NSNet: A General Neural Probabilistic Framework for Satisfiability Problems

Zhaoyu Li [1,2], Xujie Si [1,2,3]

[1] McGill University, [2] Mila – Quebec AI Institute, [3] CIFAR AI Research Chair

NeurIPS 2022

Satisfiability Problems

Boolean Satisfiability Problem (SAT):

- Determine whether there exists an assignment that can satisfy a Boolean formula.

Example:

$$(X_1 \lor \neg X_2) \land (X_1 \lor X_3) \land (\neg X_1 \lor X_2 \lor X_3)$$

A satisfying assignment

$$X_1 = 0, \qquad X_2 = 0, \qquad X_3 = 1$$

Satisfiability Problems

Boolean Satisfiability Problem (SAT):

- Determine whether there exists an assignment that can satisfy a Boolean formula.

Sharp Satisfiability Problem (#SAT, or model counting):

- Count the number of all satisfying assignments for a Boolean formula

Example:

$$(X_1 \lor \neg X_2) \land (X_1 \lor X_3) \land (\neg X_1 \lor X_2 \lor X_3)$$

All satisfying assignments:

$$\begin{array}{lll} X_1 = 0, & X_2 = 0, & X_3 = 1 \\ X_1 = 1, & X_2 = 0, & X_3 = 1 \\ X_1 = 1, & X_2 = 1, & X_3 = 0 \\ X_1 = 1, & X_2 = 1, & X_3 = 1 \end{array}$$

Factor Graph Formulation

$$p(\boldsymbol{x}) = \frac{1}{Z}\prod_{a=1}^{m} f_a(\boldsymbol{x}_a) \qquad Z = \sum_{\boldsymbol{x}}\left(\prod_{a=1}^{m} f_a(\boldsymbol{x}_a)\right)$$

where $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ denote a possible assignment, $\boldsymbol{x}_a$ denote the corresponding assignment in clause a, $f_a(\boldsymbol{x}_a) = 1$ if $\boldsymbol{x}_a$ satisfies clause a else $f_a(\boldsymbol{x}_a) = 0$, Z is the partition function, representing the number of all satisfying assignments.

We can consider $p(\boldsymbol{x})$ as a probability measure on the solution space that **has a uniform distribution for all satisfying assignments and zero probability for unsatisfying ones**.

Traditional Inference Algorithm

Belief Propagation (BP, in log space):

$$m_{i\to a}^{(k)}(x_i) = -z_{i\to a}^{(k)} + \sum_{c\in N(i)\backslash a} m_{c\to i}^{(k-1)}(x_i) \qquad m_{a\to i}^{(k)}(x_i) = -z_{a\to i}^{(k)} + \mathrm{LSE}_{\boldsymbol{x}_a\backslash x_i}\left(f_a(\boldsymbol{x}_a) + \sum_{j\in N(a)\backslash i} m_{j\to a}^{(k)}(x_j)\right)$$

- Marginal inference:

$$b_i(x_i) \propto \prod_{a\in N(i)} m_{a\to i}^{(T)}(x_i) \qquad b_a(\boldsymbol{x}_a) \propto f_a(\boldsymbol{x}_a) \prod_{j\in N(a)} m_{j\to a}^{(T)}(x_j)$$

- Partition function estimation (Bethe approximation):

$$\ln Z = -\sum_{a=1}^{m}\sum_{\boldsymbol{x}_a} b_a(\boldsymbol{x}_a)\ln\frac{b_a(\boldsymbol{x}_a)}{f_a(\boldsymbol{x}_a)} + \sum\nolimits_{i=1}^{n}(N(i)-1)\sum_{x_i} b_i(x_i)\ln b_i(x_i)$$

Drawbacks: inaccurate, not flexible, hard to converge…

NSNet's Framework

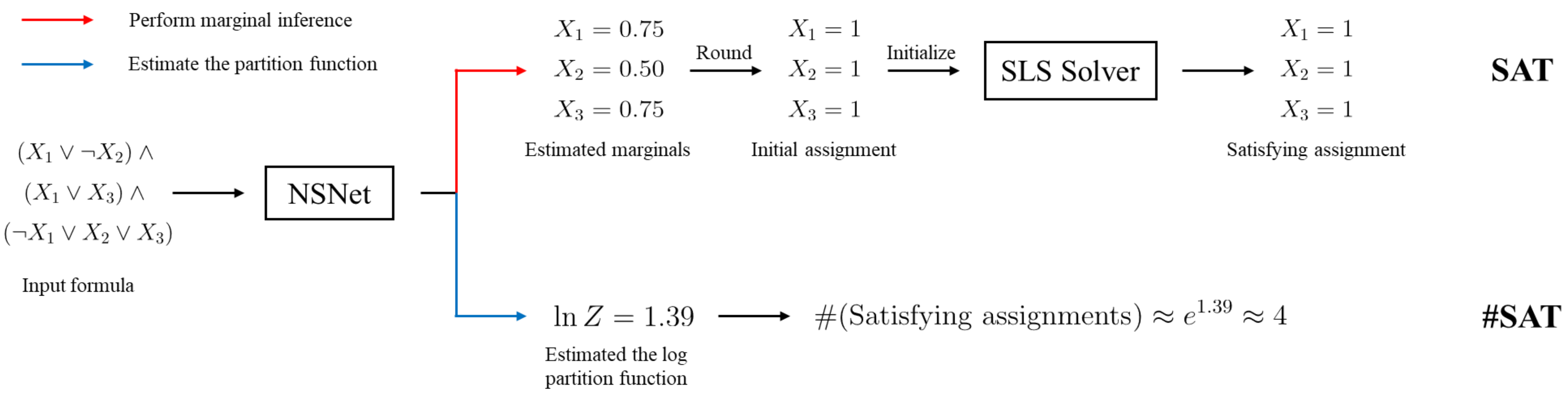
Serve as a neural generalization of BP

SAT:

- **Perform marginal inference (rather than predicting a possible assignment directly)**
- Obtain a satisfying assignment by **rounding** and **executing a local search**

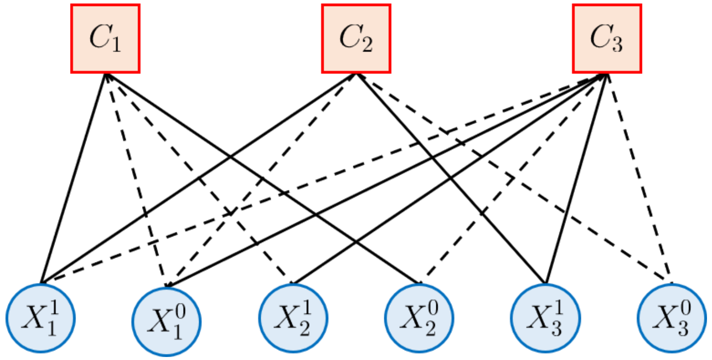
#SAT:

- **Estimate the partition function**



NSNet's Framework

Graph representation:



$$(X_1 \lor \neg X_2) \land (X_1 \lor X_3) \land (\neg X_1 \lor X_2 \lor X_3)$$

X_i^1 and X_i^0 denotes variable X_i takes values 1 and 0 respectively. The solid/dashed line indicates that the variable assignment satisfies/dissatisfies the associated clause.

NSNet’s Framework

Message passing scheme:

- Clause to variable assignment

$$\tilde{m}_{i\to a}^{(k)}(x_i) = \mathrm{MLP}\left(\sum_{c\in N(i)\backslash a} m_{c\to i}^{(k-1)}(x_i)\right) \qquad m_{i\to a}^{(k)}(x_i) = \mathrm{MLP}\left(\tilde{m}_{i\to a}^{(k)}(x_i),\ \tilde{m}_{i\to a}^{(k)}(1-x_i)\right)$$

- Variable to clause assignment

$$m_{a\to i}^{(k)}(x_i) = \mathrm{MLP}\left(\mathrm{LSE}_{x_a^*\backslash x_i}\left(\sum_{j\in N(a)\backslash i} m_{j\to a}^{(k)}(x_j)\right)\right)$$

- **Edge embeddings**
- **Use the same aggregators (summation, LSE) as BP**
- **Enforce the permutation invariance and the negation equivariance of CNF formulas**

NSNet's Framework

Readout:

SAT:

$$\widetilde{b_i}(x_i) = \mathrm{MLP}\left(\sum_{a\in N(i)} m_{a\to i}^{(T)}(x_i)\right) \qquad [b_i(1), b_i(0)] = \mathrm{softmax}\left[\widetilde{b_i}(1), \widetilde{b_i}(0)\right]$$

#SAT:

$$\widetilde{b_a}(x_a) = \mathrm{MLP}\left(\sum_{j\in N(a)} m_{j\to a}^{(T)}(x_j)\right) \qquad b_a(x_a) = \widetilde{b_a}(x_a) - \mathrm{LSE}_{x_a}\left(\widetilde{b_a}(x_a)\right)$$

$$\ln Z = -\sum_{a=1}^{m}\sum_{\boldsymbol{x}_a} b_a(\boldsymbol{x}_a)\ln b_a(\boldsymbol{x}_a) + \sum_{i=1}^{n}(N(i)-1)\sum_{x_i} b_i(x_i)\ln b_i(x_i)$$

Training:

- Using the ground truth marginals and the model counting
- KL divergence loss for SAT and MSE loss for #SAT

Experiments

SAT

- Solving accuracy of the initial assignments (without local search)
- Compared with BP and the SOTA model NeuroSAT
- Compared with the assignment supervision

Table 1: Solving accuracy (%) of the initial assignments on the synthetic datasets.

Supervision	**Method**	**Same Distribution**				**Larger Distribution**			
		SR	3-SAT	CA	Total	SR	3-SAT	CA	Total
N/A	BP	49.65	51.43	36.45	45.84	5.97	7.18	6.59	6.58
Assignment	NeuroSAT	44.16	43.74	35.37	41.09	1.60	2.52	1.64	1.92
	NSNet	39.62	57.63	47.20	48.15	3.37	8.13	3.61	5.03
Marginal	NeuroSAT	47.77	48.60	50.97	49.11	1.99	3.18	5.61	3.59
	NSNet	**63.16**	**63.52**	**56.30**	**60.99**	**9.13**	**12.07**	**8.08**	**9.76**

Experiments

SAT

- Solving accuracy with local search
- Compared with the SOTA SLS solver with different initialization methods

Table 3: Solving accuracy (%) for Sparrow with different initializations on the synthetic datasets.

Method	**Larger Distribution**			
	SR	3-SAT	CA	Total
Sparrow	8.77 ± 0.15	11.48 ± 0.26	54.25 ± 0.23	24.83 ± 0.08
BP-Sparrow	27.76 ± 0.20	35.30 ± 0.31	84.89 ± 0.19	49.32 ± 0.11
NeuroSAT-Sparrow	22.04 ± 0.30	29.03 ± 0.30	83.64 ± 0.22	44.90 ± 0.18
NSNet-Sparrow	$\mathbf{29.66 \pm 0.15}$	$\mathbf{37.24 \pm 0.18}$	$\mathbf{86.13 \pm 0.21}$	$\mathbf{51.01 \pm 0.11}$

Experiments

#SAT

- Rooted mean square error (RMSE) and runtime
- Compared with the SOTA solvers ApproxMC3, F2 and the neural baseline BPNN

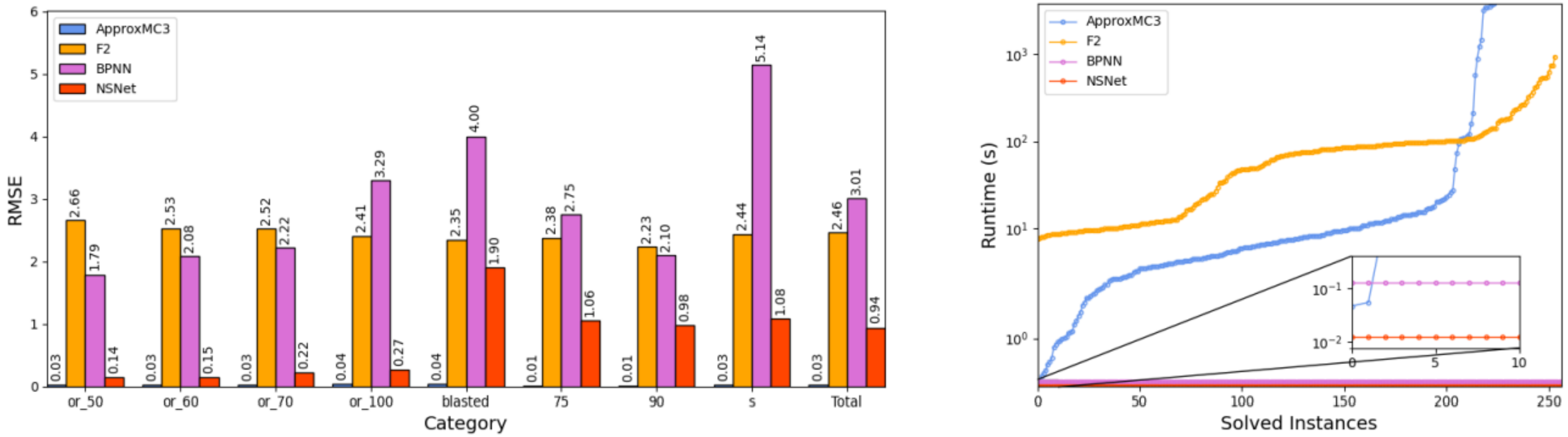


Figure 3: (Left) RMSE between estimated log countings and ground truth for each solver on the BIRD benchmark. (Right) Cactus plots of runtime for each solver on the BIRD benchmark.

Table 4: RMSE and average runtime for each solver on the SATLIB benchmark.

Method	Metric	
	RMSE	Runtime (s)
ApproxMC3	**0.05**	13.05
F2	2.36	27.79
NSNet	1.71	**< 0.01**

Thank you!

Email: **zli199@cs.mcgill.ca** or **zhaoyu.li@mila.quebec**

Github code: **https://github.com/zhaoyu-li/NSNet**