Graph Neural Networks for Dynamic Abstraction Sampling

Vincent Hsiao[1], Dana Nau[1], Rina Dechter[2]

DEPARTMENT OF COMPUTER SCIENCE

University of Maryland, College Park[1], University of California, Irvine[2]

Background

Graphical Model

- Variables: $X = \{X_1, X_2, ..., X_N\}$
- Domains: $D = \{D_{X_1}, D_{X_2}, \ldots, D_{X_N}\}$
- Functions: $\Psi = \{\Psi_1, \Psi_2, ...\Psi_M\}$

Probabilistic Inference

- Task: determine configuration probability

$$\Pr(X = \mathbf{x}) = \frac{\prod_i \Psi_i(\mathbf{x})}{\sum_x \prod_i \Psi_i(x)}$$

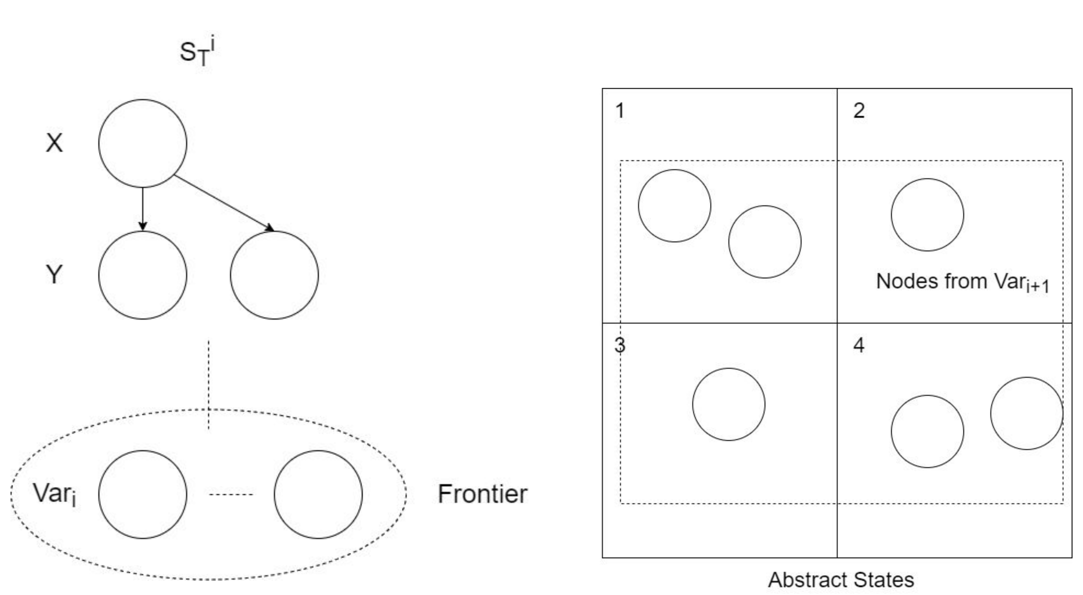
- Denominator is a normalization constant known as the **partition function (Z)**
- **Search spaces:** Alternative representation for graphical model
  - Z can be computed recursively:

$$Z(n_{v_i}) = \sum_{n_{v_j} \in ch(n_{v_i})} c(n_{v_i}) Z(n_{v_j})$$

  where Z(root) is the partition function
- Can estimate Z using sampling

Abstraction Sampling

- Sample subtrees (**probes**) from root:
  - Each level has a set of **frontier nodes**
  - **Abstraction function**: assigns each frontier node to an **abstract state:**



  - Select representative for each state using importance sampling
  - Add representatives to probe and repeat for next level

Abstract

- Abstraction sampling is a sampling scheme for computing the partition function over probabilistic graphical models.
- Its performance depends on the quality of the abstraction function that is used.
- In this paper we develop a method for learning the abstraction function using reinforcement learning and GNNs during the sampling process.
- Our initial results on several benchmarks show good performance compared with currently practiced abstraction functions.

Reinforcement Learning Problem

Key points:

- Better abstraction functions induce a lower variance in abstraction sampling.
  - Task: Minimize expected variance of our sampled Z estimates:

$$\theta' = arg \inf_{\theta} E[(\hat{Z}_\theta - Z)^2]$$

- Each frontier node has a context assignment that uniquely determines the value of its rooted subtree.

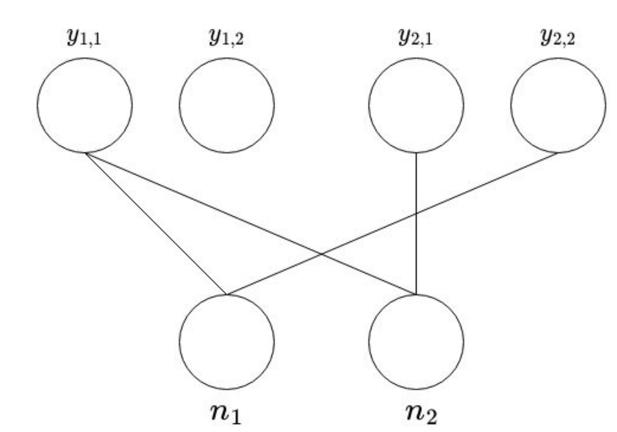
Markov Decision Process

- **States**: subtrees of the search space
- **Actions**: abstract state assignment of the frontier nodes
- **Transition**: given a current state (partial probe), we can define transition probabilities to the partial (or full) probe using importance sampling
- **Reward**: negative variance of a full probe
- An abstraction function is a policy for selecting an abstract state assignment for frontier nodes (action) conditioned on the current context (state)

Graph Neural Networks

Bipartite Context Graph

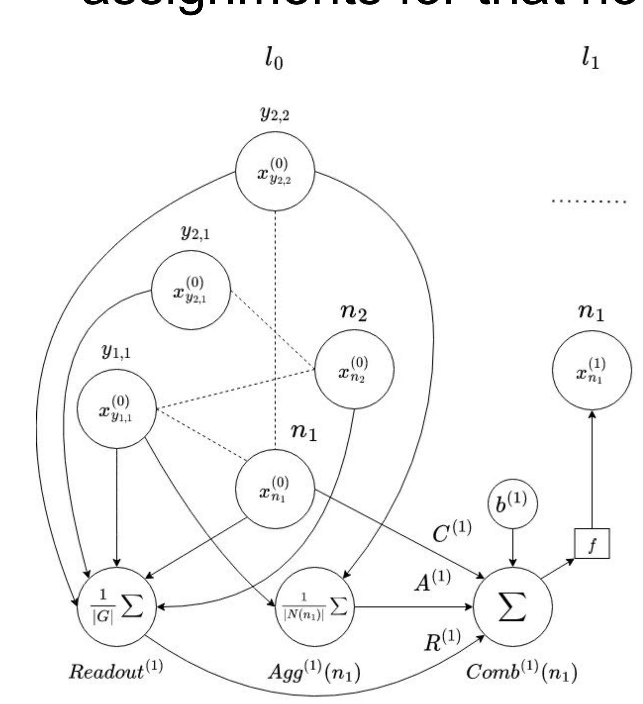
- GNN maps context representation to node-level abstract state assignments.
- **Bipartite context graph** serves as input to our GNN:
  - U: the set of frontier nodes
  - Y: a node for each value for each variable in the context of X
  - Nodes in U are connected to nodes in Y if the nodes context assignments match the variable-value node



Bipartite Context Graph

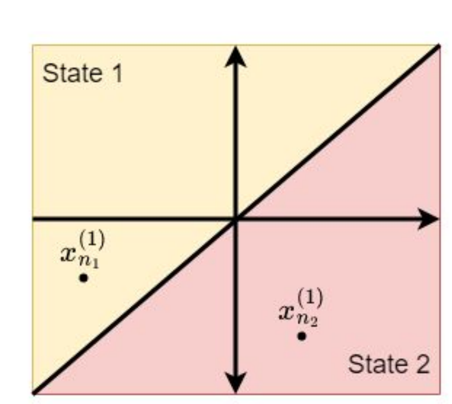
Architecture

- Aggregate-Combine GNN
- Each node in the bipartite context graph has a vector in $\mathbb{R}^d$ where d is the number of abstract states.
- Output: distribution over abstract state assignments for that node.



AC-GNN for 1 layer of computation

- For each output vector, apply H: $\mathbb{R}^d \to \{1,...,d\}$ to get the final assignment for a node to abstract state.



Example H function (argmax)

Dynamic Abstraction Sampling

Evolutionary Strategies

- We use a (r/1 + λ)-ES scheme:
  - L: population size
  - m: rollouts per iteration
  - r: number of parents
  - Fitness function:

$$f(\theta_i) = -\frac{1}{m} \sum_j^m \|Z_{est} - \hat{Z}_{i,j}\|^2$$

- We optimize with respect to the current running average:

$$Z_{est} = \frac{1}{m \cdot L \cdot g} \sum_k^g \sum_i^L \sum_j^m \hat{Z}_{k,i,j}$$

Results

- Empirical results for 1 hr of computation

Problem Instance	Truth	RandCB	GNN
grid20x20.f2	291.732	291.310	**291.537**
grid20x20.f5	665.116	661.149	**661.859**
grid20x20.f10	1311.983	1302.819	**1307.786**
grid20x20.f15	1962.977	1944.677	**1949.414**
grid20x20.f15.wrap	1979.962	1966.380	**1968.297**
rbm20	58.530	**57.354**	57.058
rbm21	63.112	57.592	**62.829**
rbm22	66.553	61.219	**65.454**
or_chain_10.fg	-8.330	-8.810	**-8.391**

- Transfer learning: small instance (1hr) to large instance (1hr) in Grid benchmark.

Base	Truth	RandCB	GNN
Trained on grid20x20.f15			
grid40x40.f5	2792.203	2743.674	**2747.990**
grid40x40.f10	5491.076	**5402.128**	5392.410
grid40x40.f15	8198.61	8022.986	**8037**
grid80x80.f5	11163	10901	**10902**
grid80x80.f10	21785.5	21225.005	**21226.268**
grid80x80.f15	32550	31655	**31750**

- Some problems show significant gain when using learned abstraction function.

Future Research

- Explore architectural changes, different optimization algorithms, more empirical evaluation

Acknowledgements

This work supported in part by NSF grant IIS-2008516 and AFOSR grant 1010GWA357.