Rethinking Belief Propagation: Can Graph Neural Networks Take the Lead?

Université de Montréal

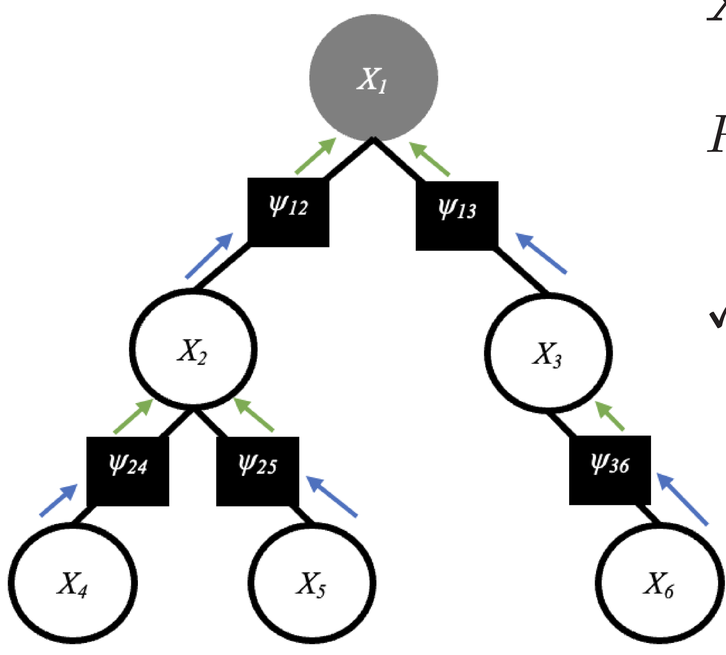
Anita Kriz[1], Clemence Granande[1], Anthony Gosselin[1], Jeremy Qin[1], Alireza Dizaji[1]

[1] *Mila*

Mila

Introduction

Problem: Performing inference tasks on graphs is HARD



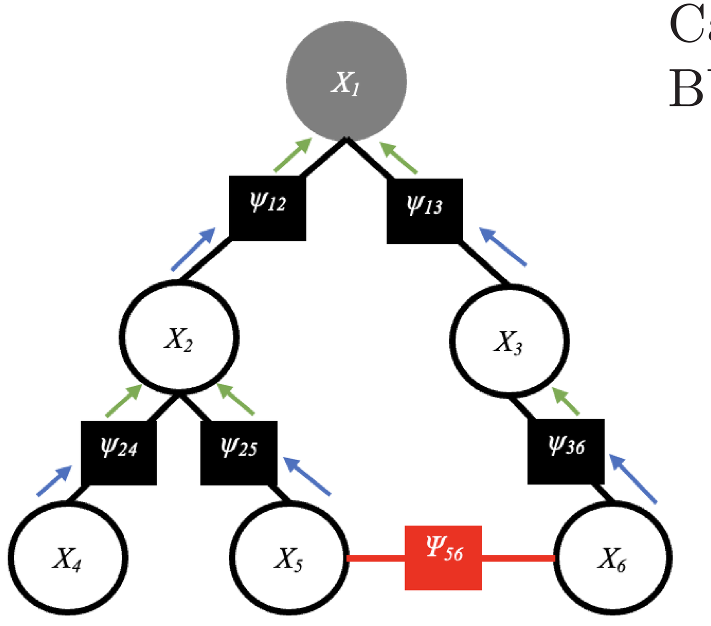
It is possible on trees...

X Naïve Marginalization: in $O(k^{|N|})$

$$P(x_i) = \sum_{x_1 \in X_1} \dots \sum_{x_N \in X_N} P(x_1, x_2, \dots, x_N)$$

✓Belief Propagation (BP) in $O(|N|)$:

- **Initiate** message passing at leaves
- **Propagate and Store** messages up to the root and back



... but not on graphs in general

Can **approximate**: run BP for t iterations, BUT:

- May not converge, even as $t \to \infty$
- May not have a closed-form solution
- May not be accurate due to the complex dependencies in graph

Research Question: How can we find better approximators for inference tasks in Probabilistic Graphical Models (PGMs)?

Background

Belief Propagation

Can be generalized to graphs with loops, i.e. *loopy BP*:

$$\mu^{(t)}_{i \to \alpha}(X_i) = \prod_{\beta \in N_i \setminus \alpha} \mu^{(t-1)}_{\beta \to i}(X_i) \tag{1}$$

$$\mu^{(t)}_{\alpha \to i}(X_i) = \sum_{X_\alpha \setminus X_i} \psi_\alpha(X_\alpha) \prod_{j \in N_\alpha \setminus i} \mu^{(t-1)}_{j \to \alpha}(X_j) \tag{2}$$

where N_i are the neighbors of variable node X_i and N_α are the neighbors of factor nodes α

GNNs

Use a message passing method $m^{(t+1)}_{i \to}$ to obtain the hidden vector states $h^{(t+1)}_i$ for a GNN node v_i at time $t+1$:

$$m^{(t)}_i = \sum_{j \in N(i)} m^{t+1}_{j \to i} = \sum_{j \in N(i)} \mathcal{M}\left(h^t_j, h^t_i, e_{ji}\right) \tag{3}$$

$$h^{(t+1)}_i = \mathcal{U}(h^{(t)}_i, m^{(t+1)}_i)$$

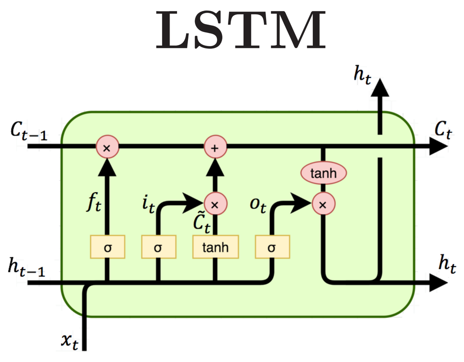
Obtain the marginal distribution at T:

$$\hat{y} = \sigma(h^{(T)})$$

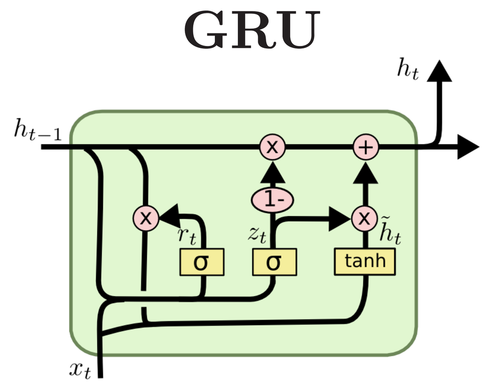
Trained with $L(\hat{y}, y) = -y_i(x_i)\log(\hat{y}_i(x_i))$

⇒ **Update Functions**

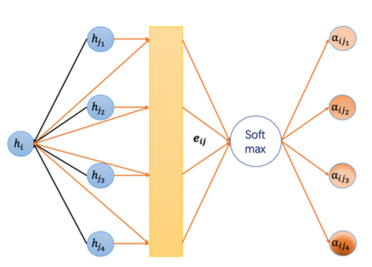
LSTM



GRU

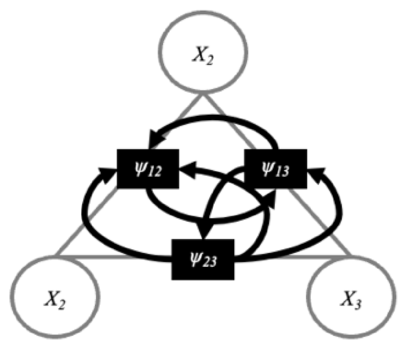


⇒ **Soft Attention Mechanism**

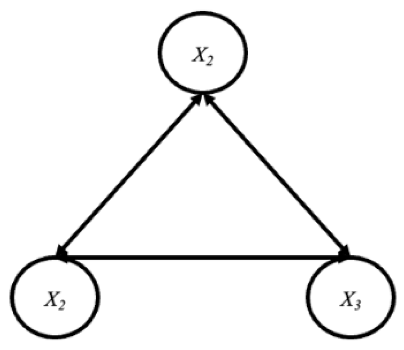


$$c'_t = \text{LeakyRelu}(e_{i,j} * A^\intercal + b)$$
$$c_t = \text{softmax}(c'_t)$$
$$e_{i,j} = c_t * e'_{i,j}$$

PGM to GNN Mapping



Message Nodes to GNN Nodes



Variable Nodes to GNN Nodes

Message Node Mapping

Message updates:

$$m^{(t+1)}_{i \to j} = \mathcal{M}\left(\sum_{k \in N_i \setminus j} h^t_{k \to i}, e_{ij}\right) \tag{4}$$

Hidden State Updates:

$$h^{(t+1)}_{i \to j} = \mathcal{U}(h^{(t)}_{i \to j}, m^{(t+1)}_{i \to j}) \tag{5}$$

Node Marginals:

$$\hat{p}_i(X_i) = \mathcal{R}\left(\sum_{j \in N_i} h^T_{j \to i}\right) \tag{6}$$

Variable Node Mapping

$$m^{(t+1)}_{i \to j} = \mathcal{M}\left(h^t_i, h^t_j, e_{ij}\right) \tag{7}$$

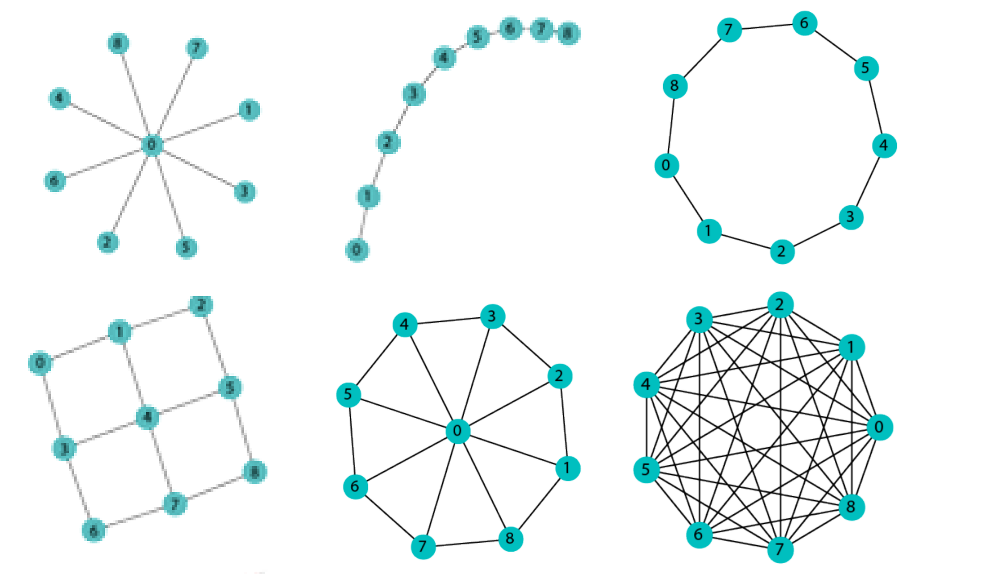
$$m^{(t+1)}_i = \sum_{j \in N_i} m^{(t+1)}_{j \to i} \tag{8}$$

$$h^{(t+1)}_{i \to j} = \mathcal{U}(h^{(t)}_i, m^{(t+1)}_i) \tag{9}$$

$$\hat{p}_i(X_i) = \mathcal{R}(h^T_i) \tag{10}$$

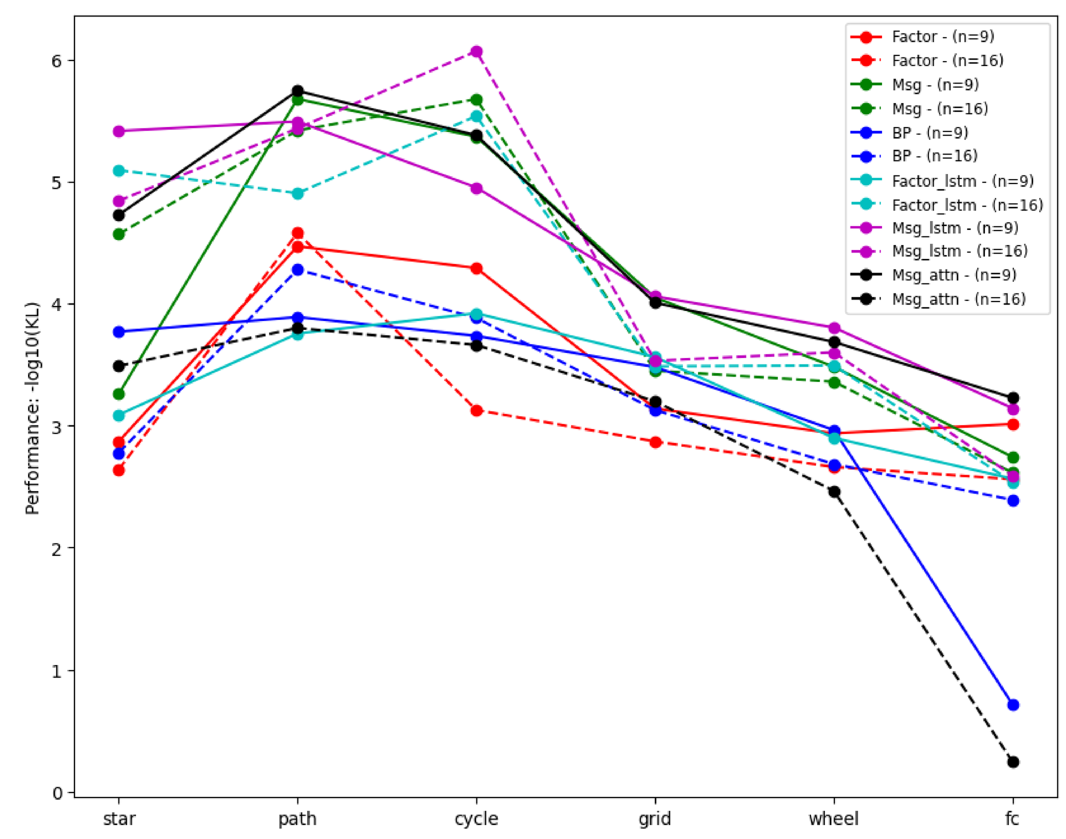
Experiments & Results

Graphs

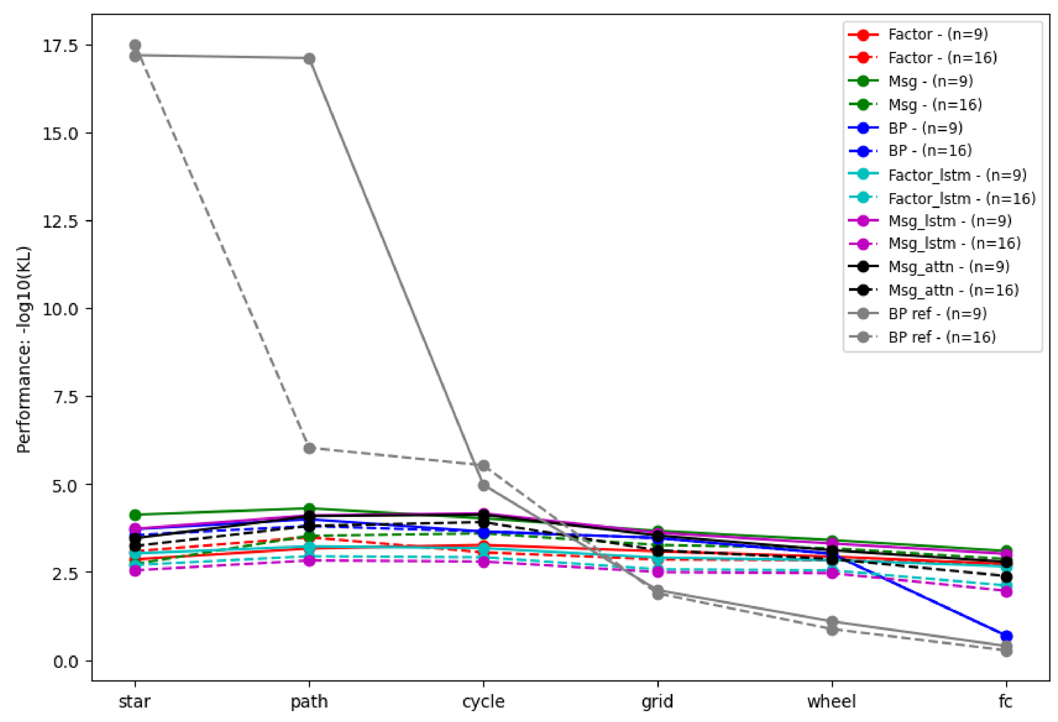


Graph structure types: star, path, cycle, grid, wheel, fc

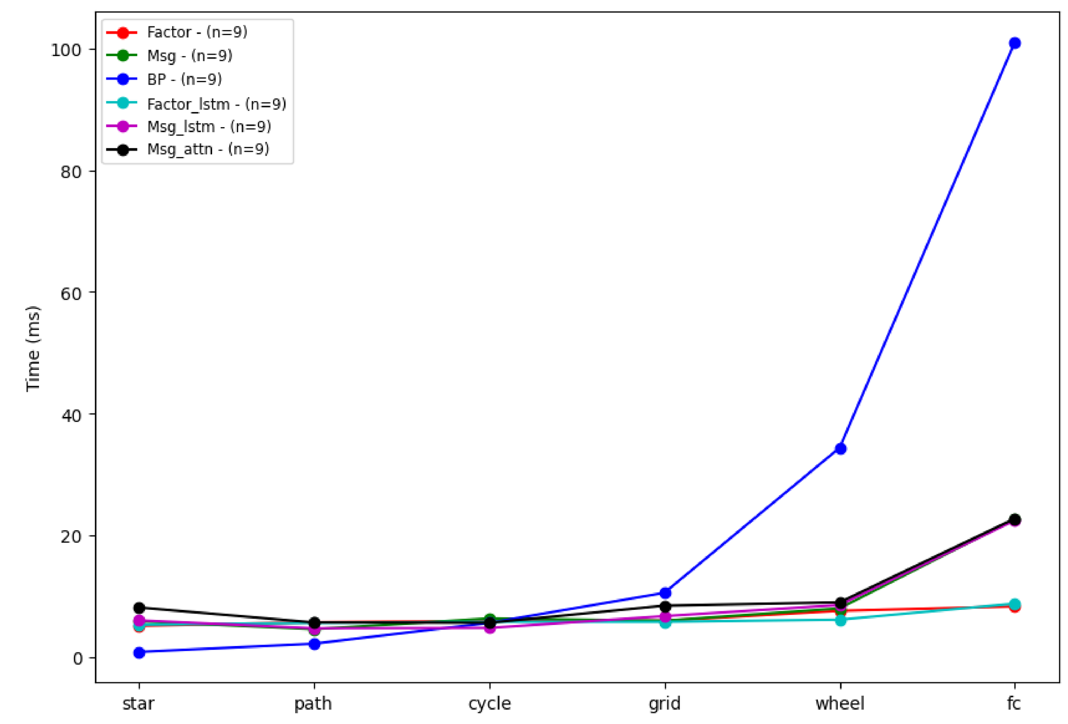
In Sample Performance



Combined Dataset Performance



Inference Time



Acknowledgements

We acknowledge the work of Yoon et al., "Inference in Probabilistic Graphical Models by Graph Neural Networks", 2019, and the use of open-source code from https://github.com/sunfanyunn/FE-GNN/tree/master.

Conclusion & Future Developments

- GNN models are more robust to increasing graph complexity w.r.t. computation time
- GNNs are great candidates when increasing the complexity of graph, with no significantly or consistantly optimal model

  Potential Developments: applying attention during $\mathcal{U}$; to other model structures (not Ising)