APPROXIMATE INFERENCE BY KULLBACK-LEIBLER TENSOR BELIEF PROPAGATION

Patrick W.A. Wijnings[⋆†] *Sander Stuijk*[⋆] *Bert de Vries*[⋆‡] *Henk Corporaal*[⋆]

[⋆] Eindhoven University of Technology, Eindhoven, The Netherlands
[†] Sorama, Eindhoven, The Netherlands [‡] GN Hearing, Eindhoven, The Netherlands

ABSTRACT

Probabilistic programming provides a structured approach to signal processing algorithm design. The design task is formulated as a generative model, and the algorithm is derived through automatic inference. *Efficient* inference is a major challenge; e.g., the *Shafer-Shenoy algorithm* (SS) performs badly on models with large treewidth, which arise from various real-world problems. We focus on reducing the size of discrete models with large treewidth by storing intermediate factors in compressed form, thereby decoupling the variables through conditioning on introduced weights.

This work proposes pruning of these weights using Kullback-Leibler divergence. We adapt a strategy from the Gaussian mixture reduction literature, leading to *Kullback-Leibler Tensor Belief Propagation* (KL-TBP), in which we use agglomerative hierarchical clustering to subsequently merge pairs of weights. Experiments using benchmark problems show KL-TBP consistently achieves lower approximation error than existing methods with competitive runtime.

***Index Terms*—** Bayes methods, Tensors, Dimensionality reduction, Approximation algorithms

1. INTRODUCTION

Modern technology relies heavily on signal processing [1]. Development of signal processing algorithms is a complex problem. To reduce the engineering cost a structured approach such as the probabilistic programming paradigm [2] can be used.

It considers all quantities to be random variables. Their associations are described with a joint probability distribution function (PDF) in a so called generative model. Running an inference query on this model results in a posterior PDF over a relevant subset of the random variables, conditioned on the observed random variables. This posterior is derived from the generative model through the axioms of probability and related identities such as Bayes' rule. Finally, the answer to the signal processing task, i.e. an estimate for the relevant quantities (such as the state in an estimation problem), follows simply by plugging in the observed values.

Domain-specific languages and tools such as ForneyLab.jl [3] aid with formulation of the generative model. Once formulated, probabilistic programming promises fully automatable derivation (*inference*) of the posterior.

Efficient inference is its main challenge: in general, it is NP-hard in the number of variables [4]. In fact, even efficient computation of approximate solutions is often problematic [4]. The reason is that the language of probability is sufficiently flexible to allow expression of known-difficult combinatorial problems such as 3SAT [5].

The only way to ameliorate this is to restrict the accepted input tasks in some way. For example, one can force all random variables to be normally distributed. In that case, many tasks related to filtering and smoothing can still be formulated intuitively [6]. However, expression of non-linear signal processing has now become difficult.

One can also use approximate inference algorithms such as sampling (e.g. Markov chain Monte Carlo [7]) or variational methods (e.g. mean field [8]). However, these methods are also limited by the fundamental hardness of inference, i.e. there exist input tasks for which they will either approximate the posterior badly or require exorbitant resources. Conversely, it is often difficult to find conditions under which the approximations are guaranteed to be accurate. This is important because if these conditions do not exist the designer cannot trust the solutions given by these methods to be correct.

Finally, one can attempt to solve inference exactly using the Shafer-Shenoy algorithm (SS) [9]. SS is efficient only when the random variables in the task exhibit strong conditional independence, or more precisely, when the model has small *treewidth* [10].

In this work, we follow the last approach and investigate the performance of SS on models with *large* treewidth. We focus on models which are the product of many factors each depending on a small number of discrete random variables. Examples of such tasks are the *Linkage* and *Promedus* problem sets from the UAI2014 inference competition [11]. These are based on real-world datasets and have been used as benchmark problems by other authors [12, 13]. Although the individual factors are small, overall treewidth of these problems is considerable due to circular dependencies. Hence, SS performs badly in terms of runtime or memory usage.

We aim to reduce model size by compressing the intermediate factors of SS, thereby improving performance. We do so through the *naive Bayes assumption*, which states that all variables in a factor can be made independent when conditioned on an introduced auxiliary variable. By pruning the alphabet of this auxiliary variable, the other variables are partially decoupled. This allows the factor to be stored in compressed form. The mathematics are directly related to mixture models [14, Ch. 9] and tensor decompositions [15].

Wrigley et al.'s *Tensor Belief Propagation* (TBP) [12] is closely related to our approach. They enforce sparsity by *sampling* from the auxiliary variables. Although this allows for derivation of conditions under which the method performs well, there exist other compression strategies that will enable us to apply TBP when sampling fails or is too computationally expensive. This is important because it allows for a more universal application of probabilistic programming, thereby reducing cost of signal processing algorithm design.

This paper substantiates that use of the Kullback-Leibler divergence (KL) [16] achieves better compression than sampling. In particular, we adapt Runnalls' Gaussian mixture reduction strategy [17], leading to *Kullback-Leibler Tensor Belief Propagation* (KL-TBP).

Our approach is as follows. First, we review SS (Sec. 2) and TBP (Sec. 3). Sec. 4 introduces KL-TBP. Next, performance with TBP is compared using the *Linkage* and *Promedus* problem sets and Ising grids in Sec. 5. Conclusions are drawn in Sec. 6.

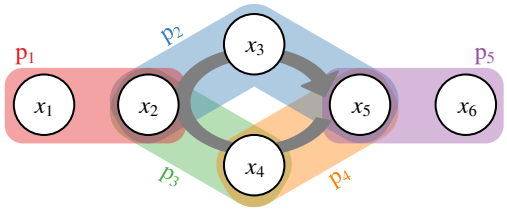
This work is funded by the NWO Perspectief program ZERO.
Correspondence to: p.w.a.wijnings@tue.nl.

2. INFERENCE USING SHAFER-SHENOY ALGORITHM

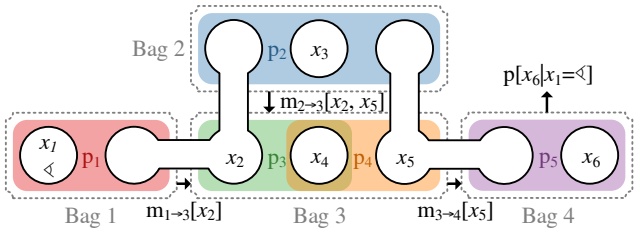
This section summarizes SS through a toy example with joint PDF:

$$\mathrm{p}[x_1,\cdots,x_6] \overset{\text{def}}{\propto} \mathrm{p}_1[x_1,x_2]\mathrm{p}_2[x_2,x_3,x_5]\mathrm{p}_3[x_2,x_4]\mathrm{p}_4[x_4,x_5]\mathrm{p}_5[x_5,x_6],$$

visualized in Fig. 1a. Each x_k is drawn from a discrete, finite alphabet $\mathcal{X}_k$. Our inference query is: $\mathrm{p}[x_6|x_1] = ?$.



(a) Venn diagram. Nodes represent the variables. Colored areas represent the factors. The gray arrow denotes a circular dependency.



(b) Junction tree. An overlay of bags is added to the Venn diagram. Factors always fit inside at least one bag; variables are shared between bags. Intermediate factors are passed over the connections (⟶) between the bags until the inference query is answered.

Fig. 1: Graphical representation of example generative model.

First, to resolve cyclic dependencies, a junction tree (JT) is built (Fig. 1b). There are many ways to do so, but it remains unclear which is best for TBP [12, Sec. 6]. We opt for Tamaki's heuristic solver [18], since it performed well in PACE2017 [19]. The JT only needs to be built once. Next, the query is answered (up to a normalization constant) for a given observation $x_1 = \triangleleft$ by passing intermediate factors (*sum-product messages*) over the JT:

$$\mathrm{m}_{2\to3}[x_2,x_5] \overset{\text{def}}{=} \sum_{x_3\in\mathcal{X}_3} \mathrm{p}_2[x_2,x_3,x_5], \qquad \mathrm{m}_{1\to3}[x_2] \overset{\text{def}}{=} \mathrm{p}_1[x_1=\triangleleft,x_2],$$

$$\mathrm{m}_{3\to4}[x_5] \overset{\text{def}}{=} \sum_{x_2\in\mathcal{X}_2}\sum_{x_4\in\mathcal{X}_4} \mathrm{m}_{1\to3}[x_2]\mathrm{m}_{2\to3}[x_2,x_5]\mathrm{p}_3[x_2,x_4]\mathrm{p}_4[x_4,x_5],$$

$$\mathrm{p}[x_6|x_1=\triangleleft] \propto \sum_{x_5\in\mathcal{X}_5} \mathrm{m}_{3\to4}[x_5]\mathrm{p}_5[x_5,x_6]. \tag{1}$$

This reveals the two main operations of SS: multiplication of factors and marginalization of nuisance variables. The latter makes SS expensive when the bags contain many variables, i.e. large treewidth.

3. TENSOR BELIEF PROPAGATION

Theorem 1 (Naive Bayes assumption)**.** *For any factor* $\mathrm{f}[x_1,\cdots,x_K]$ *with* $x_k \in \mathcal{X}_k$*, there exists a (lossless) compressed form:*

$$\mathrm{f}[x_1,\cdots,x_K] = \sum_{w\in\mathcal{W}} \mathrm{f}[w]\prod_{k=1}^{K}\mathrm{f}[x_k|w], \tag{2}$$

where $w \in \mathcal{W}$ *is an introduced auxiliary variable* (weight) *with alphabet of* $|\mathcal{W}| \le \prod_{k=1}^{K}|\mathcal{X}_k|$ *symbols. All* $\mathrm{f}[\cdot]$ *are non-negative.*

Proof. Choose $w \overset{\text{def}}{=} (w_1,\cdots,w_K) \in \mathcal{W} \overset{\text{def}}{=} \mathcal{X}_1\times\cdots\times\mathcal{X}_K$ and:

$$\mathrm{f}[w] \overset{\text{def}}{=} \mathrm{f}[x_1=w_1,\cdots,x_K=w_K], \qquad \mathrm{f}[x_k|w] \overset{\text{def}}{=} \begin{cases} 1 & \text{if } x_k = w_k \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

This is a one-hot encoding and hence equality follows directly. □

Remark. To allow interpretation as PDFs, we normalize as:

$$\sum_{x_k\in\mathcal{X}_k} \mathrm{f}[x_k|w] = 1 \;\forall k \in 1..K, \qquad \sum_{w\in\mathcal{W}} \mathrm{f}[w] = 1. \tag{4}$$

In TBP, all factors (both from the joint PDF and the intermediate messages) are forced to be in compressed form. This allows for trivial marginalization:

$$\sum_{x_{k'}\in\mathcal{X}_{k'}} \mathrm{f}[x_1,\cdots,x_K] = \sum_{w\in\mathcal{W}} \mathrm{f}[w]\prod_{\substack{k=1\\k\neq k'}}^{K}\mathrm{f}[x_k|w]. \tag{5}$$

The form is also closed under multiplication. However, the product of two factors with weight alphabets $\mathcal{W}_1$ and $\mathcal{W}_2$ will have weight alphabet $\mathcal{W}_1\times\mathcal{W}_2$, i.e. alphabet growth is exponential in the number of multiplications. Thus, TBP trades efficient marginalization for the necessity of weight pruning during multiplication.

3.1. Storage requirements

Storage of a full probability table for $\mathrm{f}[x_1,\cdots,x_K]$ requires $\prod_{k=1}^{K}|\mathcal{X}_k|$ floating point numbers. In compressed form, this becomes $|\mathcal{W}|\big(1+\sum_{k=1}^{K}|\mathcal{X}_k|\big)$. Then, required storage does not grow exponentially with dimensionality K, as long as growth of $|\mathcal{W}|$ is limited. Indeed, this is the idea of the canonical polyadic tensor decomposition [20], which in 2D reduces to the (truncated) singular value decomposition. This motivates feasibility of weight pruning.

3.2. Weight pruning by sampling

Wrigley et al. [12] choose a sampling approach: a pruned alphabet $\hat{\mathcal{W}}$ is constructed by drawing $|\hat{\mathcal{W}}| < |\mathcal{W}|$ realizations of w with probability $\mathrm{f}[w]$. Undrawn symbols of $\mathcal{W}$ are discarded. Hence, it tends to fail when many terms are equally important, i.e. when $\mathrm{f}[w]$ has high entropy. As a corollary, this means the one-hot encoding strategy (3) combines poorly with the sampling approach. Indeed, Wrigley et al. use a tensor decomposition [21] to perform the initial conversion to compressed form. Their results are sensitive to the settings of this decomposition algorithm (Fig. 2c–d, $r = 2$ vs. $r = 4$).

4. PRUNING WITH KULLBACK-LEIBLER DIVERGENCE

KL [22, (2.26)] measures how accurately $\hat{f}$ approximates factor f:

$$\mathrm{D}\big(f\|\hat{f}\big) \overset{\text{def}}{=} \sum_{x_1\in\mathcal{X}_1}\cdots\sum_{x_K\in\mathcal{X}_K} \mathrm{f}[x_1,\cdots,x_K]\log_2\frac{\mathrm{f}[x_1,\cdots,x_K]}{\hat{\mathrm{f}}[x_1,\cdots,x_K]}. \tag{6}$$

It is not an arbitrary metric and strongly founded in information theory, where it quantifies how much information (in bits) is lost if a code for $\hat{f}$ is applied to f [22, Sec. 2.3]. It also follows from a set of axioms in the context of inductive inference [23].

Ideally we would minimize (6) directly while constraining $\hat{f}$ to be of compressed form. Unfortunately, this appears to be a difficult non-convex optimization problem, unless $|\hat{\mathcal{W}}| = 1$. For that case, Th. 2 gives the solution and Th. 3 bounds the corresponding KL.

Theorem 2 (Principal component)**.** *Given a factor f in the compressed form of Th. 1, the best (in KL sense) approximation $\hat{f}^\star$ in compressed form with associated alphabet $\hat{\mathcal{W}} = \{1\}$ is:*

$$\hat{\mathrm{f}}^\star[x_k|\hat{w}=1] \overset{\text{def}}{\propto} \sum_{w\in\mathcal{W}} \mathrm{f}[w]\mathrm{f}[x_k|w], \quad \hat{\mathrm{f}}^\star[\hat{w}=1] \overset{\text{def}}{=} 1. \tag{7}$$

Proof. This follows from application of [24, Th. 2]. □

Theorem 3 (KL upper bound)**.** *Given f and $\hat{f}^\star$ from Th. 2,*

$$\mathrm{D}\big(f\|\hat{f}^\star\big) \leq \sum_{w\in\mathcal{W}} \mathrm{f}[w] \sum_{k=1}^{K} \sum_{x_k\in\mathcal{X}_k} \mathrm{f}[x_k|w] \log \frac{\mathrm{f}[x_k|w]}{\hat{\mathrm{f}}^\star[x_k]}. \tag{8}$$

Proof. This follows from convexity of KL [22, Th. 2.7.2]. □

In the context of Gaussian mixture reduction, Runnalls gives analogous results for the normal distribution in [17, Th. 2 & 3]. He observes (8) can be used as a (symmetric) distance metric between a pair of terms (corresponding to two weights $w_i, w_j \in \mathcal{W}$) from f. By running a clustering algorithm on the terms and then merging each cluster into its principal component using Th. 2, the weight alphabet is pruned while heuristically minimizing (6).

In our proposed method, *Kullback-Leibler Tensor Belief Propagation* (KL-TBP), we opt for *agglomerative hierarchical clustering* (AHC) [25, Sec. 5.1], following Runnalls approach. In AHC, (8) is evaluated for *every* pair of terms. The pair with smallest KL upper bound is greedily merged using Th. 2. This is repeated until the desired amount of terms remains. Note that for the next iteration (8) only needs to be re-evaluated for pairs involving the new term; all others can be cached from the previous iterations. Nonetheless, AHC is computationally expensive because it operates on *pairs* of terms. In the context of SS, where factors are the product of other factors, this means AHC operates on *pairs of pairs* of terms. However, our results show that this is offset by excellent clustering performance, allowing for a relatively small number of terms.

Because AHC prunes by merging instead of discarding, it combines well with the one-hot encoding strategy (3). This allows us to compute small SS intermediate messages exactly, only converting on-the-fly to compressed form once message size exceeds the desired memory limit.

Due to space limitations, we do not provide a full algorithm listing for KL-TBP. However, our C implementation is available on request.

5. RESULTS

Firstly, in Fig. 2 we compare performance of KL-TBP with TBP and other methods from the literature. To this end, we adopt the benchmark problems and error metric from Wrigley et al. [12] and Zhu and Ermon [13]. Aggregation of their results into one figure allows for direct comparison with our results.

The benchmark consists of the UAI2014 *Promedus* (medical diagnosis) and *Linkage* (genetic linkage) problem sets, as well as Ising grids (ferromagnetism) [26, Sec. 2.5] with attractive and mixed interactions with parameters cf. [12]. For each problem instance, the marginal posteriors over each random variable are queried. The error metric is the L_1 error over all variables and states:

$$L_1(p\|\hat{p}) \overset{\text{def}}{=} \frac{1}{K}\sum_{k=1}^{K} \frac{1}{|\mathcal{X}_k|} \sum_{x'_k\in\mathcal{X}_k} \big|\mathrm{p}\big[x_k = x'_k|\dots\big] - \hat{\mathrm{p}}\big[x_k = x'_k|\dots\big]\big|, \tag{9}$$

where $\hat{\mathrm{p}}[x_k|\dots]$ are our solutions, and $\mathrm{p}[x_k|\dots]$ are the true posteriors as provided by UAI2014 or exactly computed using SS (Ising grids).

For KL-TBP and TBP, weight alphabet size $\big|\hat{\mathcal{W}}\big|$ is limited for all intermediate messages in compressed form. Since KL-TBP allows on-the-fly conversion from full factors, we impose the additional constraint that these do not exceed the storage size of their compressed equivalent (see Sec. 3.1).

Our method consistently outperforms TBP. It also achieves lower error than other approximate inference methods (without weight alphabet) such as *mean field*. In terms of runtime, for $\big|\hat{\mathcal{W}}\big| = 10^5$ Wrigley et al. report an average of about 10 minutes (Linkage) and 3 minutes (Promedus) per instance (*Intel Core i5, 1.4 GHz*). We achieve comparable performance for $\big|\hat{\mathcal{W}}\big| = 10$, with corresponding runtimes of 2 minutes (Linkage) and 15 seconds (Promedus) per instance (*Intel Core i9-9900K, 3.6 GHz*). This confirms that KL-TBP (with small alphabet) is competitive with TBP (with large alphabet).

Note that our runtimes do not include the construction of the junction tree, since that task can be carried out without observations (evidence). We ran Tamaki's heuristic solver [18] with a limit of 10 minutes per instance (*Intel Core i7-7560U, 2.4 GHz*). However, good solutions were already returned within 2 minutes per instance. Wrigley et al. used a min-fill heuristic for junction tree construction which we expect to have small computational cost.

Our runtime does become excessive for large alphabets (i.e. $\big|\hat{\mathcal{W}}\big| \geq 100$ for the UAI2014 problems) due to AHC. Although not necessary to solve the representative benchmark problems, this could be ameliorated by parallelizing the distance computations (our AHC implementation used a single core), or switching to a cheaper (but less accurate) clustering algorithm.

Secondly, in Fig. 3 we investigate the relation between model size and performance of KL-TBP with a fixed storage limit. Fig. 3a shows model size of each UAI2014 problem instance, both in terms of the largest factor in the joint PDF and the largest SS intermediate message (with junction tree constructed as above). When SS is implemented in a straight-forward way (i.e. without loop transformations cf. Halide [27]), the latter is indicative of required memory.

Fig. 3b shows performance of KL-TBP with a limit of 1000 floating point numbers per factor. Thus, actual weight alphabet size $\big|\hat{\mathcal{W}}\big|$ is variable and follows from the number of variables (Sec. 3.1).

We measure performance using KL (6) for each marginal:

$$\mathrm{D}(\mathrm{p}[x_k|\dots]\|\hat{\mathrm{p}}[x_k|\dots]), \tag{10}$$

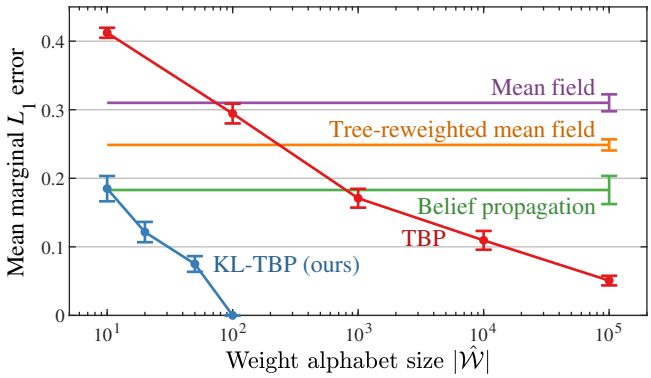
and report mean and 90th quantiles over all random variables. We exclude the few 'random' variables with $|\mathcal{X}_k| = 1$ for which KL is zero by definition. Most posterior variables in these problems have $|\mathcal{X}_k| \approx 2$ and hence contain at most 1 bit of entropy. Hence, KL divergences approaching 1 bit are significant. This way of reporting performance is more fine-grained than mean L_1 error.

Instances which fit inside the limit are solved exactly, since no messages need compression and hence KL-TBP reduces to exact SS. *Linkage* problems compress well, resulting in small errors even for problems with large storage requirement. The *Promedus* set is more difficult to compress, resulting in a trend of increasing divergence with increasing problem size, which was also reflected in mean L_1 error, which is larger for *Promedus* than for *Linkage* (see Fig. 2). Average runtime is 1¡ minute (Linkage) and 7 seconds (Promedus).

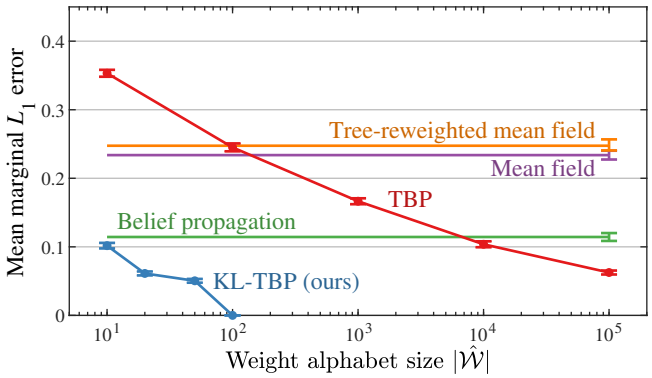
6. CONCLUSION

We have formulated Tensor Belief Propagation using Kullback-Leibler divergence for weight pruning. The experiments support our claim that it outperforms pruning by sampling.

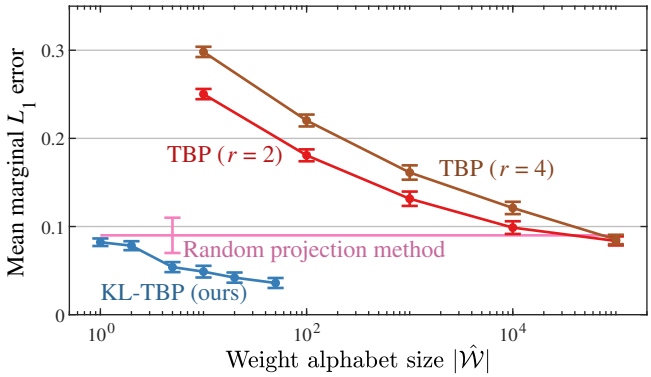
Future work includes formulation of a tractable error bound on (10) using Th. 3, and extension to continuous random variables.



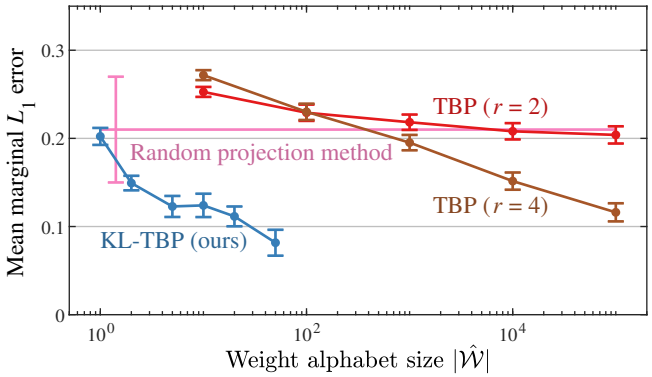
(a) Ising grids (10×10; attractive interactions; 100 random models).



(b) Ising grids (10×10; mixed interactions; 100 random models).

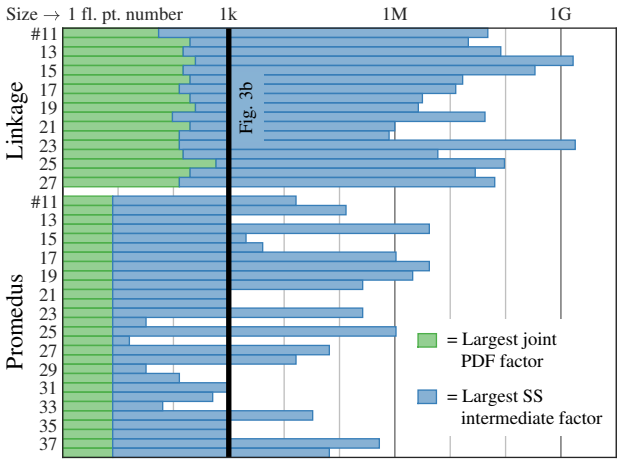


(c) Linkage (genetic linkage) problems (*UAI2014*).

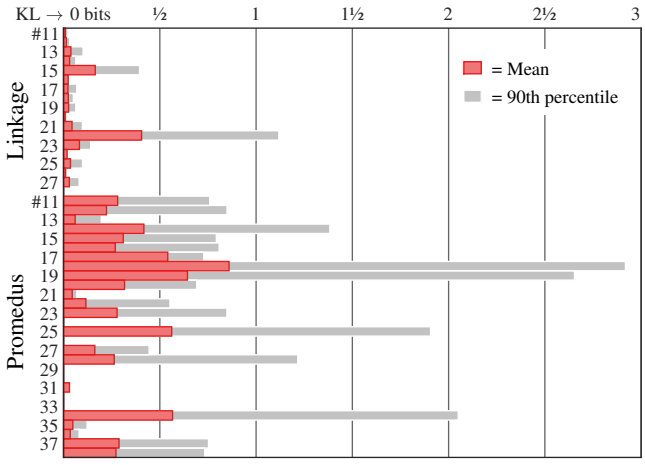


(d) Promedus (medical diagnosis) problems (*UAI2014*).

Fig. 2: Comparison of KL-TBP with other methods from the literature. Performance is measured using mean L_1 error over all variables and states. Error bars denote standard error over the problem set. *KL-TBP* denotes our results. *Random projection method* results taken from Zhu and Ermon [13, Tbl. 1]. *TBP* and other results taken from Wrigley et al. [12, Fig. 1 & 3].



(a) Model size, quantified as the amount of *floating point numbers* required to store the largest SS intermediate factor. This factor, whose size is indicative of treewidth, is considerably larger than the factors in the joint PDF due to circular dependencies.



(b) Performance of KL-TBP for enforced storage limit of 1000 floating point numbers per factor, measured using *Kullback-Leibler divergence* between true and approximated marginal posteriors. Mean and percentile are over the random variables in each problem instance.

Fig. 3: Relation between model size and KL-TBP performance.

7. REFERENCES

[1] IEEE Signal Processing Society, "What is Signal Processing?," https://signalprocessingsociety.org/our-story/signal-processing-101 [Accessed: 15 August 2019], Dec. 2015.

[2] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood, "An Introduction to Probabilistic Programming," *arXiv:1809.10756 [cs, stat]*, Sept. 2018, arXiv: 1809.10756.

[3] Thijs van de Laar, Marco Cox, and Bert de Vries, "ForneyLab.jl: a Julia Toolbox for Factor Graph-based Probabilistic Programming," in *JuliaCon*, London, UK, Aug. 2018.

[4] Johan Kwisthout, Todd Wareham, and Iris van Rooij, "Bayesian Intractability Is Not an Ailment That Approximation Can Cure," *Cognitive Science*, vol. 35, no. 5, pp. 779–784, July 2011.

[5] Gregory F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, vol. 42, no. 2, pp. 393–405, Mar. 1990.

[6] Hans-Andrea Loeliger, Justin Dauwels, Junli Hu, Sascha Korl, Li Ping, and Frank R. Kschischang, "The Factor Graph Approach to Model-Based Signal Processing," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1295–1322, June 2007.

[7] Persi Diaconis, "The Markov chain Monte Carlo revolution," *Bulletin of the American Mathematical Society*, vol. 46, no. 2, pp. 179–205, 2009.

[8] Jonathan S. Yedidia, "An Idiosyncratic Journey Beyond Mean Field Theory," in *Advanced Mean Field Methods*, pp. 37–49. 2000.

[9] Glenn R. Shafer and Prakash P. Shenoy, "Probability propagation," *Annals of Mathematics and Artificial Intelligence*, vol. 2, no. 1, pp. 327–351, Mar. 1990.

[10] Johan Kwisthout, Hans Bodlaender, and L.C. van der Gaag, "The Necessity of Bounded Treewidth for Efficient Inference in Bayesian Networks," *Frontiers in Artificial Intelligence and Applications*, pp. 237–242, 2010.

[11] Vibhav Gogate, Tahrima Rahman, Somdeb Sarkhel, David Smith, and Deepak Venugopal, "UAI 2014 Inference Competition," http://www.hlt.utdallas.edu/~vgogate/uai14-competition [Accessed: 21 June 2019].

[12] Andrew Wrigley, Wee Sun Lee, and Nan Ye, "Tensor Belief Propagation," in *International Conference on Machine Learning*, July 2017, pp. 3771–3779.

[13] Michael Zhu and Stefano Ermon, "A Hybrid Approach for Probabilistic Inference using Random Projections," in *International Conference on Machine Learning*, June 2015, pp. 2039–2047.

[14] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag New York, Inc., 2006.

[15] Lek-Heng Lim and Pierre Comon, "Nonnegative approximations of nonnegative tensors," *Journal of Chemometrics*, vol. 23, no. 7-8, pp. 432–441, 2009.

[16] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[17] A. R. Runnalls, "Kullback-Leibler Approach to Gaussian Mixture Reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 989–999, July 2007.

[18] Hisao Tamaki, "Positive-instance driven dynamic programming for treewidth," *arXiv:1704.05286 [cs]*, Apr. 2017, arXiv: 1704.05286.

[19] Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller, "The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration," in *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, Daniel Lokshtanov and Naomi Nishimura, Eds., Dagstuhl, Germany, 2018, vol. 89 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 30:1–30:12, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[20] Richard A Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis.," *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, Dec. 1970.

[21] Brett W. Bader, Tamara G. Kolda, and others, "MATLAB Tensor Toolbox Version 2.6," http://www.tensortoolbox.org/ [Accessed: 21 October 2019], Feb. 2015.

[22] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., second edition, 2005.

[23] Ariel Caticha, "Relative Entropy and Inductive Inference," *AIP Conference Proceedings*, vol. 707, no. 1, pp. 75–96, Apr. 2004.

[24] K. Huang and N. D. Sidiropoulos, "Kullback-Leibler principal component for tensors is not NP-hard," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct. 2017, pp. 693–697.

[25] Lior Rokach and Oded Maimon, "Clustering Methods," in *Data Mining and Knowledge Discovery Handbook*, Oded Maimon and Lior Rokach, Eds., pp. 321–352. Springer US, Boston, MA, 2005.

[26] Marc Mezard and Andrea Montanari, *Information, physics, and computation*, Oxford graduate texts. Oxford University Press, Oxford ; New York, 2009, OCLC: ocn234430714.

[27] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe, "Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing Pipelines," in *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2013, pp. 519–530, ACM.