Deterministic Policy Gradient for RL-Based MaxCut on Sparse Ising Graphs

Sarah Ameur
School of Computer Science
McGill University, Montréal, Canada
sarah.ameur@mail.mcgill.ca

Fadi Younes
School of Computer Science
McGill University, Montréal, Canada
fadi.younes@mail.mcgill.ca

Habiba Abdelrehim
School of Computer Science
McGill University, Montréal, Canada
habiba.abdelrehim@mail.mcgill.ca

Abstract

This study implements the deterministic REINFORCE algorithm of [4] for solving MaxCut on sparse Ising graphs using a policy-gradient approach with parallel MCMC sampling. In light of the missing architectural specifications, a systematic search was conducted to find optimal MLP and training parameters. This work proposes an initial Metropolis–Hastings burn-in to accelerate convergence; benchmarks performance against Metropolis–Hastings and pure greedy local search; and presents the first public deterministic REINFORCE code. Tests on two datasets achieved comparable MaxCut scores despite limited training time.

1 Introduction

1.1 Motivation

Given a graph $G = (V, E)$ with $n = |V|$ vertices, the MaxCut problem seeks to split V into two subsets so that the number of edges between them is maximized. This problem is NP-hard: the 2^n possible splits make exhaustive search impractical for even moderate n. Equivalently, MaxCut on G corresponds to finding the ground state of an antiferromagnetic Ising model (see Section 2). In sparse graphs ($|E| \ll n^2$), most random spin flips do not modify the cut size, providing scarce feedback for basic state-space search heuristics. Deterministic policy-gradient reinforcement learning can train a policy neural network to choose spin flips strategically, bypassing the need to try all 2^n configurations.

1.2 Contribution

The deterministic REINFORCE algorithm from [4] is implemented, with a Metropolis–Hastings burn-in introduced at the start of each training trajectory to escape shallow local minima and shorten training time. This adaptation is compared against classical Metropolis–Hastings and pure greedy local search. A systematic search of the optimal hyperparameters is conducted for each graph. All code and best hyperparameters are made public.

2 Background

In statistical mechanics, the Ising model describes n magnetic dipole moments with spins $\sigma_i \in \{\pm 1\}$ on the vertices of a graph $G = (V, E)$—each edge $(i, j) \in E$ represents a pair of neighboring spins that interact— in thermal equilibrium at temperature T. The full spin configuration is $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n) \in \{\pm 1\}^n$, and its Boltzmann probability is

$$P(\boldsymbol{\sigma}) \propto \exp\Big[-H(\boldsymbol{\sigma})/(k_B T)\Big]. \tag{1}$$

Workshop on Instruction Tuning and Instruction Following at NeurIPS 2023.

Here k_B is Boltzmann's constant. In the antiferromagnetic variant, coupling coefficients $J_{ij} < 0$ favor neighbors of opposite spins. The Hamiltonian,

$$H(\boldsymbol{\sigma}) = -\sum_{(i,j)\in E} J_{ij}\,\sigma_i\,\sigma_j,$$

quantifies the total interaction energy. Setting all weights to $w_{ij} \equiv -J_{ij} = 1$, one finds that minimizing H is equivalent to maximizing the cutsize of the graph:

$$\mathrm{Cut}(\boldsymbol{\sigma}) = \frac{1}{2}\sum_{(i,j)\in E} \big(1 - \sigma_i\sigma_j\big).$$

3 Related Work

Reinforcement learning has been applied to NP-hard graph tasks via Ising-model formulations. Bello et al. [1] used policy gradient with pointer networks (sequence models that generate solutions by pointing to input nodes) to solve vehicle routing problems. Their policies outperformed simple heuristics but training stalled when rewards were too sparse. Khalil et al. [3] employed deep Q-learning for MaxCut and related tasks. The method surpassed random and greedy searches but yielded high variance, sensitivity to hyperparameters, and difficulty scaling to large, sparse Ising graphs.

4 Methodology

4.1 Deterministic REINFORCE

The deterministic REINFORCE algorithm proposed by [4] is extended here. The policy network $\pi_\theta(\sigma_{t+1} \mid \sigma_t)$ is a four-layer MLP with n input neurons encoding $p(\sigma_{i,t} = -1)$ for each spin in $\sigma_t \in \{\pm 1\}^n$, two hidden layers of size n with ReLU activations, and n Sigmoid output neurons yielding $p(\sigma_{i,t+1} = -1 \mid \sigma_t)$. Since the parameters of the MLP were not specified in the original study, two hidden layers of size n are chosen so the network can learn spin interactions. Actions are obtained by thresholding the output probabilities at 0.5:

$$\sigma_{i,t+1} = \begin{cases} -1 & \text{if } p_\theta(\sigma_{i,t+1} = -1 \mid \sigma_t) > 0.5, \\ +1 & \text{otherwise.} \end{cases}$$

Then, the output state s_{t+1} undergoes a partial greedy local search: a small fraction of spins are flipped sequentially, and each flip is kept only if it immediately lowers the Hamiltonian. To increase sample diversity and avoid shallow local minima, each trajectory begins with a Metropolis burn-in: random single-spin flips are proposed and accepted with probability $\exp(-\Delta E/T_{\text{burn}})$, allowing occasional Hamiltonian-increasing moves so the spin configuration explores a wider range of energy levels at the burn-in temperature T_{burn} before the main search. This initial sampling can improve MaxCut by up to 500—an approach not explored in the original study.

$$r_t = H(\sigma_t) - H(\sigma_{t+1}), \quad \nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{T-1} r_t\,\nabla_\theta \log \pi_\theta(\sigma_{t+1} \mid \sigma_t)\Big].$$

The reward r_t measures the energy decrease at each step, and parameters are updated via gradient ascent on $J(\theta)$.

4.2 Datasets

Two datasets are considered. The *Optsicom* dataset [2] includes 10 dense graphs with 125 nodes and 750 edges (about 10% of all possible edges), which makes it easier to find low-energy regions. The *GSet* dataset [5] contains sparse graphs of size 800–10'000 nodes with only 0.1–1.5% of possible edges, yielding a much more challenging search space with fewer spin interactions.

Algorithm 1 Adapted deterministic REINFORCE

Input: epochs M, learning rate α, chains N, trajectory length T, spin count n
Initialize policy network π_θ with input dimension n, two hidden ReLU layers of size n, and Sigmoid output of size n
for epoch $1 \to M$ **do**
    Sample initial configurations $\{\sigma_0^i\}_{i=1}^N \subset \{\pm 1\}^n$
    Perform Metropolis burn-in: spin flips accepted with probability $\propto \exp(-\Delta E/T)$
    for $t = 0 \to T-1$ and $i = 1 \to N$ **do**
        Compute $p_\theta(\cdot \mid \sigma_t^i)$ via forward pass
        Determine σ_{t+1}^i by thresholding p_θ
        Apply greedy local search $LS(\sigma_{t+1}^i)$ to reduce energy
        Compute reward $r_t^i = H(\sigma_t^i) - H(\sigma_{t+1}^i)$
    end for
    Compute returns $R_t^i = \sum_{t'=t}^{T-1} r_{t'}^i$
    Update $\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(\sigma_{t+1}^i \mid \sigma_t^i)$
end for
Output: best spin configuration

4.3 Hyperparameters tuning

Training runs in PyTorch on Google Colab with a high-RAM T4 GPU and CUDA acceleration. Batch sizes of 32, 64, and 128 enable parallel MCMC sampling of multiple trajectories, broadening exploration of the 2^n configuration space and speeding up convergence. The hyperparameters tuned for each graph are number of epochs M, batch size N, trajectory length (`traj`), learning rate `lr`, number of Metropolis–Hastings steps (`steps`), flip fraction `flip_ratio`, burn-in temperature T_{burn}, and number of flip trials `tries`. Here, `steps` specifies how many single-spin flips are attempted during each thermal sampling trajectory. Details appear in Algorithm 1. Each configuration is repeated independently several times, and the highest MaxCut score is reported.

4.4 Benchmarks

Two benchmark experiments are conducted. In the first, the adapted deterministic REINFORCE method is compared to a Metropolis–Hastings baseline. This stochastic local optimization baseline uses the same burn-in temperature T_{burn} and accepts energy-decreasing spin flips with Boltzmann probability as in Equation 1. In the second experiment, a pure greedy local search tries all possible single-spin flips. Each flip is kept only if it reduces the Hamiltonian. This provides an estimate of the attainable MaxCut without any policy network.

5 Experiment

Table 1 reports MaxCut scores on the dense Optsicom graphs, comparing the adapted deterministic REINFORCE method, Metropolis–Hastings baseline, pure greedy local search, and the original Lu&Liu [2023] results. In this study, the training loops ran for 10–30 minutes on each graph except G70 ($\approx$ 50 minutes), limited by computational resources, whereas the authors allotted a full hour per graph. This reduced runtime explains the lower MaxCut results. Additionally, an overly long initial Metropolis burn-in could have hurt learning by causing too much random exploration before policy updates. Achieved MaxCut values correspond to 89–96% of those reported in the original paper.

Table 2 shows MaxCut performance on the sparse GSet graphs under the same four methods. Attained MaxCut values range from 77% to 89% of those reported in the original paper, with the lowest relative performance on G70, the sparse graph with the largest node count.

Table 3 lists the optimal hyperparameters selected for each graph in our adapted deterministic REINFORCE experiments.

Table 1: MaxCut results on dense Optsicom graphs

Graph	Adapted Det. REINF. (Ours)	Metropolis–Hastings	Pure greedy LS	Lu & Liu [2023]
G54100	104	110	86	110
G54200	104	112	86	112
G54300	102	106	78	106
G54400	104	114	84	114
G54500	100	112	82	112
G54600	104	110	86	110
G54700	104	112	88	112
G54800	102	108	76	108
G54900	102	110	78	110
G541000	104	112	82	112

Table 2: MaxCut results on sparse GSet graphs

Graph	Adapted Det. REINF. (Ours)	Metropolis–Hastings	Pure greedy LS	Lu & Liu [2023]
G14	2725	3015	2925	3064
G15	2707	2983	2915	3050
G22	11834	13031	12839	13359
G49	5128	5940	6000	6000
G50	5100	5830	5880	5880
G55	8660	9941	9359	10298
G70	7337	9159	8496	9583

Table 3: Optimal hyperparameters used during training in our adapted deterministic REINFORCE

Graph	M	N	traj	`lr`	`flip_ratio`	`tries`	T_{burn}	`steps`
G54100	950	128	150	9e-06	0.01	10	10.0	30
G54200	500	32	30	1e-04	0.01	10	10.0	10
G54300	1500	128	200	3e-05	0.02	10	10.0	20
G54400	950	64	150	9e-06	0.01	10	10.0	30
G54500	950	64	150	9e-06	0.01	10	10.0	30
G54600	950	64	150	9e-06	0.01	10	10.0	30
G54700	950	64	150	9e-06	0.01	10	10.0	130
G54800	950	64	150	9e-06	0.01	10	10.0	130
G54900	950	64	150	9e-06	0.01	10	10.0	30
G541000	800	64	150	9e-06	0.01	10	10.0	30
G14	950	64	150	2e-05	0.01	10	1.0	30
G15	200	64	200	1e-03	0.125	20	5.0	20
G22	600	32	50	1e-06	0.001	10	5.0	10
G49	600	16	50	1e-06	0.01	10	1.0	10
G50	600	16	50	1e-06	0.01	10	1.0	10
G55	600	16	50	1e-06	0.01	10	1.0	10
G70	600	16	50	1e-06	0.01	10	1.0	10

6 Conclusion

The adapted deterministic REINFORCE algorithm achieves competitive MaxCut scores on Optsicom and GSet with shorter training times. Actor–critic methods use a policy network together with a value network (critic) that predicts expected returns. The critic's reference value reduces gradient variance and better guides the policy network, which is can be helpful in sparse Ising graphs with many shallow minima. Future work could explore a deterministic actor–critic variant that still includes a Metropolis–Hastings burn-in at the start of each trajectory.

References

[1] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016. URL `https://arxiv.org/abs/1611.09940`.

[2] GRAFO Research Group. Optsicom benchmark instances. `https://grafo.etsii.urjc.es/optsicom/#instances`, 2025. Accessed: 2025-03-02.

[3] Elias B. Khalil, Hanjun Dai, Yiyuan Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6351–6361, 2017. URL `https://arxiv.org/abs/1704.01665`.

[4] Yicheng Lu and Xiao-Yang Liu. Reinforcement learning for ising model. 2023. URL `https://ml4physicalsciences.github.io/2023/files/NeurIPS_ML4PS_2023_248.pdf`.

[5] Yinyu Ye. GSet graph collection. `https://web.stanford.edu/~yyye/yyye/Gset/`, 2025. Accessed: 2025-03-03.