-U), and classical approaches like semidefinite programming and approximated heuristics.

Figure 1 shows the results on BA graphs, while the results on ER graphs are included in the appendix as they show similar trends. From the plot we can see iSCO achieves an optimal solution in all cases (where the error bar is barely visible), and obtains much better results than Gurobi on the large instances, where the ratio α is above 1. Note that this result improves the quality of many specially designed baselines for maxcut problems, such as LAG (Dai et al., 2020), which leverages greedy and spanning tree features for augmented expressiveness. On the largest graph iSCO runs for 10,000 steps for roughly 12s on a single GPU, which also achieves the best time-quality trade-off.

Following Khalil et al. (2017), we also include results on realistic instances, which are ten graphs from the Optsicom project [2]. The edge weights are in $\{-1, 0, 1\}$.

Table 4. Maxcut results on Optsicom.

Method	SDP	Approx	S2V-DQN	iSCO
Approximation ratio	0.526	0.780	0.978	**1.00**

The results above show that iSCO is able to achieve the optimal solution in only 1,000 steps, which translates to less than 1 second runtime on 1080Ti.

Balanced graph partition We further evaluate iSCO on graph cuts that also consider cluster balance. We follow the experiments in Nazi et al. (2019) and report the results on five different computation graphs compiled from commonly used deep neural networks. The largest graph is Inception-v3 (Szegedy et al., 2017), which contains 27,144 operations (nodes) and 40,875 edges. We compare the results with

[2]https://grafo.etsii.urjc.es/optsicom

Table 5. Graph partition.

Metric	Methods	VGG	MNIST-conv	ResNet	AlexNet	Inception-v3
Edge cut ratio ↓	hMETIS	0.05	0.05	0.04	0.05	0.04
	GAP	**0.04**	0.05	0.04	0.05	0.04
	iSCO	0.05	**0.04**	0.05	**0.04**	0.05
Balanceness ↑	hMETIS	0.99	0.99	0.99	0.99	0.99
	GAP	0.99	0.99	0.99	0.99	0.99
	iSCO	0.99	0.99	0.99	0.99	0.99

GAP (Nazi et al., 2019), a specially designed learning architecture for graph partition, together with hMETIS (Karypis & Kumar, 1999), a widely used framework for this problem. Overall iSCO achieves comparable results against alternatives in Table 5, with almost perfect balance and a low cut ratio. Though in this case iSCO needs more sampling steps and used 30 minutes for the largest graph, the fastest GAP takes around 2 minutes. We discuss the potential limitation of iSCO in Section 6.

5.3. Traveling salesman problem

Finally we evaluate iSCO on 2D-TSP, where the solution space $\mathcal{S}$ is a permutation of integers in $\{0, 1, \ldots, n-1\}$ for a problem with n nodes in the 2D plane. We follow the setting in Qiu et al. (2022) and evaluate against the same set of test graphs with $n \in \{500, 1000, 10000\}$. In this scenario, we define the neighborhood relationship between two states if they can be reached by a single 2-opt operator (Croes, 1958). We use the same k-nn graph as in Qiu et al. (2022) where k equals to 50. To make the Markov chain reversible we also allow existence of edges between random pair of nodes with $1/(k+1)$ probability. All other baseline results are obtained from Qiu et al. (2022) directly, as the same test set and hardware environment are used.

We report the best results of the variants of each baseline in Table 6 and the full results in Table 10 in appendix. Overall we can see that within the same amount of time, iSCO yields a superior result to all the learning based methods considered. Many of these methods are not able to produce solutions within a reasonable amount of time for TSP-10000. Note that iSCO is highly parallelizable where the speed is mainly limited by the number of cores in GTX 1080Ti, and can benefit from more cores. For example, the same configuration of iSCO for TSP-10000 would only take 26.8 minutes on a single Nvidia V100. Nevertheless, none of the generic methods including iSCO would be able to compare against specialized solvers yet.

5.4. Ablation study

5.4.1. Compared with Gibbs sampling

We provide a comparison to using Gibbs sampling with annealing, or equivalently the classical SA for CO. We draw the curves of solution quality with respect to the number

Table 6. Results of TSP, where the numbers of baselines are taken from Qiu et al. (2022). In addition to the approaches mentioned in Table 1, some are implemented with Beam Search (BS), Active Search (AS) or Monte Carlo Tree Search (MCTS).

Method	Type	TSP-500			TSP-1000			TSP-10000		
		Length↓	Increase↓	Time↓	Length↓	Increase↓	Time↓	Length↓	Increase↓	Time↓
Concorde	OR	16.55*	-	37.66m	23.12*	-	6.65h	N/A	N/A	N/A
Gurobi	OR	16.55	0.00%	45.63h	N/A	N/A	N/A	N/A	N/A	N/A
LKH-3	OR	16.55	0.00%	46.28m	23.12	0.00%	2.57h	71.77*	-	8.8h
Farthest Insertion	OR	18.30	10.57%	0s	25.72	11.25%	0s	80.59	12.29%	6s
EAN (Deudon et al., 2018)	RL+S+2-OPT	23.75	43.57%	57.76m	47.73	106.46%	5.39h	N/A	N/A	N/A
AM (Kool et al., 2018)	RL+BS	19.53	18.03%	21.99m	29.90	29.23%	1.64h	129.40	80.28%	1.81h
GCN (Joshi et al., 2019)	SL+G	29.72	79.61%	6.67m	48.62	110.29%	28.52m	N/A	N/A	N/A
POMO (Kwon et al., 2020)	RL+AS	24.54	48.22%	11.61h	49.56	114.36%	63.45h	N/A	N/A	N/A
Att-GCN (Fu et al., 2021)	SL+MCTS	16.97	2.54%	2.20m	23.86	3.22%	4.10m	74.93	4.39%	21.49m
DIMES (Qiu et al., 2022)	RL+AS+MCTS	16.84	1.76%	2.15h	23.69	2.46%	4.62h	74.06	3.19%	3.57h
iSCO (Ours)	Sampling	**16.64**	**0.54%**	6.94m	**23.33**	**0.91** %	7.94m	**74.02**	**3.14%**	1.01h

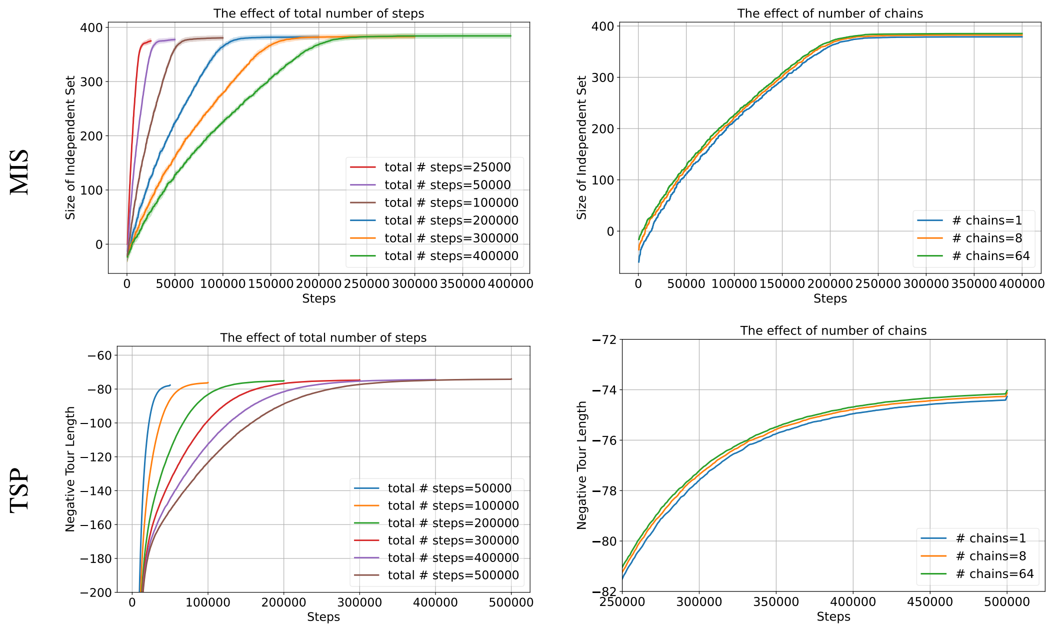

Figure 2. Ablation study on (left) number of steps T and (right) number of chain.

of sampling steps in Figure 4, on the largest graphs of MIS and maxcut problems. We can see using our variant of PAFS this can be 100x more sample efficient to achieve similar quality under a best tuned τ-scheduling, making the sampling for CO efficient enough to compare against many modern data-driven methods.

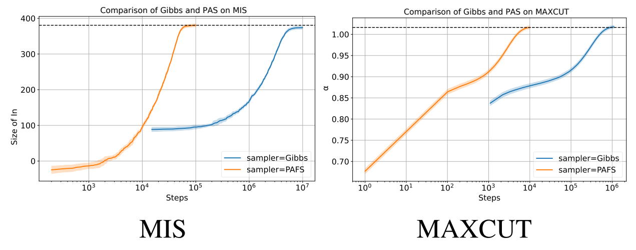

Figure 4. Sample efficiency comparison using different samplers.

5.4.2. τ SCHEDULE, MORE CHAINS/LONGER CHAINS

We provide the ablation studies on different hyper-parameters for iSCO. Results in Figure 2 and 3 are obtained after we run MIS and TSP tasks on the largest graphs.

Chain length and # chains. In general the more the total steps T the better the results should be, so as the number of Markov chains that run in parallel. We mainly study to what extent can we reduce the number of steps or number of chains required to get a reasonably good result. We can see from the first column of Figure 2, under the same annealing schedule for different T, the chain with 10x shorter length is still performing competitively compared to the other baselines in previous studies. This means one can further improve the speed-quality trade-off of iSCO. Also note that 1 chain is usually good enough to obtain reasonably

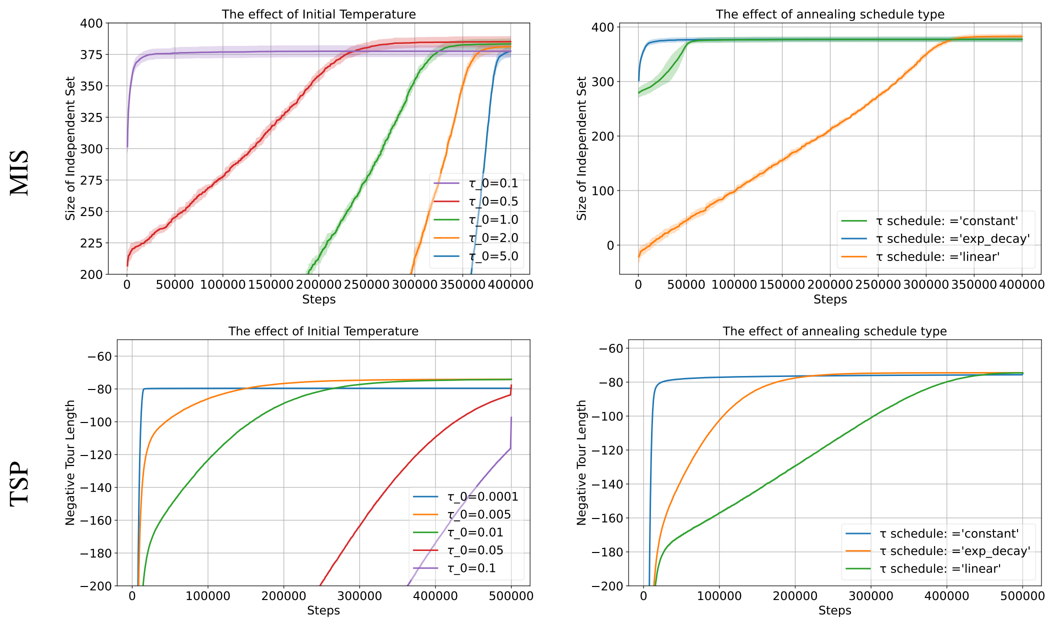

Figure 3. Ablation study on (left) initial temperature and (right) temperature annealing schedule.

good results. However despite the fact that more chains would help, the gains becomes marginal. One possible solution is to increase the diversity between different chains, which we will further investigate in future works.

τ **schedule** One potential headache in using iSCO is the setup of the initial temperature and the annealing schedule. As in Figure 3, we can see that overall the results would be dependent on the initial temperature, but the sampler is able to achieve good results in a range of initial temperatures. In practice we only did 2-3 binary search to identify a reasonably good temperature range. For annealing schedule, we find that generally the schedule has a major impact on the convergence speed, but after sufficiently many steps the final solution is not that different.

6. Limitation

Despite the effectiveness of iSCO on the problems we have considered, there are still many limitations of the current treatment. Specifically in the following situations:

- Black-box optimization: Currently the approximated gradient based samplers rely on the known form of objective function. When this becomes a black-box function, iSCO reduces to the vanilla SA which can be inefficient. One potential workaround is to learn a surrogate function and optimize on it, as inspired by (Wang et al., 2022).
- Difficult constraints: The current EBM formulation relies on the penalty form of the original problem. When it becomes nontrivial to find even just a feasible solution, the generic formulation of iSCO would probably fail.

Nevertheless, due to the appealing properties of iSCO, the surge of new sampling algorithms in discrete spaces and the ability of leveraging modern accelerators, we hope to bring the attention on sampling approaches back to this topic and improve further in future works.

7. Conclusion

We have shown that combining recently improved MCMC methods for discrete spaces with parallel neighborhood exploration on accelerators makes the generic sampling approach highly competitive on a wide range of CO problems. In fact, we find that the sampling approach often yields a superior speed-quality trade-off compared to the recent data-driven approaches. As a results, we encourage the future works on learning for CO to carefully calibrate the efficiency against this simple, generic and light-weight approach.

Meanwhile, the generic sampling approach also leaves interfaces for problem specific design. The discrete Langvein dynamics (6) allows for customized neighborhoods $N(x)$ and edge weights $c(x, y)$. In this work, we only considered the 1-Hamming ball neighborhood and a uniform edge weight. We believe that more sophisticated choices grounded in a theoretical analysis, or via learning based approaches, can further improve the solving effectiveness.

The current work represents only a tentative first step to developing efficient sampling based algorithms for discrete optimization. Future work involves extending this line of work to constrained programming problems, as well as tighter integration with learning approaches or search strategies.

References

Ahn, S., Seo, Y., and Shin, J. Learning what to defer for maximum independent sets. In *International Conference on Machine Learning*, pp. 134–144. PMLR, 2020.

Bilbro, G., Mann, R., Miller, T., Snyder, W., van den Bout, D., and White, M. Optimization by mean field annealing. *Advances in neural information processing systems*, 1, 1988.

Böther, M., Kißig, O., Taraz, M., Cohen, S., Seidel, K., and Friedrich, T. What's wrong with deep learning in tree search for combinatorial optimization. *arXiv preprint arXiv:2201.10494*, 2022.

Braunstein, A., Mézard, M., and Zecchina, R. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

Černỳ, V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

Chandy, J. A. and Banerjee, P. Parallel simulated annealing strategies for vlsi cell placement. In *Proceedings of 9th International Conference on VLSI Design*, pp. 37–42. IEEE, 1996.

Chen, X. and Tian, Y. Learning to perform local rewriting for combinatorial optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Chen, Y.-L. and Ke, Y. Multi-objective VAr planning for large-scale power systems using projection-based two-layer simulated annealing algorithms. *IEE Proceedings-Generation, Transmission and Distribution*, 151(4):555–560, 2004.

Croes, G. A. A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812, 1958.

Dai, H., Chen, X., Li, Y., Gao, X., and Song, L. A framework for differentiable discovery of graph algorithms. 2020.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., and Rousseau, L.-M. Learning heuristics for the tsp by policy gradient. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15*, pp. 170–181. Springer, 2018.

Dong, J. and Tong, X. T. Replica exchange for non-convex optimization. *J. Mach. Learn. Res.*, 22:173–1, 2021.

Dowsland, K. A. and Thompson, J. Simulated annealing. *Handbook of natural computing*, pp. 1623–1655, 2012.

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

Erdős, P., Rényi, A., et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Fu, Z.-H., Qiu, K.-B., and Zha, H. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7474–7482, 2021.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. Exact combinatorial optimization with graph convolutional neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Grathwohl, W., Swersky, K., Hashemi, M., Duvenaud, D., and Maddison, C. J. Oops I took a gradient: Scalable sampling for discrete distributions. *arXiv preprint arXiv:2102.04509*, 2021.

Gumbel, E. J. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

Gupta, P., Gasse, M., Khalil, E., Mudigonda, P., Lodi, A., and Bengio, Y. Hybrid models for learning to branch. *Advances in neural information processing systems*, 33: 18087–18097, 2020.

Hamze, F. and de Freitas, N. From fields to trees. *arXiv preprint arXiv:1207.4149*, 2012.

Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. 1970.

Hespe, D., Schulz, C., and Strash, D. Scalable kernelization for maximum independent sets. *ACM Journal of Experimental Algorithmics*, 24(1):1.16:1–1.16:22, 2019. doi: 10.1145/3355502. URL https://doi.org/10.1145/3355502.

Hoos, H. H. and Stützle, T. Satlib: An online resource for research on SAT. *Sat*, 2000:283–292, 2000.

Hopfield, J. J. and Tank, D. W. "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

Iba, Y. Extended ensemble Monte Carlo. *International Journal of Modern Physics C*, 12(05):623–656, 2001.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. Optimization by simulated annealing: An experimental evaluation; part I, graph partitioning. *Operations research*, 37(6):865–892, 1989.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Operations research*, 39(3):378–406, 1991.

Joshi, C. K., Laurent, T., and Bresson, X. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Jwo, W.-S., Liu, C.-W., Liu, C.-C., and Hsiao, Y.-T. Hybrid expert system and simulated annealing approach to optimal reactive power planning. *IEE Proceedings-Generation, Transmission and Distribution*, 142(4):381–385, 1995.

Karalias, N. and Loukas, A. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33:6659–6672, 2020.

Karypis, G. and Kumar, V. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th annual ACM/IEEE design automation conference*, pp. 343–348, 1999.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. Optimization by simulated annealing. *science*, 220(4598): 671–680, 1983.

Kool, W., Van Hoof, H., and Welling, M. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Gwon, Y., and Min, S. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Lamm, S., Sanders, P., Schulz, C., Strash, D., and Werneck, R. F. Finding near-optimal independent sets at scale. *J. Heuristics*, 23(4):207–229, 2017. doi: 10.1007/s10732-017-9337-x. URL https://doi.org/10.1007/s10732-017-9337-x.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

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

Ma, Y.-A., Chen, Y., Jin, C., Flammarion, N., and Jordan, M. I. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences*, 116(42): 20881–20885, 2019.

Manchanda, S., Mittal, A., Dhawan, A., Medya, S., Ranu, S., and Singh, A. Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs. *Advances in Neural Information Processing Systems*, 33: 20000–20011, 2020.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Nair, V., Bartunov, S., Gimeno, F., von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Wang, P., et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

Nazi, A., Hang, W., Goldie, A., Ravi, S., and Mirhoseini, A. Gap: Generalizable approximate graph partitioning framework. *arXiv preprint arXiv:1903.00614*, 2019.

Neal, R. M. Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6(4):353–366, 1996.

Qiu, R., Sun, Z., and Yang, Y. Dimes: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, 2022.

Ramanujam, J. and Sadayappan, P. Mapping combinatorial optimization problems onto neural networks. *Information sciences*, 82(3-4):239–255, 1995.

Ravanbakhsh, S. and Greiner, R. Perturbed message passing for constraint satisfaction problems. *The Journal of Machine Learning Research*, 16(1):1249–1274, 2015.

Sechen, C., Braun, D., and Sangiovanni-Vincentelli, A. Thunderbird: A complete standard cell layout package. *IEEE Journal of Solid-State Circuits*, 23(2):410–420, 1988.

Seçkiner, S. U. and Kurt, M. A simulated annealing approach to the solution of job rotation scheduling problems. *Applied Mathematics and Computation*, 188(1): 31–45, 2007.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Sun, H., Dai, H., Xia, W., and Ramamurthy, A. Path auxiliary proposal for MCMC in discrete space. In *International Conference on Learning Representations*, 2021.

Sun, H., Dai, H., Dai, B., Zhou, H., and Schuurmans, D. Discrete Langevin sampler via Wasserstein gradient flow. *arXiv preprint arXiv:2206.14897*, 2022a.

Sun, H., Dai, H., and Schuurmans, D. Optimal scaling for locally balanced proposals in discrete spaces. *arXiv preprint arXiv:2209.08183*, 2022b.

Sun, H., Guha, E. K., and Dai, H. Annealed training for combinatorial optimization on graphs. *arXiv preprint arXiv:2207.11542*, 2022c.

Swendsen, R. H. and Wang, J.-S. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical review letters*, 58(2):86, 1987.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

Tavakkoli-Moghaddam, R., Safaei, N., Kah, M., and Rabbani, M. A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing. *Journal of the Franklin Institute*, 344(5):406–425, 2007.

Thompson, J. M. and Dowsland, K. A. A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, 25(7-8):637–648, 1998.

Toenshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Graph neural networks for maximum constraint satisfaction. *Frontiers in artificial intelligence*, 3:580607, 2021.

Van Breedam, A. Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86(3):480–490, 1995.

van Hemmen, J. L. Spin-glass models of a neural network. *Physical Review A*, 34(4):3435, 1986.

Wang, C., Hyman, J. D., Percus, A., and Caflisch, R. Parallel tempering for the traveling salesman problem. *International Journal of Modern Physics C*, 20(04):539–556, 2009.

Wang, H. P., Wu, N., Yang, H., Hao, C., and Li, P. Unsupervised learning for combinatorial optimization with principled objective relaxation. In *Advances in Neural Information Processing Systems*, 2022.

Wong, D., Leong, H. W., and Liu, H. *Simulated annealing for VLSI design*, volume 42. Springer Science & Business Media, 2012.

Xu, K., Boussemart, F., Hemery, F., and Lecoutre, C. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artificial intelligence*, 171(8-9):514–534, 2007.

Zanella, G. Informed proposals for local MCMC in discrete spaces. *Journal of the American Statistical Association*, 115(530):852–865, 2020.

Zhang, R., Liu, X., and Liu, Q. A Langevin-like sampler for discrete distributions. In *International Conference on Machine Learning*, pp. 26375–26396. PMLR, 2022.

A. Implementation details

In this section we provide the actual energy function we used for each of the problems we experimented in the main paper. For a graph $G = (V, E)$ we label the nodes in V from 1 to d. The adjacency matrix is represented as A. For a weighted graph we simply let A_{ij} denote the edge weight between node i and j. For constraint problems, we follow Sun et al. (2022c) to select penalty coefficient λ as the minimum value of λ such that $x^* := \operatorname{argmin} f(x)$ is achieved at x^* satisfying the original constraints. Such a choice of the coefficient guarantees the target distribution converges to the optimal solution of the original CO problems while keeping the target distribution as smooth as possible.

MIS The MIS has the integer programming formulation as

$$\min_{x \in \{0,1\}^d} -\sum_{i=1}^{d} c_i x_i, \quad \text{s.t. } x_i x_j = 0,\ \forall (i,j) \in E \tag{18}$$

We use the corresponding energy function in the following quadratic form:

$$f(x) := -c^T x + \lambda \frac{x^T A x}{2} \tag{19}$$

In our experiments c equals to 1 and we use $\lambda = 1.0001$. In post processing, we iteratively go through all nodes x_i for $i = 1, ..., d$. If there exists $x_j = 1$ for $(x_i, x_j) \in E$, we flip its value $x_j = 0$. After post processing, the state x is guaranteed to be feasible in the original MIS problem.

Max clique The max clique problem is equivalent to MIS on the dual graph. In our experiments c equals to 1.

$$\min_{x \in \{0,1\}^d} -\sum_{i=1}^{d} c_i x_i, \quad \text{s.t. } x_i x_j = 0,\ \forall (i,j) \notin E \tag{20}$$

The energy function is

$$f(x) := -c^T x + \frac{\lambda}{2}\left(\mathbf{1}^\top x \cdot (\mathbf{1}^\top x - 1) - x^T A x\right) \tag{21}$$

In our experiments c equals to 1 and we use $\lambda = 1.0001$. In post processing, we iteratively go through all nodes x_i for $i = 1, ..., d$. If there exists $x_j = 1$ for $(x_i, x_j) \notin E$, we flip its value $x_j = 0$. After post processing, the state x is guaranteed to be feasible in the original MIS problem.

Maxcut We optimize the following problem:

$$\min_{x \in \{-1,1\}^d} - \sum_{(i,j) \in E} A_{i,j} \left(\frac{1 - x_i x_j}{2}\right) \tag{22}$$

Note that for simplicity each dimension of x is selected from $\{-1, 1\}$. To represent the corresponding energy function for $x \in \{0, 1\}^d$, we have

$$f(x) := - \sum_{(i,j) \in E} A_{i,j} \left(\frac{1 - (2x_i - 1)(2x_j - 1)}{2}\right) \tag{23}$$

In our experiments A_{ij} equals to 1. Since the problem is always feasible, the post processing is identity map.

Balanced graph partition We find the following objective for balanced graph partition gives the best result:

$$f(x) := \sum_{s=1}^{k} \sum_{(i,j) \in E} \mathbb{I}\left(x_i \neq x_j \&\& (x_i = s || x_j = s)\right) + \sum_{s=1}^{k} \left(d/k - \sum_{i=1}^{d} \mathbb{I}(x_i = s)\right)^2 \tag{24}$$

where k is the number of partitions. Since the problem is always feasible, the post processing is identity map.

TSP It is complicated to write TSP in a general integer programming form, but fortunately we only need to be able to write the energy function for a given valid solution. For TSP in 2D space any permutation of visiting orders is a valid one, so we simply use the energy function as

$$f(x) := -\sum_{i=1}^{d} L_2(x_i, x_{i+1}) \tag{25}$$

where x is a valid permutation and we overload the notation to let $x_{d+1} = x_1$. $L_2(i, j)$ denotes the Euclidean distance between node i and j. As long as we can guarantee that the sampler makes a valid jump from one state to another, we can effectively explore the feasible space for the minimum tour length. Since the problem is always feasible, the post processing is identity map.

B. More experimental details

Table 7. Synthetic data statistics.

Name	MIS		max clique	maxcut		TSP		
	ER-[700-800]	ER-[9000-11000]	RB	ER	BA	TSP-500	TSP-1000	TSP-10000
Max # nodes	800	10,915	475	1,100	1,100	500	1,000	10,000
Max # edges	47,885	1,190,799	90,585	91,239	4,384	250,000*	1,000,000*	100,000,000*
# Test instances	128	16	500	1,000	1,000	128	128	16

*conceptually fully connected

Table 8. Real-world data statistics.

Name	MIS	max clique	maxcut	balanced graph partition				
	SATLIB	Twitter	Optsicom	MNIST	VGG	ALEXNET	RESNET	INCEPTION
Max # nodes	1,347	247	125	414	1,325	798	20,586	27,114
Max # edges	5,978	12,174	375	623	2,036	1,198	32,298	40,875
# Test instances	500	196	10	1	1	1	1	1

Here we provide more details on the experiments. Firstly, the statistics of synthetic datasets, including the maximum number of nodes/edges in a graph, and the number of test instances, are provided in Table 7. Corresponding statistics of real-world graphs are in Table 8.

To get the quality metrics we run all the experiments on V100 GPUs. When reporting the runtime, we use either GTX 1080Ti or A100, depending on the actual device used in the benchmark. As the hardware keeps improving, we would favor the approaches that would benefit more from the growth of the hardware. As we demonstrated, iSCO could be one of them.

B.1. MIS

For MIS on ER-9000-11000 graphs, we use 400k steps; for SATLIB graphs we use 1M steps; for ER-700-800 graphs we use 200k steps. We use the penalty coefficient $\lambda = 1.0001$ that is slightly larger than 1, which is good enough to guarantee the feasibility of the solution. This is inspired from the derivation of minimum required penalty from Sun et al. (2022c). We further conduct experiments with different $\lambda = \{1.1, 1.01, 1.001, 1.0001, 1.00001\}$ and we report the results on ER-[9000-11000] in Table 9. Some randomness is expected as iSCO is a stochastic algorithm. Other than that we can see that for MIS the solution is pretty robust to many choices of λ.

Table 9. Performance of iSCO with different hyperparameter λ

penalty coefficient λ	1.1	1.01	1.001	1.0001	1.00001
independent set size	383.8125	384.1212	384.2496	385.1249	384.3125

Gurobi: For Gurobi, we reformulate the optimization problem for MIS in (18) as:

$$\min_{x \in \{0,1\}^d} -\sum_{i=1}^{d} c_i x_i, \quad \text{s.t. } x_i + x_j \leqslant 1,\ \forall (i,j) \in E \tag{26}$$

and for Gurobi-clique, we reformulate the problem as:

$$\min_{x \in \{0,1\}^d} -\sum_{i=1}^{d} c_i x_i, \quad \text{s.t. } \sum_{k \in C} x_k \leqslant 1,\ \forall C \in \mathcal{C} \tag{27}$$

In (27), we use greedy clique partition to obtain a clique partition $\mathcal{C} = \{C_1, ..., C_m\}$ of the nodes, such that, in each clique C_j, at most one node can be selected in an independent set. The clique formulation provides a stronger linear relaxation compared to the edge formulation in (26), hence has better performance. There

B.2. Maxcut and balanced graph partition

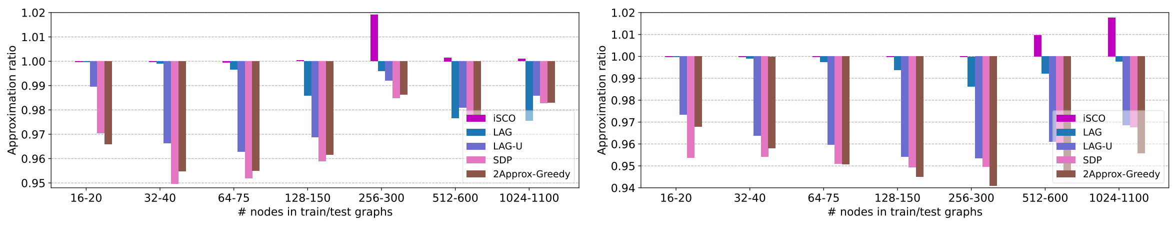

Figure 5. Results of maxcut on ER (left) and BA (right) graphs.

For maxcut, in most cases a chain length around 50k would be good enough to obtain super-Gurobi results on both of the random graph types. Since this is an unconstrained optimization problem, we find iSCO or classical sampling approaches in general are very efficient. The full results on both types of random graphs are displayed in Figure 5.

Balanced graph partition is a challenging problem where iSCO requires 800k steps to match the performance of alternatives on the largest computation graph instance. In this case we also tune the λ and find it achieves the best results when $\lambda = 0.001$.

B.3. TSP

We leverage 2-OPT to define the neighborhood structure $\mathcal{N}(x)$ for the current tour solution x. The selection of a new sample is done in the following conceptual process:

- Select a node x and its next node y in the tour;
- With probability $k/(k+1)$, select a node a that is in the knn graph of x; or with probability $1/(k+1)$, select a random node other than x. Denote the selected node as a and its neighbor as b.
- Do the 2-OPT operation between edges $x \to y$ and $a \to b$.

In practice the above process is converted into the unnormalized probability of each 2-OPT option, where the scoring of each option is done in parallel.

We follow the baseline Qiu et al. (2022) to use $k = 50$, though to make the Markov chain reversible, conceptually we are dealing with the entire fully (probabilistically) connected graph.

C. Discrete Sampler and Annealing Algorithm

Path Auxiliary Sampler

Given path length prior $\alpha(\cdot)$ and current state x, a MH step is:

Table 10. Results of TSP, where the numbers of baselines are directly taken from Qiu et al. (2022) and runtime is calibrated on GTX 1080Ti. In addition to the approaches mentioned in Table 1, some are implemented with Beam Search (BS), Active Search (AS) or Monte Carlo Tree Search (MCTS). [a]Results of POMO are obtained with different variants as detailed in Fu et al. (2021).

Method	Type	TSP-500			TSP-1000			TSP-10000		
		Length↓	Drop↓	Time↓	Length↓	Drop↓	Time↓	Length↓	Drop↓	Time↓
Concorde	OR	16.55*	-	37.66m	23.12*	-	6.65h	N/A	N/A	N/A
Gurobi	OR	16.55	0.00%	45.63h	N/A	N/A	N/A	N/A	N/A	N/A
LKH-3 (default)	OR	16.55	0.00%	46.28m	23.12	0.00%	2.57h	71.77*	-	8.8h
Farthest Insertion	OR	18.30	10.57%	0s	25.72	11.25%	0s	80.59	12.29%	6s
EAN (Deudon et al., 2018)	RL+S+2-OPT	23.75	43.57%	57.76m	47.73	106.46%	5.39h	N/A	N/A	N/A
AM (Kool et al., 2018)	RL+BS	19.53	18.03%	21.99m	29.90	29.23%	1.64h	129.40	80.28%	1.81h
GCN (Joshi et al., 2019)	SL+G	29.72	79.61%	6.67m	48.62	110.29%	28.52m	N/A	N/A	N/A
	SL+BS	30.37	83.55%	38.02m	51.26	121.73%	51.67m	N/A	N/A	N/A
POMO[a] (Kwon et al., 2020)	RL+AS	19.24	16.25%	12.80h	N/A	N/A	N/A	N/A	N/A	N/A
	RL+AS	19.35	16.92%	16.19h	N/A	N/A	N/A	N/A	N/A	N/A
	RL+AS	24.54	48.22%	11.61h	49.56	114.36%	63.45h	N/A	N/A	N/A
Att-GCN (Fu et al., 2021)	SL+MCTS	16.97	2.54%	2.20m	23.86	3.22%	4.10m	74.93	4.39%	21.49m
DIMES (Qiu et al., 2022)	RL+G	18.93	14.38%	0.97m	26.58	14.97%	2.08m	86.44	20.44%	4.65m
	RL+AS+G	17.81	7.61%	2.10h	24.91	7.74%	4.49h	80.45	12.09%	3.07h
	RL+S	18.84	13.84%	1.06m	26.36	14.01%	2.38m	85.75	19.48%	4.80m
	RL+AS+S	17.80	7.55%	2.11h	24.89	7.70%	4.53h	80.42	12.05%	3.12h
	RL+MCTS	16.87	1.93%	2.92m	23.73	2.64%	6.87m	74.63	3.98%	29.83m
	RL+AS+MCTS	16.84	1.76%	2.15h	23.69	2.46%	4.62h	74.06	3.19%	3.57h
iSCO (Ours)	Sampling	**16.64**	**0.54%**	6.94m	**23.33**	**0.91** %	7.94m	**74.02**	**3.14%**	1.01h

* indicates the baseline for computing the performance drop.

1. Sample a path length $L \sim \alpha(L)$.
2. Sample indices $\mathcal{J} = \{j_1, ..., j_L\}$ from Categorical(w_j) without replacement, where the weight

$$w_j = \sum_{s \neq x_j} g\left(\frac{p(x_{-j}, s)}{p(x)}\right) \tag{28}$$

Denote the probability to choose $\mathcal{J}$ from x as $q_x(\mathcal{J})$.

3. For $j \notin \mathcal{J}$, set $y_j = x_j$. For $j \in \mathcal{J}$, sample $y_j \sim q_x^j(s)$, for $s \neq x_j$ and

$$q_x^j(s) = g\left(\frac{p(x_{-j}, s)}{p(x)}\right) / \sum_{s \neq x_j} g\left(\frac{p(x_{-j}, s)}{p(x)}\right) \tag{29}$$

4. Accept y with probabilty

$$A(x, \mathcal{J}, y) = \min\left\{1, \frac{p(y)q_y(\mathcal{J})\prod_{j\in\mathcal{J}} q_y^j(x_s)}{p(x)q_x(\mathcal{J})\prod_{j\in\mathcal{J}} q_x^j(y_s)}\right\} \tag{30}$$

The no-replacement categorical sampling for $\mathcal{J}$ and y_j can be efficiently implemented via Gumbel-Max trick (Gumbel, 1954). In practice, we set $\alpha(\cdot)$ as a Poisson like distribution $\alpha(L) \propto \frac{\mu^L e^{-\mu}}{L!} 1_{\{L>0\}}$. Following Sun et al. (2022b), we tune the scale μ by

$$\mu \leftarrow \text{clip}(\mu + 0.001 * (\bar{A} - 0.574), \min = 1, \max = d) \tag{31}$$

which makes the average acceptance rate being 0.574. Here, $\bar{A}$ is the batch average acceptance rate. When the average acceptance rate is too large, we increase μ, hence increase the path length, and vice versa.

Annealing Algorithm

Combining the path auciliary sampler above and the standard annealing algorithm, we obtain our algorithm for solving combinatorial optimization problems. Given an instance, we have the energy function $f(x)$ as derived in Section A. For temperature τ, the corresponding target distribution is $\pi_\tau(x) \propto \exp(-\frac{f(x)}{\tau})$. The sampling algorithm is summarized in Algorithm 1. The set $\{z_i\}_{i=1}^m$ is the collected solutions.