Theory-guided Message Passing Neural Network for Probabilistic Inference

Zijun Cui[1,2] **Hanjing Wang**[1] **Tian Gao**[3] **Kartik Talamadupula**[3,4] **Qiang Ji**[1]
[1]Rensselaer Polytechnic Institute
[2]University of Southern California
[3]IBM Research
[4]Symbl.ai

Abstract

Probabilistic inference can be tackled by minimizing a variational free energy through message passing. To improve performance, neural networks are adopted for message computation. Neural message learning is heuristic and requires strong guidance to perform well. In this work, we propose a *theory-guided message passing neural network* (TMPNN) for probabilistic inference. Inspired by existing work, we consider a generalized Bethe free energy which allows for a learnable variational assumption. Instead of using a black-box neural network for message computation, we utilize a general message equation and introduce a symbolic message function with semantically meaningful parameters. The analytically derived symbolic message function is seamlessly integrated into the MPNN framework, giving rise to the proposed TMPNN. TMPNN is trained using algorithmic supervision without requiring exact inference results. Leveraging the theory-guided symbolic function, TMPNN offers strengthened theoretical guarantees compared to conventional heuristic neural models. It presents a novel contribution by demonstrating its applicability to both MAP and marginal inference tasks, outperforming SOTAs in both cases. Furthermore, TMPNN provides improved generalizability across various graph structures and enhanced data efficiency.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 INTRODUCTION

Probabilistic inference includes maximum-a-posteriori (MAP) inference and marginal inference. Given a joint probability distribution of a set of random variables, marginal inference is to infer the marginal probability of an unobserved variable and MAP inference is to infer the most probable configuration of a subset of unobserved variables. A probabilistic graphical model (PGM) is usually employed for a compact representation of joint distribution. Probabilistic inference is then carried out by leveraging the structure of a PGM. In this work, we focus on probabilistic inference in the context of PGMs. Particularly, we consider pairwise Markov random fields (MRFs).

For tackling approximate inference tasks, variational approaches have gained broad acceptance. Variational belief propagation (BP) solves for a variational distribution by minimizing the Bethe free energy through message passing (Yedidia et al., 2000, 2001, 2003). Its inference performance thus is inherently limited by the Bethe assumption which holds true only on loop free graphs. Different algorithms are proposed to adapt the Bethe assumption for addressing inference on loopy graphs (Wainwright et al., 2003; Hazan and Shashua, 2010; Riegler et al., 2012). In these algorithms, message equations are correspondingly derived, specific to the pre-defined variational assumptions. Recently, message passing neural networks (MPNNs) (Gilmer et al., 2017) are explored for improving inference performance (Yoon et al., 2019; Huynh, 2021). These models are still under the Bethe assumption and black-box neural networks are employed for computing messages in a heuristic way. Exact inference results as strong supervision are required for training. Differently, Cui et al. (2022) proposed a V-MPNN for a learnable variational free energy without being limited to a fixed variational assumption. However, black-box neural networks are still required for message computation in V-MPNN. Furthermore, it is worth noting that all the aforementioned neural

models focus on learning neural messages specifically tailored for either the marginal or MAP inference task.

In this paper, we propose a *theory-guided message passing neural network* (TMPNN) for probabilistic inference. Inspired by the existing work (Cui et al., 2022), we consider a generalized Bethe free energy for a learnable variational distribution that is not limited to a fixed variational assumption. We utilize the general message equation and introduce a symbolic message function with semantically meaningful parameters, deviating from the conventional approach of learning black-box neural messages. We seamlessly integrate the analytically derived symbolic message function into the MPNN framework, leading to the proposed TMPNN. TMPNN is trained using algorithmic supervision without requiring exact inference results. Our contributions lie in three parts:

- We design a novel symbolic message function, featuring semantically meaningful parameters. This symbolic message function enables the incorporation of learnable variational assumptions and is seamlessly integrated into the MPNN framework.
- TMPNN represents a notable advancement in theoretical guarantees over existing heuristic neural models. Moreover, TMPNN excels in its ability to handle both MAP inference and marginal inference tasks.
- TMPNN surpasses state-of-the-art methods for both inference tasks. Furthermore, TMPNN demonstrates improved generalizability across various graph structures and exhibits superior data efficiency compared to existing approaches.

2 RELATED WORKS

Belief propagation (BP) was introduced by (Pearl, 2014) and various methods have been suggested to enhance the performance of BP on loopy graphs (Koehler, 2019; Knoll et al., 2018; Elidan et al., 2012; Knoll et al., 2015; Aksenov et al., 2020; Murphy et al., 2013; Pretti, 2005). Further theoretical insights are introduced whereby the connection between BP and the Bethe free energy is established (Yedidia et al., 2000, 2001, 2003). To improve the performance of variational BP in loopy graphs, some works consider generalizing the Bethe free energy, such as the TRW-BP (Wainwright et al., 2003) for marginal inference and the TRW-MP (Wainwright et al., 2005) for MAP inference. These variational BP algorithms require a pre-defined and fixed variational assumption. Correspondingly, their derived message equations are specific to the pre-defined variational assumption. Other studies propose convexifying the Bethe free energy by introducing a set of entropy parameters (Meshi et al., 2012; Hazan and Shashua, 2010). The message equation is derived as a function of the entropy parameters. In addition, the convergence performance of message passing can be analyzed by studying the variational free energy (Savchynskyy et al., 2011, 2012; Hazan and Shashua, 2012; Weiss et al., 2012; Meshi et al., 2015).

Recent works begin to exploit neural networks for improved probabilistic inference performance. Node-GNN (Yoon et al., 2019) directly adopted MPNN to perform probabilistic inference. The architecture and the training objective of Node-GNN are tailored separately for the distinct requirements of MAP and marginal inference tasks. GNN-Mimic-BP (Huynh, 2021) improved node-GNN by customizing neural network functions based on the message equation from BP for marginal inference. Satorras and Welling (2020) proposed to combine the messages from BP with messages estimated via neural networks, where neural messages serve as a refinement of BP messages. These works employ neural networks for message computation. Belief propagation neural network (BPNN) (Kuck et al., 2020) was proposed for improving the convergence performance on loopy graphs. Besides using neural messages, a Bethe free energy is approximated via a multi-layer perceptron and is used for regularization purposes. BPNN's theoretical guarantee is limited to convergence performance. All the aforementioned works adhere to the Bethe assumption in accordance with BP. Furthermore, they require exact inference results for training.

Instead of being limited to a specific variational assumption, Cui et al. (2022) proposed a V-MPNN for a learnable variational distribution. V-MPNN is trained by minimizing a neural-augmented free energy, without requiring exact inference results. However, black-box neural networks are required for message computation. The training of V-MPNN can often converge to non-optimal points due to the absence of strong guidance. Besides, V-MPNN with neural messages can't generalize well to unseen structures and is only applied to MAP inference. Compared to V-MPNN, TMPNN employs an analytically derived message function with semantically meaningful parameters. TMPNN can better generalize to unseen structures and is more data efficient. Moreover, it can be applied to both MAP and marginal inference tasks. In Appendix A, we provide a comprehensive summary of the distinctions between the aforementioned neural methods and TMPNN.

3 PROPOSED METHOD

Consider a set of random variables $\boldsymbol{X} = \{X_1, X_2, ..., X_N\}$ in discrete space $\chi = \chi_1 \times \chi_2 \times ... \times \chi_N$.

$|\chi_i| = k$ is the number of possible states of each variable X_i. N is the total number of variables. A pairwise MRF $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ with $|\mathcal{V}| = N$ and a set of edges $\mathcal{E}$ with $|\mathcal{E}| = M$ where M is the total number of edges in the graph. MRF captures the joint probability distribution of $\boldsymbol{X}$ as $p(\boldsymbol{X} = \boldsymbol{x}) \propto \prod_{i\in\mathcal{V}} \theta_i(x_i) \prod_{(i,j)\in\mathcal{E}} \theta_{ij}(x_i, x_j)$, where θ_i and θ_{ij} are potential functions. Given a graph $\mathcal{G}$ and its probability parameters $\boldsymbol{\Theta} = \{\theta_i, \theta_{ij} : i \in \mathcal{V}, (i,j) \in \mathcal{E}\}$, marginal inference infers the marginal probability of a variable X_i, i.e., $p(X_i)$. MAP inference infers the most probable configuration, i.e., $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\chi} p(\boldsymbol{X} = \boldsymbol{x})$. For simplicity, we denote $p(\boldsymbol{X} = \boldsymbol{x})$ as $p(\boldsymbol{x})$. In the following, we first study the general message equation and then propose TMPNN whereby we introduce the symbolic message function and integrate it into an MPNN.

3.1 General Message Equations

Generalized Bethe free energy. With a variational approach, a variational distribution $q(\boldsymbol{x})$ is obtained by minimizing a variational free energy under a variational assumption. Inference is then performed using $q(\boldsymbol{x})$. In pairwise MRF, it is commonly assumed that $q(\boldsymbol{x})$ is a function of unary marginals $\boldsymbol{q}^{node} = \{q_i(X_i)\}_{i\in\mathcal{V}}$ and pairwise marginals $\boldsymbol{q}^{edge} = \{q_{ij}(X_i, X_j)\}_{(i,j)\in\mathcal{E}}$, which we refer to as the *pairwise assumption*. We term the general form of the variational free energy under the pairwise assumption as *generalized Bethe free energy*, i.e.,

$$\begin{aligned}&\mathcal{F}_{\text{general}}(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}; \mathcal{C})\\ &= U(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}) - \mathcal{T}H(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}; \mathcal{C})\end{aligned} \tag{1}$$

$\mathcal{T}$ is the temperature: $\mathcal{T} = 1$ is for marginal inference and $\mathcal{T} \to 0$ is for MAP inference (Weiss et al., 2012). U refers to the average energy. In pairwise MRF, it is computed as $U = -\sum_{i\in\mathcal{V}} \sum_{x_i} q_i(x_i) \ln \theta_i(x_i) - \sum_{(i,j)\in\mathcal{E}} \sum_{x_i,x_j} q_{ij}(x_i, x_j) \ln \theta_{ij}(x_i, x_j)$. H refers to entropy and is calculated as a linear combination of $H(q_i)$ and $H(q_{ij})$ with a set of entropy coefficients $\mathcal{C} = \{c_i, c_{ij}\}_{i\in\mathcal{V},(i,j)\in\mathcal{E}}$, i.e.,

$$H(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}; \mathcal{C}) = \textstyle\sum_{i\in\mathcal{V}} c_i H(q_i) + \sum_{(i,j)\in\mathcal{E}} c_{ij} H(q_{ij}) \tag{2}$$

where $H(q_i) = -\sum_{x_i} q_i(x_i) \ln q_i(x_i)$ and $H(q_{ij}) = -\sum_{x_i,x_j} q_{ij}(x_i, x_j) \ln q_{ij}(x_i, x_j)$. Different sets of coefficients represent different variational assumptions. For example, in BP, we have $\mathcal{C}^{\text{BP}} = \{c_i = 1 - |\mathcal{N}(i)|; c_{ij} = 1\}_{i\in\mathcal{V},(i,j)\in\mathcal{E}}$ where $\mathcal{N}(i)$ denotes a set of neighboring nodes of i-th node. $\mathcal{F}_{\text{general}}(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}; \mathcal{C}^{\text{BP}})$ is equivalent to the Bethe free energy. While $\mathcal{C}^{BP}$ ensures accurate inference on loop-free graphs, different sets of coefficients are introduced to handle loopy graphs for accurate inference, e.g., edge appearance probability based on spanning trees in TRW-MP. The entropy is strictly concave if $\mathcal{C} \in \mathbb{C}^{concave}$ where $\mathbb{C}^{concave} = \{\exists\ \hat{c}_{ij} > 0, \hat{\alpha}_{ij} \geq 0$ such that $c_i = \hat{c}_i - \sum_{j\in\mathcal{N}(i)} \hat{\alpha}_{ij}, c_{ij} = \hat{c}_{ij} + \hat{\alpha}_{ij} + \hat{\alpha}_{ji}\}$ (Heskes, 2004; Weiss et al., 2012; Cui et al., 2022).

General equations. An optimal variational distribution $\boldsymbol{q}^* = \{q_i^*, q_{ij}^*\}$ is obtained by minimizing $\mathcal{F}_{\text{general}}$ subject to the local pairwise consistency constraint $\{q_i(x_i) = \sum_{x_j} q_{ij}(x_i, x_j);\ \forall (i,j) \in \mathcal{E}\}$[1]. This constrained optimization is solved through message passing in variational BP algorithms. Traditional variational BP algorithms compute messages specific to a predefined variational assumption (Yedidia et al., 2001, 2003; Wainwright et al., 2003). Neural messages in (Cui et al., 2022), though not specific to a pre-defined variational assumption, are computed heuristically without theoretical guarantee. Instead, inspired by the work from Meshi et al. (2012), we consider the general message equation that is analytically derived from $\mathcal{F}_{\text{general}}$. The derived equation is a function of entropy coefficients $\mathcal{C}$. Detailed derivations are shown in Appendix B.1. For MAP inference ($\mathcal{T} \to 0$), the message from j-th node to i-th node, i.e., $\hat{\mu}_{ji}$, is computed as

$$\begin{aligned}&\hat{\mu}_{ji}(x_i) \propto\\ &\max_{x_j} \theta_j(x_j)^{1/\hat{c}_j} \theta_{ij}(x_i, x_j)^{1/c_{ij}} \frac{\prod_{k\in\mathcal{N}(j)} \hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}\end{aligned} \tag{3}$$

where $\hat{c}_i = c_i + \sum_{k\in\mathcal{N}(i)} c_{ik}$. The belief equations for q_i and q_{ij} are computed as

$$\begin{aligned}q_i(x_i) &\propto \theta_i(x_i)^{1/\hat{c}_i} \prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)\\ q_{ij}(x_i, x_j) &\propto \theta_i(x_i)^{1/\hat{c}_i} \theta_j(x_j)^{1/\tilde{c}_j} \theta_{ij}(x_i, x_j)^{1/c_{ij}} \times\\ &\quad \frac{\prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)}{\hat{\mu}_{ji}} \frac{\prod_{k\in\mathcal{N}(j)} \hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}\end{aligned} \tag{4}$$

For marginal inference, the message equation is closely related to the equation for MAP inference under the zero temperature lemma (Weiss et al., 2012). Specifically, the message equation is obtained by replacing the maximization operation in Eq. 3 with a summation operation (derivations are in Appendix B.2). The belief equation for marginal inference remains the same as Eq. 4. Existing variational BP algorithms can be treated as instantiations of the derived general equations with different configurations of $\mathcal{C}$. We show BP and TRW-MP as two examples in Appendix B.3. Our equations share resemblance to those presented in Meshi et al. (2012). However, it is important to note that Meshi et al. (2012) exclusively focused on

[1] Both marginal and pairwise distributions are valid, each satisfying the normalization constraint. We thus exclude this constraint for simplicity.

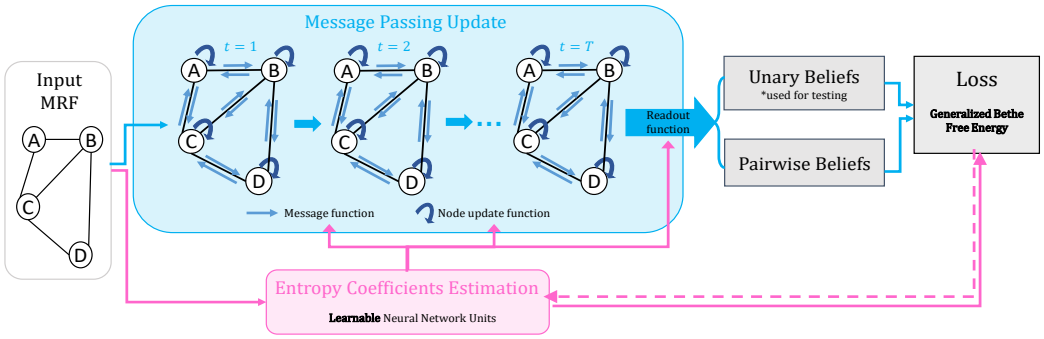

Figure 1: Overview of TMPNN which consists of two modules: the entropy coefficients estimation module and the message passing update module. Solid lines denote forward calculation and dashed lines denote backward update.

marginal inference, while our work extends to MAP inference with detailed derivations.

3.2 Theory-guided Message Passing Neural Network

An overview of TMPNN is shown in Figure 1. We first discuss the architecture in Sec. 3.2.1. The theoretical guarantees of TMPNN are offered in Sec. 3.2.2, followed by the training of TMPNN. We devote our discussion primarily to the introduction of TMPNN for MAP inference, while addressing its application in marginal inference towards the end of this section.

3.2.1 Architecture

The architecture of TMPNN consists of two modules: *entropy coefficients estimation module* and *message passing update module*. Entropy coefficients estimation module takes graph structure and probability parameters as input and outputs estimated entropy coefficients. Taking the estimated entropy coefficients as input, the message passing update module outputs the estimated unary and pairwise beliefs. The entropy coefficients estimation module is trainable through which an optimal family of variational distributions can be learned. The message passing update module is based on the derived general message equation whose parameters are entropy coefficients.

Entropy coefficients estimation. Entropy coefficients $\mathcal{C} = \{c_i,\ c_{ij} : i \in \mathcal{V}, (i,j) \in \mathcal{E}\}$ are estimated based on the graph structure and probability parameters Θ. Unary entropy coefficients c_i and pairwise entropy coefficients c_{ij} are estimated through two separate neural networks. Specifically, an MLP is adopted for the estimation of c_i:

$$c_i = \mathcal{F}_{node}(\texttt{cat}[\theta_i, |\mathcal{N}(i)|, \sum_{j \in \mathcal{N}(i)} \theta_{ij}]; \Phi_{node}) \quad (5)$$

where Φ_{node} indicates free parameters in $\mathcal{F}_{node}$. Input features consist of unary potential θ_i, the number of neighboring nodes $|\mathcal{N}(i)|$, and the summation of pairwise potentials over neighboring nodes $\sum_{j \in \mathcal{N}(i)} \theta_{ij}$. They are assembled via a concatenation operation $\texttt{cat}$. For pairwise entropy coefficients estimation, a graph attention layer (Veličković et al., 2017) is adopted and c_{ij} is estimated as:

$$\begin{aligned} c_{ij} &= \mathcal{F}_{edge}(\theta_i, \theta_j, \theta_{ij}; \Phi_{edge}) \\ &= \texttt{exp}(\texttt{LeakyReLU}(\boldsymbol{a}^T \texttt{cat}[W\theta_i, W\theta_j, \theta_{ij}])) \end{aligned} \quad (6)$$

where $\Phi_{edge} = \{W, \boldsymbol{a}\}$ are unknown parameters in $\mathcal{F}_{edge}$. We impose the strictly positive constraint on the estimation of c_{ij} by introducing $\texttt{exp}$ to the output layer of $\mathcal{F}_{edge}$. In summary, $\Phi = \{\Phi_{node}, \Phi_{edge}\}$ are free parameters in TMPNN. They are employed for modeling non-linear relationships between an input graph and a distribution family. Unlike the neural-augmented free energy in V-MPNN (Cui et al., 2022), TMPNN embraces an explicit estimation of coefficients using the entropy coefficients estimation module.

Message passing update. Given the estimated entropy coefficients, TMPNN performs message passing updates under an MPNN framework. MPNN is a recurrent network with nodes and a graph structure. Node features are updated iteratively based on the features of neighboring nodes, determined by the graph structure. The features are updated through a message function $\mathcal{M}$ followed by a node update function $\mathcal{U}$. Finally, a readout function $\mathcal{R}$ is employed to utilize the features for predictions. Typically, these functions are implemented using non-linear neural networks, such as multi-layer perceptrons with ReLU activation.

To leverage the MPNN for probabilistic inference, we map each node in MPNN to a variable in MRF with hidden feature $\boldsymbol{h}_i \in R^k$. k is the number of possible states of variable x_i. Hidden feature $\boldsymbol{h}_i$ is equivalent to unary belief up to a scale factor z_i in logarithmic space.

Instead of using black-box neural message functions, we define symbolic message function $\mathcal{M}$ based on the derived general message equation (Eq. 3). At each iteration t, each node receives a message from each of its neighboring nodes as

$$\begin{aligned} m_{j\to i}^{t+1}(x_i) &= \mathcal{M}(\boldsymbol{h}_j^t, z_j^t, m_{i\to j}^{t+1}) \\ &= \ln\{\max_{x_j} \exp(\frac{1}{c_{ij}}\phi_{ij}(x_i,x_j) + \boldsymbol{h}_j^t + z_j^t - m_{i\to j}^{t+1})\} \end{aligned} \tag{7}$$

$m_{j\to i}^{t+1}$ is equivalent to the message $\hat{\mu}_{ji}$ in the logarithmic space. $\phi_{ij} = \ln\theta_{ij}$. The messages are aggregated as $m_i^{t+1} = \frac{1}{\hat{c}_i}\sum_{k\in\mathcal{N}(i)} c_{ki} m_{k\to i}^{t+1}$. Hidden feature of each node is updated with the aggregated message through a customized node update function $\mathcal{U}$ as

$$\boldsymbol{h}_i^{t+1}(x_i) = \mathcal{U}(z_i^{t+1}, m_i^{t+1}) = -z_i^{t+1} + \tfrac{1}{\hat{c}_i}\phi_i(x_i) + m_i^{t+1}(x_i) \tag{8}$$

where $\phi_i = \ln\theta_i$. Scale factor of each node is also updated as $z_i^{t+1} = \ln(\sum_{x_i}\exp(\frac{1}{\hat{c}_i}\phi_i(x_i) + m_i^{t+1}(x_i)))$. The update process is repeated until convergence. The estimated unary and pairwise beliefs are obtained through respective customized readout functions $\mathcal{R}^{node}$ and $\mathcal{R}^{edge}$ as

$$\begin{aligned} &q_i = \mathcal{R}^{node}(\boldsymbol{h}_i^T) = \exp(\boldsymbol{h}_i^T) \\ &q_{ij}(x_i,x_j) = \mathcal{R}^{edge}(\boldsymbol{h}_i^T, \boldsymbol{h}_j^T) \\ &= \exp(\frac{1}{c_{ij}}\phi_{ij}(x_i,x_j) + \boldsymbol{h}_i^T + \boldsymbol{h}_j^T - m_{i\to j}^T - m_{j\to i}^T) \end{aligned} \tag{9}$$

T is the total number of iterations. Detailed derivations are shown in Appendix C.

3.2.2 Theoretical Guarantees

Leveraging the theory-guided functions, TMPNN's performance is supported by stronger theoretical guarantees through the following proposition compared to existing neural models.

Proposition 1. *Variational distribution $\boldsymbol{q}^*$ from TMPNN is locally optimal with pairwise consistency constraint strictly satisfied. If the set of entropy coefficients satisfy $\mathcal{C} \in \mathbb{C}^{concave}$, $\boldsymbol{q}^*$ becomes globally optimal.*
Through the proof, we can obtain that

$$\begin{aligned} &\frac{\partial \mathcal{F}_{\text{general}}}{\partial q_i^*} = 0,\ \forall i \in \mathcal{V}; \quad \frac{\partial \mathcal{F}_{\text{general}}}{\partial q_{ij}^*} = 0, \quad \forall (i,j) \in \mathcal{E} \\ &\text{subject to} \quad q_i^*(x_i) = \sum_{x_j} q_{ij}^*(x_i,x_j) \end{aligned} \tag{10}$$

It is worth noting that the pairwise consistency constraint must always be maintained as it constitutes a fundamental cornerstone of pairwise MRF. However, pairwise consistency constraint is not guaranteed to be strictly satisfied in the existing neural models. Consequently, their solutions rely on factors such as the learning rate and the gradient descent algorithms. Detailed proofs are in Appx. D.

Furthermore, the MAP inference performance guarantees from Cui et al. (2022) remain effective in TMPNN as shown in following two propositions (proofs are in Appx. D).

Proposition 2. *Given a target probability distribution p, MAP inference error is defined as $\Delta_{map}(\boldsymbol{q}_{\mathcal{C}}^*, p) = \sum_{i\in\mathcal{V}}\sum_{x_i}(p_i(x_i) - q_{\mathcal{C},i}^*(x_i))\ln\theta_i(x_i) + \sum_{(i,j)\in\mathcal{E}}\sum_{x_i,x_j}(p_{ij}(x_i,x_j) - q_{\mathcal{C},ij}^*(x_i,x_j))\ln\theta_{ij}(x_i,x_j)$ where $p_i(x_i) = \sum_{\boldsymbol{x}\setminus x_i} p(\boldsymbol{x})$ and $p_{ij}(x_i,x_j) = \sum_{\boldsymbol{x}\setminus(x_i\cup x_j)} p(\boldsymbol{x})$. With an optimal configuration of entropy coefficients $\mathcal{C}^* \in \mathbb{C}^{concave}$, the MAP inference error of TMPNN is upper bounded by an entropy approximation scaled by ϵ, i.e., $\Delta_{map}(\boldsymbol{q}_{\mathcal{C}^*}^*, p) \le \epsilon H(\boldsymbol{q}_{\mathcal{C}^*}^*; \mathcal{C}^*)$. An optimal configuration of entropy coefficients is obtained as $\mathcal{C}^* = \arg\min_{\mathcal{C}\in\mathbb{C}^{concave}} H(\boldsymbol{q}_{\mathcal{C}}^*; \mathcal{C})$.*

Recall that in *generalized Bethe free energy* (Eq. 1), temperature $\mathcal{T} \to 0$ stands for MAP inference. In Proposition 2, we symbolically denote $\mathcal{T} \to 0$ using a sufficiently small constant value ϵ for proof.

Proposition 3. *TMPNN subsumes existing variational BP algorithms (e.g., BP and TRW-MP) as a strict generalization by incorporating entropy coefficients. TMPNN tends to achieve superior or at least comparable inference performance against existing variational BPs.*

The proof of Proposition 3 is independent of the optimization approach but considers the optimal scenario where TMPNN finds the optimal configuration of entropy coefficients $\mathcal{C}^*$.

3.2.3 Training Objective

The training objective for updating Φ is defined based on the generalized Bethe free energy. Particularly, given a current entropy coefficients estimation $\mathcal{C}(\Phi)$ together with the estimated beliefs, we obtain an entropy approximation $H(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}; \mathcal{C}(\Phi))$. Based on the proposition 2, we set $H(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}; \mathcal{C}(\Phi))$ as our main loss function and Φ is thus updated toward the direction of minimizing the inference error bound. We don't use ϵ during training since ϵ is a constant. An additional regularization term is defined based on the mean squared error between estimated $\mathcal{C}(\Phi)$ and $\mathcal{C}^{\text{BP}}$. This term is to avoid unreasonable estimation of $\mathcal{C}$ that can cause numerical instability. In the end, the training loss is defined as

$$\mathcal{L} = H(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}; \mathcal{C}(\Phi)) + \lambda||\mathcal{C}(\Phi) - \mathcal{C}^{\text{BP}}||_2^2 \tag{11}$$

λ is an importance weight of the regularization term. After training, optimal parameters Φ^* are used for MAP inference and MAP configuration is obtained as $\hat{x}_i = \arg\max_{x_i\in\chi_i} q_i(x_i; \Phi^*)$.

It is worth noting that, in Proposition 2, we de-

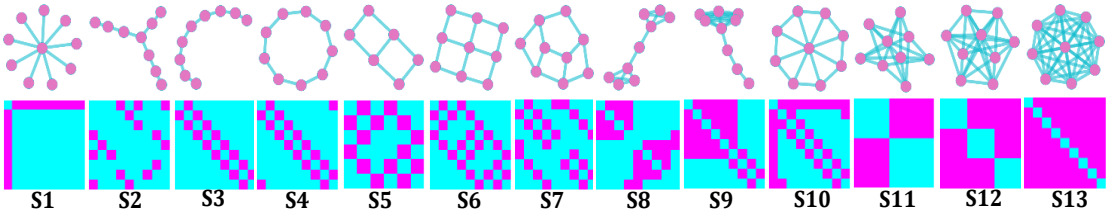

Figure 2: 13 Benchmark Graphs With 9 Nodes (Left to Right): STAR♣ (S1); TREE♣ (S2); PATH♣ (S3); CIRCLE (S4); LADDER (S5); GRID (S6); CIRCULAR LADDER (S7); BARBELL (S8); LOLLIPOP (S9); WHEEL (S10); BIPARTITE (S11); TRIPARTITE (S12); COMPLETE (S13). The top row visualizes the structure, and the bottom row shows the adjacency matrix. ♣ marks loop-free graphs.

rive $\Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}}, p) \leq \epsilon(H(\boldsymbol{q}^*_{\mathcal{C}}; \mathcal{C}) - H(\boldsymbol{p}^{node}, \boldsymbol{p}^{edge}; \mathcal{C})) \leq \epsilon H(\boldsymbol{q}^*_{\mathcal{C}}; \mathcal{C})$. Theoretically, to achieve equality (tight bound), the first inequality requires $\boldsymbol{q}^*_{\mathcal{C}}$ equals to the target distribution $\boldsymbol{p}$ while the second inequality requires $H(\boldsymbol{p}^{node}, \boldsymbol{p}^{edge}; \mathcal{C}) = 0$. The tightness of the second bound does not influence the training process since $H(\boldsymbol{p}^{node}, \boldsymbol{p}^{edge}; \mathcal{C})$ is a constant. Empirically, we report the gap $(\epsilon H(\boldsymbol{q}^*_{\mathcal{C}}) - \Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}}, p))$ in Appx. E.1. As shown, on loop free graphs, the upper bound is tight. On loopy graphs, the gap increases as the graph complexity increases, and is positively correlated to the inference error.

3.3 TMPNN for Marginal Inference

The TMPNN can straightforwardly be applied to marginal inference. The message function $\mathcal{M}$ (Eq. 7) is changed by replacing the maximization by a summation as $m^{t+1}_{j\to i}(x_i) = \mathcal{M}(\boldsymbol{h}^t_j, z^t_j, m^{t+1}_{i\to j}) = \ln\{\sum_{x_j} \exp(\frac{1}{c_{ij}}\phi_{ij}(x_i, x_j) + \boldsymbol{h}^t_j + z^t_j - m^{t+1}_{i\to j})\}$. The rest part of the architecture remains the same. For training, the loss function (Eq. 11) remains effective as demonstrated through experiments. After training, the marginal probability for x_i is estimated as $q_i(x_i; \Phi^*)$.

4 EXPERIMENTS

Datasets. Thirteen benchmark graphs are considered for evaluation following (Yoon et al., 2019; Huynh, 2021; Cui et al., 2022). These graphs consist of three loop-free graphs (STAR, TREE and PATH) and ten loopy graphs, as visualized in Figure 2. For clarification, we mark loop-free graphs with ♣. These graphs define the structures of MRFs. To generate probability parameters, following Wainwright et al. (2005), we assume $\theta_i(x_i) = b_i x_i$ and $\theta_{ij}(x_i, x_j) = J_{ij} x_i x_j$ with $x_i = \{-1, 1\}$. Pairwise parameter J_{ij} is sampled from a uniform distribution as $J_{ij} \sim U[-1, 1]$. We consider the pairwise coupling with the strength ranging from -1 to +1, which represents a general graph that has traditionally posed challenges for BP methods (Knoll et al., 2018). Unary parameter b_i is sampled from a uniform distribution as $b_i \sim U[-0.05, 0.05]$. For each type of graph, we simulate 1000 graphs for training and 100 graphs for testing. Exact inference results are computed by enumeration. Since enumeration is a computationally expensive process, we limit the sizes of graphs to 9 and 16 nodes for evaluation.

Experiment settings. In Eq. 5, $\mathcal{F}_{node}$ employs a three-layer MLP with hidden dimension being 256. In Eq. 6, $W \in R^{k\times d}$ with $k = 2$ and $d = 256$. $\boldsymbol{a}$ is a weight vector with dimension $2d + 1$ and $d = 256$. In Eq. 11, $\lambda = 0.001$. ADAM optimizer is employed with a learning rate of $1e - 5$. Messages are initialized to ones. For better training convergence, TMPNN is pre-trained on loop-free graphs (STAR, TREE, PATH) using inference results from BP as supervision. The detailed settings and the effectiveness of the pre-training are studied in Appendix E.2. The effects from message initialization are examined in Appendix E.3. Experiments are performed on a laptop with an i9 processor and CPU only.

Evaluation metrics. For MAP inference, given an exact MAP configuration $\boldsymbol{x}^* = \{x^*_1, ..., x^*_N\}$ and an estimated one $\hat{\boldsymbol{x}} = \{\hat{x}_1, ..., \hat{x}_N\}$, the accuracy is $\frac{\#(x^*_i = \hat{x}_i)}{N}$ (Yoon et al., 2019; Cui et al., 2022). For marginal inference, we use the negative log KL divergence with base 10 (i.e., $-\log_{10}\mathrm{D_{KL}}$) between the estimated marginal probability and the exact one (Huynh, 2021)[2]. For both metrics, a higher value is preferable.

4.1 Comparison to SOTA Methods

We first compare the inference performance of TMPNN against SOTA methods for both MAP inference and marginal inference. We consider both the training-free methods and the training-based methods for comparison. The training-free methods refer to the optimization algorithms that don't require training. The training-based methods refer to the neural models that require training. When comparing to training-based

[2]For exact inference, KL divergence is zero leading to an invalid value. Instead, $1e - 10$ is used.

Table 1: Comparison to SOTA Training-free Methods for MAP Inference (Accuracy; loop-free♣)

Graph	N=9				N=16			
	BP	TRW-MP	MPLP	TMPNN	BP	TRW-MP	MPLP	TMPNN
STAR (S1)♣	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
TREE (S2)♣	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
PATH (S3)♣	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
CYCLE (S4)	**.92**	.81	.89	.90	.89	.88	.82	**.92**
LADDER (S5)	.63	.58	**.79**	.78	.60	.47	.70	**.76**
GRID (S6)	.63	.55	.77	**.79**	.58	.50	.65	**.69**
CIRCULAR LADDER (S7)	.84	.85	.89	**.90**	.68	.51	.69	**.73**
BARBELL (S8)	.59	.52	.65	**.75**	.58	.49	.63	**.70**
LOLLIPOP (S9)	.78	.84	.83	**.88**	.60	.54	.63	**.68**
WHEEL (S10)	.56	.46	.61	**.73**	.59	.42	.63	**.71**
BIPARTITE (S11)	.62	.47	.64	**.77**	.61	.50	.51	**.73**
TRIPARTITE (S12)	.54	.50	.54	**.67**	.60	.49	.54	**.64**
COMPLETE (S13)	.43	.52	.50	**.63**	.49	.50	.51	**.62**
Average	.73	.70	.78	**.83**	.71	.64	.72	**.78**

methods, we present the average accuracy and standard deviation derived from 10 separate training runs with neural model parameters initialized randomly. We limit our comparisons to algorithms that also follow a message passing scheme. In addition, we also achieve competitive performance compared to sampling-based methods (detailed comparisons are in Appendix E.4).

4.1.1 MAP Inference

Training-free methods. We consider three training-free methods: BP (Murphy et al., 2013), TRW-MP (Wainwright et al., 2005) and max product linear programming (MPLP) (Globerson and Jaakkola, 2007) for comparison. We use the typical stopping criteria and break the algorithm if the number of iterations is larger than 200 or the average difference between unary beliefs from two consecutive iterations is sufficiently small, i.e., $\frac{1}{N}\sum_{i=1}^{N}||q_i^{t+1} - q_i^t||_2^2 < 1e-7$. In BP, we apply the default damping parameter 0.5. In TRW-MP (Wainwright et al., 2005), message damping is applied with damping parameter being 0.5, and edge appearance probability is set as $\rho_{ij} = \frac{|\mathcal{V}|-1}{|\mathcal{E}|}$.

As shown in Table 1, TMPNN achieves the best average performance over graphs of different structures for both N=9 and N=16. In loopy graphs, by leveraging a learnable variational distribution without being limited to a fixed variational assumption, TMPNN significantly outperforms other methods, particularly in larger and more complex graphs. In total, TMPNN outperforms in 8 out of 10 cases for N=9 and in all 10 cases for N=16. For example, in BIPARTITE with 9 nodes, TMPNN achieves 77% accuracy, which is 30% better than TRW-MP.

Training-based methods. We consider the fully-supervised method node-GNN (Yoon et al., 2019)[3] and the weakly-supervised method V-MPNN (Cui et al., 2022) for comparison. Following the original settings, V-MPNN is pre-trained. Node-GNN is trained using exact inference results without pre-training. Suggested hyper-parameters are used. As shown in Table 2, TMPNN achieves better average performance than V-MPNN, both of which do not require GT MAP configurations. By leveraging the analytically derived functions, TMPNN performs exact inference on loop-free graphs and outperforms V-MPNN on different structures. For example, on COMPLETE (S13) with 16 nodes, TMPNN is 5% better than V-MPNN. Furthermore, without using exact inference results, TMPNN even outperforms the fully-supervised method Node-GNN on average. Though Node-GNN performs well on larger graphs, it is heuristic and is limited by the graph complexity since the exact inference results are not obtainable on more complex graphs.

For better readability of Table 1 and Table 2, we visualize the values in Appendix E.5. We further examine the pre-training of Node-GNN using loop free graphs. Our results show that pre-training of Node-GNN helps improve average accuracy by 0.09 on loop-free graphs, but experiences a decrease on loopy graphs (e.g., 0.03 ↓ on the complete graph). Conclusions in comparison to TMPNN remain same.

4.1.2 Marginal Inference

Training-free methods. We consider two training-free methods: BP (Murphy et al., 2013) and TRW-BP (Wainwright et al., 2005) for comparison. Algorithm settings remain the same as we used for MAP inference evaluation. Due to space constraints, we

[3] https://github.com/ks-korovina/pgm_graph_inference.

Table 2: Comparison to SOTA Training-based Method for MAP Inference (Accuracy; loop-free♣)

Graph	N=9			N=16		
	Full supervision	Weak Supervision		Full supervision	Weak Supervision	
	Node-GNN	V-MPNN	TMPNN	Node-GNN	V-MPNN	TMPNN
STAR (S1)♣	.88±.06	.93±.02	**1.00**±.00	.74±.06	.71±.04	**1.00**±.00
TREE (S2)♣	.84±.02	.96±.02	**1.00**±.00	.83±.01	.92±.00	**1.00**±.00
PATH (S3)♣	.78±.01	.97±.00	**1.00**±.00	.78±.01	.92±.04	**1.00**±.00
CYCLE (S4)	.85±.01	.85±.01	**.90**±.00	.78±.01	.86±.03	**.92**±.01
LADDER (S5)	.75±.01	.77±.02	**.78**±.01	.72±.01	.72±.02	**.76**±.03
2D GRID (S6)	**.81**±.00	.74±.03	.79±.01	**.74**±.01	.69±.03	.69±.00
CIRCULAR LADDER (S7)	.80±.01	.83±.05	**.90**±.05	**.76**±.01	.74±.05	.73±.00
BARBELL (S8)	**.76**±.01	.71±.03	.75±.02	**.73**±.00	.69±.03	.70±.00
LOLLIPOP (S9)	.79±.00	**.88**±.02	**.88**±.00	**.69**±.01	.67±.01	.68±.00
WHEEL (S10)	**.77**±.02	.70±.04	.73±.01	**.71**±.02	.66±.04	**.71**±.01
BIPARTITE (S11)	**.83**±.01	.74±.06	.77±.02	**.79**±.00	.65±.03	.73±.04
TRIPARTITE (S12)	**.75**±.01	.68±.03	.67±.00	**.74**±.01	.60±.01	.64±.00
COMPLETE (S13)	**.77**±.01	.65±.04	.63±.00	**.71**±.01	.57±.01	.62±.00
Average	.80±.01	.80±.03	**.83**±.01	.75±.01	.72±.03	**.78**±.01

present the average performance across 13 structures. Detailed results for individual structures are in Appendix E.6. As shown in Table 3 and the appendix, TMPNN achieves better average performance for both sizes and it performs significantly better than SOTA training-free methods on loopy graphs. Specifically, TMPNN demonstrates the best performance in 10 out of 10 types of loopy graphs for N=9, and in 6 out of 10 types for N=16. On COMPLETE(S13), TMPNN achieves 3.21, outperforming BP (achieves 0.60) by about two orders of magnitude on graphs of size 16 (the measure is in logarithmic scale).

Table 3: Comparison to SOTA Training-free Methods for Marginal Inference ($-\log_{10}\mathrm{D_{KL}}$)

	N=9			N=16		
	BP	TRW-BP	TMPNN	BP	TRW-BP	TMPNN
Ave.	6.23	5.89	**6.66**	5.24	5.04	**5.70**

Training-based methods. We consider two fully-supervised training-based methods: node-GNN (Yoon et al., 2019) and GNN-Mimic-BP (Huynh, 2021) for comparisons. For GNN-Mimic-BP, without access to source codes, we only compare to its reported performance on loop-free graphs (marked with *). Both BPNN (Kuck et al., 2020) and NEBP (Satorras and Welling, 2020) focus on factor graphs. Due to the lack of access to their source codes and datasets, we exclude them from comparison. V-MPNN is only for MAP inference and thus is excluded. We present the average performance in Table 4 and the detailed results are in Appendix E.6. As shown in Table 4 and the appendix, TMPNN again achieves better average performance, particularly on loop-free graphs. For example, TMPNN outperforms node-GNN and GNN-Mimic-BP by about four orders of magnitude on STAR and TREE with 9 nodes, without requiring exact inference results.

In addition, we evaluate the efficiency of TMPNN against SOTA methods. Detailed experiment settings and results are in Appendix E.7. As shown, TMPNN is computationally efficient compared to training-free baselines. For example, on graphs with 16 nodes, TMPNN on average takes $0.0288s$ for one testing graph, while TRW-BP needs $0.2649s$, which is around 10 times longer than TMPNN. On the other hand, TMPNN's efficiency does not surpass that of neural-based models, due to the utilization of analytical message functions. Similar to conventional variational BPs with analytically derived message functions, the computational complexity of TMPNN increases with graph complexity. While neural messages' complexity only depends on the neural network architecture and doesn't change w.r.t. graph complexity. This finding reflects a typical trade-off between accuracy and efficiency, and our primary goal does not prioritize enhancing efficiency.

4.2 Ablation Studies

To further study the effectiveness of the theory-guided message function over neural messages, we perform ablation studies to evaluate the generalization ability and the data efficiency of TMPNN against V-MPNN. We consider MAP inference on graphs with 9 nodes.

Generalization. We perform a cross-graph evaluation to study the generalization ability from loop-free graphs to loopy graphs. Experiment settings and detailed results on each individual structure are shown in Appendix E.8. We here report the average performance

Table 4: Comparison to SOTA Training-based Methods for Marginal Inference ($-\log_{10}\mathrm{D_{KL}}$)

Graph	N=9			N=16		
	Full Supervision		Weak Supervision	Full Supervision		Weak Supervision
	Node-GNN	GNN-Mimic-BP	TMPNN	Node-GNN	GNN-Mimic-BP	TMPNN
Average	5.51±0.14	5.82*	**6.66**±0.03	5.12±0.13	4.91*	**5.70**±0.04

across graphs. As shown in Table 5, TMPNN has better generalization ability over V-MPNN by leveraging its analytically derived equations.

Table 5: Generalization Evaluation

Method	V-MPNN	TMPNN
Average	.73	**.77**

Data Efficiency. We study the data efficiency of TMPNN by using only 20% of the training data. We report the average MAP inference accuracy in Table 6 and detailed results on each individual structure are shown in Appendix E.9. As shown, given only 20% training data, TMPNN outperforms V-MPNN by 15%, demonstrating that TMPNN is less data dependent by leveraging the analytical equations.

Table 6: Data Efficiency Evaluation

Training Data	V-MPNN	TMPNN
100%	.80	**.83**
20%	.58	**.73**

4.3 Discussions on Larger Graphs

Systematically evaluating TMPNN on large graphs presents a considerable challenge due to the absence of exact inference results. Alternatively, we show the utility of TMPNN on larger graphs through a node classification task. We consider both real-world graphs (i.e., PolBlog (Adamic and Glance, 2005), EuEmail (Leskovec et al., 2007), and CORA (Šubelj and Bajec, 2013)), and synthetic graphs. Detailed settings and results are in Appendix E.10. As shown, TMPNN can apply to large graphs and achieve decent inference performance in terms of both accuracy and efficiency. On CORA, TMPNN achieves 27% accuracy improvement compared to V-MPNN. For efficiency, on PolBlog, the computing time of TMPNN is 14.78s, around 2× faster than TRW-MP which takes 35.34s. Nonetheless, it remains a challenging problem to systematically evaluate TMPNN's performance enhancement as the graph size and complexity increase. We intend to study this issue as part of our future work.

5 CONCLUSIONS

In this work, we proposed a theory-guided message passing neural network (TMPNN) for probabilistic inference. TMPNN is designed based on the derived general message and belief equations. Training of TMPNN is guided by the generalized Bethe free energy and doesn't require exact inference results. Theoretical performance guarantees are provided. We empirically demonstrate the effectiveness of TMPNN by comparing it against SOTA methods for both MAP inference and marginal inference. Ablation studies further show that TMPNN can generalize across graphs with different structures and has better data efficiency. The proposed TMPNN relies on the pairwise assumption, which could be a potential limitation. Extending TMPNN to cluster-based message passing with Kikuchi assumption is our future work. Another limitation arises from the computational complexity due to the utilization of the derived symbolic message function. To further improve the efficiency, our further work includes investigating existing accelerated variational BP algorithms that target mitigating the intrinsic limitation due to the utilization of analytical message functions.

Acknowledgement

This work is supported by the Rensselaer-IBM AI Research Collaboration (http://airc.rpi.edu), part of the IBM AI Horizons Network (http://ibm.biz/AIHorizons).

References

Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

Vitaly Aksenov, Dan Alistarh, and Janne H Korhonen. Relaxed scheduling for scalable belief propagation. *arXiv preprint arXiv:2002.11505*, 2020.

Zijun Cui, Hanjing Wang, Tian Gao, Kartik Talamadupula, and Qiang Ji. Variational message passing neural network for maximum-a-posteriori inference. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

Gal Elidan, Ian McGraw, and Daphne Koller. Residual belief propagation: Informed scheduling for

asynchronous message passing. *arXiv preprint arXiv:1206.6837*, 2012.

Dhivya Eswaran, Srijan Kumar, and Christos Faloutsos. Higher-order label homogeneity and spreading in graphs. In *Proceedings of The Web Conference 2020*, pages 2493–2499, 2020.

Wolfgang Gatterbauer. The linearization of belief propagation on pairwise markov random fields. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Amir Globerson and Tommi Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. *Advances in neural information processing systems*, 20:553–560, 2007.

Tamir Hazan and Amnon Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 56(12):6294–6316, 2010.

Tamir Hazan and Amnon Shashua. Convergent message-passing algorithms for inference over general graphs with convex free energies. *arXiv preprint arXiv:1206.3262*, 2012.

Tom Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, 16(11): 2379–2413, 2004.

Hieu Tri Huynh. *Inference in Ising models by graph neural networks with structural features*. PhD thesis, 2021.

Christian Knoll, Michael Rath, Sebastian Tschiatschek, and Franz Pernkopf. Message scheduling methods for belief propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 295–310. Springer, 2015.

Christian Knoll, Adrian Weller, and Franz Pernkopf. Self-guided belief propagation–a homotopy continuation method. *arXiv preprint arXiv:1812.01339*, 2018.

Frederic Koehler. Fast convergence of belief propagation to global optima: Beyond correlation decay. *Advances in Neural Information Processing Systems*, 32:8331–8341, 2019.

Jonathan Kuck, Shuvam Chakraborty, Hao Tang, Rachel Luo, Jiaming Song, Ashish Sabharwal, and Stefano Ermon. Belief propagation neural networks. *arXiv preprint arXiv:2007.00295*, 2020.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

Ofer Meshi, Ariel Jaimovich, Amir Globerson, and Nir Friedman. Convexifying the bethe free energy. *arXiv preprint arXiv:1205.2624*, 2012.

Ofer Meshi, Mehrdad Mahdavi, and Alex Schwing. Smooth and strong: Map inference with linear convergence. *Advances in Neural Information Processing Systems*, 28, 2015.

Kevin Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. *arXiv preprint arXiv:1301.6725*, 2013.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

Marco Pretti. A message-passing algorithm with damping. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11008, 2005.

Erwin Riegler, Gunvor Elisabeth Kirkelund, Carles Navarro Manchón, Mihai-Alin Badiu, and Bernard Henri Fleury. Merging belief propagation and the mean field approximation: A free energy approach. *IEEE Transactions on Information Theory*, 59(1):588–602, 2012.

Victor Garcia Satorras and Max Welling. Neural enhanced belief propagation on factor graphs. *arXiv preprint arXiv:2003.01998*, 2020.

Bogdan Savchynskyy, Jörg Kappes, Stefan Schmidt, and Christoph Schnörr. A study of nesterov's scheme for lagrangian decomposition and map labeling. In *CVPR 2011*. IEEE, 2011.

Bogdan Savchynskyy, Stefan Schmidt, Jörg Kappes, and Christoph Schnörr. Efficient mrf energy minimization via adaptive diminishing smoothing. *arXiv preprint arXiv:1210.4906*, 2012.

Lovro Šubelj and Marko Bajec. Model of complex networks based on citation dynamics. In *Proceedings of the 22nd international conference on World Wide Web*, pages 527–530, 2013.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *International Workshop on Artificial Intelligence and Statistics*, pages 308–315. PMLR, 2003.

Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE transactions on information theory*, 51(11):3697–3717, 2005.

Yair Weiss, Chen Yanover, and Talya Meltzer. Map estimation, linear programming and belief propagation with convex free energies. *arXiv preprint arXiv:1206.5286*, 2012.

Jonathan S Yedidia, William T Freeman, Yair Weiss, et al. Generalized belief propagation. In *NIPS*, volume 13, pages 689–695, 2000.

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Bethe free energy, kikuchi approximations, and belief propagation algorithms. *Advances in neural information processing systems*, 13:689, 2001.

Jonathan S Yedidia, William T Freeman, Yair Weiss, et al. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.

KiJung Yoon, Renjie Liao, Yuwen Xiong, Lisa Zhang, Ethan Fetaya, Raquel Urtasun, Richard Zemel, and Xaq Pitkow. Inference in probabilistic graphical models by graph neural networks. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 868–875. IEEE, 2019.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No] Source codes will be made publicly available upon publication.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No] Source codes will be made publicly available upon publication.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Comparison to Existing Neural Methods

Given the discussions we provide in the main body of the paper, we provide a comprehensive summary of the distinctions between the aforementioned neural methods and TMPNN in Table 7. We discuss variations in message computation, training objectives, the specific inference tasks they address, and the level of guarantee they offer in terms of inference accuracy. Particularly, for message computation, we consider if the model utilizes classical calculation of messages following variational BP algorithms or the model directly uses black-box neural networks for message computation. For training loss, we consider if the model requires exact inference results as labels to perform fully-supervised training and if the model relies on the variational free energy as a primary week supervision or a regularization term.

Table 7: Comparison to SOTA Training-based Methods

	Message Computation		*Training Loss*		*Inference*	*Accuracy*
	classical calculation	neural network	labels	variational free energy	*Tasks*	*Theoretical Guarantee*
Node-GNN	no	yes	yes	no	both*	no
GNN-Mimic-BP	no	yes	yes	no	marginal	no
NEBP	BP equations	yes	yes	no	marginal	no
BPNN	BP equations	yes	yes	Bethe	marginal	no*
V-MPNN	BP equations	yes	no	neural-augmented	MAP	yes
TMPNN (Ours)	general equations	no	no	generalized Bethe	both	yes

In particular, Node-GNN directly adopted MPNN to perform probabilistic inference. The architecture and the training objective of Node-GNN are tailored separately for the distinct requirements of MAP and marginal inference tasks. GNN-Mimic-BP improved node-GNN by customizing neural network functions based on the message equation from BP for marginal inference. NEBP proposed to combine the messages from BP with messages estimated via neural networks, where neural messages serve as a refinement of BP messages. These works employ neural networks for message computation. Belief propagation neural network (BPNN) was proposed for improving the convergence performance on loopy graphs. Besides using neural messages, a Bethe free energy is approximated via a multi-layer perceptron and is used for regularization purposes. BPNN's theoretical guarantee is limited to convergence performance. All the aforementioned works adhere to the Bethe assumption in accordance with BP. Furthermore, they require exact inference results for training. Instead of being limited to a specific variational assumption, V-MPNN proposed a V-MPNN for a learnable variational distribution. V-MPNN is trained by minimizing a neural-augmented free energy, without requiring exact inference results.

Compared to V-MPNN, TMPNN employs an analytically derived message function with semantically meaningful parameters. TMPNN can better generalize to unseen structures and is more data efficient. Moreover, it can be applied to both MAP and marginal inference tasks.

B Message Passing with the Generalized Bethe Free Energy

B.1 Derivation for general belief and message equations

We derive the belief and message update equations based on the generalized Bethe free energy. We first convert the constrained optimization into an unconstrained optimization by absorbing constraints via Lagrangian multipliers μ_{ij}, and we obtain

$$\mathcal{L}_{\text{general}} = U(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}) - \mathcal{T}H(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}) + \sum_{(i,j))\in\mathcal{E}} \sum_{x_i} \mu_{ji}(x_i)(\sum_{x_j} q_{ij}(x_i, x_j) - q_i(x_i)) \tag{12}$$

Given the fact that beliefs are local optimal solutions, beliefs are corresponding to zero-gradient points, i.e.,

$$\begin{aligned}
&\frac{\partial L_{\text{general}}}{\partial q_i(x_i)} = -\ln(\theta_i(x_i)) - \mathcal{T}\frac{\partial H}{\partial q_i(x_i)} - \sum_{j\in\mathcal{N}(i)} \mu_{ji}(x_i) = 0 \\
&\frac{\partial L_{\text{general}}}{\partial q_{ij}(x_i,x_j)} = -\ln(\theta_{ij}(x_i,x_j)) - \mathcal{T}\frac{\partial H}{\partial q_{ij}(x_i,x_j)} + \mu_{ji}(x_i) + \mu_{ij}(x_j) = 0
\end{aligned} \tag{13}$$

Given the total entropy approximation $H(\boldsymbol{q}^{node}, \boldsymbol{q}^{edge}; \mathcal{C})$, we first re-organize the entropy in the generalized Bethe free energy as follows:

$$\begin{aligned}
H(q_{ij}, q_i) &= \sum_{i\in\mathcal{V}} c_i H(q_i) + \sum_{(i,j)\in\mathcal{E}} c_{ij} H(q_i, q_j) \\
&= -\sum_{i\in\mathcal{V}} c_i \sum_{x_i} q_i(x_i)\ln q_i(x_i) - \sum_{(i,j)\in\mathcal{E}} c_{ij} \sum_{x_i,x_j} q_{ij}(x_i,x_j)\ln q_{ij}(x_i,x_j) \\
&= -\sum_{i\in\mathcal{V}} (c_i + \sum_{k\in\mathcal{N}(i)} c_{ik}) \sum_{x_i} q_i(x_i)\ln q_i(x_i) - \sum_{(i,j)\in\mathcal{E}} c_{ij} \sum_{x_i,x_j} q_{ij}(x_i,x_j)\ln \frac{q_{ij}(x_i,x_j)}{q_i(x_i)q_j(x_j)}
\end{aligned} \tag{14}$$

Given the entropy, we obtain the derivative $\frac{\partial H}{\partial q_i(x_i)}$ as

$$\begin{aligned}
\frac{\partial H}{\partial q_i(x_i)} &= -(c_i + \sum_{k\in\mathcal{N}(i)} c_{ik})(\ln(q_i(x_i)) + 1) + \sum_{k\in\mathcal{N}(i)} c_{ik} \\
&= -(c_i + \sum_{k\in\mathcal{N}(i)} c_{ik})\ln(q_i(x_i)) - c_i
\end{aligned} \tag{15}$$

Similarly, the derivative $\frac{\partial H}{\partial q_{ij}(x_i,x_j)}$ is calculated as

$$\frac{\partial H}{\partial q_{ij}(x_i,x_j)} = -c_{ij}\ln(q_{ij}(x_i,x_j)) - c_{ij} + c_{ij}\ln(q_i(x_i)) + c_{ji}\ln(q_j(x_j)) \tag{16}$$

Given the entropy derivatives Eq. 15 and Eq. 16, we plug them into Eq. 13 and obtain two equality equations based on zero-gradients as

$$\begin{aligned}
&-\ln(\theta_i(x_i)) + \mathcal{T}\{(c_i + \sum_{k\in\mathcal{N}(i)} c_{ik})\ln(q_i(x_i)) + c_i\} - \sum_{k\in\mathcal{N}(i)} \mu_{ki}(x_i) = 0 \\
&-\ln(\theta_{ij}(x_i,x_j)) + \mathcal{T}\{c_{ij}\ln(q_{ij}(x_i,x_j)) + c_{ij} - c_{ij}\ln(q_i(x_i)) - c_{ji}\ln(q_j(x_j))\} + \mu_{ji}(x_i) + \mu_{ij}(x_j) = 0
\end{aligned} \tag{17}$$

Setting $\mu_{ki}(x_i) = c_{ki}\ln(\hat{\mu}_{ki}(x_i))$ and solving above two equations, we obtain the update equation for unary belief and pairwise belief. Particularly, we first show the derivation to obtain the unary belief q_i:

$$\begin{aligned}
&-\ln(\theta_i(x_i)) + \mathcal{T}\{(c_i + \sum_{k\in\mathcal{N}(i)} c_{ik})\ln(q_i(x_i)) + c_i\} - \sum_{k\in\mathcal{N}(i)} \mu_{ki}(x_i) = 0 \\
&\Rightarrow \mathcal{T}\{\hat{c}_i\ln(q_i(x_i)) + c_i\} = \ln(\theta_i(x_i)) + \sum_{k\in\mathcal{N}(i)} \mu_{ki}(x_i) \\
&\Rightarrow \mathcal{T}\ln(q_i(x_i)) = \frac{1}{\hat{c}_i}\ln(\theta_i(x_i)) + \frac{1}{\hat{c}_i}\sum_{k\in\mathcal{N}(i)} \mu_{ki}(x_i) - \mathcal{T}\frac{c_i}{\hat{c}_i} \\
&\Rightarrow \mathcal{T}\ln(q_i(x_i)) = \frac{1}{\hat{c}_i}\ln(\theta_i(x_i)) + \frac{1}{\hat{c}_i}\sum_{k\in\mathcal{N}(i)} c_{ki}\ln(\hat{\mu}_{ki}(x_i)) - \mathcal{T}\frac{c_i}{\hat{c}_i} \\
&\Rightarrow q_i^{\mathcal{T}}(x_i) = \theta_i(x_i)^{1/\hat{c}_i} \prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)\exp(-\mathcal{T}\frac{c_i}{\hat{c}_i}) \\
&\Rightarrow q_i^{\mathcal{T}}(x_i) \propto \theta_i(x_i)^{1/\hat{c}_i} \prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)
\end{aligned} \tag{18}$$

where $\hat{c}_i = c_i + \sum_{k\in\mathcal{N}(i)} c_{ik}$. The last step is based on the fact that $\exp(-\mathcal{T}\frac{c_i}{\hat{c}_i})$ is constant with respect to different x_i. Similarly, we derive the update equation for pairwise belief q_{ij} as

$$\begin{aligned}
&-\ln(\theta_{ij}(x_i,x_j)) + \mathcal{T}\{c_{ij}\ln(q_{ij}(x_i,x_j)) + c_{ij} - c_{ij}\ln(q_i(x_i)) - c_{ji}\ln(q_j(x_j))\} + \mu_{ji}(x_i) + \mu_{ij}(x_j) = 0\\
&\Rightarrow \mathcal{T}\{\ln(q_{ij}(x_i,x_j)) + 1 - \ln(q_i(x_i)) - \ln(q_j(x_j))\} = \frac{1}{c_{ij}}\ln(\theta_{ij}(x_i,x_j)) - \frac{1}{c_{ij}}\mu_{ji}(x_i) - \frac{1}{c_{ij}}\mu_{ij}(x_j)\\
&\Rightarrow \mathcal{T}\ln(q_{ij}(x_i,x_j)) = \frac{1}{c_{ij}}\ln(\theta_{ij}(x_i,x_j)) - \frac{1}{c_{ij}}\mu_{ji}(x_i) - \frac{1}{c_{ij}}\mu_{ij}(x_j) + \mathcal{T}\ln(q_i(x_i)) + \mathcal{T}\ln(q_j(x_j)) - \mathcal{T}
\end{aligned} \tag{19}$$

where we assume $c_{ij} = c_{ji}$. Plug the equation for $\mathcal{T}\ln(q_i(x_i))$ as derived in Eq. 18 and we have

$$\begin{aligned}
&\mathcal{T}\ln(q_{ij}(x_i,x_j))\\
&= \frac{1}{c_{ij}}\ln(\theta_{ij}(x_i,x_j)) - \frac{1}{c_{ij}}\mu_{ji}(x_i) - \frac{1}{c_{ij}}\mu_{ij}(x_j)\\
&+ \frac{1}{\hat{c}_i}\ln(\theta_i(x_i)) + \frac{1}{\hat{c}_i}\sum_{k\in\mathcal{N}(i)}\mu_{ki}(x_i) - \mathcal{T}\frac{c_i}{\hat{c}_i} + \frac{1}{\hat{c}_j}\ln(\theta_j(x_j)) + \frac{1}{\hat{c}_j}\sum_{k\in\mathcal{N}(j)}\mu_{kj}(x_j) - \mathcal{T}\frac{c_j}{\hat{c}_j} - \mathcal{T}\\
&= \frac{1}{c_{ij}}\ln(\theta_{ij}(x_i,x_j)) - \ln(\hat{\mu}_{ji}(x_i)) - \ln(\hat{\mu}_{ij}(x_j))\\
&+ \frac{1}{\hat{c}_i}\ln(\theta_i(x_i)) + \frac{1}{\hat{c}_i}\sum_{k\in\mathcal{N}(i)}c_{ki}\ln(\hat{\mu}_{ki}(x_i)) + \frac{1}{\hat{c}_j}\ln(\theta_j(x_j)) + \frac{1}{\hat{c}_j}\sum_{k\in\mathcal{N}(j)}c_{kj}\ln(\hat{\mu}_{kj}(x_j))\\
&- \mathcal{T}\frac{c_i}{\hat{c}_i} - \mathcal{T}\frac{c_j}{\hat{c}_j} - \mathcal{T}
\end{aligned} \tag{20}$$

where $\hat{c}_i = c_i + \sum_{k\in\mathcal{N}(i)} c_{ik}$ and $\hat{c}_j = c_j + \sum_{k\in\mathcal{N}(j)} c_{jk}$. Since $-\mathcal{T}\frac{c_i}{\hat{c}_i} - \mathcal{T}\frac{c_j}{\hat{c}_j} - \mathcal{T}$ is a constant with respect to different x_i, we in the end have

$$q_{ij}^{\mathcal{T}}(x_i,x_j) \propto \theta_i(x_i)^{1/\hat{c}_i}\theta_j(x_j)^{1/\tilde{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)}{\hat{\mu}_{ji}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}} \tag{21}$$

In summary, we obtain the update equations for unary belief and pairwise belief as

$$\begin{aligned}
q_i^{\mathcal{T}}(x_i) &\propto \theta_i(x_i)^{1/\hat{c}_i}\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)\\
q_{ij}^{\mathcal{T}}(x_i,x_j) &\propto \theta_i(x_i)^{1/\hat{c}_i}\theta_j(x_j)^{1/\tilde{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)}{\hat{\mu}_{ji}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}
\end{aligned} \tag{22}$$

With the derived belief update equations, we can then derive message update equation. Given the local consistency constraint (i.e.,$q_i(x_i) = \sum_{x_j} q_{ij}(x_i,x_j)$), we first have

$$q_i^{\mathcal{T}}(x_i) = \{\sum_{x_j} q_{ij}(x_i,x_j)\}^{\mathcal{T}} \tag{23}$$

Plug the derived belief update equations in Eq. 22 into the above equation, we have

$$\begin{aligned}
&\theta_i(x_i)^{1/\hat{c}_i}\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)\\
&\propto \{\sum_{x_j}[\theta_i(x_i)^{1/\hat{c}_i}\theta_j(x_j)^{1/\tilde{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)}{\hat{\mu}_{ji}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}]^{1/\mathcal{T}}\}^{\mathcal{T}}\\
&\propto \{[\theta_i(x_i)^{1/\hat{c}_i}\frac{\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)}{\hat{\mu}_{ji}}]^{1/\mathcal{T}}\sum_{x_j}[\theta_j(x_j)^{1/\tilde{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}]^{1/\mathcal{T}}\}^{\mathcal{T}}\\
&\propto \theta_i(x_i)^{1/\hat{c}_i}\frac{\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)}{\hat{\mu}_{ji}}\{\sum_{x_j}[\theta_j(x_j)^{1/\tilde{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}]^{1/\mathcal{T}}\}^{\mathcal{T}}
\end{aligned} \tag{24}$$

Cancel the common terms $\theta_i(x_i)^{1/\hat{c}_i}\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)$ on both sides of the equation and through a re-organization, we obtain the message update equation as

$$\hat{\mu}_{ji}(x_i)\propto\{\sum_{x_j}[\theta_j(x_j)^{1/\hat{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}]^{1/\mathcal{T}}\}^{\mathcal{T}} \tag{25}$$

In the end, we obtain the belief and message update equations of general form given the general free energy.

B.2 Probabilistic inference with derived general equations

B.2.1 Marginal inference

To perform marginal inference, we set $\mathcal{T}=1$. The message update equation now becomes

$$\hat{\mu}_{ji}(x_i)=\sum_{x_j}\theta_j(x_j)^{1/\hat{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}} \tag{26}$$

And we obtain the belief update equations as

$$\begin{aligned}q_i(x_i)&\propto\theta_i(x_i)^{1/\hat{c}_i}\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)\\ q_{ij}(x_i,x_j)&\propto\theta_i(x_i)^{1/\hat{c}_i}\theta_j(x_j)^{1/\hat{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)}{\hat{\mu}_{ji}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}\end{aligned} \tag{27}$$

The estimated beliefs correspond to marginal probabilities.

B.2.2 MAP inference

To perform MAP inference, we consider the zero temperature limit and the message function becomes

$$\begin{aligned}\hat{\mu}_{ji}(x_i)&=\lim_{\mathcal{T}\to0}\{\sum_{x_j}[\theta_j(x_j)^{1/\hat{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}]^{1/\mathcal{T}}\}^{\mathcal{T}}\\ &=|\theta_j(x_j)^{1/\hat{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}|_\infty\\ &=\max_{x_j}\theta_j(x_j)^{1/\hat{c}_j}\theta_{ij}(x_i,x_j)^{1/c_{ij}}\frac{\prod_{k\in\mathcal{N}(j)}\hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}\end{aligned} \tag{28}$$

where $|\cdot|_\infty$ indicates the infinity norm. To show the belief equation, we first introduce a zero temperature lemma (Weiss et al., 2012):

Zero Temperature Lemma: *Suppose $\boldsymbol{q}^{node}=\{q_i\}_{i\in\mathcal{V}}$ and $\boldsymbol{q}^{edge}=\{q_{ij}\}_{(i,j)\in\mathcal{E}}$ are fixed-points of the sum-product algorithm at temperature T, Define $\hat{\boldsymbol{q}}^{node}=\{\hat{q}_i\}_{i\in\mathcal{V}}$ and $\hat{\boldsymbol{q}}^{edge}=\{\hat{q}_{ij}\}_{(i,j)\in\mathcal{E}}$ as $\hat{q}_i=q_i^T$ and $\hat{q}_{ij}=q_{ij}^T$. Then, for any $T\to0$, $\hat{\boldsymbol{q}}^{node}$ and $\hat{\boldsymbol{q}}^{edge}$ approach to the fixed-points of the max-product inference.*

Based on the zero temperature lemma, we thus have the estimated max-marginals for MAP inference is updated as

$$\begin{aligned}q(x_i)&\propto\lim_{\mathcal{T}\to0}\{[\theta_i(x_i)^{1/\hat{c}_i}\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)]^{1/\mathcal{T}}\}^{\mathcal{T}}\\ &\propto\lim_{\mathcal{T}\to0}\theta_i(x_i)^{1/\hat{c}_i}\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)\\ &\propto\theta_i(x_i)^{1/\hat{c}_i}\prod_{k\in\mathcal{N}(i)}\hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)\end{aligned} \tag{29}$$

The belief update equations (Eq. 22) remains the same and the estimated beliefs correspond to max-marginals.

B.3 Generalization over existing algorithms

For BP, we have $\hat{c}_i = c_i + \sum_{k\in\mathcal{N}(i)} c_{ik} = 1$ and $c_{ij} = 1$. Correspondingly, our belief and message update equations of general form is reduced to

$$\begin{aligned} q_i(x_i) &\propto \theta_i(x_i) \prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}(x_i) \\ q_{ij}(x_i, x_j) &\propto \theta_i(x_i)\theta_j(x_j)\theta_{ij}(x_i, x_j) \frac{\prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}(x_i)}{\hat{\mu}_{ji}} \frac{\prod_{k\in\mathcal{N}(j)} \hat{\mu}_{kj}(x_j)}{\hat{\mu}_{ij}} \\ \hat{\mu}_{ji}(x_i) &= \max_{x_j} \theta_j(x_j)\theta_{ij}(x_i, x_j) \frac{\prod_{k\in\mathcal{N}(j)} \hat{\mu}_{kj}(x_j)}{\hat{\mu}_{ij}} \end{aligned} \tag{30}$$

which are identical to the equations we have for BP algorithm. For TRW-BP, we have $\hat{c}_i = c_i + \sum_{k\in\mathcal{N}(i)} c_{ik} = 1$ and $c_{ij} = \rho_{ij}$. Correspondingly, our belief and message update equations of general form is reduced to

$$\begin{aligned} q_i(x_i) &\propto \theta_i(x_i) \prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}^{\rho_{ki}}(x_i) \\ q_{ij}(x_i, x_j) &\propto \theta_i(x_i)\theta_j(x_j)\theta_{ij}^{1/\rho_{ij}}(x_i, x_j) \frac{\prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}^{\rho_{ki}}(x_i)}{\hat{\mu}_{ji}} \frac{\prod_{k\in\mathcal{N}(j)} \hat{\mu}_{kj}^{\rho_{kj}}(x_j)}{\hat{\mu}_{ij}} \\ \hat{\mu}_{ji}(x_i) &= \max_{x_j} \theta_j(x_j)\theta_{ij}^{1/\rho_{ij}}(x_i, x_j) \frac{\prod_{k\in\mathcal{N}(j)} \hat{\mu}_{kj}^{\rho_{kj}}(x_j)}{\hat{\mu}_{ij}} \end{aligned} \tag{31}$$

which are identical to the equations we have for TRW-MP algorithm. The generalization can be straightforwardly applied to marginal inference by replacing maximization operation with summation operation.

C Architecture of TMPNN for Message Passing

Given the estimated entropy coefficients, TMPNN performs message passing update under an MPNN framework. Hidden feature $\boldsymbol{h}_i$ is equivalent to unary belief up to a scale factor z_i in logarithmic space. To derive message function and node update function, we first re-write unary belief equation in logarithmic space as

$$\ln q_i^t(x_i) = -z_i^t + \frac{1}{\hat{c}_i} \ln \theta_i(x_i) + \sum_{k\in\mathcal{N}(i)} \frac{c_{ki}}{\hat{c}_i} \ln \hat{\mu}_{ki}^t(x_i) \tag{32}$$

where $z_i^t = \ln \sum_{x_i} \theta_i(x_i)^{1/\hat{c}_i} \prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i)$. We denote probability parameters as $\phi_i = \ln \theta_i$ and messages in MPNN as $m_{k\to i} = \ln \hat{\mu}_{ki}$. The aggregated message is then denoted as

$$m_i^t(x_i) = \sum_{k\in\mathcal{N}(i)} \frac{c_{ki}}{\hat{c}_i} \ln \hat{\mu}_{ki}^t(x_i) = \frac{1}{\hat{c}_i} \sum_{k\in\mathcal{N}(i)} c_{ki} \ln m_{k\to i}^t(x_i) \tag{33}$$

With the mapping $\mathbf{h}_i = \ln q_i$, we in the end have the node update function $\mathcal{U}$ as

$$\mathbf{h}_i^t(x_i) = -z_i^t + \frac{1}{\hat{c}_i}\phi_i(x_i) + m_i^t(x_i) \tag{34}$$

with

$$z_i^t = \ln \sum_{x_i} \theta_i(x_i)^{1/\hat{c}_i} \prod_{k\in\mathcal{N}(i)} \hat{\mu}_{ki}^{c_{ki}/\hat{c}_i}(x_i) = \ln \sum_{x_i} \exp(\frac{1}{\hat{c}_i}\phi_i(x_i) + m_i^t(x_i)) \tag{35}$$

To derive the message function $\mathcal{M}$, we similarly re-write the message equation in logarithmic space as

$$\begin{aligned} \ln \hat{\mu}_{ji}(x_i) &= \ln\{\max_{x_j} \theta_j(x_j)^{1/\hat{c}_j} \theta_{ij}(x_i, x_j)^{1/c_{ij}} \frac{\prod_{k\in\mathcal{N}(j)} \hat{\mu}_{kj}^{c_{kj}/\hat{c}_j}(x_j)}{\hat{\mu}_{ij}}\} \\ &= \ln\{\max_{x_j} \exp(\frac{1}{\hat{c}_j} \ln \theta_j + \frac{1}{c_{ij}} \ln \theta_{ij} + \sum_{k\in\mathcal{N}(j)} \frac{c_{kj}}{\hat{c}_j} \ln \hat{\mu}_{kj}(x_j) - \ln \hat{\mu}_{ij}(x_i))\} \end{aligned} \tag{36}$$

With the introduced notations plugged in, we then have

$$m_{j\to i}^t(x_i) = \ln\{\max_{x_j} \exp(\frac{1}{\hat{c}_j}\phi_j + \frac{1}{c_{ij}}\phi_{ij} + m_j^t(x_j) - m_{i\to j}^t(x_i)) \tag{37}$$

where $\phi_{ij} = \ln \theta_{ij}$. Given $\frac{1}{\hat{c}_j}\phi_j + m_j^t(x_j) = \mathbf{h}_j^t(x_j) + z_j^t$, we obtain the message function $\mathcal{M}$

$$m_{j\to i}^t(x_i) = \ln\{\max_{x_j} \exp(\frac{1}{c_{ij}}\phi_{ij} + \mathbf{h}_j^t(x_j) + z_j^t - m_{i\to j}^t)\} \tag{38}$$

Given the mapping $\mathbf{h}_i = \ln q_i$, we obtain the readout function $\mathcal{R}^{node}$ for unary belief, i.e.,

$$q_i = \exp(\mathbf{h}_i^T) \tag{39}$$

For pairwise belief, we first express q_{ij} as a function of q_i as

$$q_{ij}(x_i, x)j) = \theta_{ij}(x_i, x_j)^{1/c_{ij}} \frac{q_i(x_i)}{\hat{\mu}_{ji}} \frac{q_j(x_j)}{\hat{\mu}_{ij}} \tag{40}$$

We then re-write the equation in logarithmic space as

$$\begin{aligned} \ln q_{ij} &= \frac{1}{c_{ij}} \ln \theta_{ij}(x_i, x_j) + \ln q_i + \ln q_j - \ln \hat{\mu}_{ji} - \ln \hat{\mu}_{ij} \\ &= \frac{1}{c_{ij}} \phi_{ij}(x_i, x_j) + \mathbf{h}_i + \mathbf{h}_j - m_{j\to i} - m_{i\to j} \end{aligned} \tag{41}$$

Thus, we obtain the readout function $\mathcal{R}^{edge}$ for pairwise belief as

$$q_{ij} = \exp(\frac{1}{c_{ij}}\phi_{ij}(x_i, x_j) + \mathbf{h}_i^T + \mathbf{h}_j^T - m_{j\to i}^T - m_{i\to j}^T) \tag{42}$$

D Propositions and Proofs

Proposition 1. *Variational distribution $\boldsymbol{q}^*$ from TMPNN is locally optimal with pairwise consistency constraint strictly satisfied. If the set of entropy coefficients satisfy $\mathcal{C} \in \mathbb{C}^{concave}$, $\boldsymbol{q}^*$ becomes globally optimal.*
Proof: We first show the satisfaction of the pairwise consistency constraint. We plug theory-guided message function into belief functions and we can have

$$q_i(x_i) = \sum_{x_j} q_{ij}(x_i, x_j), \quad \forall (i,j) \in \mathcal{E} \tag{43}$$

Hence, pairwise consistency constraint is always satisfied. We further take gradient of $\mathcal{F}_{\text{general}}$ w.r.t. $\boldsymbol{q}^*$ and plug the belief functions into the gradient, we can have

$$\begin{aligned} &\frac{\partial \mathcal{F}_{\text{general}}}{\partial q_i^*} = 0, \ \forall i \in \mathcal{V}; \quad \frac{\partial \mathcal{F}_{\text{general}}}{\partial q_{ij}^*} = 0, \quad \forall (i,j) \in \mathcal{E} \\ &\text{subject to} \quad q_i(x_i) = \sum_{x_j} q_{ij}(x_i, x_j) \end{aligned} \tag{44}$$

Based on (Hazan and Shashua, 2010; Meshi et al., 2012; Cui et al., 2022), if these exists a set of entropy coefficients $\mathcal{C} \in \mathbb{C}^{concave}$, the generalized Bethe free energy is provably convex. Since the variational distribution from TMPNN is always stationary point of $\mathcal{F}_{\text{general}}$, given the convexity, we can show that it is always globally optimal with the consistency constraint strictly satisfied.

D.1 Proposition 2

Given a target probability distribution p, MAP inference error is defined as

$$\Delta_{map}(\boldsymbol{q}_{\mathcal{C}}^*, p) = \sum_{i\in\mathcal{V}} \sum_{x_i} (p_i(x_i) - q_{\mathcal{C},i}^*(x_i)) \ln \theta_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \sum_{x_i, x_j} (p_{ij}(x_i, x_j) - q_{\mathcal{C},ij}^*(x_i, x_j)) \ln \theta_{ij}(x_i, x_j) \tag{45}$$

where $p_i(x_i) = \sum_{\boldsymbol{x}\setminus x_i} p(\boldsymbol{x})$ *and* $p_{ij}(x_i, x_j) = \sum_{\boldsymbol{x}\setminus(x_i \cup x_j)} p(\boldsymbol{x})$. *With an optimal set of linear coefficients* $\mathcal{C}^* \in \mathbb{C}^{concave}$, *the MAP inference error is upper bounded by a total entropy approximation scaled by* ϵ, *i.e.,*

$$\Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^*}, p) \leq \epsilon H(\boldsymbol{q}^*_{\mathcal{C}^*}; \mathcal{C}^*) \tag{46}$$

with $\mathcal{C}^* = \arg\min_{\mathcal{C}\in\mathbb{C}^{concave}} H(\boldsymbol{q}^*_{\mathcal{C}}; \mathcal{C})$.

Proof: Given the outputs from TMPNN $\boldsymbol{q}_{\mathcal{C}}$ with $\mathcal{C} \in \mathbb{C}^{concave}$, we have

$$\mathcal{F}_{\text{general}}(\boldsymbol{q}^{node,*}_{\mathcal{C}}, \boldsymbol{q}^{edge,*}_{\mathcal{C}}; \mathcal{C}) \leq \mathcal{F}_{\text{general}}(\boldsymbol{p}^{node}, \boldsymbol{p}^{edge}; \mathcal{C}) \tag{47}$$

With the definition of $\mathcal{F}_{\text{general}}$ and the fact that $H(\boldsymbol{p}^{node}, \boldsymbol{p}^{edge}; \mathcal{C}) \geq 0$, we have

$$\Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}}, p) \leq \epsilon H(\boldsymbol{q}^*_{\mathcal{C}}; \mathcal{C}) \tag{48}$$

An optimal set of entropy coefficients $\mathcal{C}^* = \arg\min_{\mathcal{C}\in\mathbb{C}^{concave}} H(\boldsymbol{q}^*_{\mathcal{C}}; \mathcal{C})$ thus gives a minimal MAP inference error, which is bounded by an total entropy approximate, i.e.,

$$\Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^*}, p) \leq \epsilon H(\boldsymbol{q}^*_{\mathcal{C}^*}; \mathcal{C}^*) \tag{49}$$

D.2 Proposition 3

TMPNN subsumes existing variational BP algorithms (e.g., BP and TRW-MP) as a generalization. The optimal MAP inference performance achieved with TMPNN is superior or comparable to existing variational BP algorithms, i.e., $\Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^*}, p) \leq \Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^{fix}}, p)$

Proof: Given different settings of $\mathcal{C}$, we can derive different existing variational BP algorithms. For BP, we have $\hat{c_i} = c_i + \sum_{k\in\mathcal{N}(i)} c_{ik} = 1$ and $c_{ij} = 1$. For TRW-BP, we have $\hat{c_i} = c_i + \sum_{k\in\mathcal{N}(i)} c_{ik} = 1$ and $c_{ij} = \rho_{ij}$. Furthermore, since $\boldsymbol{q}^*_{\mathcal{C}^*} = \arg\min_{\boldsymbol{q}} \mathcal{F}_{\text{general}}(\boldsymbol{q}; \mathcal{C}^*)$, we have

$$U(\boldsymbol{q}^*_{\mathcal{C}^*}) - \epsilon H(\boldsymbol{q}^*_{\mathcal{C}^*}; \mathcal{C}^*) \leq U(\boldsymbol{q}^*_{\mathcal{C}^{fix}}) - \epsilon H(\boldsymbol{q}^*_{\mathcal{C}^{fix}}; \mathcal{C}^*) \tag{50}$$

with $\boldsymbol{q}^*_{\mathcal{C}^{fix}}$ denotes the optimal variational distribution minimizing the generalized Bethe free energy specified with a set of fixed coefficients $\mathcal{C}^{fix}$. By subtracting $U(\boldsymbol{p}^{node}, \boldsymbol{p}^{edge})$ on both sides of Eq. 50, we have

$$\Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^*}, p) \leq \Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^{fix}}, p) + \epsilon\Delta \tag{51}$$

with $\Delta = H(\boldsymbol{q}^*_{\mathcal{C}^*}; \mathcal{C}^*) - H(\boldsymbol{q}^*_{\mathcal{C}^{fix}}; \mathcal{C}^*) \leq H(\boldsymbol{q}^*_{\mathcal{C}^*}; \mathcal{C}^*)$. We can further show that $H(\boldsymbol{q}^*_{\mathcal{C}^*}; \mathcal{C}^*) \leq \delta$ where δ is a constant and finite value. For clear derivation, we use notation $\boldsymbol{q}$ and $\mathcal{C}$ and derive for the bounds of $H(\boldsymbol{q}; \mathcal{C})$. The following derivation applies to arbitrary $\boldsymbol{q}$ and $\mathcal{C}$. To derive δ, we firstly re-organize $H(\boldsymbol{q}; \mathcal{C})$ as

$$H(\boldsymbol{q}; \mathcal{C}) = \sum_{i\in\mathcal{V}} c_i H(q_i) + \sum_{(i,j)\in\mathcal{E}} c_{ij} H(q_i, q_j) \tag{52}$$

We now show that both $H(q_i)$ and $H(q_i, q_j)$ are bounded by a constant value. For $H(q_i)$, applying the Jensen's inequality yields,

$$H(q_i) = -\sum_{x_i} q_i(x_i) \ln q_i(x_i) = \sum_{x_i} q_i(x_i) \ln \frac{1}{q_i(x_i)} \leq \ln \sum_{x_i} \frac{q_i(x_i)}{q_i(x_i)} = \ln k_i \tag{53}$$

where k_i indicates the number of states of variable x_i. Similarly, we have $H(q_i, q_j) \leq \ln k_{ij}$ where k_{ij} indicates the number of joint configurations of variables x_i and x_j. Given the bounds for $H(q_i)$ and $H(q_i, q_j)$, we can conclude

$$H(\boldsymbol{q}; \mathcal{C}) \leq \delta = \sum_{i\in\mathcal{V}} |c_i| \ln k_i + \sum_{(i,j)\in\mathcal{E}} |c_{ij}| \ln k_{ij} \tag{54}$$

δ is only a function of underlying graph and coefficients $\mathcal{C}$. Thus, we have $\Delta \leq H(\boldsymbol{q}^*_{\mathcal{C}^*}; \mathcal{C}^*) \leq \delta$. With $\Delta \leq \delta$, we can show

$$\Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^*}, p) \leq \Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^{fix}}, p) + \epsilon\delta \tag{55}$$

where δ is not a function of ϵ. Furthermore, given the mild assumption that entropy coefficients $\mathcal{C}$ are of finite values, δ is always finite. With a constant and finite upper bound δ, there always exists a sufficiently small ϵ such that $\Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^*}, p) \leq \Delta_{map}(\boldsymbol{q}^*_{\mathcal{C}^{fix}}, p)$.

E Additional Evaluations

E.1 Tightness of Error Bounds after Training

On loop free graphs (S1-3), TMPNN achieves exact inference results, i.e., $\Delta_{map}(\boldsymbol{q}_{\mathcal{C}}^*, p) = 0$, and the upper bound ($\epsilon H(\boldsymbol{q}_{\mathcal{C}}^*)$) is tight (0.000521 on average). For loopy graphs (S4-13), the respective gaps shown in Table 8. As shown, the gap is positively correlated to the inference error and increases as the graph complexity increases.

Table 8: Tightness of Error Bounds (N=9)

Graph	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
gap	0.0564	0.1557	0.8103	0.1436	1.734	0.2637	2.22	3.178	5.528	8.745

E.2 Effectiveness of the Training Objective

For better training convergence, TMPNN is pre-trained on loop-free graphs (Star, Tree, Path) using exact inference results as supervision. We simulate 3000 graphs for pre-training with 1000 graphs per type. Exact inference results are obtained from BP. The pre-trained model is employed for evaluation on both loop-free graphs and loopy graphs. To better demonstrate the effectiveness of the proposed training objective based on the generalized Bethe free energy, we perform an ablation study. Particularly, we compare the MAP inference performance without and with fine tuning through free energy objective. We consider 13 classic graphs with 9 nodes. Results are shown in Table 9.

Table 9: Effectiveness of pre-training (MAP inference accuracy with unit %)

Graph	S1♣	S2♣	S3♣	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	**Avg.**
w/o	94.67	93.73	92.44	87.11	76.22	77.00	87.89	73.56	85.33	71.44	72.56	66.44	**63.00**	80.11
w	**100.00**	**100.00**	**100.00**	**89.67**	**77.89**	**78.78**	**89.78**	**74.56**	**88.11**	**72.22**	**73.00**	**67.00**	**63.00**	**82.62**

As shown, fine-tuning through the proposed training objective improves TMPNN's average performance compared to TMPNN with pre-training only.

E.3 Effectiveness of the Message Initialization

For training-free methods, messages are initialized as ones, following standard settings. We further consider different initial messages randomly sampled from uniform distribution. Results over 10 different runs are shown in Tab. 10. As shown, the variance due to different message initialization is small.

Table 10: Multiple initializations for training-free methods (MAP inference on graphs with N=9)

Graph	Cycle	Ladder	Grid	C-ladder	Barbell	Lollipop	Wheel	Bipartite	Tripartite	Complete
BP	.85±.03	.57±.04	.61±.01	.82±.01	.54±.02	.80±.01	.54±.04	.55±.07	.54±.04	.49±.03
TRW-MP	.90±.01	.58±.02	.54±.02	.84±.01	.52±.01	.83±.01	.49±.02	.83±.01	.50±.01	.49±.02
MPLP	.82±.02	.64±.02	.64±.02	.83±.01	.53±.04	.78±.01	.52±.06	.51±.04	.52±.02	.51±.01

E.4 Comparison to Sampling-based method

Sampling-based methods adopt a fundamentally different approach from variational BP algorithms. Sampling-based methods approximate target distributions via sampling, and the inference performance highly depends on the sampling strategy. We provide additional comparison to sampling-based methods. Particularly, we compare the Gibbs sampling method on graphs with 9/16 nodes for MAP inference. We use default hyper-parameters for experiments: number of samples to be collected is 5000, the burn-in is 1000 and the stride size is 2. Results are shown in Table 11.

Table 11: Comparison to sampling-based methods for MAP inference (Accuracy %; loop-free♣)

Graph	N=9		N=16	
	Gibbs	TMPNN	Gibbs	TMPNN
STAR♣	78.22	**100.00**	64.75	**100.00**
TREE♣	81.18	**100.00**	68.81	**100.00**
PATH♣	82.33	**100.00**	69.81	**100.00**
CYCLE	80.33	**89.67**	65.81	**91.75**
LADDER	**83.78**	77.89	**70.63**	69.56
GRID	**81.22**	78.78	**73.75**	68.13
C-LADDER	83.78	**89.78**	**75.00**	73.44
BARBELL	**80.67**	74.56	**73.13**	70.38
LOLLIPOP	80.33	**88.11**	**69.00**	68.25
WHEEL	**81.22**	72.22	**71.75**	68.63
BIPARTITE	**84.89**	73.00	**69.00**	64.06
TRIPARTITE	**81.00**	67.00	**71.06**	62.75
COMPLETE	**78.67**	63.00	57.63	**61.63**
Average	81.36	**82.62**	69.24	**76.81**

Comparing TMPNN to Gibbs sampling, we can see that TMPNN outperforms Gibbs sampling on average for both N=9 and N=16. Besides, without requiring sampling, TMPNN is less time consuming than Gibbs sampling method.

E.5 Visualization of the comparison to SOTA MAP Inference Methods

Training-free methods. We visualize the comparison to training-free methods on graphs with N=9 and N=16 in Figure 3. This visualization is corresponding to the Table 1 in the main body of the paper.

Figure 3: Visualization of the comparison to SOTA training-free methods

Training-based methods. We visualize the comparison to training-based methods on graphs with N=9 and

N=16 in Figure 4. This visualization is corresponding to the Table 2 in the main body of the paper.

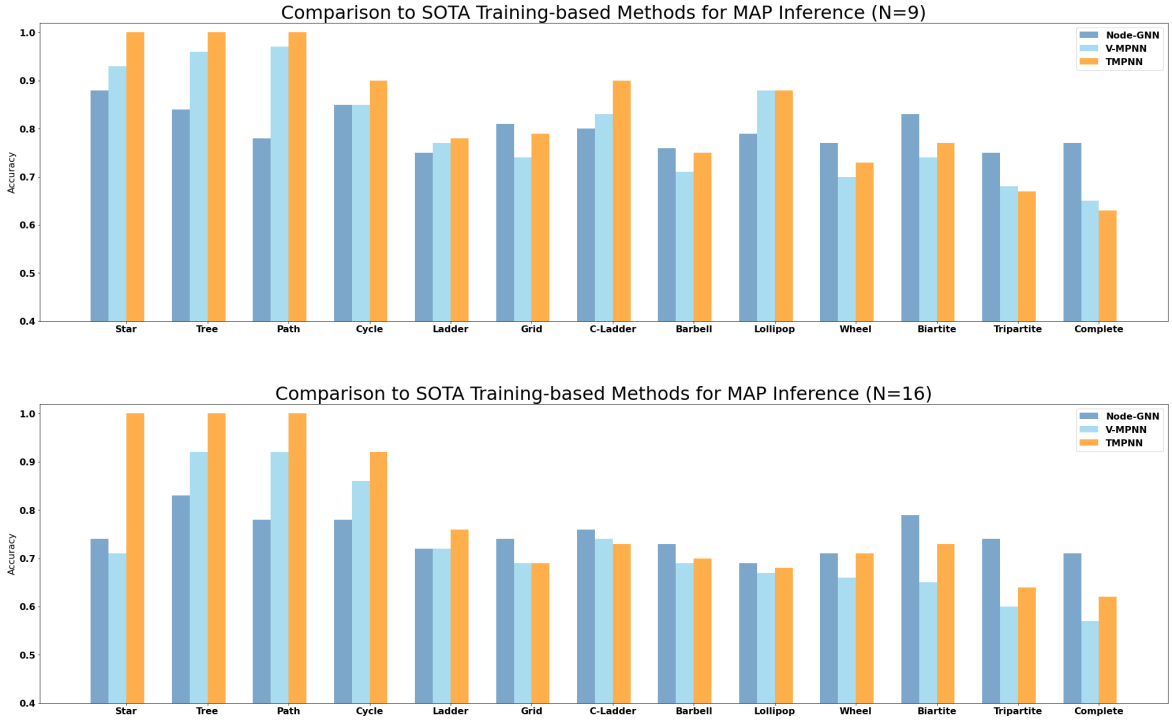

Figure 4: Visualization of the comparison to SOTA training-based methods

E.6 Comparison to SOTA Marginal Inference Methods

Training-free methods. We here show the detailed results on different structure. We compare to two training-free methods: BP and TRW-BP. As shown in Table 12, TMPNN achieves better average performance for both sizes and it performs significantly better than SOTA training-free methods in loopy graphs. For example, in COMPLETE(S13) graph, TMPNN outperforms BP by about two orders of magnitude (the measure is in logarithmic scale) on graphs of size 9 and 16. In loop-free graphs, TMPNN is nearly exact (inference error $\sim 1e-9$), and the tiny inference errors are caused by roundoff errors.

Table 12: Comparison to SOTA Training-free Methods for Marginal Inference ($-\log_{10}\mathrm{D_{KL}}$; loop-free♣)

Graph	N=9			N=16		
	BP	TRW-BP	TMPNN	BP	TRW-BP	TMPNN
S1♣	**10.00**	**10.00**	8.96	**10.00**	**10.00**	8.83
S2♣	**10.00**	**10.00**	9.81	**10.00**	**10.00**	8.81
S3♣	**10.00**	**10.00**	9.13	**10.00**	**10.00**	8.82
S4	7.06	6.07	**9.18**	7.12	6.59	**8.98**
S5	6.19	4.97	**6.26**	**5.82**	4.66	5.46
S6	5.92	4.64	**5.91**	**5.70**	4.45	4.90
S7	7.12	6.08	**8.80**	**5.67**	4.34	5.64
S8	4.84	4.35	**4.87**	**4.27**	3.85	4.20
S9	6.68	6.06	**7.25**	2.34	2.26	**3.94**
S10	4.70	4.04	**4.85**	4.29	1.57	**4.40**
S11	3.96	3.63	**4.27**	1.53	1.44	**3.53**
S12	3.34	3.52	**3.90**	0.81	3.27	**3.39**
S13	1.22	3.26	**3.39**	0.60	3.15	**3.21**
Average	6.23	5.89	**6.66**	5.24	5.04	**5.70**

Training-based methods. We here show the detailed results compared to two training-based methods: node-GNN and GNN-Mimic-BP. For GNN-Mimic-BP, without access to source codes, we only compare to its reported performance (marked with *) in loop-free graphs. As shown in Table 13, TMPNN again achieves better average performance, particularly in loop-free graphs. For example, TMPNN outperforms node-GNN and GNN-Mimic-BP by about four orders of magnitude in STAR and TREE with 9 nodes, without requiring exact inference results for training.

Table 13: Comparison to SOTA Training-based Methods for Marginal Inference ($-\log_{10}\mathrm{D_{KL}}$; loop-free♣)

Graph	N=9			N=16		
	Node-GNN	GNN-Mimic-BP	TMPNN	Node-GNN	GNN-Mimic-BP	TMPNN
STAR (S1)♣	6.40±0.22	5.74*	**8.96**±0.00	5.71±0.36	3.08*	**8.83**±0.00
TREE (S2)♣	5.96±0.14	5.58*	**9.81**±0.00	5.94±0.13	5.56*	**8.81**±0.00
PATH (S3)♣	6.05±0.19	6.14*	**9.13**±0.00	5.68±0.11	6.08*	**8.82**±0.00
CYCLE (S4)	6.31±0.21	-	**9.18**±0.00	6.22±0.27	-	**8.98**±0.00
LADDER (S5)	5.59±0.13	-	**6.26**±0.00	5.45±0.12	-	**5.46**±0.19
2D GRID (S6)	5.60±0.21	-	**5.91**±0.27	**5.53**±0.08	-	4.90±0.01
CIRCULAR LADDER (S7)	6.00±0.32	-	**8.80**±0.00	5.52±0.11	-	**5.64**±0.07
BARBELL (S8)	**5.12**±0.07	-	4.87±0.05	**4.98**±0.05	-	4.20±0.01
LOLLIPOP (S9)	5.73±0.15	-	**7.25**±0.00	**4.48**±0.14	-	3.94±0.00
WHEEL (S10)	**5.06**±0.05	-	4.85±0.04	**4.89**±0.07	-	4.40±0.17
BIPARTITE (S11)	**4.90**±0.04	-	4.27±0.00	**4.28**±0.10	-	3.53±0.01
TRIPARTITE (S12)	**4.62**±0.03	-	3.90±0.00	**4.04**±0.03	-	3.39±0.00
COMPLETE (S13)	**4.33**±0.02	-	3.39±0.00	**3.80**±0.06	-	3.21±0.00
Average	5.51±0.14	5.82*	**6.66**±0.03	5.12±0.13	4.91*	**5.70**±0.04

E.7 Computing Time Evaluation

We report the average convergence time on graphs of different types and different sizes. Testing is performed on a laptop with a 8-Core Intel Core i9 processor with CPU only. Convergence criteria is the same as stated in the paper. Results are shown in Table 14 and Table 15.

Table 14: Convergence time evaluation (N=9) (Second s; loop-free♣)

Graph	N=9					
	BP	TRW-MP	MPLP	Node-GNN	V-MPNN	TMPNN
STAR♣	0.0069	0.0022	0.0064	0.0076	0.0107	0.0042
TREE♣	0.0083	0.0027	0.0041	0.0078	0.0092	0.0089
PATH♣	0.0134	0.0050	0.0044	0.0078	0.0105	0.0135
CYCLE	0.0317	0.0198	0.0048	0.0082	0.0338	0.0143
LADDER	0.1212	0.1014	0.0056	0.0080	0.0135	0.0193
GRID	0.1249	0.0936	0.0058	0.0081	0.0109	0.0194
CIRCULAR LADDER	0.0415	0.0234	0.0055	0.0082	0.0124	0.0158
BARBELL	0.1608	0.1170	0.0059	0.0080	0.0236	0.0222
LOLLIPOP	0.0277	0.0309	0.0051	0.0077	0.0110	0.0184
WHEEL	0.1591	0.1946	0.0068	0.0097	0.0086	0.0184
BIPARTITE	0.1664	0.0573	0.0064	0.0097	0.0081	0.0216
TRIPARTITE	0.2213	0.0626	0.0075	0.0100	0.0077	0.0359
COMPLETE	0.2660	0.0812	0.0090	0.0104	0.0173	0.0481
Average	0.1038	0.0609	0.0059	0.0086	0.0136	0.0186

As shown, TMPNN is less consuming compared to BP and TRW-MP, particularly on large and dense graphs. For example, on graphs of 16 nodes, the average computing time that TMPNN takes is 0.0288s, significantly smaller than the time that TRW-MP takes (0.2649s). TMPNN is of comparable computational complexity compared to MPLP. The complexity of Node-GNN depends only on neural network architecture (i.e., the number of MLP layers) and doesn't change w.r.t. graphs' density or size. Leveraging neural networks, node-GNN achieves overall better efficiency. Similarly, V-MPNN leverages neural networks for message modeling and hence is also competitive in terms of efficiency.

Table 15: Convergence time evaluation (N=16) (Second s; loop-free♣)

Graph	N=16					
	BP	TRW-MP	MPLP	Node-GNN	V-MPNN	TMPNN
STAR♣	0.0112	0.0030	0.0190	0.0080	0.0129	0.0067
TREE♣	0.0242	0.0107	0.0076	0.0081	0.0136	0.0143
PATH♣	0.0328	0.0137	0.0077	0.0084	0.0150	0.0162
CYCLE	0.0503	0.0395	0.0082	0.0119	0.0263	0.0171
LADDER	0.2079	0.2563	0.0087	0.0110	0.0175	0.0367
GRID	0.2157	0.2802	0.0098	0.0103	0.0113	0.0219
CIRCULAR LADDER	0.2080	0.0889	0.0089	0.0100	0.0134	0.0165
BARBELL	0.3245	0.3961	0.0151	0.0106	0.0098	0.0122
LOLLIPOP	0.2925	0.5565	0.0258	0.0110	0.0126	0.0111
WHEEL	0.2375	0.3043	0.0116	0.0105	0.0096	0.0363
BIPARTITE	0.2833	0.6390	0.0147	0.0108	0.0088	0.0360
TRIPARTITE	0.5167	0.3921	0.0217	0.0118	0.0116	0.0899
COMPLETE	0.7323	0.4629	0.0341	0.0118	0.0176	0.0588
Average	0.2413	0.2649	0.0148	0.0094	0.0138	0.0288

E.8 Cross-graph Evaluation

To further demonstrate the effectiveness of TMPNN over data-driven training-based methods, we perform a cross-graph evaluation to study their generalization ability from loop-free graphs to loopy graphs. For comparison, we pre-train V-MPNN with loop-free dataset and then apply the initialized V-MPNN to loopy graphs. We consider MAP inference in graphs with 9 nodes for evaluation. Results are shown in Table 16. As shown, TMPNN has better generalization ability over graphs with different structures compared to V-MPNN. For example, in TRIPARTITE (S12), TMPNN achieves 67.00% accuracy, which is 7.22% higher than V-MPNN. V-MPNN employs neural networks for message modeling, and hence also performs poorly on cross-structure evaluation. On the other hand, by leveraging the analytical derived message equations which hold true across graphs, TMPNN generalizes well to graphs with different structures.

Table 16: Cross-graph evaluation: from loop-free to loopy (MAP inference accuracy with unit %)

Methods	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	**Avg.**
V-MPNN	86.22	75.33	77.56	73.22	74.11	73.22	**76.00**	77.56	59.78	57.22	73.02
TMPNN	**89.67**	**77.89**	78.78	**89.78**	**74.56**	**88.11**	72.22	73.00	**67.00**	**63.00**	**77.40**

E.9 Data Efficiency Evaluation

We study the data efficiency of TMPNN compared to V-MPNN by using only 20% of the training data. We report the results on each structure in Table 17. As shown, given only 20% training data, TMPNN outperforms V-MPNN on all the structures.

E.10 Discussions on Large Graphs

Systematically evaluating TMPNN on large graphs presents a considerable challenge due to the absence of exact inference results. Alternatively, we show the utility of TMPNN on larger graphs through a node classification task. We consider both real-world graphs and synthetic graphs. It is worth emphasizing that the classification accuracy evaluation is only a surrogate evaluation. Our primary goal is to show the utility of our approach on large graphs.

Real-world Graph. We consider three real-world benchmark datasets for the node classification task PolBlog (Adamic and Glance, 2005), EuEmail (Leskovec et al., 2007) and CORA (Šubelj and Bajec, 2013). POLBLOG contains $1.49K$ nodes and $19.09K$ edges. EuEmail contains 1005 nodes and 16.0K edges. CORA contains 23.1K nodes and 89.1K edges. EuEmail is of a similar size to PolBlog and CORA is much larger than PolBlog. Following Eswaran et al. (2020), we define potential functions over graphs and perform MAP inference for node classification task. We use the typical 5-fold cross evaluation protocol and compare against training-free methods

Table 17: Data Efficiency Evaluation (Accuracy %; loop-free♣)

Graph	100% data		20% data	
	V-MPNN	TMPNN	V-MPNN	TMPNN
STAR♣	.93	**1.00**	.60	**1.00**
TREE♣	.96	**1.00**	.63	**1.00**
PATH♣	.97	**1.00**	.63	**1.00**
CYCLE	.85	**.90**	.54	**.82**
LADDER	.77	**.78**	.61	**.69**
GRID	.74	**.79**	.60	**.69**
C-LADDER	.83	**.90**	.63	**.73**
BARBELL	.71	**.75**	.60	**.70**
LOLLIPOP	**.88**	**.88**	.62	**.63**
WHEEL	.70	**.72**	.62	**.68**
BIPARTITE	**.74**	.73	.50	**.54**
TRIPARTITE	**.68**	.67	.53	**.51**
COMPLETE	**.65**	.63	.49	**.56**
Average	.80	**.83**	.58	**.73**

BP, TRW-MP, and MPLP. We also compare it to the training-based method V-MPNN. Due to the lack of supervision, Node-GNN can't be applied to these graphs and is thus excluded from comparison. We report the classification accuracy as shown below.

Table 18: Accuracy Evaluation on Large Real-world Graphs

Graph	BP	TRW-MP	MPLP	V-MPNN	TMPNN
PolBlog	.88	**.89**	.81	.67	**.89**
EuEmail	.74	.74	.74	.73	**.75**
CORA	.86	.74	.62	.61	**.88**

We in addition report the computing time of different algorithms as shown below. Evaluation is performed on a laptop with a 8-Core Intel Core i9 processor with CPU only.

Table 19: Efficiency Evaluation on Large Graphs (Computing time s)

Graph	BP	TRW-MP	MPLP	V-MPNN	TMPNN
PolBlog	6.8	35.34	206.05	**1.68**	14.78
EuEmail	5.72	29.25	150.92	**1.56**	12.86
CORA	586.12	1001.22	792.56	**206.71**	751.49

From the results, we can see that TMPNN achieves overall better accuracy on different real-world benchmark graphs, without a huge loss in computational efficiency. Particularly, on PolBlog, the computing time of TMPNN is $14.78s$, around $2\times$ faster than TRW-MP which takes $35.34s$. TMPNN takes longer computing time than BP mainly due to the process of predicting optimal entropy coefficients. While by leveraging the learned entropy coefficients, TMPNN achieves outstanding accuracy, without much loss in efficiency. V-MPNN models messages via neural network and thus achieves the best efficiency compared to other methods.

From the results we also observe that, on large real graphs, TMPNN's performance improvement may not be significant compared to BP. We hypothesize that the structures of the graphs can be one of the reasons. Despite the larger size, the real-world graphs are relatively sparse. For instance, given N as the number of nodes and M as the number of links, the average degree of connectivity (M/N) of a fully connected graph is $N-1$. In contrast, for CORA with $N = 23.1K$ and $M = 89.1K$, the average degree of connectivity is 3.85, which is much smaller than $N-1$. Unfortunately, we have no control over the complexity of the real data. To go one step further, we consider synthetic graphs with varying complexity as discussed below.

Synthetic Graph. To further study along this line, we employ the graph generator from Gatterbauer (2017) to produce large synthetic graphs and compare TMPNN against BP using the surrogate classification accuracy metric. Default settings for the graph generator are used. To be specific, we use the "planted distribution model" function for the graph generation. The input of this function for the experiments is listed in the following: number of nodes n = 1000. Node label distribution alpha = [1/2, 1/2] for binary variables. Compatibility matrix P is [0.7, 0.3; 0.3, 0.7]. Total number of edges $m = n \times d$. Distribution is 'powerlaw' with exponent equals '-0.3'. 'directed' argument is set to be 'False' for generating undirected graphs. The rest of the arguments are using default values. Given the inputs, the graph generation function outputs an adjacency matrix and node class labels that are used for classification accuracy evaluation. The synthetic graphs consist of a total of N=1000 nodes. We vary the average degree of connectivity (d) as d = 10, 20, 30. Notably, these complexities surpass those of the real-world graphs we evaluated on. To better indicate the complexity, we also report the maximum degree of connectivity among nodes and the total number of edges ($N \times d$). Results are shown in Table 20.

Table 20: Case study on synthetic graph with varying average degree of connectivity

Average (Max) degree of connectivity	Total number of edges	BP	TMPNN
10 (89)	10,000	0.503	0.797
20 (180)	20,000	0.597	0.842
30 (271)	30,000	0.497	0.999

On these three specific cases, TMPNN outperforms BP significantly, achieving notably higher classification accuracy on large and complex graphs. These values show that the performance is influenced by the average degree of connectivity of graph structures. Nonetheless, more surrogate studies as future work are required to fully understand the TMPNN's performance enhancement as the graph size and complexity increase. Besdes, we want to emphasize that this evaluation is an indirect assessment of TMPNN's inference accuracy. It remains challenging to comprehensively evaluate TMPNN on large graphs due to the absence of exact inference results. We will further study these issues as our future work.

E.11 Convergence Curve

We also plot the accuracy vs iteration curve. We consider MAP inference and plot the average accuracy over graphs of 9 nodes with different structures. The curve is shown in Figure 5. As the number of iterations increases, the average accuracy increases and then converges.

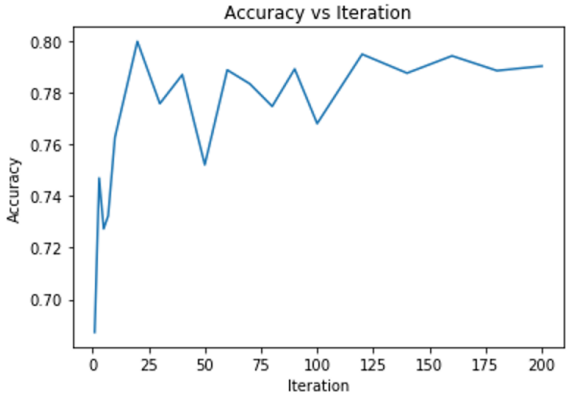

Figure 5: Accuracy vs iteration curve