The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Computing Divergences between Discrete Decomposable Models

Loong Kuan Lee^1, Nico Piatkowski^2, François Petitjean^1, and Geoffrey I. Webb^1

^{1} Department of Data Science and AI, Monash University, Melbourne, Australia
^{2}Fraunhofer IAIS, Schloss Birlinghoven, 53757 Sankt Augustin, Germany
mail@lklee.dev

Abstract

There are many applications that benefit from computing the exact divergence between 2 discrete probability measures, including machine learning. Unfortunately, in the absence of any assumptions on the structure or independencies within these distributions, computing the divergence between them is an intractable problem in high dimensions. We show that we are able to compute a wide family of functionals and divergences, such as the alpha-beta divergence, between two decomposable models, i.e. chordal Markov networks, in time exponential to the treewidth of these models. The alpha-beta divergence is a family of divergences that include popular divergences such as the Kullback-Leibler divergence, the Hellinger distance, and the chi-squared divergence. Thus, we can accurately compute the exact values of any of this broad class of divergences to the extent to which we can accurately model the two distributions using decomposable models.

Introduction

Computing the divergence, i.e. the degree of "difference", between two joint probability distributions is a problem that has many applications in the field of Machine Learning. For instance, it can be used to estimate the divergence between the underlying distributions of two data samples. This particular application is useful in the study of changing distributions, i.e. concept drift (Schlimmer and Granger 1986; Webb et al. 2018), in the detection of anomalous regions in spatio-temporal data (Barz et al. 2019; Piatkowski, Lee, and Morik 2013), and in tasks related to the retrieval, classification, and visualisation of time series data (Chen, Ye, and Li 2020).

Although there has been much work in *estimating* the divergence between 2 general *high-dimensional* discrete distributions (Bhattacharya, Kar, and Pal 2009; Abdullah et al. 2016), they do not compute the exact divergence between these distributions as it is intractable to do so without any knowledge or assumptions made regarding the structure within these distributions. Instead, approaches that do take advantage of some structural properties within the distributions for an efficient *computation* of divergences have appeared in the literature before, e.g., for computing the Kullback-Leibler (KL) divergence between Bayesian networks (BNs) (Moral, Cano, and Gómez-Olmedo 2021). It is also possible to tractably compute the KL divergence between a general Markov network (MN) and a MN where inference tasks are tractable (Koller and Friedman 2009).

However, there are situations where one might want to compute divergences other than the KL divergence (Nowozin, Cseke, and Tomioka 2016), in particular in the variational inference community where they have been employed to derive alternative evidence lower bounds (Chen et al. 2018; Li and Turner 2016; Dieng et al. 2017) or in the context of generative models (Genevay, Peyre, and Cuturi 2018). Furthermore, in natural language processing, using the KL divergence is problematic in the presence of uneven word frequencies (Labeau and Cohen 2019). Even for fundamental problems like model selection, we show that considering different types of divergences can be beneficial.

Motivated by these considerations, in this paper we show how to compute a wide family of divergences, the $\alpha\beta$-divergences, between two decomposable models (DMs). In the process of showing how the $\alpha\beta$-divergence can be computed between any two DMs, we will reach a more general result. That is, we will show how one can compute, between two DMs, the functional $\mathcal{F}$ defined in Definition 1:

Definition 1. *(Functional $\mathcal{F}$)*

$$\begin{aligned}&\mathcal{F}(\mathbb{P}, \mathbb{Q}; g, h, g^*, h^*)\\ &= \sum_{\boldsymbol{x}\in\mathcal{X}} \left[g\left[\mathbb{P}\right](\boldsymbol{x})\right] \left[h\left[\mathbb{Q}\right](\boldsymbol{x})\right] L\Big(\left[g^*[\mathbb{P}](\boldsymbol{x})\right]\left[h^*[\mathbb{Q}](\boldsymbol{x})\right]\Big)\end{aligned}$$

where, for any distribution $\mathbb{P}$ of a DM with graph structure $\mathcal{G}$, L is any function with the property $L\left(\prod_r r\right) = \sum_r L(r)$, and $f \in \{g, h, g^*, h^*\}$ are functionals with the property:

$$f\left[\prod_{C\in\boldsymbol{\mathcal{C}}(\mathcal{G})} \mathbb{P}_C\right](\boldsymbol{x}_C) = \prod_{C\in\boldsymbol{\mathcal{C}}(\mathcal{G})} f\left[\mathbb{P}_C\right](\boldsymbol{x}_C) \qquad (1)$$

An example of such a functional is the power functional: $[\prod_{C\in\boldsymbol{\mathcal{C}}(\mathcal{G})} \mathbb{P}_C(\boldsymbol{x}_C)]^2 = \prod_{C\in\boldsymbol{\mathcal{C}}(\mathcal{G})} \mathbb{P}_C(\boldsymbol{x}_C)^2$.

This result implies the possibility for the computation of divergences and functionals other than the $\alpha\beta$-divergence between two DMs. In fact, we show that $\mathcal{F}$ can be computed by running the junction tree algorithm (JTA) over a specifically constructed chordal graph and set of initial factors.

Proofs for our contributed theoretical results are deferred to the technical appendix of the extended version of this paper at: https://arxiv.org/abs/2112.04583

Background and Notation

Let us summarize the notation and background necessary for the subsequent development.

Markov Networks (MNs)

An undirected graph $\mathcal{G} = (V, E)$ consists of $n = |V|$ vertices, connected via edges $(v, w) \in E$. For two graphs $\mathcal{G}_1, \mathcal{G}_2$, we write $V(\mathcal{G}_1)$ and $V(\mathcal{G}_2)$ to denote the vertices of $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively and similar $E(\mathcal{G}_1)$ and $E(\mathcal{G}_2)$ for the edges. A clique $\mathcal{C}$ is a fully-connected subset of vertices, i.e., $\forall v, w \in \mathcal{C} : (v, w) \in E$. The set of all cliques of $\mathcal{G}$ is denoted by $\boldsymbol{\mathcal{C}}(\mathcal{G})$. Here, any undirected graph represents the conditional independence structure of an undirected graphical model or MN (Wainwright and Jordan 2008).

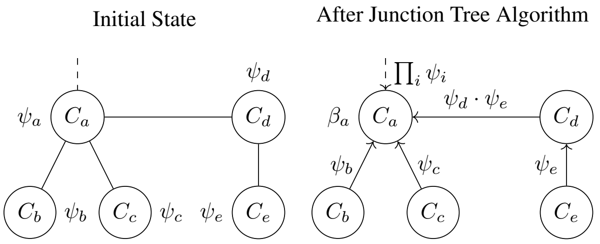
To this end, we identify each vertex $v \in V$ with a random variable X_v taking values in the state space $\mathcal{X}_v = \mathrm{Dom}(X_v)$. The random vector $\boldsymbol{X} = (X_v : v \in V)$, with probability mass function (pmf) $\mathbb{P}$, represents the random joint state of all vertices in some arbitrary but fixed order, taking values $\boldsymbol{x}$ in the Cartesian product space $\mathcal{X} = \mathrm{Dom}(\boldsymbol{X}) = \bigotimes_{v\in V} \mathcal{X}_v$. If not stated otherwise, $\mathcal{X}$ is a discrete set. Moreover, we allow to access these quantities for any proper subset of variables $S \subset V$, i.e., $\boldsymbol{X}_S = (X_v : v \in S)$, $\boldsymbol{x}_S$, and $\mathcal{X}_S = \bigotimes_{v\in S} \mathcal{X}_v$, respectively. We write $\omega(\mathcal{G})$ to indicate the treewidth of $\mathcal{G}$, i.e. $\omega(\mathcal{G}) = \max_{\mathcal{C}\in\boldsymbol{\mathcal{C}}(\mathcal{G})} |\mathcal{C}| - 1$.

According to the Hammersley-Clifford theorem (Hammersley and Clifford 1971), the probability mass of $\boldsymbol{X}$ factorizes over positive functions $\psi_\mathcal{C} : \mathcal{X} \to \mathbb{R}_+$, one for each maximal clique of the underlying graph,

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z} \prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \psi_\mathcal{C}(\boldsymbol{x}_\mathcal{C}) \ , \tag{2}$$

normalized via $Z = \sum_{\boldsymbol{x}\in\mathcal{X}} \prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \psi_\mathcal{C}(\boldsymbol{x}_\mathcal{C})$. Due to positivity of $\psi_\mathcal{C}$, it can be written as an exponential, i.e., $\psi_\mathcal{C}(\boldsymbol{x}_\mathcal{C}) = \exp(\langle \boldsymbol{\theta}_\mathcal{C}, \phi_\mathcal{C}(\boldsymbol{x}_\mathcal{C})\rangle)$ with sufficient statistic $\phi_\mathcal{C} : \mathcal{X}_\mathcal{C} \to \mathbb{R}^{|\mathcal{X}_\mathcal{C}|}$. The overcomplete sufficient statistic of discrete data is a "one-hot" vector that selects a specific weight value, e.g., $\psi_\mathcal{C}(\boldsymbol{x}_\mathcal{C}) = \exp(\boldsymbol{\theta}_{\mathcal{C}=\boldsymbol{x}_\mathcal{C}})$. The full joint can be written in the famous exponential family form $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle - \log Z)$ with $\boldsymbol{\theta} = (\boldsymbol{\theta}_\mathcal{C} : \mathcal{C} \in \boldsymbol{\mathcal{C}})$ and $\phi(\boldsymbol{x}) = (\phi_\mathcal{C}(\boldsymbol{x}_\mathcal{C}) : \mathcal{C} \in \boldsymbol{\mathcal{C}})$.

The parameters of exponential family members are estimated by minimizing the negative average log-likelihood $\ell(\boldsymbol{\theta}; \mathcal{D}) = -(1/|\mathcal{D}|) \sum_{\boldsymbol{x}\in\mathcal{D}} \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$ for some data set $\mathcal{D}$ via first-order numeric optimization methods. $\mathcal{D}$ contains samples from $\boldsymbol{X}$, and it can be shown that the estimated probability mass converges to the data generating distribution as the size of $\mathcal{D}$ increases. However, computing Z and hence performing probabilistic inference is #P-hard (Valiant 1979; Bulatov and Grohe 2004). There are approximation techniques for inference with quality guarantees (Piatkowski and Morik 2018), but for exact inference, the junction tree algorithm is needed. The junction tree representation of an undirected model is a tree, in which each vertex represents a maximal clique of a triangulation[1] of $\mathcal{G}$ (Wainwright and Jordan 2008, Sec. 2.5.2). The cutset of each pair of adjacent clique-vertices is called a separator.



$$\beta_a = \psi_a \sum_x \psi_b(x)\psi_c(x)\psi_d(x)\psi_e(x) \prod_i \psi_i(x)$$

Figure 1: Illustration of the Junction Tree Algorithm.

Nevertheless, junction trees require the underlying graphical structure of the graphical model to be *decomposable*.

Decomposable Models (DMs)

A *DM*, $\mathbb{P}_\mathcal{G}$, is a MN where the underlying conditional independence structure, $\mathcal{G}$, is a chordal graph [2].

DMs can be translated directly into an equivalent junction tree representation by finding the *maximum spanning tree* of its *clique graph* . Each vertex of the clique graph is a maximal clique in the DM and each edge is the separator between the vertex. The weight of each edge is then the number of variables in the corresponding separator. The resulting junction tree, $\mathcal{T} = (\boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{S}})$, will have vertices that are the maximal cliques, $\boldsymbol{\mathcal{C}}$, and edges that are the minimal separators, $\boldsymbol{\mathcal{S}}$, of the DM.

Beside allowing for fast inference, another benefit of a DM is that there is a closed form solution to the maximum likelihood parmeter estimation problem for the joint distribution over all the variables in the model (Haberman 1977). Therefore, the joint distribution for the DM $\mathbb{P}_\mathcal{G}$ is:

$$\mathbb{P}_\mathcal{G}(\boldsymbol{x}) = \frac{\prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \mathbb{P}_\mathcal{C}(\boldsymbol{x})}{\prod_{S\in\boldsymbol{\mathcal{S}}} \mathbb{P}_S(\boldsymbol{x})}$$

where $\mathbb{P}_d(\cdot)$ represents the marginal probability over $\mathcal{X}_d$.

Alternatively, we can also represent the joint distribution of $\mathbb{P}_\mathcal{G}$ as a product of conditional probability tables (CPTs) if we choose a maximal clique in $\boldsymbol{\mathcal{C}}$ to be the root node of $\mathbb{P}_\mathcal{G}$'s junction tree $\mathcal{T}$.

$$\mathbb{P}_\mathcal{G}(\boldsymbol{x}) = \prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \mathbb{P}_{\mathcal{C}-\mathrm{pa}(\mathcal{C})|\mathrm{pa}(\mathcal{C})}(\boldsymbol{x}_{\mathcal{C}-\mathrm{pa}(\mathcal{C})}|\boldsymbol{x}_{\mathrm{pa}(\mathcal{C})}) = \prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \mathbb{P}^\mathcal{T}_\mathcal{C}(\boldsymbol{x})$$

where $\mathbb{P}^\mathcal{T}_\mathcal{C}(\boldsymbol{x}) = \mathbb{P}_{\mathcal{C}-\mathrm{pa}(\mathcal{C})|\mathrm{pa}(\mathcal{C})}(\boldsymbol{x}_{\mathcal{C}-\mathrm{pa}(\mathcal{C})}|\boldsymbol{x}_{\mathrm{pa}(\mathcal{C})})$ and $\mathrm{pa}(\mathcal{C})$ is the parent clique of $\mathcal{C}$ in the junction tree $\mathcal{T}$. $\mathrm{pa}(\mathcal{C}) = \emptyset$ when $\mathcal{C}$ is the assigned root node of $\mathcal{T}$.

Junction Tree Algorithms (JTAs)

Recall the partition function of a general MN Z: $Z = \sum_{\boldsymbol{x}\in\mathcal{X}} \prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \psi_\mathcal{C}(\boldsymbol{x}_\mathcal{C})$. Evaluating the partition function of loopy models exactly does not necessarily require a naive

[1] A triangulation of a graph $\mathcal{G} = (V, E)$ is another graph $\mathcal{G}' = (V, E')$ with $E \subseteq E'$, such that $\mathcal{G}'$ is a chordal graph.

[2] A graph is cordial if every induced cycle has exactly 3 vertices.

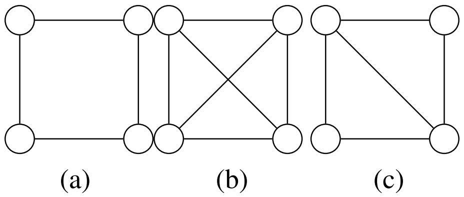


Figure 2: (a) a non-chordal MN, (b) a possible triangulation of this MN, (c) a minimal triangulation of this MN

summation over the state space $\mathcal{X}$; there is another, more efficient, technique. Any loopy graph can be triangulated and converted into a chordal graph with a junction tree representation. (Lauritzen and Spiegelhalter 1988; Wainwright and Jordan 2008). Then, as illustrated in Figure 1, the junction tree algorithm goes a step further and computes the *un-normalized* marginal "probability", β, for each maximal clique in the JT (Koller and Friedman 2009, Corollary 10.2):

$$\forall \mathcal{C} \in \boldsymbol{\mathcal{C}} : \beta_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-\mathcal{C}}} \prod_{\mathcal{C} \in \boldsymbol{\mathcal{C}}} \psi_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{x}_{\mathcal{C}})$$

As with belief propagation in ordinary trees, inference on the junction tree has a time complexity that is polynomial in the maximal state space size of its vertices. The maximal vertex state space size of a junction tree is, however, exponential in the treewidth of the triangulation of $\mathcal{G}$ used. Hence, if the treewidth of the triangulation of a loopy model is small, exact inference via the junction tree algorithm is rather efficient. Choosing a triangulation that results in a minimal treewidth is an **NP**-hard problem, but a valid triangulation can be found with time and memory complexity linear in the number of vertices (Dechter 2003; Berry et al. 2004; Heggernes 2006). See Figure 2 for an example of a non-minimal and minimal triangulation of a graph.

$\alpha\beta$-Divergence between DMs

A divergence is a measure of the "difference" between 2 probability distributions. More formally, a divergence is a function between 2 distributions as defined in Definition 2.

Definition 2. *(Divergence) Suppose S is the set of probability distributions with the same support. A divergence, D, is the function $D(\cdot \mid\mid \cdot) : S \times S \to \mathbb{R}$ such that $\forall \mathbb{P}, \mathbb{Q} \in S$: $D(\mathbb{P} \mid\mid \mathbb{Q}) \geq 0$ and $\mathbb{P} = \mathbb{Q} \Leftrightarrow D(\mathbb{P} \mid\mid \mathbb{Q}) = 0$* [3].

Furthermore, there are also generalized divergences where common divergences, such as the KL divergence, are special cases of the generalized divergence. Specifically, we will use the generalized divergence known as the $\alpha\beta$-divergence (Cichocki, Cruces, and Amari 2011).

Definition 3 ($\alpha\beta$-divergence)**.** *The $\alpha\beta$-divergence, D_{AB}, between 2 positive measures $\mathbb{P}$ and $\mathbb{Q}$ is defined by the following, where α and β are parameters:*

$$D_{AB}^{\alpha,\beta}(\mathbb{P}, \mathbb{Q}) = \sum_{\boldsymbol{x} \in \mathcal{X}} d_{AB}^{\alpha,\beta}\big(\mathbb{P}(\boldsymbol{x}), \mathbb{Q}(\boldsymbol{x})\big) \tag{3}$$

where (Cichocki, Cruces, and Amari 2011):

$$d_{AB}^{(\alpha,\beta)}(\mathbb{P}(\boldsymbol{x}), \mathbb{Q}(\boldsymbol{x})) \tag{4}$$
$$= \begin{cases} -\frac{1}{\alpha\beta}\left(\mathbb{P}(\boldsymbol{x})^{\alpha}\mathbb{Q}(\boldsymbol{x})^{\beta} - \frac{\alpha\mathbb{P}(\boldsymbol{x})^{\alpha+\beta}}{\alpha+\beta} - \frac{\beta\mathbb{Q}(\boldsymbol{x})^{\alpha+\beta}}{\alpha+\beta}\right) & \\ \qquad \textit{for } \alpha, \beta, \alpha+\beta \neq 0 & \\ \frac{1}{\alpha^2}\left(\mathbb{P}(\boldsymbol{x})^{\alpha} \log \frac{\mathbb{P}(\boldsymbol{x})^{\alpha}}{\mathbb{Q}(\boldsymbol{x})^{\alpha}} - \mathbb{P}(\boldsymbol{x})^{\alpha} + \mathbb{Q}(\boldsymbol{x})^{\alpha}\right) & \\ \qquad \textit{for } \alpha \neq 0, \beta = 0 & \\ \frac{1}{\alpha^2}\left(\log \frac{\mathbb{Q}(\boldsymbol{x})^{\alpha}}{\mathbb{P}(\boldsymbol{x})^{\alpha}} + \left(\frac{\mathbb{Q}(\boldsymbol{x})^{\alpha}}{\mathbb{P}(\boldsymbol{x})^{\alpha}}\right)^{-1} - 1\right) & \\ \qquad \textit{for } \alpha = -\beta \neq 0 & \\ \frac{1}{\beta^2}\left(\mathbb{Q}(\boldsymbol{x})^{\beta} \log \frac{\mathbb{Q}(\boldsymbol{x})^{\beta}}{\mathbb{P}(\boldsymbol{x})^{\beta}} - \mathbb{Q}(\boldsymbol{x})^{\beta} + \mathbb{P}(\boldsymbol{x})^{\beta}\right) & \\ \qquad \textit{for } \alpha = 0, \beta \neq 0 & \\ \frac{1}{2}(\log \mathbb{P}(\boldsymbol{x}) - \log \mathbb{Q}(\boldsymbol{x}))^2 & \textit{for } \alpha, \beta = 0. \end{cases}$$

The parameters α and β in the $\alpha\beta$-divergence is used to express other commonly used divergences. Specifically, the $\alpha = 1, \beta = 0$ gives the KL divergence, while the $\alpha = 0.5, \beta = 0.5$ gives the Bhattacharyya coefficient which immediately gives the Hellinger distance.

The expression of the $\alpha\beta$-divergence in Equation 4 can be expressed as a linear combination of 3 smaller functionals.

Theorem 1. *The 5 cases of the $\alpha\beta$-divergence in Equation 4 are linear combinations of the following 3 functionals:*

$$f_1(\mathbb{P}, \mathbb{Q}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2}(\log \mathbb{P}(\boldsymbol{x}) - \log \mathbb{Q}(\boldsymbol{x}))^2$$
$$f_2(\mathbb{P}, \mathbb{Q}; a, b) = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x})^a \mathbb{Q}(\boldsymbol{x})^b$$
$$f_3(\mathbb{P}, \mathbb{Q}; a, b, c, d) = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x})^a \mathbb{Q}(\boldsymbol{x})^b \log(\mathbb{P}(\boldsymbol{x})^c \mathbb{Q}(\boldsymbol{x})^d)$$

Therefore, the ability to tractably compute these functionals between 2 DMs will imply the ability to tractably compute the $\alpha\beta$-divergence between 2 DMs. Here we assume a complexity exponential to the treewidth of our DMs is *tractable*.

Theorem 2. *The time complexity for computing the functional f_1 directly between 2 DMs is $\mathcal{O}(n^2\omega 2^{\omega+1})$ where $\omega(\mathcal{G})$ is the treewidth of chordal graph $\mathcal{G}$, ω = max$(\omega(\mathcal{G}_{\mathbb{P}}), \omega(\mathcal{G}_{\mathbb{Q}}))$, and n is the number of variables.*

Since computing f_1 directly is tractable, the focus of the rest of this paper will be to show how to compute functionals f_2 and f_3 between 2 DMs. In order to simplify further exposition, it will be ideal if functionals f_2 and f_3 can be expressed by a single, more general, functional.

Theorem 3. *f_2 can be expressed by functional $\mathcal{F}$.*

Theorem 4. *f_3 can be expressed by functional $\mathcal{F}$.*

Therefore, any method that can tractably compute $\mathcal{F}$, as defined in Definition 1, between 2 DMs can also tractably compute the $\alpha\beta$-divergence between these models.

[3] Some authors also require that the quadratic part of the Taylor expansion of $D(p, p+dp)$ define a Riemannian metric on S (Amari 2016). However, this requirement is not needed by the methods described in this paper and is therefore left out.

With reasoning for the definition of $\mathcal{F}$ established, we can now substitute the maximum likelihood estimator of DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ into functional $\mathcal{F}$. But before we start, first recall the notation established in Section :

$$\mathbb{P}(\boldsymbol{x}) = \prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \mathbb{P}\left(\boldsymbol{x}_{\mathcal{C}-\text{pa}(\mathcal{C})} \mid \boldsymbol{x}_{\text{pa}(\mathcal{C})}\right) = \prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \mathbb{P}^{\mathcal{T}}_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}})$$
$$\text{where, for } \mathcal{C}\subset X : \mathbb{P}^{\mathcal{T}}_{\mathcal{C}}\left(\boldsymbol{x}_X\right) = \mathbb{P}^{\mathcal{T}}_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}})$$

and pa$(\mathcal{C})$ is the parent of the maximal clique $\mathcal{C}$ in the junction tree of $\mathbb{P}$'s and $\mathbb{Q}$'s respective chordal graph. Then continuing with the substitution we get:

$$\begin{aligned}
&\mathcal{F}(\mathbb{P},\mathbb{Q};g,h,g^*,h^*) \qquad (5)\\
&= \sum_{\boldsymbol{x}\in\mathcal{X}} \big(g\left[\mathbb{P}\right](\boldsymbol{x})\big)\big(h\left[\mathbb{Q}\right](\boldsymbol{x})\big) L\left(\big(g^*\left[\mathbb{P}\right](\boldsymbol{x})\big)\big(h^*\left[\mathbb{Q}\right](\boldsymbol{x})\big)\right)\\
&= \left[\sum_{\mathcal{C}\in\boldsymbol{\mathcal{C}}_{\mathbb{P}}}\sum_{\boldsymbol{x}\in\mathcal{X}} L\left(g^*\left[\mathbb{P}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x})\right)\big(g\left[\mathbb{P}\right](\boldsymbol{x})\big)\big(h\left[\mathbb{Q}\right](\boldsymbol{x})\big)\right] +\\
&\quad \left[\sum_{\mathcal{C}\in\boldsymbol{\mathcal{C}}_{\mathbb{Q}}}\sum_{\boldsymbol{x}\in\mathcal{X}} L\left(h^*\left[\mathbb{Q}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x})\right)\big(g\left[\mathbb{P}\right](\boldsymbol{x})\big)\big(h\left[\mathbb{Q}\right](\boldsymbol{x})\big)\right]\\
&= \sum_{\mathcal{C}\in\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}})}\sum_{\boldsymbol{x}_{\mathcal{C}}\in\mathcal{X}_{\mathcal{C}}} L\left(g^*\left[\mathbb{P}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x}_{\mathcal{C}})\right) SP_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}) +\\
&\quad \sum_{\mathcal{C}\in\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}})}\sum_{\boldsymbol{x}_{\mathcal{C}}\in\mathcal{X}_{\mathcal{C}}} L\left(h^*\left[\mathbb{Q}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x}_{\mathcal{C}})\right) SP_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}})
\end{aligned}$$

where, for ease of notation:

$$\begin{aligned}
&SP_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}})\\
&= \sum_{\boldsymbol{x}\in\mathcal{X}_{X-\mathcal{C}}} \Big(g\left[\mathbb{P}\right](\boldsymbol{x}_{\mathcal{C}},\boldsymbol{x})\Big)\Big(h\left[\mathbb{Q}\right](\boldsymbol{x}_{\mathcal{C}},\boldsymbol{x})\Big) \qquad (6)\\
&= \sum_{\boldsymbol{x}\in\mathcal{X}_{X-\mathcal{C}}} \left[\prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}_{\mathbb{P}}} g\left[\mathbb{P}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x})\right]\left[\prod_{\mathcal{C}\in\boldsymbol{\mathcal{C}}_{\mathbb{Q}}} h\left[\mathbb{Q}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x})\right]
\end{aligned}$$

which represents the marginalisation of all the variables that are not in the clique $\mathcal{C}$ over the all the non-log factors produced by $\mathcal{F}$. The equality in Equation 5 holds mainly due to the associativity of summations.

Remark 1. *The lower bound complexity of directly computing Equation 5 is $\Omega(2^n)$ where n is the number of variables. Therefore directly computing the functional $\mathcal{F}$ between 2 DMs is intractable.*

Consequently, in order to compute $\mathcal{F}(\mathbb{P},\mathbb{Q})$, and therefore $D_{\text{AB}}(\mathbb{P},\mathbb{Q})$, while avoiding complexity exponential to n, we require a more sophisticated method for its computation.

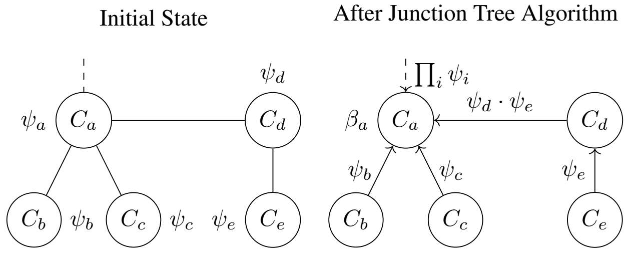
Computing Functional $\mathcal{F}$ between DMs

In order to tractably compute $\mathcal{F}$ between DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$, and therefore the $\alpha\beta$-divergence, we first require knowledge of a *computation graph* between DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$.

Definition 4 (strictly larger, clique mapping α)**.** *A chordal graph $\mathcal{H}$ is strictly larger than chordal graphs $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$ if all the maximal cliques in both chordal graphs is either a subset or equal to a maximal clique in $\mathcal{H}$. In other words, $\mathcal{H}$ is strictly larger than $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$ if and only if there exists a mapping α such that:*

$$\begin{aligned}
&\alpha : \boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}},\mathcal{G}_{\mathbb{Q}}) \rightarrow \boldsymbol{\mathcal{C}}(\mathcal{H})\\
&\textit{s.t.}\quad \forall\mathcal{C}\in\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}},\mathcal{G}_{\mathbb{Q}}) : \mathcal{C}\subseteq\alpha(\mathcal{C})
\end{aligned}$$

where $\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}},\mathcal{G}_{\mathbb{Q}}) = \boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}})\cup\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}})$ is the set of maximal cliques in chordal graphs $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$.



$$\psi_i = \prod_{C_{\mathbb{P}}\in A_i(\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}}))} g\left(\mathbb{P}^{\mathcal{T}}_{C_{\mathbb{P}}}\right) \prod_{C_{\mathbb{Q}}\in A_i(\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}}))} h\left(\mathbb{Q}^{\mathcal{T}}_{C_{\mathbb{Q}}}\right)$$
$$\text{where} : A_i(\boldsymbol{\mathcal{C}}) = \{\mathcal{C} : \mathcal{C}\in\boldsymbol{\mathcal{C}} \wedge \alpha(\mathcal{C}) == \mathcal{C}_i\}$$
$$\beta_a = \psi_a \sum_{\boldsymbol{x}\in\mathcal{X}} \psi_b(\boldsymbol{x})\psi_c(\boldsymbol{x})\psi_d(\boldsymbol{x})\psi_e(\boldsymbol{x}) \prod_i \psi_i(\boldsymbol{x})$$

Figure 3: Junction Tree Algorithm to compute the functional $\mathcal{F}$ between 2 DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ using computation graph $\mathcal{H}$, assuming $\mathcal{H}$ is a connected graph.

Definition 5 (computation graph)**.** *If a chordal graph, $\mathcal{H}$, is strictly larger than chordal graphs $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$, then $\mathcal{H}$ is a computation graph of DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$.*

We can obtain the computation graph $\mathcal{H}$ by first taking the graph union of $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$, and then triangulating $\mathcal{G}_{\mathbb{P}}\cup\mathcal{G}_{\mathbb{Q}}$.

For the rest of this section, we will provide details on how our method, Junction Forest Computation (JFComp), uses the *junction tree algorithm* to compute the functional $\mathcal{F}$ between 2 DMs. We will first describe how JFComp works on a connected computation graph $\mathcal{H}$ before generalising this to cases when $\mathcal{H}$ is a disconnected graph.

Junction Tree Computation (JTComp)

Recall we want to compute $\mathcal{F}(\mathbb{P},\mathbb{Q})$ as expressed in Equation 5. Observe that the 2 nested sums in $\mathcal{F}(\mathbb{P},\mathbb{Q})$ has innermost sums over $\mathcal{X}_{\mathcal{C}}$ with similar forms but over different sets of maximal cliques, $\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}})$ and $\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}})$ respectively. Therefore, we want to re-express $\mathcal{F}(\mathbb{P},\mathbb{Q})$ such that the two sums over $\mathcal{X}_{\mathcal{C}}$ are over the same set of maximal cliques, $\boldsymbol{\mathcal{C}}(\mathcal{H})$.

Theorem 5. *Assume we have 2 DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ and a computation graph $\mathcal{H}$ for both models. By Definition 4, we also have a mapping α from maximal cliques in $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ to maximal cliques in $\mathcal{H}$. Then the following equivalences holds for $\mathbb{D}\in\{\mathbb{P},\mathbb{Q}\}$:*

$$\begin{aligned}
&\sum_{\mathcal{C}\in\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{D}})}\sum_{\boldsymbol{x}_{\mathcal{C}}\in\mathcal{X}_{\mathcal{C}}} L\left(g^*\left[\mathbb{D}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x}_{\mathcal{C}})\right) SP_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}) \qquad (7)\\
&= \sum_{\mathcal{C}\in\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{D}})}\sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{C})}\in\\ \mathcal{X}_{\alpha(\mathcal{C})}}} L\left(g^*\left[\mathbb{D}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x}_{\alpha(\mathcal{C})})\right) SP_{\alpha(\mathcal{C})}(\boldsymbol{x}_{\alpha(\mathcal{C})})
\end{aligned}$$

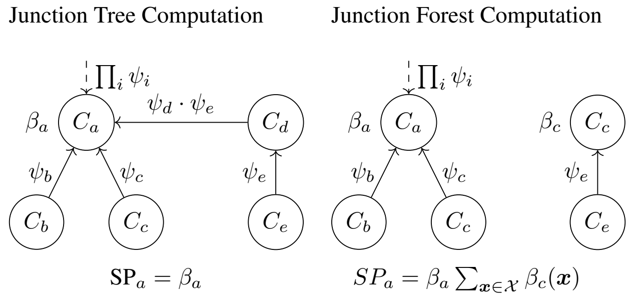


Figure 4: Differences in getting the clique beliefs over each maximal clique in the computation graph $\mathcal{H}$ between a connected and a disconnected computation graph

With that, we will now show how the computation of $SP_{\alpha(\mathcal{C})}, \forall \mathcal{C} \in \boldsymbol{\mathcal{C}}(\mathcal{H})$ is equivalent to the running the *junction tree algorithm* over the junction tree of $\mathcal{H}$ with a set of specific initial factors. Figure 3 shows an illustration of this procedure that we will now describe.

Theorem 6. *Given 2 DMs, $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, and a computation graph of both models, $\mathcal{H}$. Let Ψ be a set of factors defined as follows:*

$$\Psi := \left\{ g \circ \mathbb{P}^{\mathcal{T}}_{\mathcal{C}_\mathbb{P}} : \mathcal{C}_\mathbb{P} \in \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{P}) \right\} \bigcup \left\{ h \circ \mathbb{Q}^{\mathcal{T}}_{\mathcal{C}_\mathbb{Q}} : \mathcal{C}_\mathbb{Q} \in \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{Q}) \right\}$$

After running the junction tree algorithm over the junction tree of $\mathcal{H}$ with factors Ψ, we will get the following beliefs over each maximal clique in $\mathcal{H}$:

$$\forall \mathcal{C} \in \boldsymbol{\mathcal{C}}(\mathcal{H}) : \beta_\mathcal{C}(\boldsymbol{x}_\mathcal{C}) = SP_\mathcal{C}(\boldsymbol{x}_\mathcal{C})$$

Therefore, using the junction tree algorithm, we obtain beliefs over each maximal clique in $\mathcal{H}$ that we can use to substitute for *SP* in Equation 5. However, this assumes that $\mathcal{H}$ is a connected graph, which might not always be the case.

Junction Forest Computation (JFComp)

We now show how JFComp can be extended to handle cases where $\mathcal{H}$ is disconnected.

When the computation graph $\mathcal{H}$ is disconnected, $\mathcal{H}$ can be represented as a list of chordal graphs, $\mathcal{H} = \{\mathcal{H}_i\}$. Therefore, we also have a list of clique trees for each chordal graph in $\mathcal{H}$, $\mathcal{T} = \{\mathcal{T}_i\}$, as well.

Since the $\mathcal{H}$ is still strictly larger than the chordal graph structure of $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, by Definition 4, there is still a mapping α from $\boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{P}) \cup \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{Q})$ to $\boldsymbol{\mathcal{C}}(\mathcal{H})$. However, since chordal graph $\mathcal{H}$ is now comprised of multiple chordal graphs, and therefore clique trees, we are unable to apply Theorem 6 directly to compute *SP* from Equation 6. The reason for this is because there is no single clique tree to run the junction tree algorithm on, therefore factors from different clique trees are unable to propagate to each other.

Instead, we show in Theorem 7, that having a disconnected computation graph $\mathcal{H}$, and therefore a set of clique trees, $\mathcal{T}$, which are disconnected from each other, essentially breaks up *SP* into smaller sub-problems over each clique tree in $\mathcal{T}$. The results of these sub-problems can then be combined via multiplication to compute *SP*. An illustration of the result from Theorem 7 and its difference in computing *SP* on a connected $\mathcal{H}$ can be found in Figure 4.

Definition 6 (τ, clique to clique tree mapping)**.** *Let τ be a mapping from the maximal cliques of $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ to a clique tree in $\mathcal{T}$ that contains the maximal clique given by the clique mapping, α, from Definition 4:*

$$\begin{aligned} &\tau : \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{P}) \cup \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{Q}) \to \mathcal{T} \\ \textit{s.t.} \quad &\forall \mathcal{C} \in \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{P}) \cup \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{Q}) : \alpha(\mathcal{C}) \in \boldsymbol{\mathcal{C}}(\tau(\mathcal{C})) \end{aligned}$$

Theorem 7. *If the computation graph $\mathcal{H}$ for DMs $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ is disconnected, $SP_{\alpha(\mathcal{C})}$ in Equation 7 can be re-expressed as follows:*

$$\begin{aligned} &SP_{\alpha(\mathcal{C})}(\boldsymbol{x}_{\alpha(\mathcal{C})}) \\ &= \beta_{\alpha(\mathcal{C})}(\boldsymbol{x}_{\alpha(\mathcal{C})}) \prod_{\mathcal{T}_i \in \mathcal{T} - \tau(\mathcal{C})} \sum_{\boldsymbol{x} \in \mathcal{X}_{\mathcal{C}(\mathcal{T}_i)}} \beta_{\mathcal{C}(\mathcal{T}_i)}(\boldsymbol{x}_{\alpha(\mathcal{C})}, \boldsymbol{x}) \\ &= \beta_{\alpha(\mathcal{C})}(\boldsymbol{x}_{\alpha(\mathcal{C})}) R_{\tau(\mathcal{C})} \end{aligned}$$

where $\mathcal{C}(\mathcal{T}_i)$ represents any clique in the set of maximal cliques in clique tree $\mathcal{T}_i$ and $R_{\tau(\mathcal{C})} \in \mathbb{R}$.

Then we can obtain the required beliefs by running the junction tree algorithm for each junction tree in $\mathcal{T}$ separately. Therefore, even if the computation graph $\mathcal{H}$ is disconnected, we can compute *SP* and thus $\mathcal{F}(\mathbb{P}, \mathbb{Q})$. Substituting these beliefs back into Equation 5 we get:

$$\begin{aligned} &\mathcal{F}(\mathbb{P}, \mathbb{Q}) \qquad (8) \\ &= \sum_{\mathcal{C} \in \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{P})} R_{\tau(\mathcal{C})} \sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{C})} \in \\ \mathcal{X}_{\alpha(\mathcal{C})}}} L\left(g^* \left[\mathbb{P}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x}_{\alpha(\mathcal{C})})\right) \beta_{\alpha(\mathcal{C})}(\boldsymbol{x}_{\alpha(\mathcal{C})}) \\ &+ \sum_{\mathcal{C} \in \boldsymbol{\mathcal{C}}(\mathcal{G}_\mathbb{Q})} R_{\tau(\mathcal{C})} \sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{C})} \in \\ \mathcal{X}_{\alpha(\mathcal{C})}}} L\left(h^* \left[\mathbb{Q}^{\mathcal{T}}_{\mathcal{C}}\right](\boldsymbol{x}_{\alpha(\mathcal{C})})\right) \beta_{\alpha(\mathcal{C})}(\boldsymbol{x}_{\alpha(\mathcal{C})}) \end{aligned}$$

Section will show that the complexity of computing the expression in Equation 8, and in general, that the complexity of JFComp is more efficient than computing $\mathcal{F}$ directly.

Computational Complexity

We can determine the computational complexity of computing the $\alpha\beta$-divergence between 2 DMs by first checking what the given values for α and β are. This step takes $\mathcal{O}(1)$ time. When $\alpha, \beta = 0$, from Theorem 2 we know that the complexity of computing $D^{0,0}_{AB}(\mathbb{P} \mid\mid \mathbb{Q})$ is:

$$D^{0,0}_{AB}(\mathbb{P}, \mathbb{Q}) \in \mathcal{O}(n^2 \omega 2^{\omega+1})$$

where $\omega(\mathcal{G})$ is the treewidth of chordal graph $\mathcal{G}$ and $\omega = \max(\omega(\mathcal{G}_\mathbb{P}), \omega(\mathcal{G}_\mathbb{Q}))$.

When α and β takes values other than 0, we require the use of JFComp to compute parameterisations of $\mathcal{F}$ between $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$. In general, the $\alpha\beta$-divergence is a linear combination of different parameterisations of $\mathcal{F}$. Therefore, the complexity of computing the $\alpha\beta$-divergence is equivalent to computing $\mathcal{F}$ in big-O notation. As such, for the remainder of this section, we will discuss the overall complexity of JFComp for computing $\mathcal{F}$ between 2 DMs.

The first step of JFComp involves assigning factors constructed from the CPTs over $\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}})$ and $\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}})$ to $\boldsymbol{\mathcal{C}}(\mathcal{H})$ Therefore, for each factor ψ, and therefore for each $\mathcal{C} \in \boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}}) \cup \boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}})$, we need to search through $\boldsymbol{\mathcal{C}}(\mathcal{H})$ to find a suitable clique to assign ψ to. This results in the complexity:

$$\mathcal{O}(|\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}})| \cdot |\boldsymbol{\mathcal{C}}(\mathcal{H})|) + \mathcal{O}(|\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}})| \cdot |\boldsymbol{\mathcal{C}}(\mathcal{H})|) \in \mathcal{O}(n^2)$$

since the number of maximal cliques in any chordal graph is bounded by the number of vertices in the graph.

Once all ψs have been assigned to a maximal clique in $\mathcal{H}$, we then run the junction tree algorithm to calibrate the clique tree(s) of $\mathcal{H}$ with these factors. The complexity of this is:

$$\mathcal{O}(|\boldsymbol{\mathcal{C}}(\mathcal{H})| \cdot 2^{\omega(\mathcal{H})+1}) \in \mathcal{O}(n \cdot 2^{\omega(\mathcal{H})+1})$$

Once the clique tree/forest is calibrated and we know $\beta_{\alpha(\mathcal{C})}$ for all $\mathcal{C} \in \boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}}) \cup \boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}})$, we can then compute Equation 8:

$$\begin{aligned}
&\mathcal{F}(\mathbb{P}, \mathbb{Q}) \\
&= \sum_{\mathcal{C} \in \boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}})} R_{\tau(\mathcal{C})} \sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{C})} \in \\ \mathcal{X}_{\alpha(\mathcal{C})}}} L\left(g^* \left[\mathbb{P}_{\mathcal{C}}^{\mathcal{T}}\right] (\boldsymbol{x}_{\alpha(\mathcal{C})})\right) \beta_{\alpha(\mathcal{C})}(\boldsymbol{x}_{\alpha(\mathcal{C})}) \\
&+ \sum_{C \in \boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}})} R_{\tau(\mathcal{C})} \sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{C})} \in \\ \mathcal{X}_{\alpha(\mathcal{C})}}} L\left(h^* \left[\mathbb{Q}_{\mathcal{C}}^{\mathcal{T}}\right] (\boldsymbol{x}_{\alpha(\mathcal{C})})\right) \beta_{\alpha(\mathcal{C})}(\boldsymbol{x}_{\alpha(\mathcal{C})}) \\
&\in \mathcal{O}(\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{P}}) \cdot 2^{\omega(\mathcal{H})+1}) + \mathcal{O}(\boldsymbol{\mathcal{C}}(\mathcal{G}_{\mathbb{Q}}) \cdot 2^{\omega(\mathcal{H})+1}) \\
&\in \mathcal{O}(n \cdot 2^{\omega(\mathcal{H})+1})
\end{aligned}$$

Adding up the computational complexity of each step in JFComp results in the final complexity of computing the functional $\mathcal{F}$ between 2 DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$:

$$\begin{aligned}
&\mathcal{O}(n^2) + \mathcal{O}(n \cdot 2^{\omega(\mathcal{H})+1}) + \mathcal{O}(n \cdot 2^{\omega(\mathcal{H})+1}) \\
&\in \mathcal{O}(n \cdot 2^{\omega(\mathcal{H})+1})
\end{aligned}$$

which is more efficient than $\Omega(2^n)$, the complexity of computing $\mathcal{F}$ directly.

Therefore, the computational complexity of computing the $\alpha\beta$-divergence between $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ is:

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P} \,||\, \mathbb{Q}) \in \begin{cases} \mathcal{O}(n^2 \cdot \omega 2^{\omega+1}) & \alpha, \beta = 0 \\ \mathcal{O}(n \cdot 2^{\omega(\mathcal{H})+1}) & \text{otherwise} \end{cases}$$

Runtime Comparison with mcgo

Recall that a method already exists for computing the KL divergence between 2 BNs (Moral, Cano, and Gómez-Olmedo 2021) which we will refer to as mcgo. Also note that it is possible to take a distribution represented by a BN and, in exchange for some loss in independence information, represent it using a DM instead (Koller and Friedman 2009, p.p. 134). Therefore, one might ask, how does the practical runtime of JFComp compare to mcgo when computing the KL divergence between 2 BNs.

To answer this question, we will replicate the experiment used by Moral, Cano, and Gómez-Olmedo. They chose a set of BNs from the *bnlearn* (Scutari 2010) repository (https://www.bnlearn.com/bnrepository/) to sample from and estimated a second BN from these samples. The authors have provided these *estimated* BNs for each of the BN from *bnlearn* used in their experiments: https://github.com/mgomez-olmedo/KL-pgmpy. Therefore, we will use this set of BNs from their repository in our own experiments.

Network	mcgo (secs)		JFComp (secs)	
	mean	sd	mean	sd
cancer	**0.0117**	0.0026	0.0132	0.0033
earthquake	0.0104	0.0025	**0.0075**	0.0009
survey	0.0140	0.0032	**0.0081**	0.0002
asia	0.0163	0.0001	**0.0137**	0.0007
sachs	0.0464	0.0106	**0.0151**	0.0001
child	0.0778	0.0101	**0.0402**	0.0013
insurance	0.3838	0.0051	**0.1590**	0.0029
water	**6.9326**	0.0329	7.6454	0.0637
mildew	**19.326**	0.1318	19.459	0.0852
alarm	0.3177	0.0099	**0.0875**	0.0018
hailfinder	0.8543	0.0243	**0.1672**	0.0052
hepar2	1.3058	0.0307	**0.2403**	0.0140
win95pts	1.0256	0.0289	**0.3538**	0.0049

Table 1: Mean runtimes in seconds and their standard deviation for mcgo and JFComp on computing the KL divergence between 2 BN. The lower the better. Fastest times are bold.

Now that we have multiple pairs of BNs, one original and one estimated from samples, we then compute the KL divergence between each BN pair using both mcgo and JFComp. We repeat this 10 times in order to get an estimate of both methods' runtime in seconds. We also do not factor in the conversion of these BNs into DMs in the final runtime.

We run the experiments on an Intel NUC-10i7FNH with 64GB of RAM. The implementation of both methods are in Python and use the pgmpy library (Ankan and Panda 2015). The repository for the implementation for JFComp can be found at: https://lklee.dev/pub/2023-aaai/code

From the results in Table 1, we can observe that despite mcgo containing numerous computation optimisations, our direct application of belief propagation to carry out the computation has a practical runtime that is comparable to mcgo. Furthermore, on some networks, JFComp is faster than mcgo, probably due to having a lower overhead and being better able to leverage the optimized code in the pgmpy library for the bulk of the computation.

Case Study in Model Selection

Although allowing for a simpler implementation that can leverage existing library implementations of the junction tree algorithm for most of the computation is a satisfactory result by itself, recall that the original motivation of JFComp is to compute a wider range of divergences between graphical models. Therefore, in order to motivate the need of using divergences other than the KL divergence, we now present a case study on the application of computing divergences between BNs for the problem of model selection, a problem that the KL divergence is normally well suited for.

Consider a scientist who, in an attempt to model a natural phenomenon that they have samples from, constructs 2 can-

run	Kullback-Leibler		Hellinger	
	$A \mid\mid E$	$B \mid\mid E$	A, E	B, E
1	**0.4021**	0.5169	0.3027	**0.2915**
2	**0.3993**	0.5182	0.3009	**0.2895**
3	**0.3979**	0.5234	0.3014	**0.2906**
4	**0.4018**	0.5219	0.3022	**0.2904**
5	**0.3996**	0.5275	0.3018	**0.2908**

Table 2: Divergence between the candidate models and a Bayesian network estimated from randomly sampled datasets of size 10000. Lower numbers indicate a better fit and are bold.

	sachs$\mid\mid A$	sachs$\mid\mid B$
Kullback-Leibler	0.3687	**0.3090**
Hellinger	0.3013	**0.2921**

Table 3: Divergence between the candidate models and the original Bayesian network sachs. Lower numbers indicate a better fit. Lowest divergences are bold.

didate BNs, A and B. They then wish to determine, using the samples, which candidate model is a better representation of the phenomenon they wish to model. One way to do this, is to estimate a new BN, E, from the samples and compute the divergence between E and the candidate models.

In order to recreate this scenario synthetically, we use the BN sachs from the bnlearn repository (Scutari 2010) as the "phenomenon" the scientist wishes to model. The scientist's "candidate models" are then constructed by removing edges from sachs and marginalising the CPTs according to (Choi, Chan, and Darwiche 2005). Further details regarding the construction of A and B can be found in Appendix I of the extended version of this paper.

Sampling 100000 samples from sachs, we then learn BN E from these samples using the constraint-based structure learner in *pgmpy* and *maximum likelihood estimation* with Laplace smoothing for learning the parameters of E. The use of a smoothing technique is to ensure that the KL divergence is defined. We then compute the Hellinger and KL divergence between the candidate models and the estimated model: D(A || E) and D(B || E). We repeat the experiment 5 times, with different random samples from sachs.

From the results in Table 2, we can observe that the KL divergence indicates that A is the BN closest to E and that the scientist should choose A, while the Hellinger distance indicates the opposite, choosing B instead. With this discrepancy, the question then is, which candidate model, A or B, is actually the closer approximation to the actual phenomenon, and therefore, which divergence is "correct".

Since, for the purpose of this case study, we already have the true model of the "phenomenon" we are modelling, we can just compute the divergence between our candidate models and sachs to get an answer. From Table 3, we can observe that when computing the divergence between sachs and the candidate models, both divergences agree that B is closer to sachs. Consequently, in our case study, our scientist would have chosen the incorrect model if they only used the KL divergence in their experiment.

Of course, it might be possible to avoid such a scenario if a different smoothing technique is used to learn the parameters of E. However, the use of multiple divergences is still needed in order for the scientist to even be aware of possible issues in the smoothing technique used in the first place. In general, the main takeaway from this example should be that, one must not be over-reliant on just a single divergence, and that the use of a wide array of divergences can be helpful in avoiding mistakes in model selection.

Conclusion

In conclusion, we showed how computing the functional $\mathcal{F}$, and therefore the $\alpha\beta$-divergence, between 2 DMs is equivalent to belief propagation on a junction tree/forest with a set of specific initial factors defined based on how the MLE of DMs decomposes the functional $\mathcal{F}$. The result is a method with complexity exponential to the treewidth of the computation graph $\mathcal{H}$ of these models. Therefore, the proposed method is more efficient than computing the $\mathcal{F}$ between the DMs directly unless $\mathcal{H}$ is a fully saturated graph.

One advantage of JFComp is that it can be easily implemented in any environment that has a pre-existing implementation of the junction tree algorithm. Furthermore, since JFComp can compute the general functional $\mathcal{F}$ between 2 DMs, it can compute, or approximate, other divergences or functionals, and not just the $\alpha\beta$-divergence.

However, recall that in order to obtain the computation graph $\mathcal{H}$, we take the graph union of the two DMs we wish to compute the divergence between, and triangulate the resulting graph union to form a chordal computation graph. Therefore, one potential area of concern is the possibility for the triangulation step to produce a computation graph $\mathcal{H}$ that has a large treewidth. Due to the complexity being exponential to the treewidth of $\mathcal{H}$, this will result in the exact value of $\mathcal{F}$ taking a long time to compute. Therefore, one avenue of further research is doing away with the requirement that the computation graph $\mathcal{H}$ has to be a chordal graph in exchange for an approximation of $\mathcal{F}$ instead of an exact computation. In principle, this can be done by not triangulating $\mathcal{G}_{\mathbb{P}} \cup \mathcal{G}_{\mathbb{Q}}$, and instead running approximate inference algorithms on the graph $\mathcal{G}_{\mathbb{P}} \cup \mathcal{G}_{\mathbb{Q}}$ using the set of factors Ψ defined in Theorem 6.

Furthermore, throughout this work, we only considered estimating the divergence between 2 discrete distributions. Therefore, more work is needed to investigate how one might extend this approach for divergence estimation to numeric or even mixed type data.

Additionally, in this work we were only concerned with computing the divergence between the joint distributions of 2 DMs. However, in practice, it is common to encounter situations where one might want to compute the divergence between 2 conditional distributions. Therefore, more work is needed to investigate this particular problem either by extending the current work or via some new method that only draw inspiration from the current work.

Acknowledgments

This research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence, LAMARR22B, the Australian Research Council under award DP210100072, as well as the Australian Government Research Training Program (RTP) Scholarship.

References

Abdullah, A.; Kumar, R.; McGregor, A.; Vassilvitskii, S.; and Venkatasubramanian, S. 2016. Sketching, Embedding and Dimensionality Reduction in Information Theoretic Spaces. In *Artificial Intelligence and Statistics*, 948–956.

Amari, S.-i. 2016. *Information Geometry and Its Applications*. Springer. ISBN 978-4-431-55978-8.

Ankan, A.; and Panda, A. 2015. Pgmpy: Probabilistic Graphical Models Using Python. *Proceedings of the 14th Python in Science Conference*, 6–11.

Barz, B.; Rodner, E.; Garcia, Y. G.; and Denzler, J. 2019. Detecting Regions of Maximal Divergence for Spatio-Temporal Anomaly Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(5): 1088–1101.

Berry, A.; Blair, S. J. R.; Heggernes, P.; and Peyton, W. B. 2004. Maximum Cardinality Search for Computing Minimal Triangulations of Graphs. *Algorithmica*, 39(4): 287–298.

Bhattacharya, A.; Kar, P.; and Pal, M. 2009. On Low Distortion Embeddings of Statistical Distance Measures into Low Dimensional Spaces. In Bhowmick, S. S.; Küng, J.; and Wagner, R., eds., *Database and Expert Systems Applications*, Lecture Notes in Computer Science, 164–172. Berlin, Heidelberg: Springer. ISBN 978-3-642-03573-9.

Bulatov, A.; and Grohe, M. 2004. The Complexity of Partition Functions. In *Automata, Languages and Programming*, volume 3142 of *Lecture Notes in Computer Science*, 294–306. Heidelberg, Germany: Springer.

Chen, L.; Tao, C.; Zhang, R.; Henao, R.; and Carin, L. 2018. Variational Inference and Model Selection with Generalized Evidence Bounds. In *International Conference on Machine Learning (ICML)*, 892–901.

Chen, Y.; Ye, J.; and Li, J. 2020. Aggregated Wasserstein Distance and State Registration for Hidden Markov Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9): 2133–2147.

Choi, A.; Chan, H.; and Darwiche, A. 2005. On Bayesian Network Approximation by Edge Deletion. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, 128–135. Arlington, Virginia, USA: AUAI Press. ISBN 978-0-9749039-1-0.

Cichocki, A.; Cruces, S.; and Amari, S.-i. 2011. Generalized Alpha-Beta Divergences and Their Application to Robust Nonnegative Matrix Factorization. *Entropy*, 13(1): 134–170.

Dechter, R. 2003. *Constraint processing*. Elsevier Morgan Kaufmann. ISBN 978-1-55860-890-0.

Dieng, A. B.; Tran, D.; Ranganath, R.; Paisley, J. W.; and Blei, D. M. 2017. Variational Inference via χ Upper Bound Minimization. In *Advances in Neural Information Processing Systems*, 2732–2741.

Genevay, A.; Peyre, G.; and Cuturi, M. 2018. Learning Generative Models with Sinkhorn Divergences. In Storkey, A.; and Perez-Cruz, F., eds., *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84 of *Proceedings of Machine Learning Research*, 1608–1617. PMLR.

Haberman, S. J. 1977. *The Analysis of Frequency Data*. University of Chicago Press. ISBN 978-0-226-31185-2.

Hammersley, J. M.; and Clifford, P. 1971. Markov fields on finite graphs and lattices. *Unpublished manuscript*.

Heggernes, P. 2006. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3): 297–317.

Koller, D.; and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press. ISBN 978-0-262-01319-2.

Labeau, M.; and Cohen, S. B. 2019. Experimenting with Power Divergences for Language Modeling. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *Empirical Methods in Natural Language Processing (EMNLP)*, 4102–4112. Association for Computational Linguistics.

Lauritzen, S. L.; and Spiegelhalter, D. J. 1988. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2): 157–224.

Li, Y.; and Turner, R. E. 2016. Rényi Divergence Variational Inference. In *Advances in Neural Information Processing Systems*, 1073–1081.

Moral, S.; Cano, A.; and Gómez-Olmedo, M. 2021. Computation of Kullback–Leibler Divergence in Bayesian Networks. *Entropy*, 23(9): 1122.

Nowozin, S.; Cseke, B.; and Tomioka, R. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Piatkowski, N.; Lee, S.; and Morik, K. 2013. Spatio-Temporal Random Fields: Compressible Representation and Distributed Estimation. *Machine Learning*, 93(1): 115–139.

Piatkowski, N.; and Morik, K. 2018. Fast Stochastic Quadrature for Approximate Maximum-Likelihood Estimation. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 715–724.

Schlimmer, J. C.; and Granger, R. H. 1986. Incremental Learning from Noisy Data. *Machine Learning*, 1(3): 317–354.

Scutari, M. 2010. Learning Bayesian Networks with the Bnlearn R Package. *Journal of Statistical Software*, 35: 1–22.

Valiant, L. G. 1979. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3): 410–421.

Wainwright, M. J.; and Jordan, M. I. 2008. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1–2): 1–305.

Webb, G. I.; Lee, L. K.; Goethals, B.; and Petitjean, F. 2018. Analyzing Concept Drift and Shift from Sample Data. *Data Mining and Knowledge Discovery*, 32(5): 1179–1199.