A Monte Carlo Policy Gradient Method with Local Search for Binary Optimization

Zaiwen Wen

Beijing International Center For Mathematical Research
Peking University

Joint work with Cheng Chen, Ruitao Chen, Tianyou Li, Ruicheng Ao

https://github.com/optsuite/MCPG

Our group

(a) 陈铖

(b) 陈锐韬

(c) 李天佑

(d) 敖睿成

Outline

Binary Optimization

Let f be arbitrary (even non-smooth) cost function:

$$\min \quad f(x), \quad \text{s.t.} \quad x \in \mathcal{B}_n = \{-1, 1\}^n.$$

- Example: maxcut problem on $G = (V, E)$

$$\max \quad \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j), \quad \text{s.t.} \quad x \in \{-1, 1\}^n.$$

- Example: maxSAT problem:

$$\begin{aligned} \max_{x\in\{-1,1\}^n} \quad & \sum_{c^i\in C_1} \max\{c_1^i x_1, c_2^i x_2, \cdots, c_n^i x_n, 0\}, \\ \text{s.t.} \quad & \max\{c_1^i x_1, c_2^i x_2, \cdots, c_n^i x_n, 0\} = 1, \quad \text{for } c^i \in C_2 \end{aligned}$$

- Binary optimization is NP-hard due to the combinatorial structure.

Algorithms for Solving Binary Optimization

- **Relaxation methods**
  - SDP-based approches for binary quadratic optimization.
- **Search methods**
  - Branch-and-bound, Cutting Plane.
  - Heurisitics for specified problems.
- **Learning methods:**
  - Supervised Learning: PtrNet, Neural Diving/Neural Branching.
    - Labels for hard problem instances is infeasible to be obtained.
  - Reinforcement Learning: Learning for MaxSAT, Neural Rewriter.
    - Regard the solving procedure as a game.
  - Unsupervised Learning: Erdos
    - Train neural networks in a differentiable and end-to-end manner.

Maxcut: 0.878 bounds

- For graph (V, E) and weights $w_{ij} = w_{ji} \geq 0$, the maxcut problem is

$$(Q) \quad \max_x \sum_{i<j} w_{ij}(1 - x_i x_j), \text{ s.t. } x_i \in \{-1, 1\}$$

- SDP relaxation

$$(SDP) \quad \max_{X \in S^n} \sum_{i<j} w_{ij}(1 - X_{ij}), \text{ s.t. } X_{ii} = 1, X \succeq 0$$

Compute the decomposition $X = V^\top V$, where $V = [v_1, v_2, \ldots, v_n]$

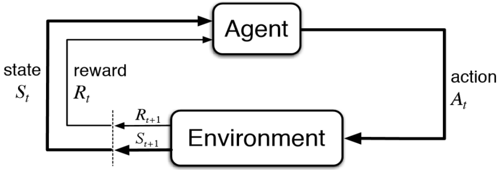
- Rounding: generate a vector r uniformly distributed on the unit sphere, i.e., $\|r\|_2 = 1$, set

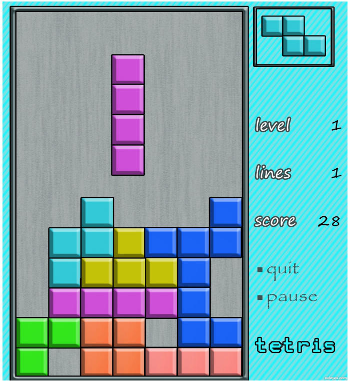
$$x_i = \begin{cases} 1 & v_i^\top r \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- Let $Z^*_{(SDP)}$ and $Z^*_{(Q)}$ be the optimal values of (SDP) and (Q)

$$E(W) \geq 0.878 Z^*_{(SDP)} \geq 0.878 Z^*_{(Q)}$$

Consider an infinite-horizon discounted Markov decision process (MDP), usually defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$;



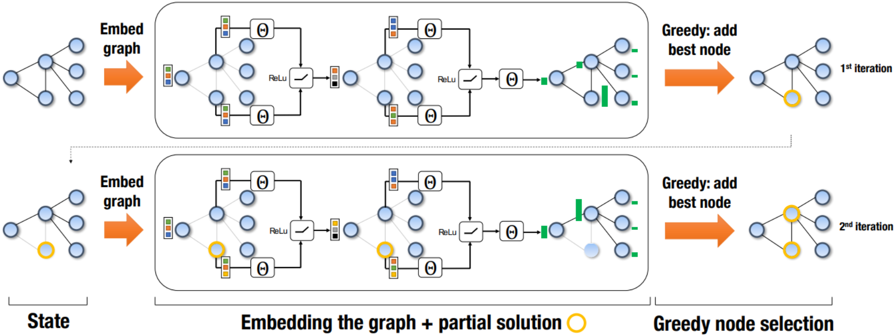


- The policy is supposed to maximize the total expected reward:

$$\max_{\pi} \; \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \text{ with } s_0 \sim \rho_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t).$$

S2V-DQN: Value-based algorithm for CO on graphs

- Key Observation: Q function in DQN is similar to the manually designed evaluation function in Greedy algorithms.
- A state is composed of a problem instance $\mathcal{P}$ and a partial solution.
- A generic greedy algorithm selects a node v to add next such that v maximizes the evaluation Q function.



Erdos Goes Neural

- The probability distribution $\mathcal{D}$ in $Erdos$ is learned by a GNN.
- A "good" probability distribution leads to higher quality solutions.

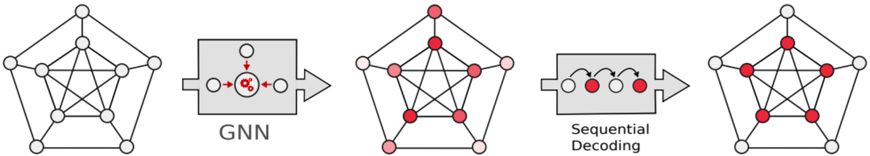


Figure: Illustration of the "Erdos goes neural" pipeline.

- Optimization on explicit formulation of the expectation.
- Maximum clique problem:

$$\ell(\mathcal{D}) = \gamma - (\beta + 1) \sum_{(v_i, v_j) \in E} w_{ij} p_i p_j + \frac{\beta}{2} \sum_{v_i \neq v_j} p_i p_j .$$

Outline

Parameterized Probabilistic Model

- **MCPG**: construct a parameterized model with parameter θ to output p_θ and generate $x \sim p_\theta$ by Monte Carlo sampling

- MCPG: optimization over the probabilistic space.
- Erdos: optimization on the expectation of objective function.

Probabilistic Approach

Let $\mathcal{X}^*$ be the set of optimal solutions and consider the distribution,

$$q^*(x) = \frac{1}{|\mathcal{X}^*|}\mathbf{1}_{\mathcal{X}^*}(x) = \begin{cases} \frac{1}{|\mathcal{X}^*|}, & x \in \mathcal{X}^*, \\ 0, & x \notin \mathcal{X}^*. \end{cases}$$

Motivation: Searching for optimal points $\mathcal{X}^*$ $\Rightarrow$ Constructing a distribution $p_\theta(x)$ converging to $q^*(x)$.

- A universal approach for various binary optimization problems.
- Algorithms for continuous optimization can be applied.
- The optimal points set $\mathcal{X}^*$ is unknown.

Gibbs distributions

- To approximate q^*, we introduce Gibbs distributions,

$$q_\lambda(x) = \frac{1}{Z_\lambda} \exp\left(-\frac{f(x)}{\lambda}\right), \quad x \in \mathcal{B}_n,$$

where $Z_\lambda = \sum_{x \in \mathcal{B}_n} \exp\left(-\frac{f(x)}{\lambda}\right)$ is the normalizer.

- Given the optimal objective value f^*, for any $x \in \mathcal{B}_n$,

$$q_\lambda(x) = \frac{\exp\left(\frac{f^* - f(x)}{\lambda}\right)}{\sum_{x \in \mathcal{B}_n} \exp\left(\frac{f^* - f(x)}{\lambda}\right)} = \frac{\exp\left(\frac{f^* - f(x)}{\lambda}\right)}{|\mathcal{X}^*| + \sum_{x \in \mathcal{B}_n / \mathcal{X}^*} \exp\left(\frac{f^* - f(x)}{\lambda}\right)}$$
$$\to \frac{1}{|\mathcal{X}^*|} \mathbf{1}_{\mathcal{X}^*}(x) = q^*, \quad \text{as } \lambda \to 0.$$

- The calculation of q_λ does not require knowledge of $\mathcal{X}^*$.

- KL divergence:

$$\mathrm{KL}\left(p_\theta \,\|\, q_\lambda\right) = \sum_{x \in \mathcal{B}_n} p_\theta(x) \log \frac{p_\theta(x)}{q_\lambda(x)}.$$

- In order to reduce the discrepancy between p_θ and q_λ, the KL divergence is supposed to be minimized:

$$\begin{aligned}\mathrm{KL}\left(p_\theta \,\|\, q_\lambda\right) &= \frac{1}{\lambda} \sum_{x \in \mathcal{B}_n} p_\theta(x) f(x) + \sum_{x \in \mathcal{B}_n} p_\theta(x) \log p_\theta(x) + \log Z_\lambda \\ &= \frac{1}{\lambda} \left(\mathbb{E}_{p_\theta}\left[f(x)\right] + \lambda \mathbb{E}_{p_\theta}\left[\log p_\theta(x)\right]\right) + \log Z_\lambda.\end{aligned}$$

- Loss Function (Z_λ is a constant):

$$\boxed{\min_\theta \quad L_\lambda(\theta) = \mathbb{E}_{p_\theta}\left[f(x)\right] + \lambda \mathbb{E}_{p_\theta}\left[\log p_\theta(x)\right]}$$

Gradient for the Loss Function

Lemma

Suppose for any $x \in \mathcal{B}_n$, $p_\theta(x)$ is differentiable with respect to θ. For any constant $c \in \mathbb{R}$, we denote the advantage function

$$A_\lambda(x;\theta,c) := f(x) + \lambda \log p_\theta(x) - c.$$

Then, the gradient of the loss function is given by

$$\nabla_\theta L_\lambda(\theta) = \mathbb{E}_{p_\theta}\left[A_\lambda(x;\theta,c)\nabla_\theta \log p_\theta(x)\right].$$

One candidate for c is

$$c = \mathbb{E}_{p_\theta}[f(x)].$$

Very similar to the policy gradient in reinforcement learning!

Extension: general constrained problem

- Consider

$$x^* = \arg\min_{x} \quad f(x), \text{ s.t. } c(x) = 0, \; x \in \mathcal{B}_n$$

- L1 exact penalty problem

$$x_\sigma^* = \arg\min_{x\in\mathcal{B}_n} f_\sigma(x) := f(x) + \sigma\|c(x)\|_1$$

- Let $\varpi := \min_{x\in\mathcal{B}_n}\{\|c(x)\|_1 \mid \|c(x)\|_1 \neq 0\}$ and $f^* = \min_{x\in\mathcal{B}_n} f(x)$. Define $\bar{\sigma} = (f_\sigma(x^*) - f^*)/\varpi \geq 0$.

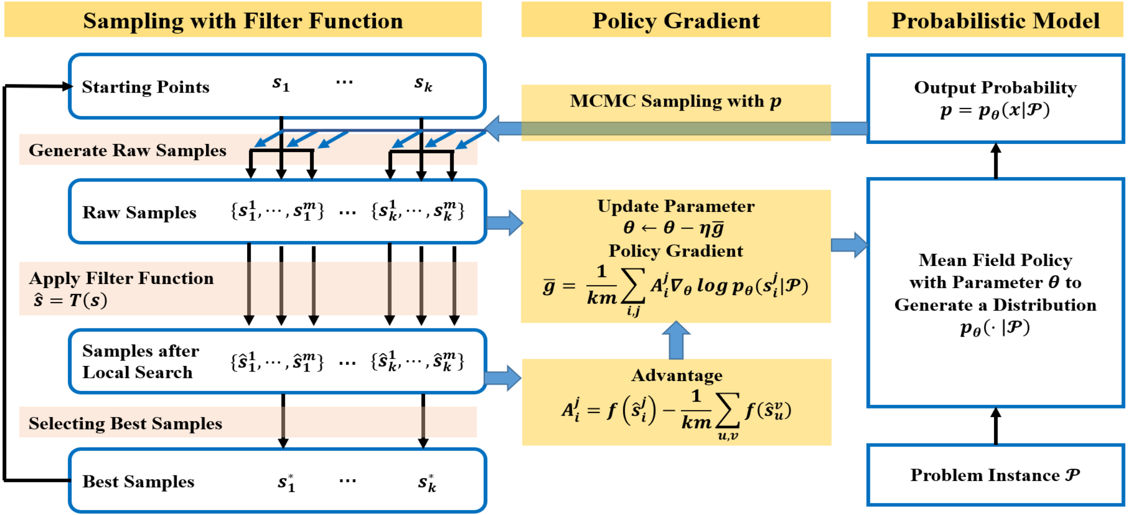
- For all $\sigma \geq \bar{\sigma}$, x^* is a global minima of the penalty problem and x_σ^* is also a global minima of the constrained problem.

Outline

Challenges for prototype algorithm

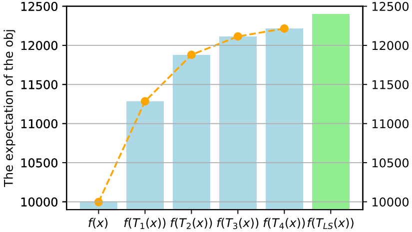
Challenges for prototype algorithm:

- High probability to be quickly stuck in the local minima.
  - **MCPG**: apply a filter function T to enhances the objective function.
- Poor diversity of samples at the later iterations.
  - **MCPG**: large-scale parallel sampling on GPU.
- Discarding all the previous samples.
  - **MCPG**: MCMC sampling starts from the best solutions found in previous steps.

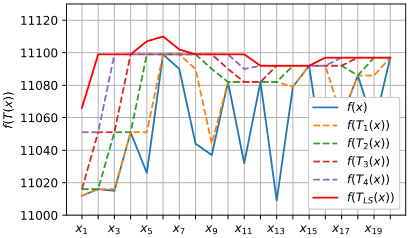

Sampling with Filter Function
Policy Gradient
Probabilistic Model
Starting Points s_1 ⋯ s_k
MCMC Sampling with p
Output Probability $p = p_\theta(x|\mathcal{P})$
Generate Raw Samples
Raw Samples $\{s_1^1, \cdots, s_1^m\}$ ⋯ $\{s_k^1, \cdots, s_k^m\}$
Update Parameter $\theta \leftarrow \theta - \eta \bar{g}$
Policy Gradient $\bar{g} = \frac{1}{km}\sum_{i,j} A_i^j \nabla_\theta \log p_\theta(s_i^j|\mathcal{P})$
Mean Field Policy with Parameter θ to Generate a Distribution $p_\theta(\cdot|\mathcal{P})$
Apply Filter Function $\hat{s} = T(s)$
Samples after Local Search $\{\hat{s}_1^1, \cdots, \hat{s}_1^m\}$ ⋯ $\{\hat{s}_k^1, \cdots, \hat{s}_k^m\}$
Advantage $A_i^j = f(\hat{s}_i^j) - \frac{1}{km}\sum_{u,v} f(\hat{s}_u^v)$
Selecting Best Samples
Best Samples s_1^* ⋯ s_k^*
Problem Instance $\mathcal{P}$

Filter Function

- The filter function T projects x to a better one in the neighborhood.
- Applied with the filter function, $f(T(x))$ has fewer local minima and the same global minimum as the original one.



(a) Expectation of the objective function.



(b) A selected sequence of solutions.

Filter Function

Definition (Filter Function)

For each $x \in \mathcal{B}_n$, let $\mathcal{N}(x) \subset \mathcal{B}_n$ be a neighborhood of x such that $x \in \mathcal{N}(x)$, $|\mathcal{N}(x)| \geq 2$ and any point in $\mathcal{N}(x)$ can be reached by applying a series of "simple" operations to x. A filter function $T(x)$ is defined as

$$T(x) \in \arg\min_{\hat{x} \in \mathcal{N}(x)} f(\hat{x}),$$

where $T(x)$ is arbitrarily chosen if there exists multiple solutions.

- Projection to the best solution on the neighborhood:

$$T_k(x) = \arg\min_{\|\hat{x}-x\|_1 \leq 2k} f(\hat{x}), \quad \mathcal{N}(x) = \{\hat{x} \mid \|\hat{x} - x\|_1 \leq 2k\}.$$

- Algorithms serves as the filter function:

$$T_{LS}(x) = \texttt{LocalSearch}_f(x).$$

Local Search

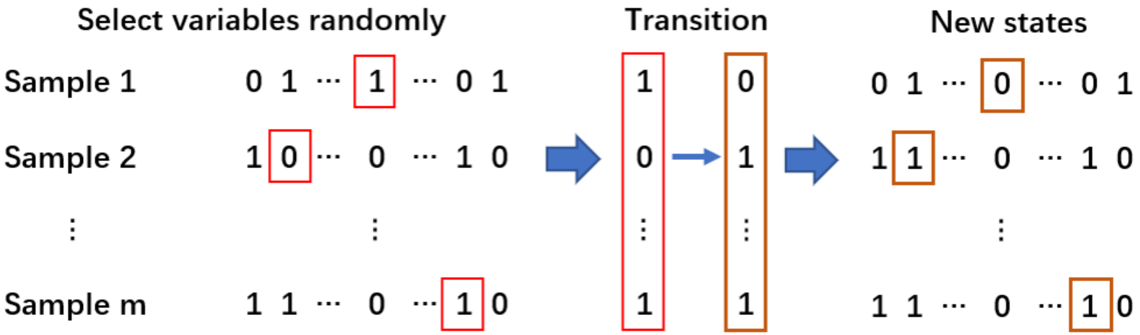
Local Search:

- **Generality**: Local search works for various kinds of problem.
- **Efficiency**: GPUs allow parallel access to the same indexed variable for a large number of samples.

Pipeline of Local Search with flipping operation:

1. Choose a single variable from the current solution x.
2. Flip the variable to its opposite value.
3. Evaluate the new solution to determine if it is improvement.
4. If it is, the variable is flipped to its opposite value
5. Back to Step 1 and continues to the next index in I.

Large-Scale Parallel Sampling on GPU



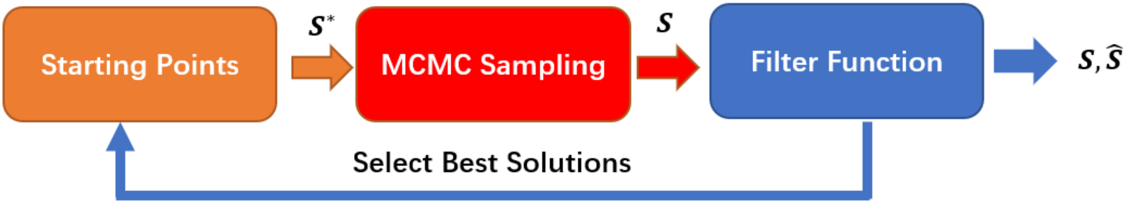
- GPU: quick for parallel accessing but slow for memory copying.

Sampling in MCPG

- constructs large number of short chains,
- discards all previous states in transition (no memory copying),
- outputs the last states for all chains.

Sampling with Filter Function

The pipeline of the sampling procedure:



- Sampling procedure based on Metropolis-Hastings (MH) algorithm.
- S is the raw samples.
- $\hat{S}$ is the samples obtained by applying filter function.
- Samples in S is of high diversity.
- Samples in $\hat{S}$ is of high quality.

Algorithm 1: Parallel Metropolis-Hastings algorithm with filter function

Input: The starting state x_0, transition number t, number of chain m, proposal dist. $Q(x'|x)$.

for $j = 1$ **to** m **do in parallel**

  Copy the starting state $x_0^j = x_0$ for this chain;

  for $v = 0$ **to** $t - 1$ **do in parallel**

    Propose a new state x' by sampling from $Q(x'|x_v^j)$;

    Compute the acceptance $\alpha(x'|x_v^j) = \min\left(1, \frac{p_\theta(x_v^j|\mathcal{P})Q(x_v^j|x')}{p_\theta(x'|\mathcal{P})Q(x'|x_v^j)}\right)$;

    Generate a random number $u \sim \text{Uniform}(0, 1)$;

    if $u < \alpha(x'|x_v^j)$ **then**

      Set $x_{v+1}^j = x'$;

    else

      Set $x_{v+1}^j = x_v^j$;

  Obtain $s^j = x_t^j$;

  Apply filter function and obtain $\hat{s}^j = T(s^j)$;

return the sample set $S = \{s^1, s^2, \cdots, s^m\}$ and $\hat{S} = \{\hat{s}^1, \hat{s}^2, \cdots, \hat{s}^m\}$;

Probabilistic Model Applied with Filter Function

- MCPG focuses on the following modified binary optimization:

$$\min \quad f(T(x)), \quad \text{s.t.} \quad x \in \mathcal{B}_n.$$

- The probabilistic model is equivalent to

$$\min_{\theta} \quad L_\lambda(\theta; \mathcal{P}) = \mathbb{E}_{p_\theta}\left[f(T(x))\right] + \lambda \mathbb{E}_{p_\theta}\left[\log p_\theta(x|\mathcal{P})\right].$$

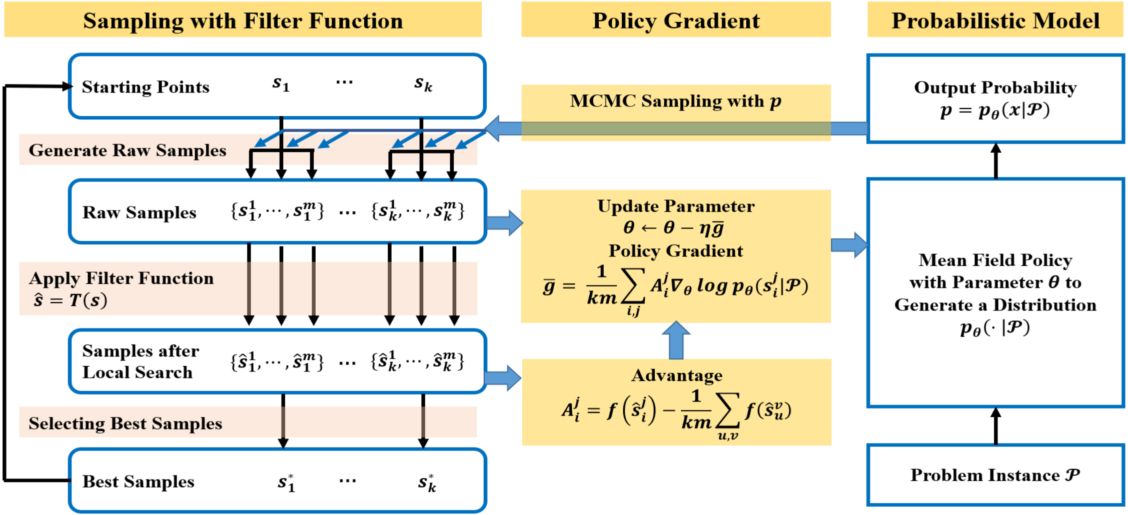
- Empirical gradient:

$$\bar{g}_\lambda(\theta) = \frac{1}{|S|} \sum_{x \in S} A_\lambda(x; \theta) \nabla_\theta \log p(x|\theta; \mathcal{P}).$$

where S is the sample set extracted from distribution $p_\theta(\cdot|\mathcal{P})$ and

$$A_\lambda(x; \theta) := f(T(x)) + \lambda \log p_\theta(x|\mathcal{P}) - \frac{1}{|S|} \sum_{x \in S} f(T(x)).$$

Pipeline of MCPG



Binary Optimization and Probabilistic model

For an arbitrary function f on $\mathcal{B}_n$, we define the B as

$$\mathcal{G}(f) = \min_{x \in \mathcal{B}_n \setminus \mathcal{X}^*} f(x) - f^*. \tag{1}$$

Proposition

For any $0 < \delta < 1$, *suppose* $L_\lambda(\theta) - f^* < (1-\delta)\mathcal{G}(f)$, *then*

$$\mathbb{P}(x \in \mathcal{X}^*) > \delta.$$

Therefore, for $x^1, \ldots, x^m$ *independently sampled from* p_θ, $\min_k f(x^k) = f^*$ *with probability at least* $1 - (1-\delta)^m$.

The above proposition shows that with a optimized probabilistic model, the obtained probability from the optimal solutions is linearly dependent on the gap between the expectation and the minimum of f.

Impact of the Filter Function

- When $T(x) = x$, it means that x is a local minimum point.
- For any given $x \in \mathcal{B}_n$, there exists a corresponding local minimum point by applying the filter function T to x for many times.
- We can divide the set $\mathcal{B}_n$ into subsets with respect to the classification of local minima.

Let $X_1, X_2, ..., X_r$ be a partition of $\mathcal{B}_n$ such that for any $j \in \{1, \ldots, r\}$, every $x \in X_j$ has the same corresponding local minimum point.

Proposition

If there exists some $x \in \mathcal{B}_n$ such that $p_\theta(x) > 0$ and $f(x) > f(T(x))$, then for any sufficiently small $\lambda > 0$ satisfying

$$\mathbb{E}_{p_\theta}[f(x) - f(T(x))] \geq \lambda \log(\max_{1 \leq i \leq r} |X_i|),$$

it holds that

$$\mathrm{KL}\left(p_\theta \,\|\, \hat{q}_\lambda\right) \leq \mathrm{KL}\left(p_\theta \,\|\, q_\lambda\right).$$

Denote $N = 2^n$ and sort all possible points in $\mathcal{B}_n = \{s_1, \ldots, s_N\}$ such that $f(s_1) \le f(s_2) \le \cdots \le f(s_N)$. The bounds of $f(T(x))$ and $\mathbb{E}_{p_\theta}[f(T(x))]$, for a large probability, are not related to samples $s_{M+1}, s_{M+2}, ..., s_N$ for an integer M.

Proposition

Suppose that the cardinality of each neighborhood $\mathcal{N}(s_i)$ is fixed to be $|\mathcal{N}(s_i)| \ge X \ge n+1$ and all elements in $\mathcal{N}(s_i)$ except s_i are chosen uniformly at random from $\mathcal{B}_n \backslash \{s_i\}$. For $\delta \in (0,1)$, let $M = \left\lceil \frac{\log(N/\delta)}{X-1} N \right\rceil + 1$. Then, with probability at least $1-\delta$ over the choice of $T(x)$, it holds:

1) $f(T(x)) \in [f(s_1), f(s_M)], \ \forall x \in \mathcal{B}_n$;
2) $\mathbb{E}_{p_\theta}[f(T(x))] \le \sum_{i=1}^{M-1} p_\theta(s_i) f(s_i) + (1 - \sum_{i=1}^{M-1} p_\theta(s_i)) f(s_M) \le f(s_M)$.

Convergence of MCPG

Assumption: Let $\phi(x;\theta) = \log p_\theta(x|\mathcal{P})$. There exists some constants $M_1, M_2, M_3 > 0$ such that, for any $x \in \mathcal{B}_n$,

1. $\sup_{\theta\in\mathbb{R}^d} |\phi(x;\theta)| \le M_1$,
2. $\sup_{\theta\in\mathbb{R}^d} \|\nabla_\theta \phi(x;\theta)\| \le M_2$,
3. $\|\nabla_{\theta_1}\phi(x;\theta) - \nabla_{\theta_2}\phi(x;\theta)\| \le M_3 \|\theta_1 - \theta_2\|, \forall \theta_1, \theta_2, \in \mathbb{R}^d$.

Theorem

Let the assumption holds and $\{\theta_t\}$ be generated by MCPG. If the stepsize is chosen as $\eta^t = \frac{c\sqrt{mk}}{\sqrt{t}}$ with $c \le \frac{1}{2l}$, then we have

$$\min_{1\le t\le\tau} \mathbb{E}\left[\left\|\nabla_\theta L_\lambda(\theta^t)\right\|^2\right] \le O\left(\frac{\log \tau}{\sqrt{mk\tau}} + \frac{1}{m^2}\right).$$

Outline

Parameterization of sampling policy

- Mean field (MF) approximation:

$$p_\theta(x|\mathcal{P}) = \prod_{i=1}^{n} \mu_i^{(1+x_i)/2}(1-\mu_i)^{(1-x_i)/2}, \quad \mu_i = \phi_i(\theta; \mathcal{P})$$

- Parameterization of μ_i:

$$\mu_i = \phi_i(\theta_i) = \frac{1-2\alpha}{1+\exp(-\theta_i)} + \alpha, \quad 1 \leq i \leq n.$$

  The probability is scaled to the range $(\alpha, 1-\alpha)$, where $0 < \alpha < 0.5$ is given.

- For problems graph structures, combining advanced neural networks such as GNN can also be a good choice.

Maxcut

- The maxcut problem aims to divide a given weighted graph $G = (V, E)$ into two parts, and maximize the total weight of the edges connecting two parts.
- This problem can be expressed as a binary programming problem:

$$\max \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j), \quad \text{s.t.} \quad x \in \{-1, 1\}^n.$$

- We use the results reported by BLS as benchmark. Denoting UB as the results achieved by BLS and obj as the cut size, the gap reported is defined as follows:

$$\mathrm{gap} = \frac{\mathrm{UB} - \mathrm{obj}}{\mathrm{UB}} \times 100\%.$$

Maxcut

- On the Gset instance, MCPG finds all the best-known results.
- For G55 and G70, the results obtained by MCPG is better than all the previous reported results.

Graph	Nodes	Edges	BLS	MCPG	DSDP	RUN-CSP	PI-GNN	EO	EMADM
G14	800	4,694	**3,064**	**3,064**	2,922	2,943	3,026	3047	3045
G15	800	4,661	**3,050**	**3,050**	2,938	2,928	2,990	3028	3034
G22	2,000	19,990	**13,359**	**13,359**	12,960	13,028	13,181	13215	13297
G49	3,000	6,000	**6,000**	**6,000**	**6,000**	**6,000**	5,918	**6000**	**6000**
G50	3,000	6,000	**5,880**	**5,880**	**5,880**	**5,880**	5,820	5878	5870
G55	5,000	12,468	10,294	**10,296**	9,960	10,116	10,138	10107	10208
G70	10,000	9,999	9,541	**9595**	9,456	-	9,421	8513	9557

Table: Computational results on selected Gset instances. The result is sourced from references.

Maxcut

Numerical results on Gset instances within limited time.

Problem		MCPG			MCPG-U			MCPG-P			EO		EMADM	
name	UB	gap (best,	mean)	time	gap (best,	mean)	time	gap (best,	mean)	time	gap mean	time	gap	time
G62	4868	**1.36**	1.57	197	1.89	2.11	189	3.92	4.37	217	2.63	509	2.30	402
G63	26997	**0.25**	0.33	224	0.30	0.37	214	0.93	1.51	242	0.92	542	0.89	576
G64	8735	**0.45**	1.25	225	0.98	1.25	216	2.43	3.08	242	5.21	542	2.70	630
G65	5558	**1.37**	1.74	223	1.94	2.16	215	3.82	4.40	240	3.04	465	2.55	690
G66	6360	**1.54**	1.86	254	2.20	2.43	241	4.30	4.72	267	3.18	510	2.55	1032
G67	6940	**1.38**	1.57	286	1.99	2.19	270	3.79	4.53	295	2.96	550	2.51	1110
G72	6998	**1.51**	1.76	284	2.17	2.40	270	4.73	5.29	294	3.09	554	2.66	1110
G77	9926	**1.49**	1.81	400	2.38	2.58	378	4.81	5.43	425	3.24	755	2.68	2646
G81	14030	**1.74**	1.98	571	2.65	2.80	559	5.26	5.57	634	3.46	1038	2.78	6144

- MCPG-U fixes the vertice-wise distribution $p_i = p(x_i) = 0.5$ instead of $p_\theta(x|G)$ during the sampling procedure.
- MCPG-P uses the fixed vertice-wise distribution output by a trained model similar to Erdos.

Cheeger Cut

- Cheeger cut is a kind of balanced graph cut, which are widely used in classification tasks and clustering.
- Given a graph $G = (V, E, w)$, the ratio Cheeger cut (RCC) are defined as

$$RCC(S, S^c) = \frac{\mathrm{cut}(S, S^c)}{\min\{|S|, |S^c|\}},$$

  where S is a subset of V and S^c is its complementary set.
- The task is to find the minimal ratio Cheeger cut or normal Cheeger cut, which can be converted into a binary optimization problem.

$$\begin{aligned} \min \quad & \frac{\sum_{(i,j)\in E}(1 - x_i x_j)}{\min\left\{\sum_{i=1}^n (1 + x_i), \sum_{i=1}^n (1 - x_i)\right\}}, \\ \text{s.t.} \quad & x \in \{-1, 1\}^n. \end{aligned}$$

Cheeger Cut

- The objective function of the Cheeger cut is not differentiable.
- Most algorithms designed for maxcut is not applicable to the Cheeger cut problem.
- MCPG maintains good performance on Cheeger cut problem.

Problem	MCPG			MCPG-U			MCPG-P			pSC	
Name	RCC (best,	mean)	time	RCC (best,	mean)	time	RCC (best,	mean)	time	RCC	time
G35	2.931	**3.039**	66	3.036	3.163	65	3.113	3.216	65	3.864	127
G36	2.858	**2.894**	67	2.922	2.982	65	2.972	3.004	68	3.794	131
G37	2.847	**2.899**	68	2.984	3.130	69	3.050	3.173	67	3.895	134
G38	2.835	**2.861**	67	2.875	2.897	69	2.913	2.927	68	3.544	125
G48	0.084	**0.085**	93	0.085	0.088	93	0.085	0.085	94	0.109	184
G49	0.151	**0.158**	96	0.157	0.159	96	0.163	0.168	94	0.188	145
G51	2.890	**2.908**	33	2.914	2.960	34	2.996	3.137	31	3.997	52
G52	2.970	**2.990**	35	3.083	3.149	37	3.220	3.364	33	3.993	53
G53	2.827	**2.846**	33	2.892	2.896	31	2.910	2.980	34	3.441	54
G54	2.852	**2.918**	34	2.885	3.018	32	2.928	3.016	35	3.548	57
G63	3.107	**3.164**	242	3.233	3.358	244	3.334	3.481	241	4.090	373

Table: Detailed result for obtaining ratio Cheeger cut (RCC).

Classical MIMO Detection

- The goal for mimo detection is to recover $x_C \in \mathcal{Q}$ from the linear model

$$y_C = H_C x_C + \nu_C.$$

- Our aim is to maximize the likelihood, that is equivalent to

$$\min_{x_C \in \mathbb{C}^N} \|H_C x_C - y_C\|_2^2, \quad \text{s.t.} \quad x_C \in \mathcal{Q}.$$

- By separating the real and imaginary parts, the problem is equivalent to the following:

$$\min_{x \in \mathbb{R}^{2N}} \|Hx - y\|_2^2, \quad \text{s.t.} \quad x \in \{\pm 1\}^{2N}.$$

- To evaluate the performance, we examine the bit error rate (BER) performance with respect to the signal noise ratio (SNR).

$$\mathrm{BER} = \frac{\mathrm{card}(\{x \neq x^*\})}{2N}, \quad \mathrm{SER} = \frac{\mathbb{E}[\|H_C x_C\|_2^2]}{\mathbb{E}[\|\nu_C\|_2^2]} = \frac{M\sigma_x^2}{\sigma_v^2},$$

Type	LB	MCPG		HOTML		MMSE	
	BER	BER	time	BER	time	BER	time
800-2	0.103731	**0.174669**	0.50	0.192981	10.63	0.177175	0.10
800-4	0.056331	**0.126675**	1.00	0.146444	11.88	0.140519	0.10
800-6	0.023131	**0.069094**	3.96	0.082063	13.47	0.105463	0.10
800-8	0.006300	**0.012150**	3.29	0.012188	6.22	0.074900	0.10
800-10	0.000969	**0.001144**	1.61	0.001363	3.35	0.049256	0.10
800-12	0.000031	**0.000031**	1.31	0.000044	2.35	0.030075	0.09
1200-2	0.104883	**0.174588**	1.00	0.193192	82.46	0.177675	0.45
1200-4	0.056400	**0.127004**	1.94	0.145813	77.83	0.140567	0.47
1200-6	0.023179	**0.070346**	6.47	0.083738	73.94	0.105979	0.47
1200-8	0.006179	**0.012529**	7.39	0.012654	61.59	0.075567	0.47
1200-10	0.000875	**0.001050**	5.03	0.001338	22.17	0.050167	0.46
1200-12	0.000058	**0.000054**	3.16	0.000071	15.70	0.030388	0.46

Table: Results on classical MIMO detection problems when $M = N = 800$ and $M = N = 1200$.

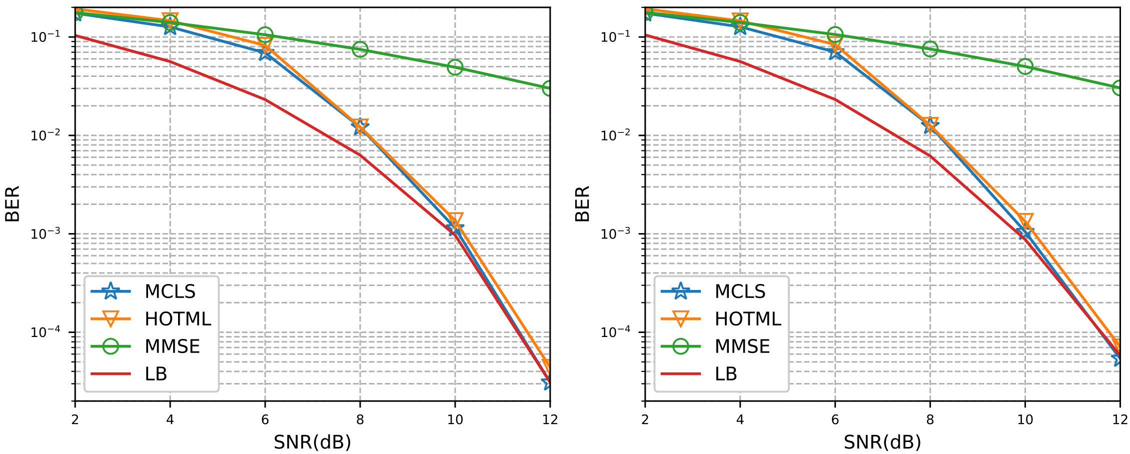


Figure: Results on classical MIMO detection problems when $M = N = 800$ and $M = N = 1200$.

MaxSAT

- Consider the partial MaxSAT problems

$$\begin{aligned} \max \quad & \sum_{c^i \in C_1} \max\{c_1^i x_1, c_2^i x_2, \cdots, c_n^i x_n, 0\}, \\ \text{s.t.} \quad & \max\{c_1^i x_1, c_2^i x_2, \cdots, c_n^i x_n, 0\} = 1, \quad \text{for } c^i \in C_2, \\ & x \in \{-1, 1\}^n, \end{aligned}$$

- Exact penalty problem:

$$\begin{aligned} \max \quad & \sum_{c^i \in C_1 \cup C_2} w_i \max\{c_1^i x_1, c_2^i x_2, \cdots, c_n^i x_n, 0\}, \\ \text{s.t.} \quad & x \in \{-1, 1\}^n, \end{aligned}$$

where $w_i = 1$ for $c_i \in C_1$ and $w_i = |C_i| + 1$ for $c_i \in C_2$.

MaxSAT

- Consider the partial MaxSAT problems

$$\begin{aligned} \max \quad & \sum_{c^i \in C_1} \max\{c_1^i x_1, c_2^i x_2, \cdots, c_n^i x_n, 0\}, \\ \text{s.t.} \quad & \max\{c_1^i x_1, c_2^i x_2, \cdots, c_n^i x_n, 0\} = 1, \quad \text{for } c^i \in C_2, \\ & x \in \{-1, 1\}^n, \end{aligned}$$

- Exact penalty problem:

$$\begin{aligned} \max \quad & \sum_{c^i \in C_1 \cup C_2} w_i \max\{c_1^i x_1, c_2^i x_2, \cdots, c_n^i x_n, 0\}, \\ \text{s.t.} \quad & x \in \{-1, 1\}^n, \end{aligned}$$

where $w_i = 1$ for $c_i \in C_1$ and $w_i = |C_i| + 1$ for $c_i \in C_2$.

MaxSAT without hard clauses on randomly generated instances

Problem			MCPG			MCPG-U			WBO/inc		SATLike	
n	$\|C_1\|$	UB	gap (best,	mean)	time	gap (best,	mean)	time	gap	gap	gap	time
2000	8000	7211	**0.00**	0.00	36	0.04	0.06	36	8.17	5.74	0.08	60
2000	8000	7204	**0.00**	0.01	36	0.06	0.08	35	8.65	6.18	0.08	60
2000	10000	8972	**0.00**	0.01	39	0.02	0.03	38	6.88	5.49	0.12	60
2000	10000	8945	**0.00**	0.01	39	0.02	0.05	38	6.47	5.62	0.15	60
3000	12000	10811	**0.00**	0.00	166	0.02	0.04	165	8.46	6.26	0.15	300
3000	12000	10823	**0.00**	0.00	166	0.01	0.03	166	8.20	5.89	0.12	300
3000	12000	10792	**0.00**	0.00	165	0.02	0.03	166	8.53	5.93	0.06	300
3000	15000	13712	**0.00**	0.00	186	0.01	0.01	185	6.66	5.49	0.13	300
3000	15000	13705	**0.00**	0.00	186	0.00	0.01	185	5.72	5.44	0.16	300
5000	20000	18032	**0.00**	0.01	341	0.05	0.06	344	7.83	6.32	0.11	500
5000	20000	18008	**0.00**	0.00	344	0.04	0.06	342	7.16	6.26	0.11	500

MaxSAT: Statistics on the random track datasets in MSE2016

Problem Set	Range	MCPG best num	MCPG best pct	MCPG mean num	MCPG mean pct	SATlike num	SATlike pct	WBO num	WBO pct	WBO-inc num	WBO-inc pct
min2sat	0.00	53	0.88	22	0.37	18	0.30	2	0.03	1	0.02
	(0.00,1.00]	7	0.12	38	0.63	31	0.52	0	0.00	8	0.13
	(1.00,2.00]	0	0.00	0	0.00	11	0.18	0	0.00	27	0.45
	(2.00,3.00]	0	0.00	0	0.00	0	0.00	0	0.00	19	0.32
	> 3.00	0	0.00	0	0.00	0	0.00	58	0.97	5	0.08
min3sat	0.00	50	0.83	33	0.55	50	0.83	0	0.00	1	0.02
	(0.00,1.00]	5	0.08	16	0.27	3	0.05	0	0.00	2	0.03
	(0.00,2.00]	4	0.07	9	0.15	6	0.10	0	0.00	9	0.15
	(2.00,3.00]	1	0.02	1	0.02	1	0.02	0	0.00	8	0.13
	> 3.00	0	0.00	1	0.02	0	0.00	60	1.00	40	0.67

partial MaxSAT from the MSE 2016 competition

Problem				MCPG			WBO/inc		SATLike	
name	$\|C_2$	$\|C_1\|$	UB	gap (best,	mean)	time	gap	gap	gap	time
min2sat-800-1	4013	401	340	0.00	**0.00**	27	24.41	2.35	1.47	60
min2sat-800-2	3983	401	352	0.00	**0.06**	27	24.43	0.85	0.85	60
min2sat-800-3	3956	400	340	0.00	**0.03**	27	22.35	1.76	0.59	60
min2sat-800-4	3933	398	349	0.00	**0.00**	27	26.36	2.58	1.72	60
min2sat-800-5	3871	402	353	0.00	0.42	27	20.11	1.70	**0.28**	60
min2sat-1040-1	4248	525	458	0.00	**0.12**	32	21.62	2.40	0.22	60
min2sat-1040-2	4158	528	473	0.00	**0.18**	33	22.83	1.27	0.21	60
min2sat-1040-3	4194	527	473	0.00	**0.29**	33	17.12	0.21	0.42	60
min2sat-1040-4	4079	520	474	0.00	**0.14**	33	18.57	1.69	0.21	60
min2sat-1040-5	4184	523	465	0.43	0.47	33	17.42	1.29	**0.00**	60

Many Thanks For Your Attention!

- 教材：刘浩洋, 户将, 李勇锋，文再文，最优化：建模、算法与理论; http://bicmr.pku.edu.cn/~wenzw/optbook.html