

Efficient Model Evidence Computation in Tree-structured Factor Graphs

Hoang M.H. Nguyen
TU Eindhoven
Eindhoven, Netherlands
m.h.n.hoang@tue.nl

Bart van Erp
TU Eindhoven
Eindhoven, Netherlands
b.v.erp@tue.nl

İsmail Şenöz
TU Eindhoven
Eindhoven, Netherlands
i.senoz@tue.nl

Bert de Vries
TU Eindhoven
Eindhoven, Netherlands
bert.de.vries@tue.nl

Abstract—Model evidence is a fundamental performance measure in Bayesian machine learning as it represents how well a model fits an observed data set. Since model evidence is often an intractable quantity, the literature often resorts to computing instead the Bethe Free Energy (BFE), which for cycle-free models is a tractable upper bound on the (negative log-) model evidence. In this paper, we propose a different and faster evidence computation approach by tracking local normalization constants of sum-product messages, termed *scale factors*. We tabulate scale factor update rules for various elementary factor nodes and by experimental validation we verify the correctness of these update rules for models involving both discrete and continuous variables. We show how tracking scale factors leads to performance improvements compared to the traditional BFE computation approach.

Index Terms—Message Passing, Model Evidence, Probabilistic Inference, Scale Factors

I. INTRODUCTION

Building probabilistic models to explain sets of observations is at the core of machine learning, and model evidence is a powerful metric to evaluate model performance in the presence of data constraints [1]. The model evidence is the main mechanism behind model comparison techniques such as model selection, averaging [2] and combination [3]. Its computation can be carried out by integrating or summing over all latent variables. Unfortunately, this naive approach often suffers from the curse of dimensionality and can lead to an intractable calculation.

A generic strategy for model evidence computation is to calculate Bethe free energy (BFE) [4], an approximation of (negative log) model evidence. If a probabilistic model submits to exact inference and is tree-structured, the BFE is equal to the exact value of (negative log) evidence [5]–[7]. Calculating the BFE requires first computing the marginal posterior distributions of all latent variables in the model, before computing the actual model evidence. Despite the generality of this approach, in certain applications, such as event detection, we might only be interested in computing the model evidence and do not wish to compute marginal posterior distributions for efficiency considerations.

In this paper, we provide a different perspective on computing model evidence for belief propagation. More specifically, we extend sum-product messages with scale factors to simultaneously track the local evolution of the model evidence and

posterior distributions. Our perspective is inspired from the previous research performed in [8, Ch.6]. The scope of the current paper limits to conjugate probabilistic models whose underlying factor graph is a tree.

After motivating our perspective with a concrete example where the general model evidence computation strategy, i.e. computing BFE, is suboptimal in Section II, we make the following contributions:

- In Section III-C, we introduce scale factors for the local tracking of the normalization constant and tabulate new scale factor message computation rules in Table I in addition to the ones presented in [8, Ch.6]. These new rules add to the already presented factor nodes involving continuous variables and also extend towards factor nodes involving discrete variables.
- The new update rules are verified in Section IV for models containing both discrete and continuous variables. The results are compared to the general BFE computation strategy as specified in [5].
- We demonstrate how the use of scale factors results into a more efficient model performance evaluation of models submitting to exact inference in Section IV.

Section V discusses our approach and concludes the paper.

II. MOTIVATION

Bethe free energy (BFE) [4] has been shown as a generic tractable objective for many message passing algorithms in factor graphs. For example, [5] and [9] indicate that the update rules of well-known algorithms, e.g. belief propagation [10] and variational message passing [11], can be derived by minimizing BFE with appropriate local constraints on the marginal posterior distributions. Furthermore, BFE defines an approximate bound to the (negative log-) model evidence, and both of them are equal when a tree-structured model submits to exact inference [5]–[7]. Therefore, BFE computation is a generic approach to compute model evidence in machine learning.

The BFE computation is a two-step procedure in which marginal posterior distributions are first computed and then the BFE. Thus, in the BFE approach, the inference process is always carried out before computing the (approximate) model evidence. For exact implementation details of this approach we refer the interested reader to [5, Sec. 5].

The BFE computation strategy, however, can be inefficient in certain models. For example, in applications involving event detection using fixed-lag smoothers, we are mainly interested in the model evidence over a sliding window of data. If the model suddenly badly predicts the data resulting in a drop in the model evidence, we might speak of an event. The probabilistic model for a fixed-lag smoother is defined as follows

$$p(\mathbf{y}, \mathbf{z}_{t:t+T}) = p(\mathbf{y}_{1:t}, \mathbf{z}_t) \prod_{\tau=t+1}^T p(\mathbf{y}_\tau | \mathbf{z}_\tau) p(\mathbf{z}_\tau | \mathbf{z}_{\tau-1}),$$

where $\mathbf{y} \triangleq \mathbf{y}_{1:T} = [\mathbf{y}_1, \dots, \mathbf{y}_T]^\top$ denotes a sequence of observations and \mathbf{z}_t is the latent state. Using the BFE strategy for this application is highly inefficient, especially for large segments lengths and small step sizes, since the computation of the BFE requires the smoothing over the latent states of the entire segment, an operation which has to be performed for every sliding window again.

In Section III, we present an efficient strategy of model evidence computation for belief propagation in tree-structured models. This strategy only requires one step to compute model evidence, and thus is more suitable for applications which mainly need the knowledge of model evidence.

III. METHODS

In this section, we first briefly review Forney-style factor graphs and the sum-product algorithm [12], which encompasses belief propagation [10]. Next we introduce scale factors and present our new message computation rules including scale factors.

A. Forney-style factor graphs

Consider a factorizable function f that factorizes as

$$f(s) = \prod_{a \in \mathcal{V}} f_a(s_a), \quad (1)$$

where s_a refers to a set of factor-bound local variables such that $\cup_a s_a = s$. The individual factors f_a are indexed by a from the set of factor indices \mathcal{V} and are assumed to be integrable. Throughout this paper we will use Forney-style factor graphs (FFGs) [13] with notational conventions adopted from [14] to visualize these factorizable functions. An FFG is a graphical model that visualizes the factorization of a function as a graph, where nodes and edges represent factors and variables, respectively. An edge connects to a node only if the variable associated with the edge is an argument of the function associated with the node.

B. Sum-product message passing

Consider the global integration of (1) over all variables except for s_j as $\int f(s) ds_{\setminus j}$ ¹. As a result of the assumed factorization of f , this global integration can be performed through a set of smaller local computations, which summarize parts of the graph that has been integrated over. These

¹Integrals are taken over the support of the variables. If a variable is discrete valued, integral operators will be replaced with summation operators. For simplicity, we use integral signs throughout the paper.

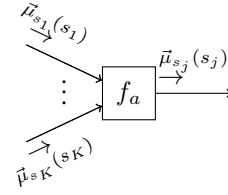


Fig. 1. A node $f_a(s_1, s_2, \dots, s_j, \dots, s_K)$. The sum-product message $\vec{\mu}_{s_j}(s_j)$ is specified by (2).

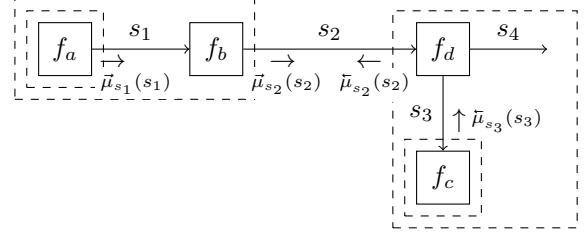


Fig. 2. A Forney-style factor graph representation of the factorized function in (3). The messages are specified in the example of Section III-B.

summaries are termed messages and are propagated along the edges of the graph. These messages are denoted by μ and can be locally computed on the graph. The sum-product message $\vec{\mu}_{s_j}(s_j)$ flowing out of the node $f_a(s_1, s_2, \dots, s_K)$ with incoming messages $\vec{\mu}_{s_k}(s_k)$ is given by [12]

$$\vec{\mu}_{s_j}(s_j) = \int f_a(s_1, s_2, \dots, s_K) \prod_{k \neq j} \vec{\mu}_{s_k}(s_k) ds_{\setminus j}. \quad (2)$$

Figure 1 visualizes this sum-product message computation rule. We represent the edges by arbitrarily directed arrows in order to distinguish between forward and backward messages propagating in or against the direction of an edge s_j as $\vec{\mu}_{s_j}(s_j)$ and $\bar{\mu}_{s_j}(s_j)$, respectively. Following this approach, the global integration reduces to the product of the messages of the variable of interest as $\int f(s) ds_{\setminus j} = \vec{\mu}_{s_j}(s_j) \bar{\mu}_{s_j}(s_j)$.

Example: Given a model $f(s_1, s_2, s_3, s_4)$ with factorization $f(s_1, s_2, s_3, s_4) = f_a(s_1) f_b(s_1, s_2) f_c(s_3) f_d(s_2, s_3, s_4)$ (3) The FFG representation of (3) is shown in Fig. 2. We can integrate over all variables except for s_2 by executing

$$\begin{aligned} f(s_2) &= \int f(s_1, s_2, s_3, s_4) ds_{\setminus 2}, \\ &= \underbrace{\int \underbrace{f_a(s_1)}_{\vec{\mu}_{s_1}(s_1)} f_b(s_1, s_2) ds_1}_{\vec{\mu}_{s_2}(s_2)} \underbrace{\int \underbrace{f_c(s_3)}_{\vec{\mu}_{s_3}(s_3)} f_d(s_2, s_3, s_4) ds_3 ds_4}_{\vec{\mu}_{s_2}(s_2)}}. \end{aligned}$$

Here, the integration is decomposed into the product of nested integrations, representing sum-product messages, as illustrated in Figure 2.

C. Scale factors

The probability distribution $p(s)$ of a model $f(s)$ can be computed as $p(s) = f(s)/Z$, where

$$Z = \int f(s) ds \quad (4)$$

is the normalization constant, under the assumption that $f(s)$ is integrable. When $f(s)$ is a probabilistic model constrained

by observed data, the normalization constant is better known as the model evidence. Often we are interested in calculating the marginal distributions of the variables in our model, for applications such as latent state tracking or parameter estimation. The marginal distribution of s_j can be computed as $p(s_j) = \int p(s) ds_{\setminus j}$. However, as the normalization constant Z is not directly available, we instead compute the product

$$Zp(s_j) = \int f(s) ds_{\setminus j}, \quad (5)$$

where the normalization constraint of the marginal distribution $p(s_j)$ allows for the retrieval of the normalization constant Z of the entire model $f(s)$ at the edge representing s_j .

The literature on message passing-based inference is largely focused on inferring the marginal distributions $p(s_j)$ only and not on tracking Z (e.g. [15], [16]). This means that only functions are propagated along the graph, often in the form of (normalized) distributions. The normalization constant then becomes a secondary objective which is often determined post-hoc using the methodology as BFE.

However, we argue that the computation of the normalization constant is an essential objective of probabilistic inference for the simple reason that the posterior distributions are much less useful for models with low evidence. Consequently, we follow [8, Ch.6] to jointly update the marginal distributions and the normalization constant using (5) through (2) by tracking the scaling of the messages.

From that perspective, a message $\vec{\mu}_{s_j}(s_j)$ can be decomposed as

$$\vec{\mu}_{s_j}(s_j) = \vec{\beta}_{s_j} \vec{p}_{s_j}(s_j), \quad (6)$$

where $\vec{p}_{s_j}(s_j)$ denotes the probability distribution representing the normalized functional form of the message $\vec{\mu}_{s_j}(s_j)$. The term $\vec{\beta}_{s_j}$ denotes the scaling of the message $\vec{\mu}_{s_j}(s_j)$, also known as the scale factor. Based on this definition, the normalization constant of a tree-structured model f can be computed at any edge in the corresponding FFG as follows [8, Ch.6]

$$Z = \int \vec{\mu}_{s_j}(s_j) \vec{\mu}_{s_j}(s_j) ds_j = \vec{\beta}_{s_j} \vec{\beta}_{s_j} \int \vec{p}_{s_j}(s_j) \vec{p}_{s_j}(s_j) ds_j. \quad (7)$$

The introduction of scale factors in the messages of (2) allows for the automatable joint computation of the marginal posterior distributions and the normalization constant using message passing. Probabilistic inference can then be automated by deriving the message computation rules for specific factor-message combinations, in which the scaling of the messages is no longer neglected. The rules can be saved in a look-up table and can be used to automatically perform probabilistic inference in an arbitrary probabilistic model as a result of the locality property of the sum-product algorithm. The main contribution of this paper is Table I, in which we have derived the message computation rules² for various factor nodes and message combinations as an extension to [8, Ch.6].

²All derivations of the message computation rules are available at https://github.com/biaslab/SiPS2022-EfficientModelEvidenceComputation/blob/main/sips2022_scalefactor_supplement.pdf.

In the next section we will show the benefits of the scale factor approach.

IV. EXPERIMENTS

In this section we describe a set of verification experiments³ of our newly derived message computation rules including scale factors. Sections IV-A-IV-C formally introduce the three probabilistic models used in the experiments, consisting of a linear Gaussian state space model, a hidden Markov model and a coin toss model. Finally, Section IV-D presents and discusses the results.

All experiments have been performed using the scientific programming language Julia [17] with the state-of-the-art probabilistic programming package `ReactiveMP.jl` [18]. `BenchmarkTools.jl` [19] has been used for benchmarking the different model evidence computation strategies. The experiments were performed on an Intel Core i7-9750H Processor with 32 GB of RAM running Windows 10 Enterprise.

A. Linear Gaussian state space model

The first probabilistic model concerns a linear Gaussian state space model (LGSSM), defined as

$$p(\mathbf{y}, \mathbf{z}) = p(\mathbf{z}_0) \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{z}_{n-1}),$$

with observed and latent variables $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]^\top$ and $\mathbf{z} = [\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_N]^\top$, respectively. N denotes the number of observations. The individual factors are specified as

$$\begin{aligned} p(\mathbf{y}_n | \mathbf{z}_n) &= \mathcal{N}(\mathbf{y}_n | \mathbf{B}\mathbf{z}_n, \mathbf{P}), \\ p(\mathbf{z}_n | \mathbf{z}_{n-1}) &= \mathcal{N}(\mathbf{z}_n | \mathbf{A}\mathbf{z}_{n-1}, \mathbf{Q}), \\ p(\mathbf{z}_0) &= \mathcal{N}(\mathbf{z}_0 | \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z), \end{aligned}$$

where $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The (square) state transition and observation matrices are denoted by \mathbf{A} and \mathbf{B} , respectively. \mathbf{P} and \mathbf{Q} represent the covariance matrices of the observation and process noise, respectively. Finally $\boldsymbol{\mu}_z$ and $\boldsymbol{\Sigma}_z$ denote the parameters of the prior distribution of \mathbf{z}_0 . In the experiments of this model the latent and observed variables are both set to be two-dimensional.

B. Hidden Markov model

The hidden Markov model (HMM) represents the second probabilistic model, whose factorization is defined as

$$p(\mathbf{y}, \mathbf{z}) = p(\mathbf{z}_0) \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{z}_{n-1}),$$

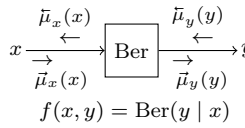
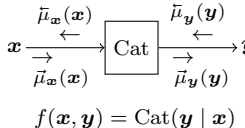
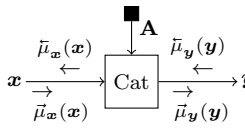
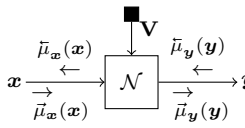
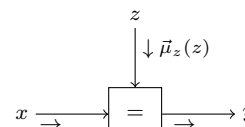
with observed and latent variables $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]^\top$ and $\mathbf{z} = [\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_N]^\top$, respectively. The individual factors are defined as

$$\begin{aligned} p(\mathbf{y}_n | \mathbf{z}_n) &= \text{Cat}(\mathbf{y}_n | \mathbf{B}\mathbf{z}_n), \\ p(\mathbf{z}_n | \mathbf{z}_{n-1}) &= \text{Cat}(\mathbf{z}_n | \mathbf{A}\mathbf{z}_{n-1}), \\ p(\mathbf{z}_0) &= \text{Cat}(\mathbf{z}_0 | \boldsymbol{\pi}_z), \end{aligned}$$

³All experiments are available at <https://github.com/biaslab/SiPS2022-EfficientModelEvidenceComputation>.

TABLE I

MESSAGE COMPUTATION RULES FOR VARIOUS NODE-MESSAGE COMBINATIONS INVOLVING SCALE FACTORS. THE INCOMING MESSAGES ARE CONSTRAINED TO BE PROPER PROBABILITY DISTRIBUTIONS AND IN PARTICULAR $\delta(\cdot)$ DENOTES A PROPER REALIZATION OF THE STOCHASTIC PROCESS.

Factor node	Incoming messages	Outgoing messages
	$\vec{\mu}_x(x) = \vec{\beta}_x \text{Beta}(x a, b)$ $\vec{\mu}_y(y) = \vec{\beta}_y \delta(y - \hat{y})$	$\vec{\mu}_y(y) = \vec{\beta}_x \text{Ber}\left(y \mid \frac{a}{a+b}\right)$ $\vec{\mu}_x(x) = \frac{1}{2} \vec{\beta}_y \text{Beta}(x \hat{y} + 1, 2 - \hat{y})$
	$\vec{\mu}_x(x) = \vec{\beta}_x \text{Dir}(x \pi)$ $\vec{\mu}_y(y) = \vec{\beta}_y \delta(y - \hat{y})$	$\vec{\mu}_y(y) = \vec{\beta}_x \text{Cat}\left(y \mid \frac{\pi}{\ \pi\ _1}\right)$ $\vec{\mu}_x(x) = \frac{1}{K!} \vec{\beta}_y \text{Dir}(x \hat{y} + 1)$ where $K = \dim_{\mathbb{R}}(\hat{y})$
	$\vec{\mu}_x(x) = \vec{\beta}_x \text{Cat}(x \pi)$ $\vec{\mu}_y(y) = \vec{\beta}_y \text{Cat}(y \pi)$	$\vec{\mu}_y(y) = \vec{\beta}_x \text{Cat}(y \mathbf{A}\pi)$ $\vec{\mu}_x(x) = \vec{\beta}_y \ \mathbf{A}^\top \pi\ _1 \text{Cat}\left(x \mid \frac{\mathbf{A}^\top \pi}{\ \mathbf{A}^\top \pi\ _1}\right)$
	$\vec{\mu}_x(x) = \vec{\beta}_x \mathcal{N}(x \mathbf{m}, \Sigma)$ $\vec{\mu}_y(y) = \vec{\beta}_y \mathcal{N}(y \mathbf{m}, \Sigma)$	$\vec{\mu}_y(y) = \vec{\beta}_y \mathcal{N}(y \mathbf{m}, \Sigma + \mathbf{V})$ $\vec{\mu}_x(x) = \vec{\beta}_y \mathcal{N}(x \mathbf{m}, \Sigma + \mathbf{V})$
	$\vec{\mu}_x(x) = \vec{\beta}_x \text{Beta}(x a_x, b_x)$ $\vec{\mu}_z(z) = \vec{\beta}_z \text{Beta}(z a_z, b_z)$ $\vec{\mu}_x(x) = \vec{\beta}_x \text{Ber}(x \pi_x)$ $\vec{\mu}_z(z) = \vec{\beta}_z \text{Ber}(z \pi_z)$ $\vec{\mu}_x(x) = \vec{\beta}_x \text{Dir}(x \pi_x)$ $\vec{\mu}_z(z) = \vec{\beta}_z \text{Dir}(z \pi_z)$ $\vec{\mu}_x(x) = \vec{\beta}_x \text{Cat}(x \pi_x)$ $\vec{\mu}_z(z) = \vec{\beta}_z \text{Cat}(z \pi_z)$ $\vec{\mu}_x(x) = \vec{\beta}_x \text{Gam}(x a_x, b_x)$ $\vec{\mu}_z(z) = \vec{\beta}_z \text{Gam}(z a_z, b_z)$	$\vec{\mu}_y(y) = \vec{\beta}_x \vec{\beta}_z \frac{B(a_y, b_y)}{B(a_x, b_x) B(a_z, b_z)} \text{Beta}(y a_y, b_y)$ where $a_y = a_x + a_z - 1$ and $b_y = b_x + b_z - 1$ $\vec{\mu}_y(y) = \vec{\beta}_x \vec{\beta}_z a \text{Ber}\left(y \mid \frac{\pi_x \pi_z}{a}\right)$ where $a = ((1 - \pi_x)(1 - \pi_z) + \pi_x \pi_z)$ $\vec{\mu}_y(y) = \vec{\beta}_x \vec{\beta}_z \frac{B(\pi_y)}{B(\pi_x) B(\pi_z)} \text{Dir}(y \pi_y)$ where $\pi_y = \pi_x + \pi_z - 1$ $\vec{\mu}_y(y) = \vec{\beta}_x \vec{\beta}_z \pi_x^\top \pi_z \text{Cat}\left(y \mid \frac{\pi_x \circ \pi_z}{\pi_x^\top \pi_z}\right)$ $\vec{\mu}_y(y) = \vec{\beta}_x \vec{\beta}_z \frac{\Gamma(a_y) b_x^{a_x} b_z^{a_z}}{\Gamma(a_x) \Gamma(a_z) b_y^{a_y}} \text{Gam}(y a_y, b_y)$ where $a_y = a_x + a_z - 1$ and $b_y = b_x + b_z$
<p>The equality node is symmetric in its arguments. The outgoing messages $\vec{\mu}_x(x)$ and $\vec{\mu}_z(z)$ can be computed similarly as $\vec{\mu}_y(y)$ by using the parameters of the 2 incoming messages on the other edges.</p>	$\vec{\mu}_x(x) = \vec{\beta}_x \mathcal{W}_p(x \mathbf{V}_x, n_x)$ $\vec{\mu}_z(z) = \vec{\beta}_z \mathcal{W}_p(z \mathbf{V}_z, n_z)$	$\vec{\mu}_y(y) = \vec{\beta}_x \vec{\beta}_z \frac{B}{A} \mathcal{W}(y \mathbf{V}_y, n_y)$ where $A = 2^{\frac{p(n_x + n_z)}{2}} \mathbf{V}_x ^{\frac{n_x}{2}} \mathbf{V}_y ^{\frac{n_y}{2}} \Gamma_p\left(\frac{n_x}{2}\right) \Gamma_p\left(\frac{n_y}{2}\right)$ $B = 2^{\frac{p n_y}{2}} \mathbf{V}_y ^{\frac{n_y}{2}} \Gamma_p\left(\frac{n_y}{2}\right)$ $n_y = n_x + n_z - p - 1; \mathbf{V}_y = (\mathbf{V}_x^{-1} + \mathbf{V}_z^{-1})^{-1}$

where $\text{Cat}(\mathbf{x} \mid \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{x_k}$ denotes the categorical distribution with event probabilities $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_K]^\top$, with K denoting the number of categories. The modeled variable $\mathbf{x} = [x_1, \dots, x_K]^\top$ denotes a 1-of- K binary vector with elements $x_k \in \{0, 1\}$, constrained by $\sum_k x_k = 1$ [1]. \mathbf{A} and \mathbf{B} denote the transition matrices, whose elements are constrained to $\mathbf{A}_{ij} \in [0, 1]$ and whose columns are also constrained by $\sum_i \mathbf{A}_{ij} = 1$. In the experiments using this model we set $K = 3$ for both the latent and observed variables.

C. Coin toss model

Lastly we introduce the coin toss model (CTM), which models the binary outcome of a coin toss. This model factorizes as

$$p(\mathbf{y}, \theta) = p(\theta) \prod_{n=1}^N p(y_n \mid \theta),$$

with outcomes $\mathbf{y} = [y_1, y_2, \dots, y_N]^\top$, where $y_n \in \{0, 1\}$. θ denotes the overall latent success probability. The factorization is further specified by

$$p(y_n \mid \theta) = \text{Ber}(y_n \mid \theta),$$

$$p(\theta) = \text{Beta}(\theta \mid \alpha_\theta, \beta_\theta),$$

where $\text{Ber}(x \mid \theta)$ denotes the Bernoulli distribution with success probability $\theta \in [0, 1]$. $\text{Beta}(x \mid \alpha, \beta)$ represents the Beta distribution with shape parameters $\alpha, \beta \in \mathbb{R}_{>0}$.

D. Results and benchmarks

For each of the models from Sections IV-A to IV-C data is generated for $N = \{10, 100, 1000\}$. Using these observations both the marginal distributions of the latent variables in the model and the corresponding model evidence is calculated. These quantities are computed simultaneously for the scale factor approach. Here, the model evidence is available at all edges in the graph. The Bethe free energy approach first computes the marginal distributions and sequentially computes the model evidence according to [5, Sec.5] Both approaches are solely implemented in `ReactiveMP.jl` as it provides superior performance with respect to alternative probabilistic programming packages such as `ForneyLab.jl` [20] or `Turing.jl` [21] as demonstrated in [18].

The computed model evidences are presented in Table II for all three model and different values of N , accompanied by the duration of their computations. For numerical stability we report the natural logarithm of the model evidence in both cases. Table II shows that both the approaches of Section III-C and BFE return the same value of the model evidence. Furthermore, we observe a decrease in run-time of 58.3%, averaged over the different models and values of N . This improvement illustrates the usefulness and applicability of scale factors for computing the model evidence for purposes such as model comparison or selection.

V. DISCUSSION AND CONCLUSIONS

For most applications involving scale factors where numerous observations are present, it is beneficial to propagate the logarithms of the scale factors instead of the scale factors

themselves. This creates a more numerically stable algorithm that is less prone to exceed the maximum and minimum values of the floating points number representing the scale factor. Consequently, the update rules for the scale factors will involve summations instead of product, possibly simplifying some update rules.

The current approach is tailored for probabilistic model without loops submitting to tractable exact inference. Interesting directions for future research are to extend the current methodology to models with loops and to models that employ alternative inference procedures based on constrained Bethe free energy minimization [5], such as (structured) variational message passing [11], [22], expectation propagation [23] and expectation maximization [24].

This paper has extended the set of scale factor update rules in [8, Ch.6] to factor nodes involving discrete random variables. These update rules can be used as an extension to conventional message passing-based probabilistic inference algorithms, allowing for the automatable computation of the model evidence in probabilistic models submitting to exact inference. The automated model evidence computation strategy using scale factors has been implemented as an extension to the probabilistic programming package `ReactiveMP.jl`. Here, we highlighted the benefits of scale factors in terms of computational speed with respect to the universal Bethe free energy computation strategy for the model evidence. Inference run times for three different probabilistic model for a varying number of observations are benchmarked, demonstrating an average computation speed increase of 58.3%.

REFERENCES

- [1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. [Online]. Available: <http://www.springer.com/computer/image+processing/book/978-0-387-31073-2>
- [2] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian Model Averaging: A Tutorial," *Statistical Science*, vol. 14, no. 4, pp. 382–401, 1999, publisher: Institute of Mathematical Statistics. [Online]. Available: <https://www.jstor.org/stable/2676803>
- [3] K. Monteith, J. L. Carroll, K. Seppi, and T. Martinez, "Turning Bayesian model averaging into Bayesian model combination," in *The 2011 International Joint Conference on Neural Networks*, San Jose, CA, USA, Jul. 2011, pp. 2657–2663, iSSN: 2161-4407.
- [4] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Bethe free energy, Kikuchi approximations, and belief propagation algorithms," *Advances in neural information processing systems*, vol. 13, p. 24, 2001.
- [5] İ. Şenöz, T. van de Laar, D. Bagaev, and B. de Vries, "Variational Message Passing and Local Constraint Manipulation in Factor Graphs," *Entropy*, vol. 23, no. 7, p. 807, Jul. 2021, number: 7 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: <https://www.mdpi.com/1099-4300/23/7/807>
- [6] M. J. Wainwright and M. I. Jordan, "Graphical Models, Exponential Families, and Variational Inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, Nov. 2008. [Online]. Available: <https://www.nowpublishers.com/article/Details/MAL-001>
- [7] M. Chertkov and V. Y. Chernyak, "Loop Calculus in Statistical Physics and Information Science," *Physical Review E*, vol. 73, no. 6, Jun. 2006, arXiv: cond-mat/0601487. [Online]. Available: <http://arxiv.org/abs/cond-mat/0601487>
- [8] C. Reller, "State-Space Methods in Statistical Signal Processing: New Ideas and Applications," Ph.D. dissertation, ETH Zurich, 2012.
- [9] D. Zhang, W. Wang, G. Fettweis, and X. Gao, "Unifying Message Passing Algorithms Under the Framework of Constrained Bethe Free Energy Minimization," *arXiv:1703.10932 [cs, math]*, Mar. 2017, arXiv: 1703.10932. [Online]. Available: <http://arxiv.org/abs/1703.10932>

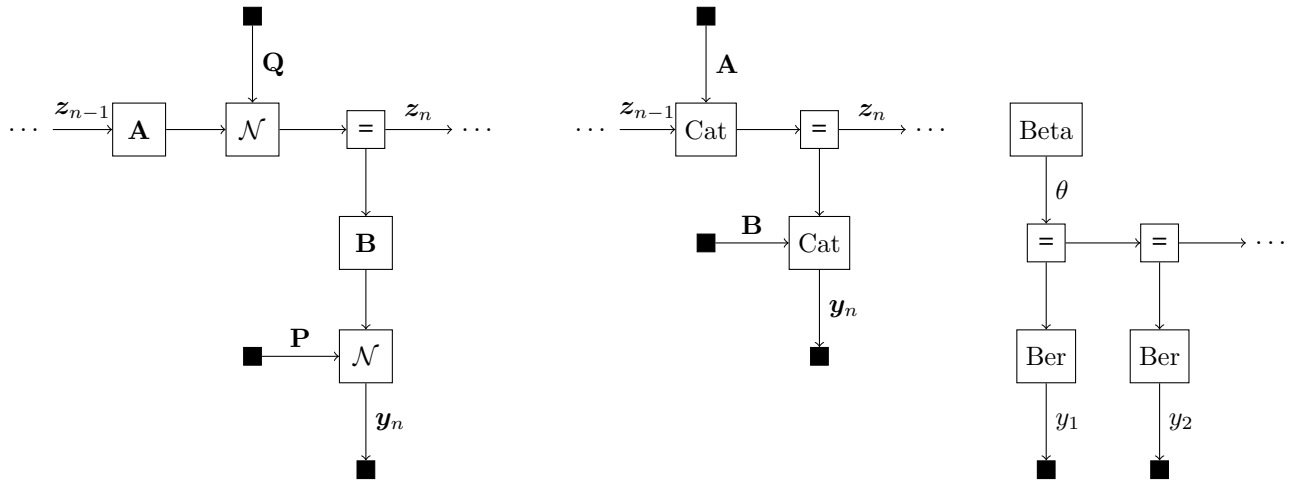


Fig. 3. Factor graph representation of the different model used throughout the experiments. (left) Single time slice of the linear Gaussian state space model of Section IV-A. (middle) Single time slice of the hidden Markov model of Section IV-B. (right) Two observations of the coin toss model of Section IV-C.

TABLE II

OVERVIEW OF THE OBTAINED (LOG) MODEL EVIDENCES AND CORRESPONDING MEDIAN RUNTIMES FOR THE MODELS SPECIFIED IN SECTION IV. THE RESULTS ARE PRESENTED FOR BOTH THE BETHE FREE ENERGY COMPUTATIONS AND THE PROPOSED SCALE FACTORS OF SECTION III-C. THE FASTEST RUNTIME IS PRESENTED IN BOLD.

model		LGSSM			HMM			CTM		
N		10	100	1000	10	100	1000	10	100	1000
Bethe free energy	$\ln p(\hat{y})$ [nats]	-6.54e1	-6.37e2	-6.53e-3	-1.08e1	-1.05e2	-1.06e3	-7.35e0	-5.76e1	-6.17e2
	runtime [ns]	2.57e5	2.84e6	3.65e7	9.66e4	9.82e5	1.53e7	2.83e4	2.45e5	3.78e6
scale factors	$\ln p(\hat{y})$ [nats]	-6.54e1	-6.37e2	-6.53e3	-1.08e1	-1.05e2	-1.06e3	-7.35e0	-5.76e1	-6.17e2
	runtime [ns]	1.30e5	1.49e6	2.03e7	4.16e4	4.71e5	8.37e6	6.96e3	6.79e4	6.99e5

- [10] J. Pearl, "Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach," in *Proceedings of the Second AAAI Conference on Artificial Intelligence*, ser. AAAI'82. Pittsburgh, Pennsylvania: AAAI Press, 1982, pp. 133–136. [Online]. Available: <http://www.aaai.org/Papers/AAAI/1982/AAAI82-032.pdf>
- [11] J. Winn and C. M. Bishop, "Variational Message Passing," *Journal of Machine Learning Research*, vol. 6, no. 23, pp. 661–694, 2005. [Online]. Available: <http://jmlr.org/papers/v6/winn05a.html>
- [12] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=910572
- [13] G. Forney, "Codes on graphs: normal realizations," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/910573>
- [14] H.-A. Loeliger, "An introduction to factor graphs," *Signal Processing Magazine, IEEE*, vol. 21, no. 1, pp. 28–41, Jan. 2004. [Online]. Available: <https://ieeexplore.ieee.org/document/1267047>
- [15] R. J. Drost and A. C. Singer, "Factor-Graph Algorithms for Equalization," *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 2052–2065, May 2007. [Online]. Available: <http://ieeexplore.ieee.org/document/4156381/>
- [16] S. Kori, H. Loeliger, and A. Lindgren, "AR model parameter estimation: from factor graphs to algorithms," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5. Montreal, Que., Canada: IEEE, 2004, pp. V–509–12. [Online]. Available: <http://ieeexplore.ieee.org/document/1327159/>
- [17] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah, "Julia: A Fresh Approach to Numerical Computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, Jan. 2017. [Online]. Available: <https://epubs.siam.org/doi/abs/10.1137/141000671>
- [18] D. Bagaev and B. de Vries, "Reactive Message Passing for Scalable Bayesian Inference," *arXiv:2112.13251 [cs]*, Dec. 2021, arXiv: 2112.13251. [Online]. Available: <http://arxiv.org/abs/2112.13251>
- [19] J. Chen and J. Revels, "Robust benchmarking in noisy environments," *arXiv:1608.04295 [cs]*, Aug. 2016, arXiv: 1608.04295. [Online]. Available: <http://arxiv.org/abs/1608.04295>
- [20] M. Cox, T. van de Laar, and B. de Vries, "A factor graph approach to automated design of Bayesian signal processing algorithms," *International Journal of Approximate Reasoning*, vol. 104, pp. 185–204, Jan. 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0888613X18304298>
- [21] H. Ge, K. Xu, and Z. Ghahramani, "Turing: A Language for Flexible Probabilistic Inference," in *International Conference on Artificial Intelligence and Statistics*, Mar. 2018, pp. 1682–1690, iSSN: 1938-7228 Section: Machine Learning. [Online]. Available: <http://proceedings.mlr.press/v84/ge18b.html>
- [22] J. Dauwels, "On Variational Message Passing on Factor Graphs," in *IEEE International Symposium on Information Theory*, Nice, France, Jun. 2007, pp. 2546–2550. [Online]. Available: <http://ieeexplore.ieee.org/abstract/document/4557602>
- [23] T. P. Minka, "Expectation Propagation for Approximate Bayesian Inference," in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 362–369. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2074022.2074067>
- [24] J. Dauwels, S. Kori, and H.-A. Loeliger, "Expectation maximization as message passing," in *International Symposium on Information Theory, 2005. ISIT 2005. Proceedings*, Sep. 2005, pp. 583–586.