Project no.: 619209

Project full title: Analysis of Massive Data STreams

Project Acronym: AMIDST

Deliverable no.: D3.4

Title of the deliverable: Efficient inference in hybrid domains with static and dynamic models

Contractual Date of Delivery to the CEC:	**30.11.2015**
Actual Date of Delivery to the CEC:	**30.11.2015**
Organisation name of lead contractor for this deliverable:	**AAU**
Author(s):	**Antonio Salmerón, Hanen Borchani, Helge Langseth, Anders L. Madsen, Ana M. Martínez, Andrés Masegosa, Thomas D. Nielsen, Darío Ramos López**
Participants(s):	**P01, P02, P03, P04**
Work package contributing to the deliverable:	**WP3**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**55**
Start date of project:	**1st January 2014 Duration: 36 month**

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This document describes the methodological developments within WP3 (exact and approximate inference), implemented and made available within the AMIDST toolbox. Procedures for belief updating in static AMIDST models based on variational message passing (VMP) and importance sampling (IS) are described. Belief updating in dynamic AMIDST models is carried out by extending VMP and IS to dynamic Bayesian networks using the factored frontier algorithm for reducing the problem complexity. Maximum a posteriori and the most probable explanation problems are approached using optimisation algorithms and Monte Carlo methods in static models while mean field approximations are used in dynamic models. Developments described in this document will be intensively used in the use case provided by WP8, and as a methodological basis for further developments in WP4.

Keyword list: Inference in hybrid static and dynamic Bayesian networks, importance sampling, variational message passing, maximum a posteriori (MAP), most probable explanation (MPE)

Contents

Document history

Version	Date	Author (Unit)	Description
v0.3	17/11 2015	Antonio Salmerón, Hanen Borchani, Helge Langseth, Anders L. Madsen, Ana M. Martínez, Andrés R. Masegosa, Thomas D. Nielsen, Darío Ramos-López	First draft finished
v0.6	29/11 2015	Antonio Salmerón, Hanen Borchani, Helge Langseth, Anders L. Madsen, Ana M. Martínez, Andrés R. Masegosa, Thomas D. Nielsen, Darío Ramos-López	Initial draft finished and reviewed by the PSRG
v1.0	30/11 2015	Antonio Salmerón, Hanen Borchani, Helge Langseth, Anders L. Madsen, Ana M. Martínez, Andrés R. Masegosa, Thomas D. Nielsen, Darío Ramos-López	Final version of document

1 Executive summary

The aim of this document is to describe the methodological developments carried out within Work package 3, corresponding to exact and approximate inference in hybrid domains with static and dynamic models. All the algorithms described in this deliverable operate over conditional linear Gaussian Bayesian networks and are available within the open source AMIDST toolbox. The document describes procedures for probabilistic inference (belief updating) in static models based on variational message passing and importance sampling. Also in static domains, maximum a posteriori (MAP) and the most probable explanation (MPE) problems are approached using optimisation algorithms and Monte Carlo methods. Belief updating in dynamic domains is carried out by extending variational message passing and importance sampling to dynamic Bayesian networks using the factored frontier algorithm for reducing the problem complexity. MAP in dynamic domains is solved using combinations of mean field approximations.

2 Introduction

The aim of this document is to describe the methodological developments carried out within Work package 3, corresponding to exact and approximate inference in hybrid domains with static and dynamic models. All the algorithms described in this deliverable operate over conditional linear Gaussian Bayesian networks and are available within the open source AMIDST toolbox. The document describes procedures for probabilistic inference (belief updating) in static AMIDST models [1] based on variational message passing (VMP) and importance sampling (IS). Belief updating in dynamic AMIDST models [1] is carried out by extending VMP and IS to dynamic Bayesian networks using the factored frontier algorithm for reducing the problem complexity. Maximum a posteriori (MAP) and the most probable explanation (MPE) problems are approached using optimisation algorithms and Monte Carlo methods in static models while mean field approximations are used in dynamic models.

The developments in Work package 3 are strongly connected to the use case provided by Work package 8. Inference (belief updating and MAP) in static models will be used in task 8.2 of Work package 8 for profile extraction of potentially good customers, where static AMIDST models will be used. Dynamic inference will be used in Task 8.2 as well, for predicting risk in credit operations. At the time of preparing this document, the real AMIDST models from Work package 8 were not yet available, as their elicitation will require the use of learning algorithms to be developed in Work package 4, and will be constructed in forthcoming stages of Work package 8. Nevertheless, all the tests reported in this document have been carried out over artificial AMIDST models designed to resemble as much as possible the range of models that will be built in Work package 8. The final and thorough testing of the algorithms described in this document, with the real data and models, will be done within the context of that work package and reported in Deliverable 8.3.

The remainder of the document is organised as follows. Section 3 describes the inference tasks addressed in Work package 3 as well as the approaches to probabilistic inference based on VMP and IS. Scalable inference in static models is the subject of Section 4, which covers probabilistic inference, MPE and MAP algorithms. The AMIDST contributions to inference in dynamic models are described in Section 5. The document ends with conclusions in Section 6.

Part of the contributions described in this document have already been published in references [2, 3]. A review of inference in hybrid static and dynamic Bayesian networks can be found in Deliverable 3.1 [4] and in [5].

3 Preliminaries

Today, omnipresent sensors are continuously providing streaming data on the environments in which they operate. For instance, a typical monitoring and analysis system may use streaming data generated by sensors to monitor the status of a particular device and to make predictions about its future behaviour, or diagnostically infer the most likely system configuration that has produced the observed data. Sources of streaming data with even a modest updating frequency can produce extremely large volumes of data, thereby making efficient and accurate data analysis and prediction difficult. One of the main challenges is related to handling uncertainty in data, where principled methods and algorithms for dealing with uncertainty in massive data applications are required. Probabilistic graphical models (PGMs) provide a well-founded and principled approach for performing inference and belief updating in complex domains endowed with uncertainty.

Bayesian networks (BNs) [6–10] are a particular type of PGM that has enjoyed widespread attention in the last two decades. Attached to each node, there is a conditional probability distribution given its parents in the network, so that in general, for a BN with N variables $\mathbf{X} = \{X_1, \ldots, X_N\}$, the joint distribution factorizes as $p(\mathbf{X}) = \prod_{i=1}^{N} p_i(X_i|\text{pa}\,(X_i))$, where $\text{pa}\,(X_i)$ denotes the set of parents of X_i in the network. A BN is called *hybrid* if some of its variables are discrete while others are continuous.

Our goal is to analyse the performance of probabilistic inference in hybrid Bayesian networks in scenarios where data come in streams at high speed, and therefore a quick response is required. Because of that, we will focus our analysis on *conditional linear Gaussian* (CLG) models [11, 12], instead of more expressive alternatives such as mixtures of exponentials [13], mixtures of polynomials [14] and mixtures of truncated basis functions in general [15], as inference in the latter models is often more time consuming due to the higher number of parameters required for specifying the distributions [16]. A CLG model is a hybrid Bayesian network where the joint distribution is a conditional linear Gaussian [12]. We will use lowercase letters to refer to values or configurations of values, so that x denotes a value of X and boldface $\mathbf{x}$ is a configuration of the variables in $\mathbf{X}$. In the CLG model, the conditional distribution of each discrete variable $X_D \in \mathbf{X}$ given its parents is a multinomial, whilst the conditional distribution of each continuous variable $Z \in \mathbf{X}$ with discrete parents $\mathbf{X}_D \subset \mathbf{X}$ and continuous parents $\mathbf{X}_C \subset \mathbf{X}$, is given as a normal density by

$$p(z|\mathbf{X}_D = \mathbf{x}_D, \mathbf{X}_C = \mathbf{x}_C) = \mathcal{N}(z; \alpha_{\mathbf{x}_D} + \boldsymbol{\beta}_{\mathbf{x}_D}^{\mathrm{T}} \mathbf{x}_C, \sigma_{\mathbf{x}_D})\,, \tag{1}$$

for all $\mathbf{x}_D \in \Omega_{\mathbf{X}_D}$ and $\mathbf{x}_C \in \Omega_{\mathbf{X}_C}$, where α and $\boldsymbol{\beta}$ are the coefficients of a linear regression model of Z given its continuous parents; this model can differ for each configuration of the discrete variables $\mathbf{X}_D$. Therefore, the conditional mean of Z is a linear model on its continuous parents, while its standard deviation, σ_D, only depends on the discrete parents.

After fixing any configuration of the discrete variables, the joint distribution of any subset $\mathbf{X}_C \subseteq \mathbf{X}$ of continuous variables is a multivariate Gaussian whose parameters

can be obtained from the ones in the CLG representation. For a set of M continuous variables $Z_1, \ldots, Z_M$ with a conditionally specified joint density $p(z_1, \ldots, z_M) = \prod_{k=1}^{M} p(z_k|z_{k+1}, \ldots, z_M)$, where the k-th factor, $1 \leq k \leq M$, is such that

$$p(z_k|z_{k+1}, \ldots, z_M) = \mathcal{N}(z_k; \mu_{z_k|z_{k+1},\ldots,z_M}, \sigma_{z_k}),$$

it holds that the joint is $p(z_1, \ldots, z_M) = \mathcal{N}(z_1, \ldots, z_M; \boldsymbol{u}, \boldsymbol{\Sigma})$, where $\boldsymbol{u}$ is the n-dimensional vector of means and $\boldsymbol{\Sigma}$ is the covariance matrix of the multivariate distribution over random variables $Z_1, \ldots, Z_M$ and both $\boldsymbol{u}$ and $\boldsymbol{\Sigma}$ are derived from the parameters in Equation (1) [17].

3.1 The inference tasks

Given a set of observed variables $\mathbf{X}_E \subset \mathbf{X}$ and a set of variables of interest $\mathbf{X}_I \subseteq \mathbf{X} \setminus \mathbf{X}_E$, *probabilistic inference* (also called *belief updating* or *probability propagation*) consists of computing the posterior distribution $p(x_i|\mathbf{x}_E)$ for each $i \in I$, where X_i can be either discrete or continuous. We denote by $\mathbf{X}_{C\setminus E}$ and $\mathbf{X}_{D\setminus E}$ the set of continuous and discrete variables not in $\mathbf{X}_E$, respectively. Furthermore, $\mathbf{X}_{C\setminus\{E\cup\{i\}\}}$ and $\mathbf{X}_{D\setminus\{E\cup\{i\}\}}$ denote the set of continuous and discrete variables not in $\mathbf{X}_E \cup \{\mathbf{X}_i\}$. Now, the goal of inference can be formulated as computing

$$p(x_i|\mathbf{x}_E) = \frac{p(x_i, \mathbf{x}_E)}{p(\mathbf{x}_E)} = \frac{\displaystyle\sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_{D\setminus\{E\cup\{i\}\}}}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_{C\setminus\{E\cup\{i\}\}}}} p(\mathbf{x}; \mathbf{x}_E)\, \mathrm{d}\mathbf{x}_C}{\displaystyle\sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_{D\setminus E}}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_{C\setminus E}}} p(\mathbf{x}; \mathbf{x}_E)\, \mathrm{d}\mathbf{x}_C}, \tag{2}$$

where $\Omega_{\mathbf{X}}$ is the set of possible values of a set of variables $\mathbf{X}$ and $p(\mathbf{x}; \mathbf{x}_E)$ is the joint distribution in the BN instantiated according to the observed values $\mathbf{x}_E$.

If X_i in Equation (2) is continuous, the result of the belief propagation is the evaluation of a density function. Of even higher interest in this case is typically the probability of the variable taking values on a given interval (a, b), which amounts to computing

$$p(X_i \in (a, b)|\mathbf{x}_E) = \frac{\displaystyle\int_{x_i=a}^{b} \sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_{D\setminus\{E\cup\{i\}\}}}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_{C\setminus\{E\cup\{i\}\}}}} p(\mathbf{x}; \mathbf{x}_E)\, \mathrm{d}\mathbf{x}_C\, \mathrm{d}x_i}{\displaystyle\sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_{D\setminus E}}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_{C\setminus E}}} p(\mathbf{x}; \mathbf{x}_E)\, \mathrm{d}\mathbf{x}_C}, \tag{3}$$

We call the probabilistic inference tasks described in Equation (2) (for discrete X_i) and Equation (3) (for continuous X_i) a *query*.

A particularly complex kind of inference in BNs is the so-called *maximum a posteriori (MAP)* problem. For a set of target variables $\mathbf{X}_I \subseteq \mathbf{X} \setminus \mathbf{X}_E$, the goal of MAP inference

is to compute

$$\mathbf{x}_I^* = \arg \max_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} p(\mathbf{x}_I | \mathbf{X}_E = \mathbf{x}_E), \tag{4}$$

where $p(\mathbf{x}_I|\mathbf{X}_E = \mathbf{x}_E)$ is obtained by first marginalizing out the variables in $\mathbf{X} \setminus \{\mathbf{X}_I \cup \mathbf{X}_E\}$ from the joint distribution $p(\mathbf{x})$. A related problem is *MPE* that stands for finding the *most probable explanation* to an observation $\mathbf{X}_E = \mathbf{x}_E$. It is a particular case of MAP, where $\mathbf{X}_I = \mathbf{X} \setminus \mathbf{X}_E$. Both MAP and MPE belong to the class of problems known as *abductive inference* [18].

3.2 Variational message passing in static Bayesian networks

Variational inference is a deterministic approximate inference technique, where we seek to iteratively optimise a variational approximation to the posterior distribution of interest [19]. Let $\mathcal{Q}$ be the set of possible approximations; then the variational approximation to a posterior distribution $p(\mathbf{x}_I|\mathbf{X}_E = \mathbf{x}_E)$ is defined as

$$q^*_{\mathbf{x}_E}(\mathbf{x}_I) = \arg \min_{q \in \mathcal{Q}} D(q(\mathbf{x}_I)||p(\mathbf{x}_I|\mathbf{X}_E = \mathbf{x}_E)),$$

where $D(q||p)$ is the KL divergence between q and p.

A common approach is to employ a *variational mean-field* approximation of the posterior distribution, so that the approximation factorises over the individual variables involved, i.e.,

$$q^*_{\mathbf{x}_E}(\mathbf{x}_I) = \prod_{i \in I} q^*_{\mathbf{x}_E}(x_i). \tag{5}$$

During the optimisation of the variational mean-field one performs a coordinate ascent by iteratively updating the individual variational distributions while holding the others fixed [20]. Updating a variational distribution essentially involves calculating the variational expectation of the logarithm of the original conditional distributions of the model. This can be done efficiently and in closed form when the distributions involved are conjugate-exponential [21]. A conditional distribution $p(\boldsymbol{X}|\boldsymbol{Y})$ is said to be on *exponential form* if it can be written as

$$\ln p(\boldsymbol{X}|\boldsymbol{Y}) = \boldsymbol{\theta}^c_{\boldsymbol{X}}(\boldsymbol{Y}) s(\boldsymbol{X}) - A^c_{\boldsymbol{X}}(\boldsymbol{\theta}^c_{\boldsymbol{X}}(\boldsymbol{Y})) + h^c(\boldsymbol{X}), \tag{6}$$

where $\boldsymbol{\theta}^c_{\boldsymbol{X}}(\boldsymbol{Y})$ is the *natural parameter* vector and $s(\boldsymbol{X})$ is the *natural statistic* vector.[1] The function $A^c_{\boldsymbol{X}}(\boldsymbol{\theta}_{\boldsymbol{X}}(\boldsymbol{Y}))$ is the *log-partition* function, which ensures that the distribution is normalized for all values of the parent variables $\boldsymbol{Y}$. $h^c(\boldsymbol{X})$ is called the *underlying measure*, typically the Lebesgue measure. A prior distribution $p(\boldsymbol{Y})$ is said to be a conjugate prior for the conditional probability distribution $p(\boldsymbol{X}|\boldsymbol{Y})$ if the posterior

[1] The superscript c is used to signify that a component relates to the conditional distribution.

distribution $p(\boldsymbol{Y}|\boldsymbol{X}) \propto p(\boldsymbol{X}|\boldsymbol{Y})p(\boldsymbol{Y})$ has the same functional form as the prior $p(\boldsymbol{Y})$. More specifically, assuming that $p(\boldsymbol{Y})$ is given by

$$\ln p(\boldsymbol{Y}) = \boldsymbol{\theta_Y} s(\boldsymbol{Y}) - A_{\boldsymbol{Y}}(\boldsymbol{\theta_Y}) + h(\boldsymbol{Y}),$$

then one should be able to write $\ln p(\boldsymbol{X}|\boldsymbol{Y})$ in terms of $s(\boldsymbol{Y})$:

$$\ln p(\boldsymbol{X}|\boldsymbol{Y}) = \boldsymbol{\theta}^p_{\boldsymbol{X}}(\boldsymbol{X}) s(\boldsymbol{Y}) - A^p_{\boldsymbol{X}}(\boldsymbol{\theta}^p_{\boldsymbol{X}}(\boldsymbol{Y})) + h^p(\boldsymbol{X}). \tag{7}$$

To make the distinction between the two forms in Equation (6) and Equation (7) clear, we will refer to them as the conditional and the posterior form, respectively, and indicate this by superscripts c and p associated with their respective components.

A large number of distributions belong to the family of conjugate-exponential distributions, including

- the Gaussian distribution with Gaussian and Gamma prior distributions for the mean and the precision, respectively.
- the Gaussian distribution conditioned on a categorical variable.
- the categorial distribution with a Dirichlet distribution over the parameters.

The family of conjugate-exponential models also include a range of other important probability distributions, but in the AMIDST toolbox we will mainly consider the ones mentioned above. As an example, consider a binary variable Y with natural parameters $[\ln(p_1), \ln(p_2)]^T$ and natural statistics $[I(Y = y_1), I(Y = y_2)]^T$, where $I(\cdot)$ is the indicator function. If Y has a child X such that

$$\ln p(X \mid Y) = I(Y = y_1) \ln p(X|Y = y_1) + I(Y = y_2) \ln p(X|Y = y_1), \tag{8}$$

then the conditional exponential form of Equation (8) is given by

$$\begin{aligned} \ln p(X \mid Y) &= \Big(I(Y = y_1) \cdot \boldsymbol{\theta}^c_X(Y = y_1) + I(Y = y_2) \cdot \boldsymbol{\theta}^c_X(Y = y_2)\Big)^T s(X) \\ &\quad - I(Y = y_1) \cdot A^c_X(\boldsymbol{\theta}^c_X(Y = y_1)) - I(Y = y_2) \cdot A^c_X(\boldsymbol{\theta}^c_X(Y = y_2)) \end{aligned}$$

and the posterior form is given by

$$\begin{aligned} \ln p(X \mid Y) &= \boldsymbol{\theta}^p_X(X)^T s(Y) - A^p_X(\boldsymbol{\theta}_X(Y)) \\ &= \begin{pmatrix} \boldsymbol{\theta}^c_X(Y = y_1)^T s(X) - A^c_X(\boldsymbol{\theta}_X(Y = y_1)) \\ \boldsymbol{\theta}^c_X(Y = y_2)^T s(X) - A^c_X(\boldsymbol{\theta}_X(Y = y_2)) \end{pmatrix}^T \begin{pmatrix} I(Y = y^1) \\ I(Y = y^2) \end{pmatrix}. \end{aligned}$$

As mentioned above, when the distributions of the model are all conjugate exponential, one can efficiently optimize the variational distributions through coordinate ascent [21]. Nevertheless, since the variational updating equations often have to be hand-tailored to

the model at hand, the specification process can be both time-consuming and potentially error-prone. Instead, a variational message passing scheme was proposed in [22], where, in the spirit of belief propagation, messages are being passed around between nodes so that a node can update its variational distribution based on local messages received from its parents and children. Each update is guaranteed to increase a lower bound on the log-probability of the evidence. The message being sent from a parent Y to a child X consists of the expected natural statistic vectors for the parent variables, where the expectations are taken wrt. the variational distribution. The message from a child X to a parent Y is given by the expected natural parameters for the child variable in its posterior form wrt. the parent variable Y to which the message is being send. Note that the latter requires that X has received messages from the co-parents of Y. Appendix A contains a specification of the conditional and posterior exponential forms of common distributions. These specifications form the basis for the variational message passing procedure implemented in the AMIDST framework.

The message passing scheme supported by variational message passing is very flexible; a variable can update its variational distribution when it has received messages from all its parent and child nodes. This feature can be exploited for devising parallel scalable message passing schemes for AMIDST models. In particular, within a Bayesian learning context, the general AMIDST model structure defines local models only connected through global parameter nodes, thus enabling independent and parallel message passing in these local models (synchronization between the local models is achieved by the passing messages over the global parameter nodes).

3.3 Inference in static Bayesian networks using importance sampling

Exact inference in CLG networks is a computationally expensive task that requires the construction of a *strong* junction tree in order to guarantee that the continuous variables are marginalised out first [11]. Hence, in scenarios as stream processing, where quick responses are required, the use of approximate algorithms becomes necessary. In this section we analyse an approach to approximate inference in CLG networks based on a general technique for probabilistic inference able to provide quick answers to queries, namely *importance sampling* [23].

Importance sampling is a versatile simulation technique that in the case of inference in BNs amounts to transforming the numerator in Equation (3) by multiplying and dividing by a distribution p^* that, unlike $p(\mathbf{x}, \mathbf{x}_E)$, is easy to handle and, more precisely, from which samples can easily be drawn.

Let θ denote the numerator of Equation (3), i.e. $\theta = \int_a^b h(x_i)\mathrm{d}x_i$ with

$$h(x_i) = p(x_i, \mathbf{x}_E) = \sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_{D\setminus\{E\cup\{i\}\}}}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_{C\setminus\{E\cup\{i\}\}}}} p(\mathbf{x}; \mathbf{x}_E)\, \mathrm{d}\mathbf{x}_C.$$

Then, we can write θ as

$$\theta = \int_a^b h(x_i)\mathrm{d}x_i = \int_a^b \frac{h(x_i)}{p^*(x_i)} p^*(x_i)\,\mathrm{d}x_i = \mathbb{E}_{p^*}\left[\frac{h(X_i^*)}{p^*(X_i^*)}\right],$$

where p^* is a probability density function on (a, b) called the *sampling distribution*, and X_i^* is a random variable with density p^*. Let $X_i^{*(1)}, \ldots, X_i^{*(m)}$ be a sample drawn from p^*. Then it is easy to prove that

$$\hat{\theta} = \frac{1}{m}\sum_{j=1}^{m} \frac{h(X_i^{*(j)})}{p^*(X_i^{*(j)})} \tag{9}$$

is an unbiased estimator of θ. As $\hat{\theta}$ is unbiased, the error of the estimation is determined by its variance, which is

$$\begin{aligned} \mathrm{Var}(\hat{\theta}) &= \mathrm{Var}\left(\frac{1}{m}\sum_{j=1}^{m} \frac{h(X_i^{*(j)})}{p^*(X_i^{*(j)})}\right) = \frac{1}{m^2}\sum_{j=1}^{m} \mathrm{Var}\left(\frac{h(X_i^{*(j)})}{p^*(X_i^{*(j)})}\right) \\ &= \frac{1}{m^2} m \mathrm{Var}\left(\frac{h(X_i^*)}{p^*(X_i^*)}\right) = \frac{1}{m}\mathrm{Var}\left(\frac{h(X_i^*)}{p^*(X_i^*)}\right). \end{aligned} \tag{10}$$

The key point in importance sampling is the selection of the sampling distribution since, according to Equation (10), it determines the accuracy of the estimation. The rule is that the closer p^* is to h, the lower the variance is [24].

A simple procedure for selecting the sampling distribution is the so-called *evidence weighting* (EW) [25]. In EW, each variable is sampled from a conditional density given its parents in the network. The sampling order is therefore from parents to children. The observed variables are not sampled, but instead they are instantiated to the observed value. A version of this algorithm in which the conditional densities are dynamically updated during the simulation procedure was introduced in [26].

Hence, adopting EW means that h involves the product of all the conditional distributions in the Bayesian network, while p^* involves the same conditional distributions except those ones corresponding to observed variables.

Note that the denominator in Equation (3) is just the probability of evidence, which has to be estimated as well in order to have an answer to a query (recall that $\hat{\theta}$ is just an estimator of the numerator). It was shown in [24] that numerator and denominator can be estimated using the same sample. To achieve this, instead of taking a sampling distribution defined on (a, b) it must be defined on the entire range of X_i. In such case, the estimator in Equation (9) becomes an estimator of the denominator (probability of evidence) and the same estimator, evaluated only in the points in the sample that fall inside (a, b), is an estimator of θ.

The details of the inference procedure are given in Algorithm 1. Here, w_1 and w_2 represent, respectively, the values of h and p^* for the simulated configurations. Each variable is

Function EW($\mathbf{X}$,P,$\mathbf{x}_E$,X,a,b,M)
Input: The set of variables in the network, $\mathbf{X} = \{X_1, \ldots, X_N\}$ in topological order. The distributions in the network $P = \{p_1, \ldots, p_N\}$. Evidence $\mathbf{X}_E = \mathbf{x}_E$. The target variable X. Sample size M.
Output: An estimation of $P(a < X < b|\mathbf{X}_E = \mathbf{x}_E)$

```
1  begin
2      Initialization:
3      s_1 ← 0 ; s_2 ← 0.
4      for j ← 1 to M do
5          Sample generation:
6          w_1 ← 1 ; w_2 ← 1.
7          for i ← 1 to N do
8              if X_i ∉ X_E then
9                  Simulate a value x_i^(j) for X_i using p_i(x_i|pa(x_i)).
10                 w_2 ← w_2 * p_i(x_i^(j)|pa(x_i)).
11             end
12             else
13                 Let x_i^(j) be the value of X_i in X_E.
14             end
15             w_1 ← w_1 * p_i(x_i^(j)|pa(x_i)).
16         end
17         if w_1 ≠ 0 then
18             Let x^(j) be the value of X in the simulated configuration x_1^(j), ..., x_N^(j).
19             if x^(j) ∈ (a, b) then
20                 s_1 ← s_1 + w_1/w_2
21             end
22             s_2 ← s_2 + w_1/w_2
23         end
24     end
25     return s_1/s_2 .
26 end
```

Algorithm 1: The EW algorithm for answering a probabilistic query.

simulated using its conditional distribution. In fact, we can only simulate from marginals rather than conditional distributions. This is why EW starts simulating from root nodes, where marginal distributions are attached to them. Once root nodes are simulated (i.e. we have a value for them), their children are simulated by first instantiating their conditional distributions to the simulated values for the roots, obtaining, therefore marginal densities. As we are operating with CLG networks, the marginal distribution for each discrete variable is a multinomial while it is a normal for each continuous variable. In both cases, simulating values from them is straightforward [27].

4 Scalable inference in static models

4.1 Probabilistic inference

4.1.1 Opportunities for parallelization

In regards of scalability of importance sampling as described in Algorithm 1, it is worth pointing out that the iterations in the **for**-loop for sample generation (starting from Line 4) can be executed in parallel. This is due to the fact that the items in the sample are independent of each other. As that loop constitutes the fundamental workload of the algorithm, the scalability is potentially high, using, for instance, a multi-threaded implementation. Our proposal for scaling up the algorithm consists of using parallelisation in the above mentioned **for**-loop. We have used Java 8 streams in our implementation. A Map/Reduce description of the parallelisation scheme used is depicted in Figure 1.

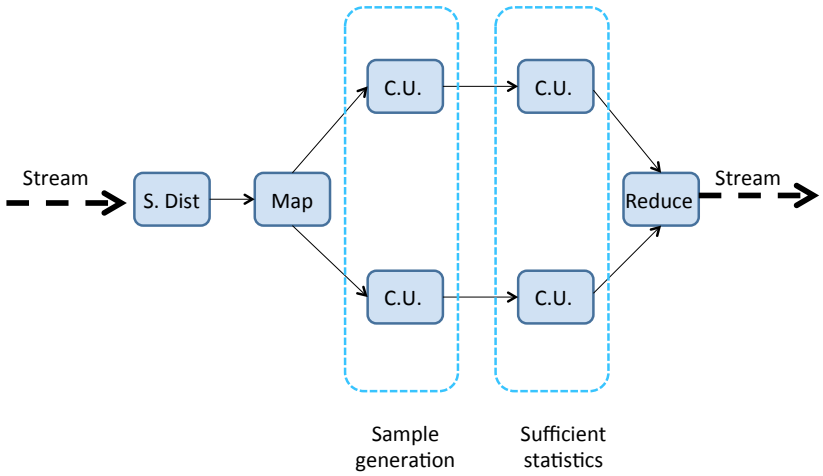


Figure 1: A Map/Reduce scheme of the design of the parallel importance sampling algorithm. Acronym c.u. stands for *computing unit* and S. Dist means computation of the sampling distributions.

Even though Algorithm 1 focused on queries involving a continuous variable (see in particular Line 19), a similar argument can be developed for discrete queries, where

instead of an interval, we seek the probability of a variable taking on a fixed value. For the sake of simplicity, we omit here the details for the discrete case. Nevertheless, some configurations can be useless for the estimation procedure when using discrete variables. That is the case in which w_1 becomes zero, which happens when a simulated configuration is incompatible with the observations (cf. Line 17). As an example, consider a BN with two binary variables X and Y, where $P(X = 0) = 0.9$, $P(Y = 0|X = 0) = 0$ and $P(Y = 0|X = 1) = 0.5$. It means that, approximately, 90% of the times the value simulated for X will be 0, and consequently Y will be 1. Assume that we have observed $Y = 0$. As $P(Y = 0|X = 0) = 0$, 90% of the times the simulated configuration will be discarded. This problem only arises when simulating discrete variables, as the normal density for a continuous variable is never equal to 0. Finally, we note that we use a static sampling distribution (i.e. it is not updated during the sample generation phase), as this is the fastest alternative.

4.1.2 Experimental evaluation

In order to test the accuracy and scalability of EW with respect to available computing resources, we conducted an experiment over two randomly generated CLG networks with 10, 100 and 500 variables respectively, half of them continuous and the rest binary discrete variables. The aim of this choice is to test the behaviour of the algorithm when dealing with small as well as with large models. The number of links was set to double the number of variables. We have not considered any parallelisation issue for VMP, but we have included it in the experimental analysis as a benchmark.

Variables	10	100	500
Run time (seconds)	0.0739	3.2119	9.6917
Error	0.4657	0.2679	2.2759

Table 1: Error and run times for VMP.

For each network, we randomly generated a set of observations for 5% of the variables (rounded upwards). Queries were also selected at random, by choosing a variable and generating a number α from a standard normal distribution and taking the interval (a, b) with $a = \alpha - 0.5$ and $b = \alpha + 0.5$. Each query was answered using VMP and EW, the latter with samples of size 1000, 5000 and 10000. Each experiment was replicated for an increasing number of cores ranging from 1 to 20. The experiments were run on a dual-processor AMD Opteron 2.8GHz server with 32 cores and 64GB of RAM, running Linux Ubuntu 14.04.1 LTS.

Each run was repeated 10 times and the run time and error of the estimations were averaged over the 10 runs. The error of the estimations was computed using the χ^2 divergence. Let p_i, $i = 1, \ldots, 10$ be the exact probability corresponding to the query in run i, and let q_i be the estimated value. The χ^2 divergence is computed as

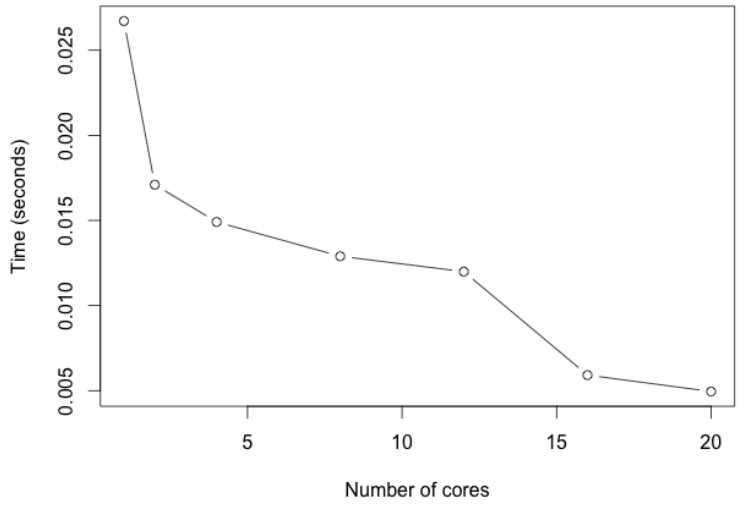


Figure 2: Run time in wall-clock seconds as a function of the number of cores for EW over a randomly generated CLG network with 10 variables (right).

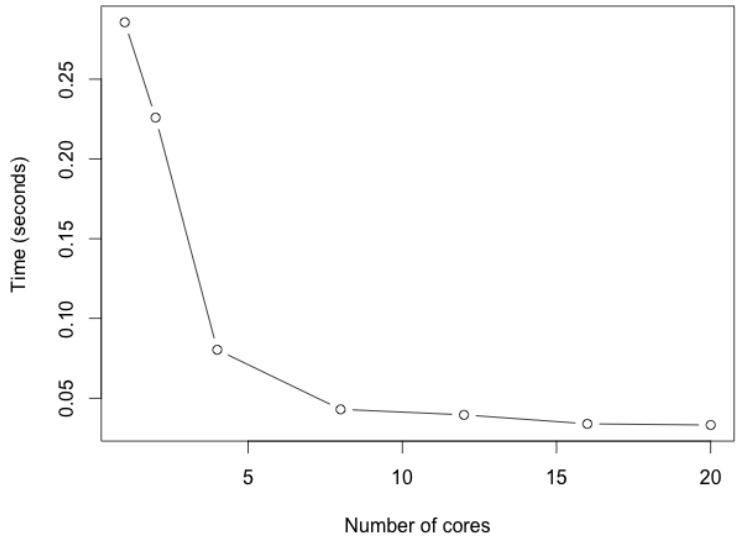


Figure 3: Run time in wall-clock seconds as a function of the number of cores for EW over a randomly generated CLG network with 100 variables (right).

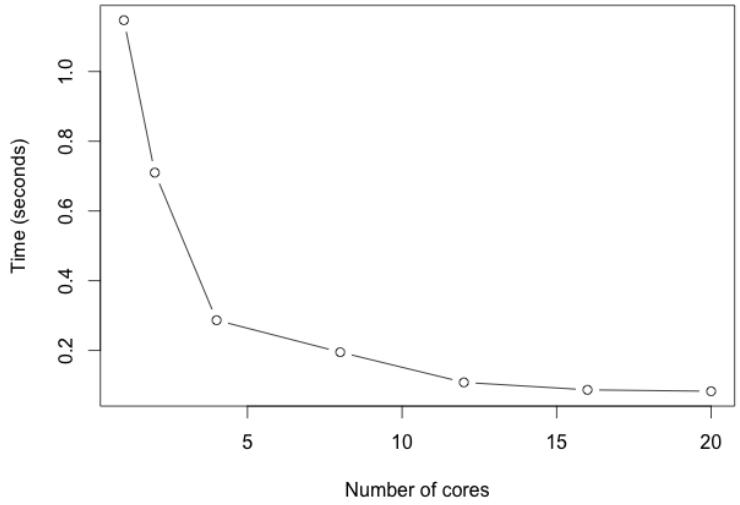


Figure 4: Run time in wall-clock seconds as a function of the number of cores for EW over a randomly generated CLG network with 500 variables (right).

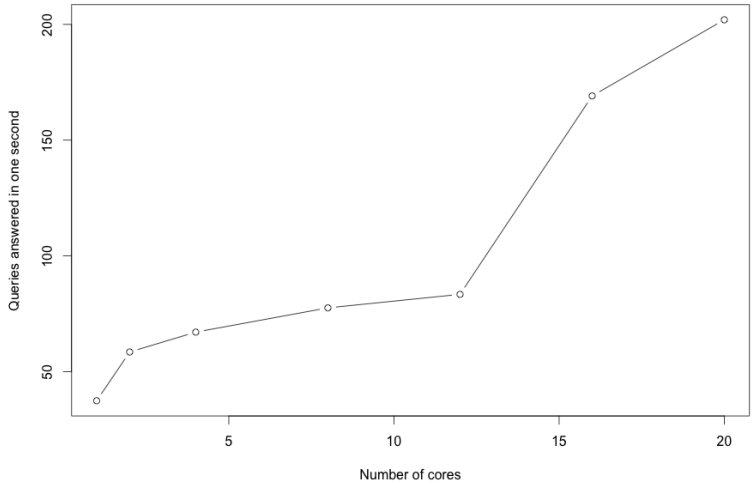


Figure 5: Number of queries answered (number of stream items processed) per wall-clock second as a function of the number of cores for EW over a randomly generated CLG network with 10 variables.

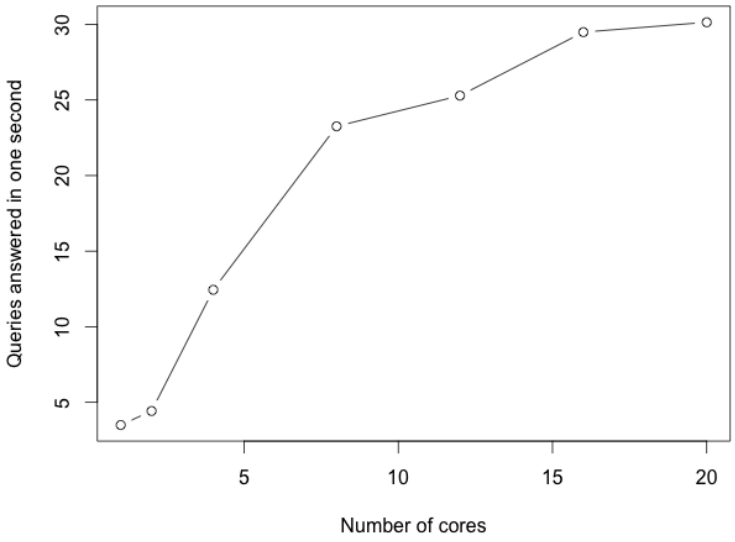


Figure 6: Number of queries answered (number of stream items processed) per wall-clock second as a function of the number of cores for EW over a randomly generated CLG network with 100 variables.

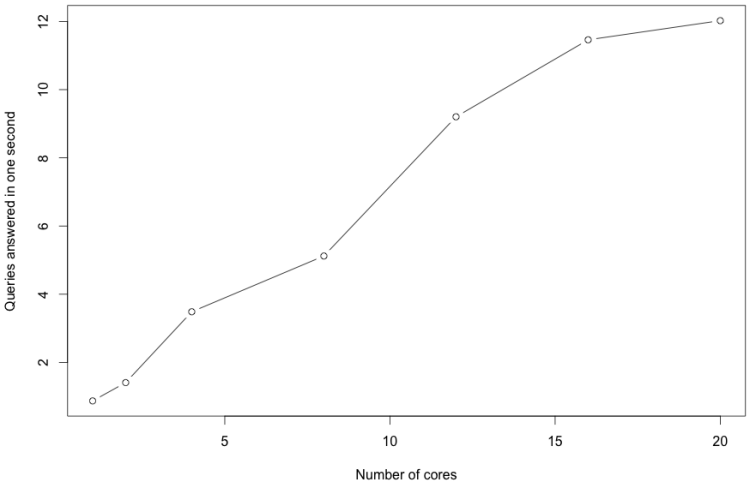


Figure 7: Number of queries answered (number of stream items processed) per wall-clock second as a function of the number of cores for EW over a randomly generated CLG network with 500 variables.

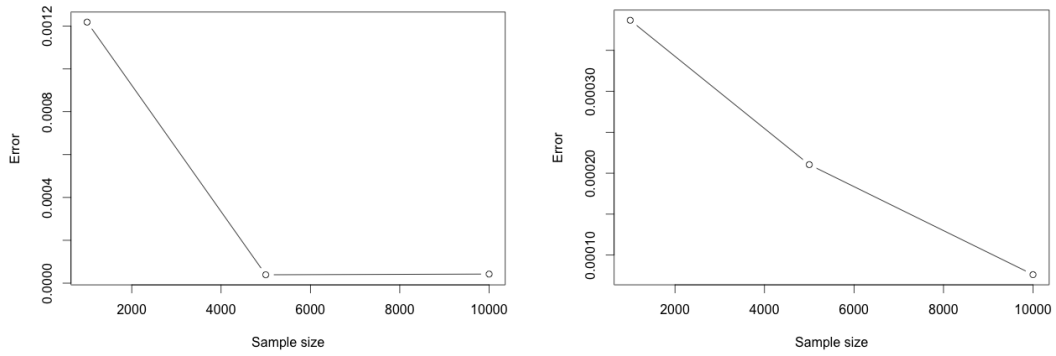


Figure 8: Error attained by EW as a function of the sample size for a network with 10 variables (left) and with 500 variables (right).

$$\chi^2 = \frac{1}{10}\sum_{i=1}^{10}\frac{(q_i - p_i)^2}{p_i}.$$

The χ^2 divergence is specially appropriate for measuring errors in probability estimations, as it is measured taking into account the magnitude of the value to estimate, and not simply the absolute or square deviation. The intuition behind the values of χ^2 is illustrated in the following example.

Example 1 *Assume we have 3 runs of the experiments with exact probabilities equal to 0.9, 0.8 and 0.01, and we have obtained two estimates, namely* $(0.91, 0.81, 0.01)$ *and* $(0.9, 0.81, 0.02)$*. Note that both estimates return one of the exact values and make an absolute error of* 0.01 *for the other values. The root mean square errors for both estimates are therefore identical, and equal to* 0.0082*. However, the* χ^2 *penalises the second estimate, as it yields an absolute error of* 0.01 *for a small probability value (correct value* 0.01*) unlike estimate 1, where the same error was produced, but for a higher value (correct value* 0.9*). The* χ^2 *errors for estimates 1 and 2 are* 0.000078 *and* 0.0033*, respectively.*

The results of the experiments in terms of run time for EW are shown in Figures 2, 3 and 4. The results for VMP are given in Table 1. The plots correspond to a sample of size 1000 for EW, and show the evolution of the run time as a function of the number of cores used during the computation. It can be seen how in both networks EW scales up with respect to the number of cores. We conjecture that the jump in the curve at 12 cores is probably due to the small magnitude of the run time (of the order of miliseconds) and hence, any small variation due to any issue external to the algorithm can cause it, specially taking into account that the server where the experiments were run was shared with other users.

The ability of the algorithm for processing streams is illustrated in Figures 5, 6 and 7 where the number of queries answered per second is given as a function of the number of cores. It can be seen that the algorithm is able to process up to 12 queries (i.e. to process up to 12 items from a stream) in a second for the 500 variable network, and over 200 per second for the 10 variable network, when using 20 cores.

There is a big difference in favour of EW with respect to VMP in what concerns computing time, according to Table 1. For instance, when using 20 cores, EW gives an answer in less than 0.1 seconds, while VMP takes around 9, in the large network. In the small network, with only 10 variables, the results are similar; EW is again faster than VMP, reaching response times for 20 cores below 0.005 seconds. It means that the method is able to answer around 200 queries in 1 second.

The behaviour of the EW algorithm in terms of error is summarised in the plots in Figure 8, where the χ^2 divergence is represented versus the sample size. As expected, the error goes down as the sample size increases. Even with the lowest sample size considered (1000), the errors are fairly low for the large and small networks. The errors reported by VMP are considerably higher, as reported in Table 1. It must be taken into account that VMP returns a posterior density that is a single Gaussian, out of which the probability of interest is estimated. However, it is known that the posterior density of a continuous variable in a CLG network is in fact a mixture of Gaussians.

4.2 Abductive inference in Conditional Linear Gaussian Networks

Given a Bayesian network where a subset of the variables is observed, we may, e.g., query the network for the posterior marginal distributions of the remaining variables or for a maximum a posteriori probability configuration for a subset of the variables. If this subset is a proper subset of the non-observed variables, then the problem is referred to as a maximum a posteriori hypothesis problem [28]. On the other hand, if the variables of interest correspond to the complement of the observation set, then the problem is referred to as that of finding the most probable explanation [29,30]; MPE can therefore be considered a specialization of MAP.

For Bayesian networks containing only discrete variables, there has been a substantial amount of work on devising both exact and approximate algorithms for performing MAP and MPE inference. However, for hybrid Bayesian networks, with both discrete and continuous variables, these types of inference problems have received only little attention [31]. In this section we consider the problem of performing MPE inference in conditional linear Gaussian networks [12]. We propose an MPE algorithm based on bucket-elimination, which has the same computational complexity as that of standard inference for posterior marginals [32]. In contrast to the proposal in [31], we study the effect of entering evidence and also avoid the use of piece-wise defined functions by using an auxiliary tree structure keeping track of the functions used in previous calculations. The algorithm is illustrated using a detailed numerical example.

4.2.1 Exact MPE Inference in CLG Networks

Exact MPE inference (i.e., solving Equation (4) when $\mathbf{X}_I = \mathbf{X} \setminus \mathbf{X}_E$) can be carried out by adapting generic inference algorithms like *Bucket Elimination* [33]. The choice of bucket elimination as the underlying inference scheme for our proposal is motivated by its simplicity and flexibility, as well as the fact that it has been successfully employed in the MPE problem for discrete variables. The bucket elimination algorithm computes the MPE using local computations. A *bucket* containing probability functions is kept for each variable. Initially, an ordering of the variables in the network is established, and each conditional distribution in the network is assigned to the bucket corresponding to the variable in its domain holding the highest rank. Afterwards, the buckets are processed in a sequence opposite to the initial ordering of the variables. Each bucket is processed by combining all the functions it contains and by marginalizing the main variable in that bucket by maximization. The details of the algorithm are given in Algorithm 2.

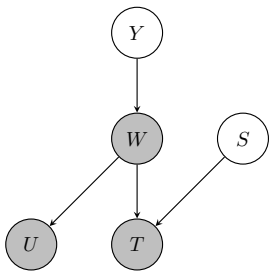
Example 2 *Consider the network in Figure 9 and the ordering* $\langle Y, S, W, T, U\rangle$*. According to such ordering, the initial setting of the buckets would be* $B_Y = \{P(Y)\}$*,* $B_S = \{P(S)\}$*,* $B_W = \{f(w|Y)\}$*,* $B_T = \{f(t|w, S)\}$ *and* $B_U = \{f(u|w)\}$*. The* backward *phase in Algorithm 2 conveys the processing of the buckets as follows. The first bucket to be processed is* B_U*. It is done by maximizing out* u *from* $f(u|w)$*. As* $f(u|w) = \mathcal{N}(u; w, 1)$*, the maximum is reached at the mean, which means that* U *is maximized out by replacing* u *in* $f(u|w)$ *by* w*, which results in a function* $h^U(w) = \frac{1}{\sqrt{2\pi}}$*. Hence, the obtained function is in fact a constant, that is shifted to bucket* B_W*. The next bucket to handle is* B_T*, where* T *is removed from* $f(t|w, S)$ *by replacing* t *by the mean of the conditional distribution, resulting again in a constant function* $h^T(w, S) = \frac{1}{\sqrt{2\pi}}$*. After this calculation,* h^T *is stored in* B_W*, which is itself processed by multiplying* $f(w|Y), h^T(w, S)$ *and* $h^U(w)$ *and maximizing out* W *from the result. Since* h^T *and* h^U *are constant, we just have to maximize* $f(x|Y)$ *and multiply by the constants afterwards. The result is* $h^W(Y, S) = (\frac{1}{\sqrt{2\pi}})^3$*, that is stored in* B_S*. Bucket* B_S *contains* $P(S)$ *and* $h^W(Y, S)$*, whose product is equal to* $0.1(\frac{1}{\sqrt{2\pi}})^3$ *when* $S = 0$ *and* $0.9(\frac{1}{\sqrt{2\pi}})^3$ *when* $S = 1$*. Hence, maximizing with respect to* S *yields* $h^S(Y) = 0.9(\frac{1}{\sqrt{2\pi}})^3$*, that is sent to bucket* B_Y*. The MPE configuration is actually obtained in the* forward *phase of the algorithm, where the bucket processing step is traced back.*

The example above shows how maximizing out continuous variables is an easy task if the continuous variables are always removed first, as it just amounts to replacing the variable being removed by its mode (which in the Gaussian case is equal to its mean). The price to pay is that, in the worst case, a function containing all the discrete variables would be created, as is the case of $h^W(Y, S)$. It is an undesirable event, as the size of a probability function of discrete variables is exponential in the number of variables. This complexity blow-up can be avoided in many cases by allowing orderings for constructing the buckets

Function Elim-MPE($\mathbf{X}$,P,σ,$\mathbf{x}_E$)
Input: The set of variables in the network, $\mathbf{X} = \{X_1, \ldots, X_N\}$. The distributions in the network $P = \{p_1, \ldots, p_N\}$. An ordering, σ, of the variables in $\mathbf{X}$. Evidence $\mathbf{X}_E = \mathbf{x}_E$.
Output: $\mathbf{x}^{mpe}$, the configuration for which the posterior density reaches its maximum, and mpe, the density value at that point.

begin
  Initialization:
  Partition P into buckets $B_1, \ldots, B_N$, where B_i contains the conditional distributions in P whose highest index variable is X_i.
  Backward phase:
  for $p \leftarrow N$ **to** 2 **do**
    if $X_p \in \mathbf{X}_E$ **then**
      Replace X_p by $\mathbf{x}_{E_p}$ in each $h \in B_p$, and insert the resulting h in the bucket corresponding to its highest ranked variable according to ordering σ.
    end
    else
      $h^p \leftarrow \max_{x_p} \prod_{h \in B_p} h$
      Insert h_p in the bucket corresponding to its highest ranked variable.
    end
  end
  Forward phase:
  for $p \leftarrow 1$ **to** n **do**
    Let $h^{R(x_1,\ldots,x_p)}$ denote the restriction of each function $h \in B_p$ to the values $(x_1, \ldots, x_p)$.
    $x_p^{mpe} \leftarrow \arg\max_{x_p} \prod_{h \in B_p} h^{R(x_1,\ldots,x_p)}$.
  end
  return $\mathbf{x}^{mpe} = \{x_1^{mpe}, \ldots, x_N^{mpe}\}$ *and* $mpe = \max_{x_1} \prod_{h \in B_1} h$.
end

Algorithm 2: The Bucket elimination algorithm for computing the MPE as described in [33].



$$P(Y) = (0.5, 0.5)$$
$$P(S) = (0.1, 0.9)$$
$$f(w|Y = 0) = \mathcal{N}(w; -1, 1)$$
$$f(w|Y = 1) = \mathcal{N}(w; 2, 1)$$
$$f(t|w, S = 0) = \mathcal{N}(t; -w, 1)$$
$$f(t|w, S = 1) = \mathcal{N}(t; w, 1)$$
$$f(u|w) = \mathcal{N}(u; w, 1)$$

Figure 9: A hybrid Bayesian network with two discrete and three continuous (shaded) variables.

where discrete and continuous variables can be arranged with no restrictions. But then a new problem arises, as the maximization operation becomes more complex. Assume, for instance, that we reach a point where Y is maximized out before W in Figure 9. This amounts to computing

$$h^Y(w) = \max_y \{P(Y = y)f(w|Y = y)\} = \max\{0.5\mathcal{N}(w; -1, 1), 0.5\mathcal{N}(w; 2, 1)\}.$$

Therefore, h^Y is not a function with a single analytical expression, but it is piece-wise defined instead. We show in the next section how it is possible to avoid piece-wise representations of the result of maximizing out discrete variables. Instead, we will keep lists of the functions that take place in the max operation. In other words, the max operation is carried out in a lazy way. The counterpart is that the forward phase in Algorithm 2 requires us to keep track of the operations carried out over the potentials in the backward phase. We propose to use a tree structure to keep track of the functions involved in intermediate calculations as illustrated in Figure 10 and which corresponds to Example 2.

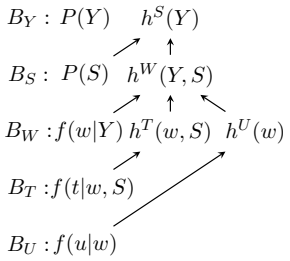


Figure 10: Tree structure keeping track of the functions involved in the intermediate calculations performed during the backward phase of the bucket elimination algorithm.

If a variable is observed, no bucket is created for it. Instead, the variable is replaced by its observed value in every function where it appears. Assume a continuous variable X that is observed taking on value $X = x_0$. If the parents of X are $Y_1, \ldots, Y_n$, replacing variable X by value x_0 in its conditional density results in a function

$$\phi(y_1, \ldots, y_n) = \frac{1}{\sigma_x \sqrt{2\pi}} \exp\left\{ -\frac{(x_0 - (\beta_0 + \sum_{i=1}^{n} \beta_i y_i))^2}{2\sigma_x^2} \right\}. \tag{11}$$

Eventually, function ϕ will be passed to the bucket corresponding to one of its parents, where it will be multiplied by the parent's density prior to maximization. Let Y_j be such a parent of X. Its conditional density can be written as

$$f(y_j | Pa(Y_j)) = \frac{1}{\sigma_{y_j} \sqrt{2\pi}} \exp\left\{ -\frac{(y_j - \mu_{y_j|pa(y_j)})^2}{2\sigma_{y_j}^2} \right\}. \tag{12}$$

Maximizing the product of the functions in Equations (11) and (12) with respect to y_j is equivalent to maximizing the sum of their respective logarithms. It is obtained by solving the equation

$$\frac{\partial}{\partial y_j} \left(-\frac{(x_0 - (\beta_0 + \sum_{i=1}^{n} \beta_i y_i))^2}{2\sigma_x^2} - \frac{(y_j - \mu_{y_j|pa(y_j)})^2}{2\sigma_{y_j}^2} \right) = 0, \tag{13}$$

which simply amounts to maximizing a quadratic function.

As a final comment, we would like to reemphasize that the exact MPE inference scheme as proposed in this section, and illustrated above, follows the same structure as standard algorithms for performing belief updating in CLG networks. Thus, the algorithms share the same computational complexity. In particular, in the example above we see that the elimination order is able to exploit the conditional independencies in the model structure, and we therefore avoid the computational blow-up of having to consider all combinations of the discrete variables, cf. the discussion in Section 4.2.1. Furthermore, when identifying MPE configurations for the continuous variables we see that these configurations can easily be identified as either corresponding to the conditional means of the densities involved or they can be found by maximizing a quadratic function.

Nevertheless, the complexity of exact MPE inference can render the procedure describe above useless when dealing with data streams. Therefore, methods able to give approximate solutions but in a shorted time are required within the context of the AMIDST toolbox.

The MAP problem is significantly more difficult than the MPE problem, as we will have to do both summation and maximization operations over the discrete variables. Consider again the model in Figure 9, and assume we are interested in the MAP configuration over Y and T. Eliminating S (by summation) will result in a *mixture* of Gaussians potential, while eliminating T (by maximization) results in a *maximum* of Gaussians

potential; the two potentials should later be combined. Maintaining these two separate types of potentials is inconvenient, as they are not closed under the required operations, something that is highly unsatisfactory from a computational point of view. This makes even more necessary the definition of approximate methods for (approximately) solving the MPE and MAP problems. Such methods are discussed in Section 4.2.2.

4.2.2 Approximate MPE Inference

We now turn to finding an *approximate* solution to the MPE problem. MAP inference has been formally introduced in Section 3.1, and the term "approximate" here refers to the fact that the given solutions are not guaranteed to be optimal. This is the price to pay for the ability to process large amounts of queries arriving as a data stream.

Suppose we have a CLG BN with joint probability distribution $p(\mathbf{x})$ and observed variables $\mathbf{X}_E$. By definition of a CLG network, discrete variables are always placed above the continuous ones (i.e., they have no continuous parents). Thus, we can choose a topological order of the nodes in the DAG so that the discrete variables always appear before the continuous ones, i.e., we can find some suitable ordering of the variables in the network such that it can be traversed so that parents are always visited before their children.

As a consequence, if there is no evidence ($\mathbf{X}_E = \varnothing$) and we have a fixed configuration of values for all the discrete variables, it is easy to show that the configuration of continuous variables of maximum probability (density) is obtained just by taking the mean of each variable conditional on the fixed discrete configuration. It can be easily computed visiting the continuous variables in topological order and evaluating their conditional means, which can always be computed as they are functions of their parent nodes.

This fact is also true in more general situations, for instance if there is evidence only for discrete variables ($\mathbf{X}_E \subset \mathbf{X}_D$) or in continuous variables that are direct descendant of discrete ones. In general, this is true whenever

$$\mathrm{pa}\left(\mathbf{X}_C \cap \mathbf{X}_E\right) \subseteq \mathbf{X}_D \cup \mathbf{X}_E, \tag{14}$$

i.e., the parents of any observed continuous variable are either discrete or observed. On the contrary, the previous procedure cannot be applied if there are observed continuous variables with continuous parents that are unobserved.

In those situations, the topological order of the continuous variables can be modified by using the so-called *inversion of arcs* [34,35] or *arc reversal*, resulting in a new DAG where observed continuous variables ($\mathbf{X}_C \cap \mathbf{X}_E$) appear before the unobserved ones ($\mathbf{X}_C \setminus \mathbf{X}_E$) in the order. However, this procedure can be computationally demanding in practice, and more efficient approximate techniques are to be sought.

4.2.2.1 General setup

Taking the comments above into account, we have developed proposals for finding an approximate solution to the MPE problem using well-known optimization algorithms, more precisely, our proposals rely on simulated annealing [36] and hill-climbing [37]. Both methods can be seen as search procedures that explore the space of possible solutions and require a method for generating the next potential solution to explore, given the current potential solution. In this case, a potential solution is a configuration of the unobserved variables in the network. Our proposal for generating a new potential solution is given in Algorithm 3.

Note that if we move from one configuration of the discrete variables to another by selecting new values for all of them, we are performing *global search*. On the contrary, if we only change values of a few (randomly chosen) discrete variables, then we are doing *local search* [37].

Input: A Bayesian network of variables $\mathbf{X} = \mathbf{X}_D \cup \mathbf{X}_C$ and joint probability distribution $p(\mathbf{x})$, some evidence $\mathbf{X}_E = \mathbf{x}_E$ (optional), and an initial assignment $\mathbf{X} = \mathbf{x}$

Result: A new configuration $\mathbf{x}_1$

1 $\mathbf{x}_1 \leftarrow \mathbf{x}$;
2 `/* to start, change some (or all) of the discrete variables: */`
3 choose randomly a discrete variable $X_j \in \mathbf{X}_D \setminus \mathbf{X}_E$;
4 pick a valid state x_j of variable X_j ;
5 change the value of variable X_j to x_j in $\mathbf{x}_1$;
6 repeat lines 3 – 5 a fixed number m of times (local search) or for every discrete variable not in the evidence (global search) ;
7 `/* now obtain values for the continuous variables, in two steps */`
8 `/* first, simulate values for variables not satisfying Equation (14) */`
9 determine the set $\mathbf{X}_{C_1}$ containing variables $Z \in \mathbf{X}_C$ with $Pa(Z) \not\subseteq \mathbf{X}_D \cap \mathbf{X}_E$ and its complementary $\mathbf{X}_{C_2} = \mathbf{X}_C \setminus \mathbf{X}_{C_1}$;
10 for every variable in $\mathbf{X}_{C_1} \setminus \mathbf{X}_E$, following the causal order, simulate a value according to its conditional distribution given $\mathbf{x}_1$, and add it to $\mathbf{x}_1$;
11 `/* finally, assign the mean of the CLG model for the rest of continuous variables */`
12 for every variable in $\mathbf{X}_{C_2} \setminus \mathbf{X}_E$, following the causal order, pick its conditional mean given $\mathbf{x}_1$ and add it to $\mathbf{x}_1$;
13 `/*` $\mathbf{x}_1$ `now holds a new instantiation of the variables */`
14 **return** $\mathbf{x}_1$

Algorithm 3: Procedure to move from one configuration of variables in a hybrid BN to another

In the case of local search, our experience suggest that a small value of m (the number

of discrete variables whose value is changed), between 2 and 5, usually provides good results. The choice $m = 1$ is not adequate for large networks since the number of configurations to explore is very large, and each "step" in this case changes very little. Of course m must be less or equal than the number of unobserved discrete variables ($|\mathbf{X}_D \setminus \mathbf{X}_E|$), and global search is performed if m is set to this value. In the experiments reported in Section 4.2.5 we chose the value $m = 3$.

4.2.2.2 Simulated annealing methods

Our proposal for MPE using simulated annealing is described in Algorithm 4. The simulated annealing algorithm was designed to avoid local maxima during the search process, allowing to move to worse potential solutions than the current one, specially during the initial simulation steps. This is controlled by the so-called *temperature* parameter. The algorithm uses a standard stopping criterion: it stops when the system is *cold* enough ($R < \varepsilon$). Nonetheless, the three free parameters R, α and ϵ must be set a priori, which can provoke the anticipated termination of the algorithm if they are not chosen appropriately.

4.2.2.3 Hill climbing-based method

Unlike simulated annealing, the *hill-climber* method never moves to a configuration worse than the current one when exploring the search space. The pseudocode, given in Algorithm 5, is similar to the simulated annealing method, but simpler because of this. Another significant difference between simulated annealing and hill climber is the user must specify a stopping criterion before executing a hill climbing. Some popular choices are to establish a maximum number of iterations, a maximum number of consecutive non-improving iterations, or a target probability threshold (the latter is difficult to implement, due to the variability in size and structure of the BNs). In practice, we found that setting a maximum number of about $30 - 50$ iterations produces good results.

Again, different versions of the algorithm can be obtained by changing the stopping criterion or the type of search used by Algorithm 3 (global or local with a specific number of movements m).

4.2.2.4 Other methods: sampling and deterministic

We have also explored other possibilities to obtain an MPE estimate. A simple approximate procedure is just to draw a large sample from the network according to the evidence, and pick the configuration that gives the highest joint probability. We call this the *plain sampling method*. The method is simple to implement, but may obviously require an unpractically large number of samples to obtain good results. Nevertheless. it is included in the AMIDST toolbox, where it serves as a naïve baseline for testing

Input: A Bayesian network of variables $\mathbf{X} = \mathbf{X}_D \cup \mathbf{X}_C$ and joint probability distribution $p(\mathbf{x})$, some evidence $\mathbf{X}_E = \mathbf{x}_E$ (optional), and an initial assignment $\mathbf{X} = \mathbf{x}_0$ (a guess of the MPE estimate)

Result: An approximate MPE estimate

```
/* fix the temperature parameters or use default values:          */
R ← 10000 ;
α ← 0.95 ;
ε ← 0.01 ;
obtain an initial assignment X = x_0, according the evidence x_E ;
while R ≥ ε do
    x_1 ← x_0 ;
    modify x_1 according to Algorithm 3, with local or global search ;
    if p(x_1) > p(x_0) then
        x_0 ← x_1 ;
    else
        /* take a worse solution with certain probability:        */
        simulate a random number τ uniformly in [0, 1] ;
        if τ < e^{-(p(x_0)-p(x_1))/R} then
            x_0 ← x_1 ;
        end
    end
    R ← αR ;
end
/* x_0 holds the MPE estimate                                     */
return x_0
```

Algorithm 4: Generic *simulated annealing* algorithm for MPE on a Bayesian network

more sophisticated algorithms.

Another alternative is to explore all the possible configurations of the discrete variables. The main use of this approach is for validation purposes, since it will only be feasible for very small networks. This *exhaustive search* approach is implemented as follows. For any configuration of the discrete variables, we assign values to every continuous variable using Algorithm 3. Afterwards, we choose between all the possible configurations the one maximizing the joint distribution. Therefore, it is a deterministic method. It can be implemented recursively, and it is exact when there is no evidence (or when observed continuous variables do not have continuous unobserved parents, i.e., when Equation (14) holds).

This recursive implementation of the sequential search of the MPE can be initialized by using `bestConfig`$(\mathbf{X} = \boldsymbol{x}, j = 1)$, and it returns the exact MPE if Equation (14) is satisfied.

Input: A Bayesian network of variables $\mathbf{X} = \mathbf{X}_D \cup \mathbf{X}_C$ and joint probability distribution $p(\mathbf{x})$, some evidence $\mathbf{X}_E = \mathbf{x}_E$ (optional).
Result: An approximate MPE estimate

```
obtain an initial assignment X = x_0, according to the evidence x_E ;
while stopping criterion not satisfied do
    x_1 ← x_0 ;
    modify x_1 according to Algorithm 3, with local or global search ;
    if p(x_1) > p(x_0) then
        x_0 ← x_1 ;
    end
end
/* x_0 holds the MPE estimate                                        */
return x_0
```

Algorithm 5: Generic *hill climbing* algorithm for MPE on a Bayesian network

4.2.3 Approximate MAP inference

To perform MAP inference, the previously discussed algorithms for MPE can be used with some modifications. The main problem is that in MAP, we cannot use the probability distribution $p(\mathbf{x})$ of the Bayesian network directly, since we have partial assignments (we lack of values for some variables). Thus, to compute or estimate the probability of a partial instantiation, the original distribution has to be marginalized down to the variables of interest. Following the previous notation, if we are interested in some specific variables in the network $\mathbf{X}_I \subset \mathbf{X} = \mathbf{X}_D \cup \mathbf{X}_C$, and assuming there is no evidence ($\mathbf{X}_E = \varnothing$), then

$$p(\mathbf{x}_I) = \sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_D \setminus \mathbf{x}_I}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_C \setminus \mathbf{x}_I}} p(\mathbf{x}) \mathrm{d}\mathbf{x}_C,$$

which is in general difficult to compute. Alternatively, if we consider that some evidence $\mathbf{X}_E = \mathbf{x}_E$ is present, instead of the expression above we have

$$p(\mathbf{x}_I|\mathbf{x}_E) = \frac{p(\mathbf{x}_I, \mathbf{x}_E)}{p(\mathbf{x}_E)} = \frac{\sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_D \setminus (\mathbf{x}_I \cup \mathbf{x}_E)}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_C \setminus (\mathbf{x}_I \cup \mathbf{x}_E)}} p(\mathbf{x}, \mathbf{x}_E) \mathrm{d}\mathbf{x}_C}{\sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_D \setminus \mathbf{x}_I}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_C \setminus \mathbf{x}_I}} p(\mathbf{x}, \mathbf{x}_E) \mathrm{d}\mathbf{x}_C}. \tag{15}$$

As the aim of the MAP technique is finding the configuration $\mathbf{x}_I^*$ of the variables of interest with highest probability, the denominator in the previous expression can be discarded, since it is constant given the evidence. Therefore, the MAP problem amounts to finding the configuration

$$\mathbf{x}_I^* = \arg \max_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} p(\mathbf{x}_I|\mathbf{x}_E),$$

```
Function bestConfig(x, j) {
    Input: A Bayesian network of variables X = X_D ∪ X_C with
           X_D = {Y_1, ..., Y_{N_D}} and joint probability distribution p(x), some
           evidence X_E = x_E (optional), a full assignment X = x, and a discrete
           variable index j.
1   if j = N_D then
2       /* Y_j is the last discrete variable                              */
3       return x ;
4   end
5   if Y_j ∈ X_E then
6       return bestConfig(x , j + 1) ;
7   end
8   s ← number of states of Y_j, the j-th discrete variable;
9   modify in x the value of Y_j, to obtain assignments x_1, ..., x_s with x_i
    satisfying Y_j = i;
10  compute b_i= bestConfig(x_i, j + 1), ∀i = 1, ..., s;
11  for each b_i, assign values to the continuous variables X_C;
12  return b = arg max_{b_i} p(b_i);
13 }
```

Algorithm 6: The recursive function `bestConfig` is used to perform the exhaustive search for the MPE

where $p(\mathbf{x}_I|\mathbf{x}_E)$ is calculated using Equation (15).

In order to simplify the problem of finding $\mathbf{x}_I^*$, the integrals or summations above can be approximated, using the importance sampling method. As an illustration, assume a simplified scenario in which we no evidence and there is only one variable that is not of interest, namely the continuous variable Z (i.e. $\mathbf{X} = \mathbf{X}_I \cup Z$ and $Z \notin \mathbf{X}_I$). Then,

$$p(\mathbf{x}_I) = \int_{z \in \Omega_Z} p(\mathbf{x}_I, z) \mathrm{d}z \propto \sum_{i=1}^{n} p(\mathbf{x}_I, z_i) \tag{16}$$

where z_i is drawn according to some distribution f^* (to be discussed later). This can be done since

$$\begin{aligned} \int_{z \in \Omega_Z} p(\mathbf{x}_I, z) \mathrm{d}z &= \int_{z \in \Omega_Z} \frac{p(\mathbf{x}_I, z)}{f^*(\mathbf{x}_I, z)} f^*(\mathbf{x}_I, z) \mathrm{d}z \\ &= \mathbb{E}_{f^*}\left[\frac{p(\mathbf{x}_I, z)}{f^*(\mathbf{x}_I, z)}\right] \approx \frac{1}{n} \sum_{i=1}^{n} \frac{p(\mathbf{x}_I, z_i)}{f^*(\mathbf{x}_I, z_i)} \end{aligned}$$

whenever z_i are i.i.d. samples drawn from distribution of f^*.

Similarly to the importance sampling algorithm in Section 4.1, we can choose f^* as the product of the conditional distributions in the network. As soon as we have an

approximate way of computing the probability of a candidate configuration over the variables of interest $\mathbf{x}_I$, the algorithms given for MPE in the previous section can be easily adapted to MAP We only have to change the use of the joint probability distribution $p(\mathbf{x})$ of the Bayesian network for $p(\mathbf{x}_I)$, where the variables not of interest have been marginalized out. The previous ideas can be easily extended to more general situations, when there is evidence, and in which marginalization has to be made over any number of discrete and continuous variables.

The MAP version of the sampling procedure described before for MPE involves drawing a large sample from the Bayesian network $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M \sim \mathbf{X}$ (according to the evidence $\mathbf{X}_E = \mathbf{x}_E$), then grouping samples by the values of variables of interest, and averaging the probabilities for each found configuration:

$$p(\mathbf{x}_I) \propto \text{average}(p(\mathbf{x}_k)), \quad \text{where } \mathbf{x}_k = (\mathbf{x}_I, \text{rest}), \quad \forall \mathbf{x}_I \in \Omega_{\mathbf{X}_I}$$

Then, the MAP estimate is chosen as

$$\mathbf{x}_I^* = \arg \max_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} p(\mathbf{x}_I) = \arg \max_{\mathbf{x}_k = (\mathbf{x}_I, \text{rest})} \text{average}(p(\mathbf{x}_k))$$

Afterwards, if the actual probabilities are required, they can be computed, normalizing the values above with their sum:

$$\alpha = \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} \text{average}(p(\mathbf{x}_k)) \Rightarrow p(\mathbf{x}_I) = \frac{1}{\alpha} \text{average}(p(\mathbf{x}_k))$$

Therefore, this sampling method for MAP is adequate when the variables of interest are all discrete (sampled values for continuous variables will not be repeated along the sample, avoiding the possibility of making groups by their values).

4.2.4 MPE and MAP in AMIDST

In the AMIDST toolbox, the methods described above are implemented using Java 8 Streams. Since hill climbing and simulated annealing are iterative algorithms, a simple manner of parallelizing the execution is the following. First, draw a certain number of samples from the Bayesian network according to the probability distribution $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M \sim \mathbf{X}$ (all of them satisfying the evidence $\mathbf{X}_E = \mathbf{x}_E$, if there is any). Then use each sample as the initial guess for any of the optimization algorithms (hill climbing or simulated annealing), which can be run in parallel to obtain M estimates of the MPE/MAP:

$$\mathbf{x}_j^* = \text{optimizationAlgorithm}(\mathbf{x}_j), \forall j = 1, 2, \ldots, M$$

which are candidates to optimal solution. Finally, pick the best found solution

$$\mathbf{x}^* = \arg \max_{1 \leq j \leq M} p(\mathbf{x}_j^*),$$

as the MPE/MAP or, if interested, pick the set of K best-performing solutions, those with the greatest probability.

Figure 11 shows the stream flow of the Map/Reduce scheme for MPE and MAP Inference. If the parallel mode is activated, then several instantiations of the selected search algorithm are run, based on different starting points. Multiple instantiations of the local variations can also be executed.

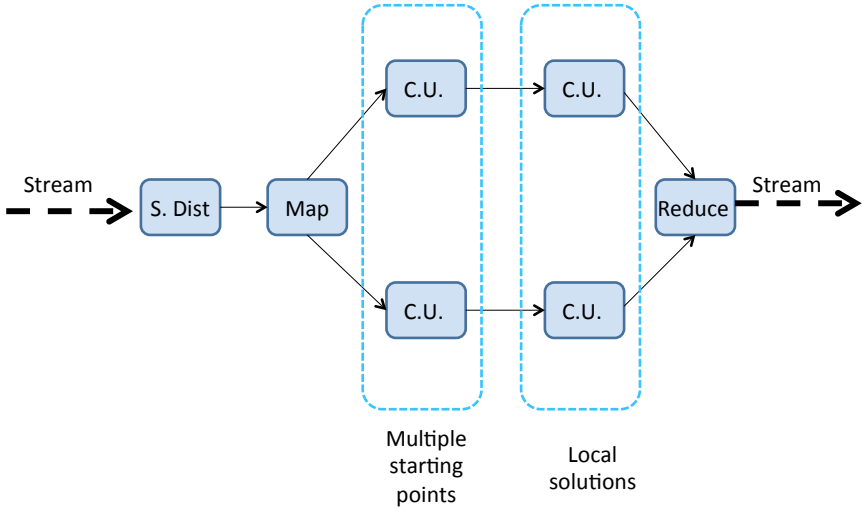


Figure 11: Illustration of the Map/Reduce scheme for MPE and MAP Inference.

To summarize, the following algorithms for MPE and MAP are currently available in the AMIDST Toolbox:

- Simulated annealing with global search
- Simulated annealing with local search
- Hill climbing with global search
- Hill climbing with local search
- Plain sampling method
- Exhaustive search (not parallel; exact if Equation (14) holds; only available for MPE)

4.2.5 Experimental evaluation

The algorithms for finding MPE/MAP were computationally validated through a number of experiments.

In the first set-up, a hybrid CLG Bayesian network of 20 discrete binary variables and 20 continuous variables was randomly generated. Then, 5% of the variables were randomly observed and given (random) evidence. To measure the execution time for simulated annealing (SA) and hill-climbing (HC), either with local or global search, 100 starting points were given and the algorithms ran for 200 iterations. For the sampling method, 10.000 samples were generated. Also, the influence of the number of cores was evaluated, by using $1, 2, 4, 8, 16, 24$ and 32. Each run was repeated 10 times and the run-time average was computed and it is shown in Figure 12, left panel.

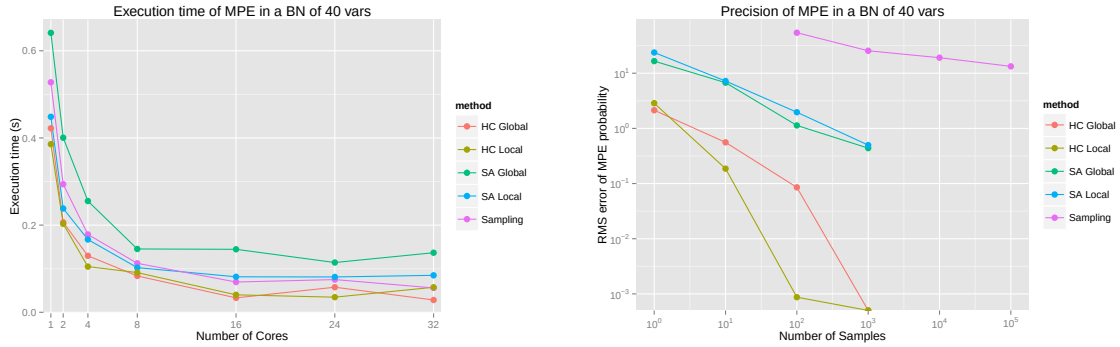


Figure 12: Performance of MPE methods in a hybrid CLG Bayesian Network of 20 discrete and 20 continuous variables

Since the BN size in this case is relatively small, the exhaustive search algorithm was employed to find the exact MPE. Then, the precision of the MPE estimations given by each method was analyzed, computing the RMS error of the log-probabilities of every found MPE estimator versus the actual MPE, as a function of the number of starting points (for SA and HC, 1, 10, 100 and 1000 initial guesses) and of the number of samples (for the sampling method, 100, 1000, 10.000 and 100.000 samples). The accuracy results are shown in Figure 12, right panel. Notice that the values on the x-axis has a different meaning for the optimization algorithms and for the sampling method.

Another experiment was carried out with the same set-up, but using a different, larger network of 100 discrete binary variables and 100 continuous variables. Again, the model randomly generated. The rest of the parameters were set to the values used above. In this case, the model was too large to find the *actual* MPE, and it was therefore estimated by running the optimization algorithms with a huge number of iterations and choosing the best found estimator. The results for execution time and precision are depicted in Figure 13

The results show that the hill-climbing method with local search outperforms the others

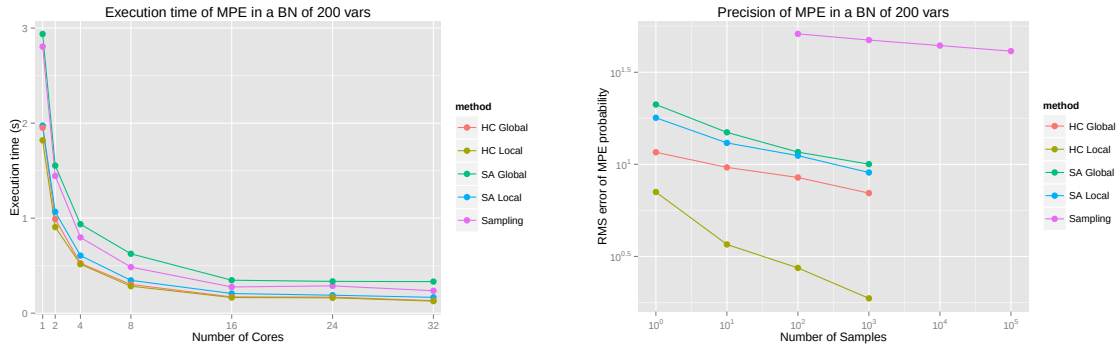


Figure 13: Performance of MPE methods in a hybrid CLG Bayesian Network of 100 discrete and 100 continuous variables

both in precision, whereas the execution times are similar with respect to its global search version, and better than the simulated annealing algorithms. The results from the naïve sampling are not directly comparable to the others since this is a totally different approach. Thus, its sample size was set to yield execution times in the same order of the optimization methods, but it clearly gives much worse approximations to the MPE estimate in both cases.

Finally, the MAP algorithms were also tested in a randomly generated CLG model with 50 discrete binary variables and 50 continuous variables. For SA and HC, 50 starting points were given and the algorithms ran for 100 iterations. For the sampling method, a sample of size 20.000 was generated. Each experiment was run with the several number of cores and in each case the runtime was averaged after 10 executions. Similarly, the *actual* MAP estimate needed to calculate the precisions was estimated by running the optimization algorithms with a extremely large number of iterations and choosing the best found estimator. Then, the accuracy of each method was calculated by the RSM error of the log-probability of the found estimator, as a function of the number of starting points (for SA and HC, 1, 10, 100 and 1000 initial guesses) and of the number of samples (for the sampling method, 100, 1000, 10.000 and 100.000 samples). The MAP results are plotted in Figure 14.

The results are consistent with those of the MPE methods, with a higher execution time due to the intrinsic difficulty of estimating the probabilities of partial assignments required for MAP calculations. As a general conclusion, the best-performing method was hill-climbing with local search, which gives good estimates of MPE/MAP in a very reasonable time, outperforming the rest of the analyzed alternatives.

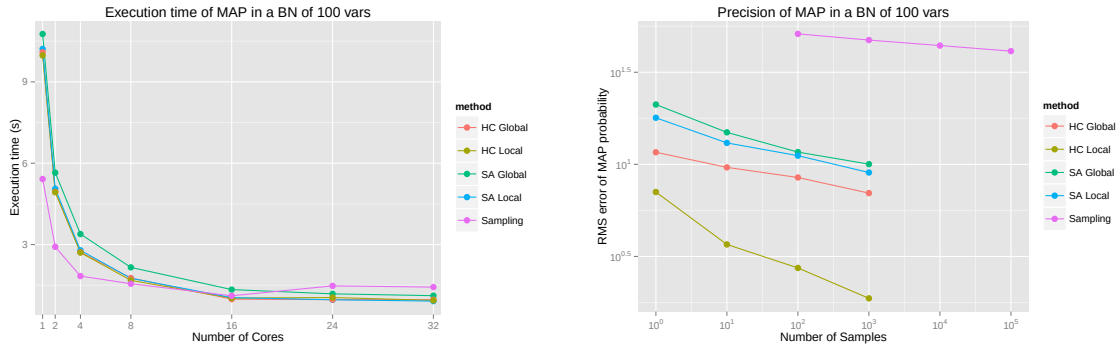


Figure 14: Performance of MAP methods in a hybrid CLG Bayesian Network of 50 discrete and 50 continuous variables

5 Inference in dynamic models

5.1 Probabilistic inference in dynamic models

5.1.1 Problem definition and opportunities for scalabillity

By a *dynamic* Bayesian networks we understand a Bayesian network, which explicitly models development over time. An example of a dynamic Bayesian network (DBN) is given in Figure 15. This is an *input-output* model, where the control (or input) variables at time t, $\boldsymbol{u}_t$, and the output variables, $\boldsymbol{e}_t$, are observed. Probabilistic inference in these models amount to calculating the posterior distribution over the unobserved *belief-state* at time t given the observable variables, i.e., $p(\mathbf{X}_t|\mathbf{e}_{1:t}, \boldsymbol{u}_{1:t})$.

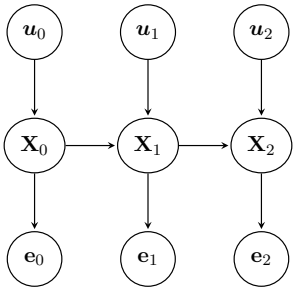


Figure 15: A three time slice dynamic Bayesian network. This is a so-called input-output model.

When working with data streams in AMIDST, we utilize the *Markov* assumption encoded in the AMIDST model class (described in Deliverable 2.2 [1]). This assumption states that at any time t, the variables prior to t, i.e., $\{\mathbf{X}_{0:t-1}, \mathbf{E}_{0:t-1}, \mathbf{U}_{0:t-1}\}$ are conditionally independent of the future variables $\{\mathbf{X}_{t+1}, \mathbf{E}_{t+1}, \mathbf{U}_{t+1}\}$ given the belief state at time t,

$\mathbf{X}_t$. To calculate $p(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}, \boldsymbol{u}_{1:t+1})$ we therefore store $p(\mathbf{X}_t|\mathbf{e}_{1:t}, \boldsymbol{u}_{1:t})$, the posterior over the belief state at time t, also known as the *frontier*. The posterior is used as input to the *prediction*, which generates $p(\mathbf{X}_{t+1}|\mathbf{e}_{1:t}, \boldsymbol{u}_{1:t})$, followed by the *updating*, which is utilizing information from the observed values $\{\mathbf{u}_{t+1}, \mathbf{e}_{t+1}\}$ to generate the desired $p(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}, \boldsymbol{u}_{1:t+1})$. One of the main drivers of computational complexity is to represent and update the frontier. If the frontier contains N discrete variables each having Q states, and we cannot utilize (conditional) independences between the variables in the frontier, the complexity of the operation is $O\left(Q^N\right)$.

Inference in dynamic Bayesian networks obviously shares the computational difficulties of regular Bayesian networks. Additionally, in the dynamic case we are also often faced with extra challenges. One particular difficulty is the *entanglement problem*, which is exemplified in Figure 16. In this model the belief-state at time t contains the variables $\mathbf{X}_t = \{X_t^1, X_t^2, X_t^3\}$, there are no control-variables in this model, and the output variables at time t are in the vector $\mathbf{e}_t$. The model for the belief-state is sparsely connected, and one may expect that this would lead to reduced computational complexity when calculating the posterior over the belief-state. Unfortunately, this is not the case, as we can see from Figure 16, , all variables $\mathbf{x}_t$ describing the belief state have become dependent after observing $\{\mathbf{e}_{1:t}\}$ starting from a certain time step t. Consequently, we cannot represent the exact belief state $p(\mathbf{x}_t|\mathbf{e}_{1:t}, \mathbf{u}_{1:t})$ in factorised form [38, 39].

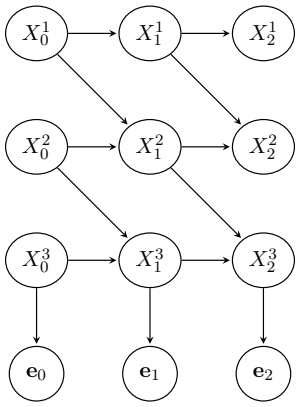


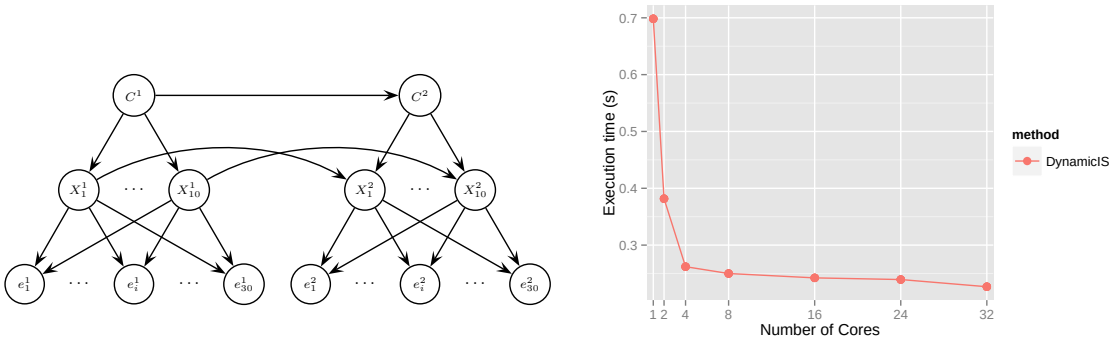
Figure 16: A three time slice dynamic Bayesian network. All variables used to describe the belief state (i.e. X_2^1, X_2^2, and X_2^3) have become correlated at time $t = 2$.

Our solution is to utilize the *factored frontier* algorithm [40], where the distribution over the belief-state is represented in factorized form, using the approximation $p(\mathbf{X}_t|\mathbf{e}_{1:t}, \mathbf{u}_{1:t}) \approx \prod_i p(\mathbf{X}_t^i|\mathbf{e}_{1:t}, \mathbf{u}_{1:t})$, which can be represented in $O\left(Q \cdot N\right)$ instead of the $O\left(Q^N\right)$ required for representing the exact frontier. This vehicle for increased efficiency is used both by the VMP and IS implementation to perform efficient probabilistic inference in dynamic models.

5.1.2 Experimental evaluation

In this section we discuss a simple experiment to show the scalability (in terms of available processors) of the dynamic inference in the AMIDST Toolbox provided by the factored frontier algorithm.

In the experiment we do probabilistic inference in the dynamic Bayesian network model displayed in Figure 17 (a). This model has 30 observed variables $\{e_1, \ldots, e_{30}\}$ and 11 unobserved or hidden variables $\{C, X_1, \ldots, X_{10}\}$ per time-step. The conditional distributions were chosen randomly. A set of randomly generated samples were fed into the model at each time-step, and we measured the time in seconds required to process each sample; the reported results were averages over 10 independent runs. We utilized the factored frontier algorithm to speed up the calculations as described above, and employed importance sampling as the core inference component in order to leverage the parallelization potential described in Section 4.1.



(a) DBN example for Dynamic IS inference.

(b) Time in seconds required to process each sample.

Figure 17: DBN model for inference using Importance Sampling with the Factored Frontier algorithm (left) and experimental results (right).

Figure 17 (b) shows the average time in seconds required to filter a single sample. This time is reduced from 0.7 seconds using a single core to around 0.25 seconds with 4 cores, and continues to decrease, although at a slower pace, as the number of core increases. We note again that the higher efficiency is achieved thanks to two key aspects of the inference process, namely the factored frontier scheme, and the parallelization capabilities of the underlying inference algorithm.

5.2 Abductive inference in dynamic models

5.2.1 Introduction

We will now sketch an approach for finding the *maximum a posteriori* (MAP) configuration of a subset of the variables in an AMIDST model. We shall focus on models with a dynamic structure, and where the variables of interest consist of discrete variables (one for each time-step) that define the hidden states of the model.

Parts of the methodological developments will assume that a mean-field based variational message passing procedure constitute the underlying inference engine as described in Section 3.2. Thus, we will try to address the apparent conflict in requesting a MAP configuration based on a mean field approximation; straight-forward application of variational mean-field for finding MAP configurations will only provide an approximation of the marginal MAP configuration, which is not sufficient.

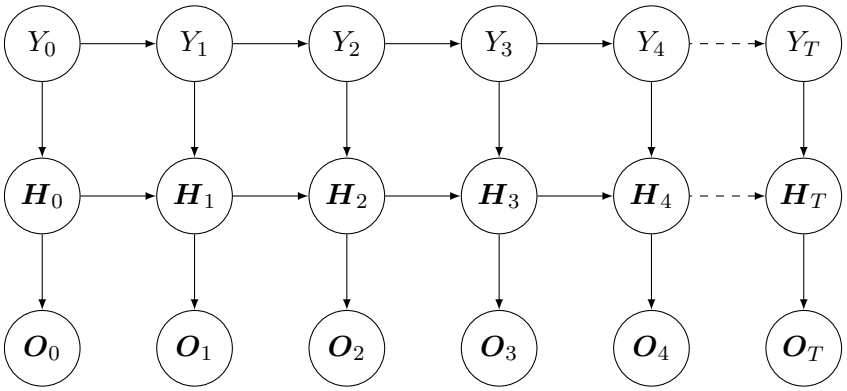


Figure 18: A simplified AMIDST model.

We shall only consider dynamic AMIDST models as specified in Deliverable 2.2. Furthermore, for ease of presentation, the description of the method will be given relative to the simplified model structure shown in Figure 18, i.e., some of the inter- and intra-time slice relations in the original model have been dropped. The proposed method does, however, readily generalize to full AMIDST models. For a given time step t, we denote the observable variables with $\boldsymbol{O}_t$, the hidden variables with $\boldsymbol{H}_t$, and the control variables $\boldsymbol{Y}_t$. The model can, in principle, support a structured representation of $\boldsymbol{Y}_t$, but for ease of exposition we shall assume that $\boldsymbol{Y}_t$ is a singleton and emphasize this by dropping the boldface notation and simply use Y_t. MAP inference takes place after model construction, so we will assume access to a fully specified AMIDST model.

5.2.2 MAP in dynamic AMIDST models

The general idea pursued is to establish (variational) posterior distributions over subsets of MAP variables. Once these posterior distributions (over possibly non-disjoint subsets) have been found we may define a joint distributions over all the MAP variables based on a suitable combination of the posterior distributions over these subsets.

5.2.2.1 Generalized mean field

One approach to this problem is to apply a generalized mean-field algorithm for different partitionings of the MAP variables $\boldsymbol{Y}_{0:T}$. Each partitioning P_i will produce a posterior variational distribution over $\boldsymbol{Y}_{0:T}$ that factorizes according to the partitioning and captures a subset of the posterior dependencies between the MAP variables.

Two examples of pair-wise partitionings of the MAP variables are illustrated in Figure 19 (a), one where the partitions are indicated using a blue line, the other with red. We have grouped together pairs of MAP variables appearing in consecutive time slices. The two partitionings not only capture different dependency relations between the MAP variables, they also represent the only possible pair-wise partitionings of consecutive MAP variables.

The variational posterior distributions obtained for different partitionings of the MAP variables $\boldsymbol{Y}_{0:T}$ can be used to define a posterior joint distribution over $\boldsymbol{Y}_{0:T}$. For example, we can combine the results from, say, D different partitionings as

$$Q^*(Y_0, \ldots, Y_T|\boldsymbol{o}_{0:T}) = \frac{1}{D}\sum_{i=1}^{D} Q_{P_i}(Y_0, \ldots, Y_T|\boldsymbol{o}_{0:T}), \tag{17}$$

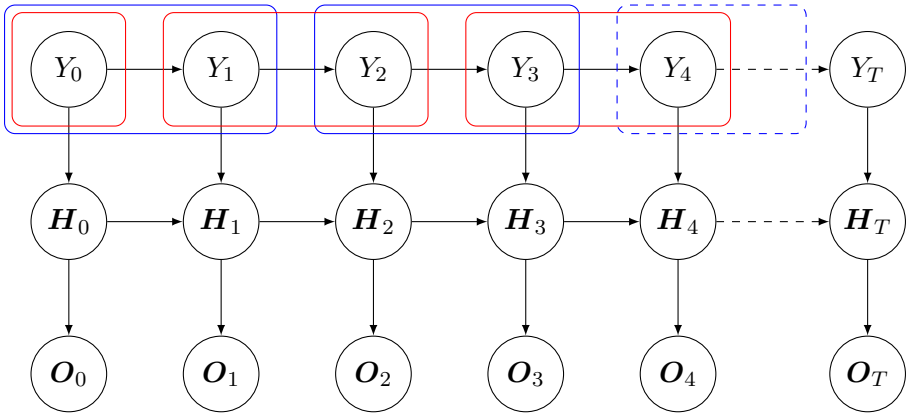
where $Q_{P_i}(Y_0, \ldots, Y_T|\boldsymbol{o}_{1:T})$ is the variational posterior distribution captured by partitioning P_i. We can then find an approximation to the original MAP problem by finding a configuration maximizing $Q^*(Y_0, \ldots, Y_T|\boldsymbol{o}_{1:T})$. In the remainder of this section we will drop the explicit conditioning on $\boldsymbol{o}_{1:T}$ when referring to the variational distributions.

For the two partitionings, P_1 and P_2, in Figure 19 (a) we can exploit the factorization of the variational distributions to obtain

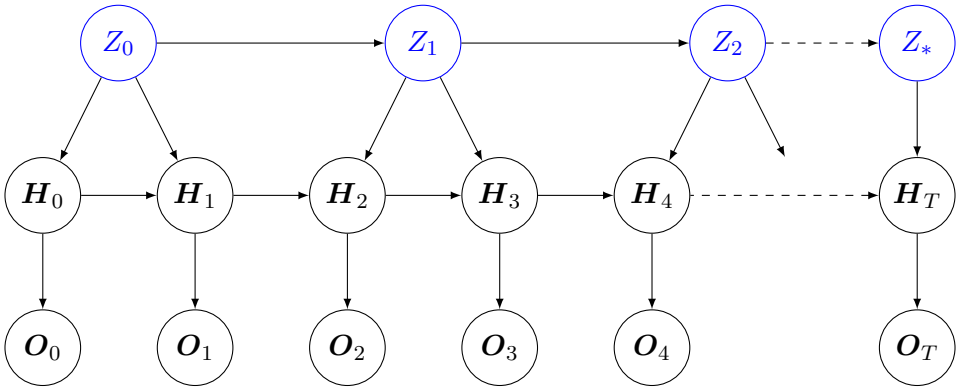
$$Q^*(Y_0, \ldots, Y_T) = \frac{1}{2}\sum_{i=1,2}\prod_{j=0}^{|P_i|} Q_{P_i}(Z_j^{(i)}), \tag{18}$$

where $Z_j^{(i)}$ is the j'th set of MAP variables under the i'th partitioning.

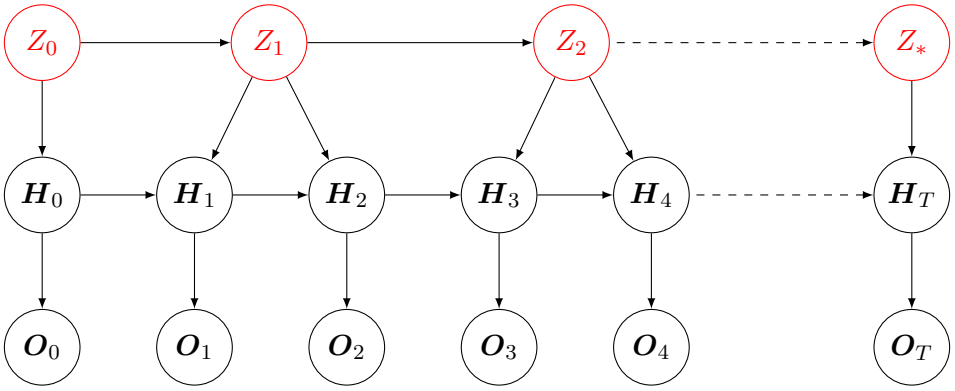
The joint distribution defined in Equation (18) follows a first order Markov model, where



(a) Original model. The two only possible partitions for pair-wise relations of MAP variables are marked in red and blue.



(b) Local-dependency model equivalent to the *blue* partitioning.



(c) Local-dependency model equivalent to the *red* partitioning.

Figure 19: The figure illustrates the two possible pair-wise partitionings and combinations of the MAP variables in the simplified AMIDST model.

the conditional distribution $Q^*(Y_i|Y_{i-1})$ is given by

$$\begin{aligned} Q^*(Y_i|Y_{i-1}) &\propto \sum_{\boldsymbol{Y}_{0:T}\setminus\{Y_i,Y_{i-1}\}} Q^*(Y_0,\ldots,Y_T) \\ &= \frac{1}{2}Q_{P_1}(Y_{i-1},Y_i) + \frac{1}{2}\sum_{Y_{i-2}} Q_{P_2}(Y_{i-2},Y_{i-1}) \sum_{Y_{i+1}} Q_{P_2}(Y_i,Y_{i+1}), \end{aligned} \tag{19}$$

assuming that partitioning P_1 defines the subset $\{Y_{i-1}, Y_i\}$; otherwise the indexes should be reversed.

An approximate MAP configuration can now be found by applying the standard Viterbi algorithm based on the Markov model defined by the distributions above.

Note: Rather than working with pair-wise relations, one could instead consider triples of consecutive MAP variables. This would produce three different partitionings, which would induce a second-order Markov chain.

5.2.2.2 Model transformation

Rather than working with generalized mean-field algorithms, that would potentially have to be tailored to the different partitionings, we can instead capture the local dependencies between the MAP variables by aggregating/combining variables that appear together in a subset of a given partitioning. Subsequent mean-field inference would then provide posterior distributions over the aggregated variables or, equivalently, the corresponding MAP-subsets. That is, any given partitioning of the MAP variables has an equivalent aggregated model, which we shall refer to as a *local-dependency* model. To illustrate the approach, Figure 19 (b) shows the structure of the local dependency model induced by the *blue* partitioning in Figure 19 (a), and the structure in Figure 19 (c) reflects the *red* partitioning.

In order to obtain a probabilistic graphical model for the model structure in Figure 19 (b), we need to populate the model with probability distributions and density functions. To simplify notation, let $Z_{j-1,j}$ denote the aggregated variable corresponding to $Y_{j-1}\times Y_j$ and let $\boldsymbol{H}_k$ be a child of $Z_{j-1,j}$ (thus $k = j-1$ or $k = j$). For these variables the required distributions and density functions can readily be calculated based on the original model:

$$P_i(Z_{j-1,j} = (y_{j-1}, y_j)|Z_{j-3,j-2} = (y_{j-3}, y_{j-2})) = P(y_j|y_{j-1})P(y_{j-1}|y_{j-2})$$

$$P_i(\boldsymbol{H}_k|Z_{j-1,j} = (y_{j-1}, y_j)) = \begin{cases} P(\boldsymbol{H}_k|Y_j = y_{j-1}) & \text{if } k = j-1 \\ P(\boldsymbol{H}_k|Y_j = y_j) & \text{if } k = j \end{cases}$$

The remaining distributions in the aggregated model can directly be read off the original model.

Using this aggregated model we can perform mean-field variational inference and still capture local dependencies between the MAP variables in the original model. The vari-

ational posterior distributions over these aggregated variables would then play the role of the generalized variational distributions in Equations (18) and (19).

The above consideration are, in principle, not limited to AMIDST models. The trick will in the end be to define a collection of partitionings of the MAP variables and combine the resulting distributions induced by these partitioning. Due to the way in which we defined the partitionings for the AMIDST model we were able to subsequently exploit the Viterbi algorithm for doing MAP inference. This will, however, not be possible in general, and one would then need to look for other (search) strategies for coming up with a MAP configuration.

5.2.3 Experimental evaluation

We have conducted two experiments, one aimed at testing the accuracy of the proposed method, and the other at checking its time complexity. For the first experiment (testing accuracy) we used 20 randomly generated CLG networks with 10 variables (7 continuous and 3 discrete) and 20 networks with 20 variables (15 continuous and 5 discrete). We chose small networks in order to be able to get the exact probability of the chosen MAP configurations. We compared the results obtained by the dynamic MAP method described above and the hill climbing method for MAP in static networks described in Section 4.2.4. The number of partitions for the dynamic MAP algorithm was set to $D = 2$. Note that the static MAP algorithm is run over the unrolled dynamic network, which is equivalent to the one depicted in Figure 19 (a). In all the test with 10 variables, 2 of them were observed, while the number of observed variables for the 20-variable networks was set to 5.

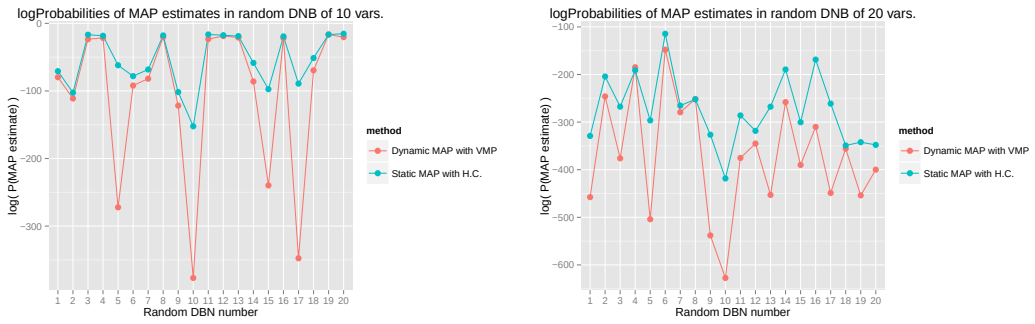


Figure 20: Log-probability of the MAP configurations obtained over randomly generated networks with 10 (left) and 20 (right) variables.

The result of the comparison is shown in Figure 20 for the 10-variable networks (left) and for the 20-variable networks (right). It can be seen how the static MAP algorithm with hill climbing achieves better results in most of the cases for both network sizes, and with more prominent differences for the larger network structure.

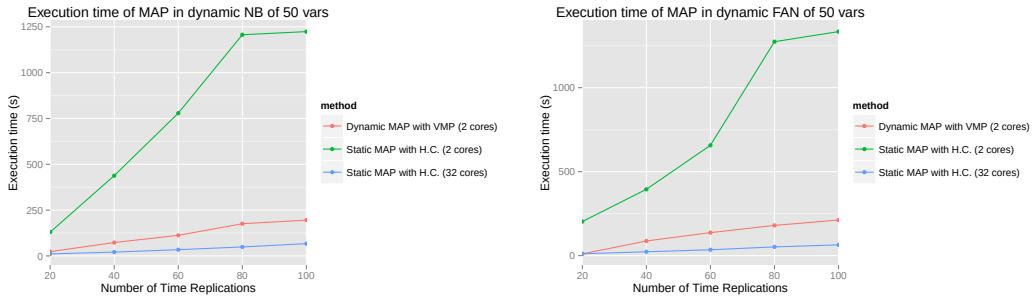


Figure 21: Run time of the tested MAP algorithms over networks with naive Bayes structure (left) and naive Bayes structure augmented with 25 extra links between features (right).

For testing the time complexity, we considered two networks, both of them resembling the models to be employed in the use cases corresponding to Work package 8. Both networks consist of 50 feature variables plus the class. The first one has a naive Bayes structure, while the second has a more complex structure, obtained by augmenting the naive Bayes with 25 extra links besides the 50 links connecting each feature with the class. We tested the same two algorithms as in the first experiments, considering time windows ranging from 20 to 100 time replications. The number of partitions for the dynamic MAP algorithm was again set to $D = 2$. It means that the only parallelisation applicable is the simultaneous execution of the VMP algorithm over the two partitionings. On the contrary, the static MAP algorithm, based on hill climbing, allows a much stronger parallelisation, by choosing different starting points to initiate the search, see Figure 11.

The results of the experiments are shown in Figure 21 for the naive Bayes (left) and augmented structures (right). The experiments were run on a dual-processor AMD Opteron 2.8GHz server with 32 cores and 64GB of RAM, running Linux Ubuntu 14.04.1 LTS. For the static MAP algorithm we tried 2 and 32 cores in order to have upper and lower bounds of the run times according to the level of parallelisation. For dynamic MAP, we chose only 2 cores, for the reason explained above.

It can be seen how the dynamic MAP algorithm is clearly more efficient when the number of cores is set to 2. This somehow pays off the difference in accuracy detected in the first experiment. However, the results show how the static algorithm scales up and is able to beat the dynamic method when the number of computing units (cores) increases.

6 Conclusions

This document summarises the different algorithms that have been used and implemented in the AMIDST toolbox in order to perform efficient inference in static and

dynamic conditional linear Gaussian Bayesian networks. For static models, we considered both variational message passing and importance sampling algorithms, and we approached the maximum a posteriori and most probable explanation using optimisation algorithms and Monto Carlo methods. For dynamic models, we used the factored frontier algorithm to define the dynamic counterpart of variational message passing and importance sampling algorithms, and we resorted to the mean field approximations to solve the maximum a posteriori problem.

The developments reported in this document correspond to Work package 3 and are strongly connected to the use case provided by Work package 8. At the time of preparing this document, the real AMIDST models from Work package 8 were not yet available, as their elicitation will require the use of learning algorithms to be developed in Work package 4, and will be constructed in forthcoming stages of Work package 8. Nevertheless, all the tests reported in this document have been carried out over artificial AMIDST models designed to resemble as much as possible the range of models that will be built in Work package 8. The final and thorough testing of the algorithms described in this document, with the real data and models, will be done within the context of that work package and reported in Deliverable 8.3.

A Exponential family representations

In this appendix we present the conditional and posterior exponential forms of selected distributions conditional on different parent configurations. For ease of exposition, we will refer to the variables/nodes based on the distributions assigned to them. Thus, we may, for instance, have a categorical child with a Dirichlet parent, which refers to a model fragment in which a (child) variable follows a categorical distribution conditional on a given parameter vector and that the distribution of this parameter vector in turn follows a Dirichlet distribution. In order to avoid notational clutter we will also sometimes simplify notation and drop indexes when these are clear from the context.

A.1 EF representation: A binary child given a binary parent

Let X and Y be two binary variables. The log-conditional probability of the child-node X given its parent-node Y is expressed as follows:

$$\begin{aligned}\ln p(X \mid Y) = I(X = x_1)I(Y = y_1) \ln P(x_1 \mid y_1) + I(X = x_2)I(Y = y_1) \ln P(x_2 \mid y_1) \\ +I(X = x_1)I(Y = y_2) \ln P(x_1 \mid y_2) + I(X = x_2)I(Y = y_2) \ln P(x_2 \mid y_2)\end{aligned}$$

This conditional probability distribution can be expressed in different exponential forms as follows:

C-form:

$$\ln p(X \mid Y) = \begin{pmatrix} I(Y = y_1) \ln P(x_1 \mid y_1) + I(Y = y_2) \ln P(x_1 \mid y_2) \\ I(Y = y_1) \ln P(x_2 \mid y_1) + I(Y = y_2) \ln P(x_2 \mid y_2) \end{pmatrix}^T \begin{pmatrix} I(X = x_1) \\ I(X = x_2) \end{pmatrix} - 0$$

P-form:

$$\ln p(X \mid Y) = \begin{pmatrix} I(X = x_1) \ln P(x_1 \mid y_1) + I(X = x_2) \ln P(x_2 \mid y_1) \\ I(X = x_1) \ln P(x_1 \mid y_2) + I(X = x_2) \ln P(x_2 \mid y_2) \end{pmatrix}^T \begin{pmatrix} I(Y = y_1) \\ I(Y = y_2) \end{pmatrix} - 0$$

A.2 EF representation: A categorical child given a set of categorical parents

Let X be a categorical variable with k possible values such that $k \geq 2$, and let $\mathbf{Y} = \{Y_1, \ldots, Y_n\}$ denote the set of parents of X, such that all of them are categorical. Each

parent Y_i, $1 \geq i \geq n$, has r_i possible values or states such that $r_i \geq 2$. A parental configuration for the child-node X is then a set of n elements $\{Y_1 = y_1, \ldots, Y_i = y_i, \ldots, Y_n = y_n\}$ such that y_i denotes a potential value of variable Y_i such that $1 \leq y_i \leq r_i$. Let $q = r_1 \times \ldots \times r_n$ denote the total number of parental configurations, and let $\mathbf{y}^l$ denote the l^{th} parental configuration such that $1 \leq l \leq q$.

The log-conditional probability of the child-node X given its parent-nodes $\boldsymbol{Y}$ can be expressed as follows:

$$\ln p(X \mid \boldsymbol{Y}) = \sum_{j=1}^{k} \sum_{l=1}^{q} I(X = x^j) I(\boldsymbol{Y} = \boldsymbol{y}^l) \ln p(x^j \mid \boldsymbol{y}^l)$$

Similarly the above log-conditional probability can be expressed in the following exponential forms:

C-form:

$$\begin{aligned} &\ln p(X \mid \boldsymbol{Y}) \\ &= \begin{pmatrix} I(\boldsymbol{Y} = \boldsymbol{y}^1) \ln p(x^1 \mid \boldsymbol{y}^1) + \ldots + I(\boldsymbol{Y} = \boldsymbol{y}^q) \ln p(x^1 \mid \boldsymbol{y}^q) \\ \vdots \\ I(\boldsymbol{Y} = \boldsymbol{y}^1) \ln p(x^k \mid \boldsymbol{y}^1) + \ldots + I(\boldsymbol{Y} = \boldsymbol{y}^q) \ln p(x^k \mid \boldsymbol{y}^q) \end{pmatrix}^T \begin{pmatrix} I(X = x^1) \\ \vdots \\ I(X = x^k) \end{pmatrix} - 0 \end{aligned}$$

P-form:

$$\begin{aligned} &\ln p(X \mid \boldsymbol{Y}) \\ &= \begin{pmatrix} I(X = x^1) \ln p(x^1 \mid \boldsymbol{y}^1) + \ldots + I(X = x^k) \ln p(x^k \mid \boldsymbol{y}^1) \\ \vdots \\ I(X = x^1) \ln p(x^1 \mid \boldsymbol{y}^q) + \ldots + I(X = x^k) \ln p(x^k \mid \boldsymbol{y}^q) \end{pmatrix}^T \begin{pmatrix} I(\boldsymbol{Y} = \boldsymbol{y}^1) \\ \vdots \\ I(\boldsymbol{Y} = \boldsymbol{y}^q) \end{pmatrix} - 0 \end{aligned}$$

$$\begin{aligned} \ln p(X \mid \boldsymbol{Y}) &= \theta(X, \mathbf{Y}')^T s(Y_i) - A(X) \text{ such that } \mathbf{Y}' = \boldsymbol{Y} \setminus Y_i \\ &= \begin{pmatrix} m_1^X \cdot \mathbf{m}_1^{\mathbf{Y}'} \cdot \theta'_{11} + \ldots + m_k^X \cdot \mathbf{m}_1^{\mathbf{Y}'} \cdot \theta'_{k1} \\ \vdots \\ m_1^X \cdot \mathbf{m}_{q'}^{\mathbf{Y}'} \cdot \theta'_{1q'} + \ldots + m_k^X \cdot \mathbf{m}_{q'}^{\mathbf{Y}'} \cdot \theta'_{kq'} \end{pmatrix}^T \begin{pmatrix} I(Y_i = y_i^1) \\ \vdots \\ I(Y_i = y_i^{r_i}) \end{pmatrix} - 0 \end{aligned}$$

where $\theta'_{ij} = \ln p(x^i|\boldsymbol{y}'^j)$, $\mathbf{m}_1^{\mathbf{Y}'} = I(\mathbf{Y}' = \mathbf{y}'^1) = I(Y_1 = y_1^1) \cdot \ldots I(Y_{i-1} = y_{i-1}^1) \cdot I(Y_{i+1} = y_{i+1}^1) \cdot \ldots I(Y_n = y_n^1)$ denotes the expected sufficient statistics for the first configuration of the parent set $\mathbf{Y}' = \boldsymbol{Y} \setminus Y_i$, and $\mathbf{m}_{q'}^{\mathbf{Y}'} = I(\mathbf{Y}' = \mathbf{y}'^{q'}) = I(Y_1 = y_1^{q'}) \cdot \ldots I(Y_{i-1} = y_{i-1}^{q'}) \cdot I(Y_{i+1} = y_{i+1}^{q'}) \cdot \ldots I(Y_n = y_n^{q'})$ denotes the expected sufficient statistics for the last configuration of the parent set $\mathbf{Y}'$, with $q' = q/r_i$ denotes the total number of configurations of the parent set $\mathbf{Y}'$.

A.3 EF representation: A categorical child given a Dirichlet parent

Let X be a categorical variable with k possible values such that $k \geq 2$, and let ρ denote a dirichlet parent of X.

The log-conditional probability of the child-node X given ρ can be expressed as follows:

$$\ln p(X \mid \rho) = \sum_{j=1}^{k} I(X = x^j) \ln p(x^j).$$

Similarly the above log-conditional probability can be expressed in the following exponential forms:

C-form:

$$\ln p(X \mid \rho) = \begin{pmatrix} \ln p(x^1) \\ \vdots \\ \ln p(x^k) \end{pmatrix}^T \begin{pmatrix} I(X = x^1) \\ \vdots \\ I(X = x^k) \end{pmatrix} - 0$$

P-form:

$$\ln p(X \mid \rho) = \begin{pmatrix} I(X = x^1) \\ \vdots \\ I(X = x^k) \end{pmatrix}^T \begin{pmatrix} \ln p(x^1) \\ \vdots \\ \ln p(x^k) \end{pmatrix} - 0$$

A.4 Dirichlet distribution

Let ρ be a dirichlet variable over $\boldsymbol{p}$ with parameters $\mathbf{u}$. The log-conditional probability of ρ can be expressed as follows:

$$\begin{aligned}\ln p(\rho) &= \ln\left(\frac{\Gamma(\sum_{i=1}^{k} u_i)}{\prod_{i=1}^{k}\Gamma(u_i)}\prod_{i=1}^{k} p_i^{u_i-1}\right)\\ &= \begin{pmatrix}u_1-1\\ \vdots\\ u_k-1\end{pmatrix}^T\begin{pmatrix}\log p_1\\ \vdots\\ \log p_k\end{pmatrix} - \left(\sum_{i=1}^{k}\log\Gamma(u_i) - \log\Gamma(\sum_{i=1}^{k} u_i)\right) + 0\end{aligned}$$

A.5 EF representation: A normal child given a set of normal parents and an inverse-gamma parent

Let X be a normal variable with variance γ and let $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ denote the set of parents of X, such that all of them are normal distributed. We let β_0, and $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_n\}$ denote the coefficients of these parent variables (with normal prior distributions) and we assume that γ follows an inverse gamma distribution.

The log-conditional probability of X given its parents $\boldsymbol{Y}, \boldsymbol{\beta}, \sigma^2$ can be expressed as follows:

$$\ln p(X|Y_1, \ldots, Y_n, \boldsymbol{\beta}, \gamma) = -\ln\sigma - 0.5\ln(2\pi) - \frac{(x - \beta_0 - \boldsymbol{\beta}^T\boldsymbol{Y})^2}{2\gamma}$$

C-form (messages from $\boldsymbol{\beta}$ and γ variables to X):

$$\begin{aligned}\ln p(X \mid \boldsymbol{Y}) &= \theta(\boldsymbol{Y})^T s(X) - A\big(\theta(\boldsymbol{Y})\big) + h(\boldsymbol{X})\\ &= \begin{pmatrix}\frac{\beta_0}{\gamma} + \frac{\boldsymbol{\beta}}{\gamma}Y\\ \frac{-1}{2\gamma}\end{pmatrix}^T\begin{pmatrix}X\\ X^2\end{pmatrix}\\ &- \left(0.5\ln\gamma + \frac{\beta_0\boldsymbol{\beta}}{\gamma}Y + \frac{\boldsymbol{\beta}\boldsymbol{\beta}^T}{2\gamma}YY^T + \frac{\beta_0^2}{2\gamma}\right) - \tfrac{1}{2}\ln(2\pi)\end{aligned}$$

P-form (messages from X to β_0):

$$\begin{aligned}\ln p(X \mid \boldsymbol{Y}, \beta_0, \boldsymbol{\beta}, \gamma) &= \theta(X, Y, \boldsymbol{\beta}, \gamma)^T s(\beta_0) - A\big(\theta(X, Y, \boldsymbol{\beta}, \gamma)\big) + h(\mathbf{Y}, \beta_{\mathbf{0}}, \boldsymbol{\beta}, \gamma) \\ &= \begin{pmatrix} X\sigma^{-2} - \boldsymbol{\beta}^T Y \sigma^{-2} \\ -0.5\sigma^{-2} \end{pmatrix}^T \begin{pmatrix} \beta_0 \\ \beta_0^2 \end{pmatrix} \\ &- X^2 0.5\sigma^{-2} - \boldsymbol{\beta}^T \boldsymbol{\beta} Y^T Y 0.5\sigma^{-2} + X\boldsymbol{\beta}^T Y \sigma^{-2} - \ln\sigma - 0.5\ln(2\pi)\end{aligned}$$

P-form (messages from X to β_i):

$$\begin{aligned}\ln p(X \mid \boldsymbol{Y}, \beta_0, \boldsymbol{\beta}, \gamma) &= \theta(X, Y, \beta_0, \boldsymbol{\beta}', \gamma)^T s(\beta_i) - A\big(\theta(X, Y, \beta_0, \boldsymbol{\beta}', \gamma)\big) + h(\mathbf{Y}, \beta_{\mathbf{0}}, \boldsymbol{\beta}, \gamma) \\ &= \begin{pmatrix} -\beta_0 Y_i \sigma^{-2} + Y_i X \sigma^{-2} - \boldsymbol{\beta}_i' Y_i Y' \sigma^{-2} \\ -0.5 Y_i Y_i \sigma^{-2} \end{pmatrix}^T \begin{pmatrix} \beta_i \\ {\beta_i}^2 \end{pmatrix} \\ &+ \boldsymbol{\theta}'_{\beta_0 \boldsymbol{\beta}} Y + 2\boldsymbol{\theta}'_{\boldsymbol{\beta}} XY + \boldsymbol{\theta}'_{\boldsymbol{\beta}\boldsymbol{\beta}} YY^T + \theta_{\beta_0} X + \theta_{\text{-}1} XX \\ &- \begin{pmatrix} \frac{\theta_{\beta_0}^2}{4\theta_{\text{-}1}} \end{pmatrix} - \tfrac{1}{2}\ln(2\pi)\end{aligned}$$

where $\boldsymbol{\beta}_i' = \boldsymbol{\beta}$ where $\beta_i = 0$,

P-form (messages from X to γ):

$$\begin{aligned}\ln p(X|Y_1, \ldots, Y_n, \boldsymbol{\beta}, \gamma) &= \theta(X, Y, \beta_0, \boldsymbol{\beta})^T s(\gamma) - A\big(\theta(X, Y, \beta_0, \boldsymbol{\beta})\big) + h(\mathbf{Y}, \beta_{\mathbf{0}}, \boldsymbol{\beta}) \\ &= \begin{pmatrix} -\frac{1}{2} \\ -\frac{(X - \beta_0 - \boldsymbol{\beta} Y)^2}{2} \end{pmatrix}^T \begin{pmatrix} \ln\gamma \\ \frac{1}{\gamma} \end{pmatrix} - 0.5\ln(2\pi)\end{aligned}$$

P-form (for each parent Y_i, $Y' = Y \setminus Y_i$):

$$\begin{aligned}\ln p(X \mid \boldsymbol{Y}) &= \theta(X, Y')^T s(Y_i) - A\big(\theta(X, Y')\big) + h(\boldsymbol{Y}) \\ &= \begin{pmatrix} -\beta_0 \beta_i \sigma^{-2} + \beta_i X \sigma^{-2} - \beta_i \boldsymbol{\beta}_i' Y' \sigma^{-2} \\ -0.5\beta_i \beta_i \sigma^{-2} \end{pmatrix}^T \begin{pmatrix} Y_i \\ {Y_i}^2 \end{pmatrix} \\ &+ \boldsymbol{\theta}'_{\beta_0 \boldsymbol{\beta}} Y + 2\boldsymbol{\theta}'_{\boldsymbol{\beta}} XY + \boldsymbol{\theta}'_{\boldsymbol{\beta}\boldsymbol{\beta}} YY^T + \theta_{\beta_0} X + \theta_{\text{-}1} XX \\ &- \begin{pmatrix} \frac{\theta_{\beta_0}^2}{4\theta_{\text{-}1}} \end{pmatrix} - \tfrac{1}{2}\ln(2\pi)\end{aligned}$$

where $\boldsymbol{\beta}_i' = \boldsymbol{\beta} \setminus \beta_i$

A.6 Inverse gamma distribution

Let γ be an inverse gamma variable with parameters α, β. The log-conditional probability of γ can be expressed as follows:

$$\begin{aligned}\ln p(\gamma) &= \ln\left(\frac{\beta^\alpha}{\Gamma(\alpha)}\gamma^{-\alpha-1}e^{-\frac{\beta}{\gamma}}\right) \\ &= \begin{pmatrix}-\alpha-1 \\ -\beta\end{pmatrix}^T \begin{pmatrix}\ln\gamma \\ \frac{1}{\gamma}\end{pmatrix} - (\ln\Gamma(\alpha) - \alpha\ln\beta) + 0\end{aligned}$$

A.7 EF representation: A base distribution given a binary parent

Let X be any base distribution variable, and let Y be a binary variable. The log-conditional probability of the child-node X given its binary parent-node Y is expressed as follows:

$$\ln p(X \mid Y) = I(Y = y^1)\ln p(X \mid y^1) + I(Y = y^2)\ln p(X \mid y^2)$$

This conditional probability distribution can be expressed in different exponential forms as follows:

C-form:

$$\begin{aligned}\ln p(X \mid Y) &= \Big(I(Y = y^1)\cdot\theta_X(Y = y^1) + I(Y = y^2)\cdot\theta_X(Y = y^2)\Big)s(X) \\ &\quad -I(Y = y^1)\cdot A(\theta_X(Y = y^1)) - I(Y = y^2)\cdot A(\theta_X(Y = y^2))\end{aligned}$$

P-form:

$$\ln p(X \mid Y) = \begin{pmatrix}\theta_X(Y = y^1)s(X) - A(\theta_X(Y = y^1)) \\ \theta_X(Y = y^2)s(X) - A(\theta_X(Y = y^2))\end{pmatrix}^T \begin{pmatrix}I(Y = y^1) \\ I(Y = y^2)\end{pmatrix} - 0$$

A.8 EF representation: A base distribution given a set of categorical parents

Let X be any base distribution, and let $\boldsymbol{Y} = \{Y_1, \ldots, Y_n\}$ denote the set of parents of X, such that all of them are categorical. Each parent Y_i, $1 \geq i \geq n$, has r_i possible values or states such that $r_i \geq 2$. A parental configuration for the child-node X is then a set of n elements $\{Y_1 = y_1, \ldots, Y_i = y_i, \ldots, Y_n = y_n\}$ such that y_i denotes a potential value of variable Y_i such that $1 \leq y_i \leq r_i$. Let $q = r_1 \times \ldots \times r_n$ denote the total number of parental configurations, and let $\boldsymbol{y}^l$ denote the l^{th} parental configuration such that $1 \leq l \leq q$.

The log-conditional probability of the child-node X given its parent-nodes $\boldsymbol{Y}$ can be expressed as follows:

$$\ln p(X \mid \boldsymbol{Y}) = \sum_{l=1}^{q} I(\boldsymbol{Y} = \boldsymbol{y}^l) \cdot \ln p(X \mid \boldsymbol{y}^l)$$

This conditional probability distribution can be expressed in different exponential forms as follows:

C-form:

$$\ln p(X \mid \boldsymbol{Y}) = \Big(\sum_{l=1}^{q} I(\boldsymbol{Y} = \boldsymbol{y}^l) \cdot \theta_X(\boldsymbol{Y} = \boldsymbol{y}^l) \Big)^T s(X) - \sum_{l=1}^{q} I(\boldsymbol{Y} = \boldsymbol{y}^l) \cdot A(\theta_X(\boldsymbol{Y} = \boldsymbol{y}^l))$$

P-form:

$$\begin{aligned} \ln p(X \mid \boldsymbol{Y}) &= \theta(X)^T s(\boldsymbol{Y}) - A(X) \\ &= \begin{pmatrix} -A(\theta_{X1}) \\ \vdots \\ -A(\theta_{Xq}) \\ s(X) \cdot \theta_{X1} \\ \vdots \\ s(X) \cdot \theta_{Xq} \end{pmatrix}^T \begin{pmatrix} I(\boldsymbol{Y} = \boldsymbol{y}^1) \\ \vdots \\ I(\boldsymbol{Y} = \boldsymbol{y}^q) \\ I(\boldsymbol{Y} = \boldsymbol{y}^1) \\ \vdots \\ I(\boldsymbol{Y} = \boldsymbol{y}^q) \end{pmatrix} - 0 \end{aligned}$$

$$\begin{aligned}
\ln p(X \mid \boldsymbol{Y}) &= \theta(X, \mathbf{Y}')^T s(Y_i) - A(X) \text{ such that } \mathbf{Y}' = \boldsymbol{Y} \setminus Y_i \\
&= \begin{pmatrix} -A(\theta_{X1}) \\ \vdots \\ -A(\theta_{Xq}) \\ s(X) \cdot \mathbf{m}_1^{\mathbf{Y}'} \cdot \theta'_{X1} + \ldots + s(X) \cdot \mathbf{m}_1^{\mathbf{Y}'} \cdot \theta'_{X1} \\ \vdots \\ s(X) \cdot \mathbf{m}_{q'}^{\mathbf{Y}'} \cdot \theta'_{Xq'} + \ldots + s(X) \cdot \mathbf{m}_{q'}^{\mathbf{Y}'} \cdot \theta'_{Xq'} \end{pmatrix}^T \begin{pmatrix} I(Y_i = y_i^1) \\ \vdots \\ I(Y_i = y_i^{r_i}) \\ I(Y_i = y_i^1) \\ \vdots \\ I(Y_i = y_i^{r_i}) \end{pmatrix} - 0
\end{aligned}$$

References

[1] Borchani, H., Fernández, A., Gundersen, O.E., Hovda, S., Langseth, H., Madsen, A.L., Martínez, A.M., Sáez-Martínez, R., Masegosa, A.R., Nielsen, T.D., Salmerón, A., Sørmo, F., Weidl, G.: The AMIDST modelling framework – final report (2015) Deliverable 2.2 of the AMIDST project, `amidst.eu`.

[2] Salmerón, A., Rumí, R., Langseth, H., Madsen, A.L., Nielsen, T.D.: MPE inference in conditional linear Gaussian networks. ECSQARU'2015. Lecture Notes in Artificial Intelligence **9161** (2015) 407–416

[3] Salmerón, A., Ramos-López, D., Borchani, H., Masegosa, A.R., Fernández, A., Langseth, H., Madsen, A.L., Nielsen, T.D.: Parallel importance sampling in conditional linear Gaussian networks. CAEPIA'2015. Lecture Notes in Artificial Intelligence **9422** (2015) 36–46

[4] Langseth, H., Madsen, A.L., Nielsen, T.D., Rumí, R., Salmerón, A.: State of the art of inference in hybrid and dynamic models (October 2014) Deliverable 3.1 of the AMIDST project (FP7 project funded by the EC under contract 616209), `amidst.eu`.

[5] Salmerón, A., Rumí, R., Langseth, H., Nielsen, T.D., Madsen, A.L.: A review of inference algorithms for hybrid Bayesian networks. Journal of Artificial Intelligence Research **Submitted** (2015)

[6] Cowell, R.G., Dawid, A.P., Lauritzen, S.L., Spiegelhalter, D.J.: Probabilistic Networks and Expert Systems. Statistics for engineering and information science. Springer (1999) ISBN 0-387-98767-3.

[7] Jensen, F.V., Nielsen, T.D.: Bayesian networks and decision graphs. Second edn. Springer Publishing Company, Incorporated (2007)

[8] Kjærulff, U., Madsen, A.L.: Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis. Springer (2013)

[9] Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)

[10] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Mateo, CA. (1988)

[11] Lauritzen, S.L., Jensen, F.: Stable local computation with conditional Gaussian distributions. Statistics and Computing **11**(2) (2001) 191–203

[12] Lauritzen, S.L., Wermuth, N.: Graphical models for associations between variables, some of which are qualitative and some quantitative. The Annals of Statistics **17** (1989) 31–57

[13] Moral, S., Rumí, R., Salmerón, A.: Mixtures of truncated exponentials in hybrid Bayesian networks. In: EQSCARU'2001. Volume 2143 of Lecture Notes in Artificial Intelligence. Springer, Berlin, Germany (2001) 145–167

[14] Shenoy, P.P., West, J.C.: Inference in hybrid Bayesian networks using mixtures of polynomials. International Journal of Approximate Reasoning **52** (2011) 641–657

[15] Langseth, H., Nielsen, T.D., Rumí, R., Salmerón, A.: Mixtures of truncated basis functions. International Journal of Approximate Reasoning **53**(2) (2012) 212–227

[16] Rumí, R., Salmerón, A.: Approximate probability propagation with mixtures of truncated exponentials. International Journal of Approximate Reasoning **45** (2007) 191–210

[17] Shachter, R.D., Kenley, C.: Gaussian influence diagrams. Management Science **35** (1989) 527–550

[18] Gámez, J.A.: Abductive inference in Bayesian networks: a review. In Gámez, J., Moral, S., Salmerón, A., eds.: Advances in Bayesian Networks. Springer (2004) 101–117

[19] Attias, H.: A variational Bayesian framework for graphical models. Advances in neural information processing systems (2000) 209—215

[20] Jaakkola, T., Qi, Y.: Parameter expanded variational Bayesian methods. In: Advances in Neural Information Processing Systems. (2006) 1097–1104

[21] Beal, M.: Variational algorithms for approximate Bayesian inference. PhD thesis, Gatsby Computational Neuroscience Unit, University College London (2003)

[22] Winn, J., Bishop, C.: Variational message passing. Journal of Machine Learning Research **6** (2005) 661–694

[23] Hammersley, J., Handscomb, D.: Monte Carlo Methods. Chapman & Hall (1964)

[24] Fernández, A., Rumí, R., Salmerón, A.: Answering queries in hybrid Bayesian networks using importance sampling. Decision Support Systems **53** (2012) 580–590

[25] Fung, R., Chang, K.C.: Weighting and integrating evidence for stochastic simulation in Bayesian networks. In Henrion, M., Shachter, R., Kanal, L., Lemmer, J., eds.: Uncertainty in Artificial Intelligence. Volume 5., North-Holland (Amsterdam) (1990) 209–220

[26] Sun, W., Chang, K.C.: Probabilistic inference using linear Gaussian importance sampling for hybrid Bayesian networks. In: Signal Processing, Sensor Fusion, and Target Recognition XIV. Proc. of SPIE. Volume 5809. (2005) 322–329

[27] Rubinstein, R.Y.: Simulation and the Monte Carlo Method. Wiley (New York) (1981)

[28] Park, J.D.: MAP complexity results and approximation methods. In Darwiche, A., Friedman, N., eds.: UAI, Morgan Kaufmann (2002) 388–396

[29] Dawid, A.P.: Applications of a general propagation algorithm for a probabilistic expert system. Statistics and Computing **2** (1992) 25–36

[30] Kwisthout, J.: Most probable explanations in Bayesian networks: Complexity and tractability. International Journal of Approximate Reasoning **52** (2011) 1452–1469

[31] Sun, W., Chang, K.C.: Study of the most probable explanation in hybrid Bayesian networks. In: Signal Processing, Sensor Fusion, and Target Recognition XX. Proc. of SPIE. Volume 8050. (2011)

[32] Lerner, U., Parr, R.: Inference in hybrid networks: Theoretical limits and practical algorithms. In: In UAI. (2001) 310–318

[33] Dechter, R.: Bucket elimination: a unifiying framework for reasoning. Artificial Intelligence **113** (1999) 41–85

[34] Madsen, A.L.: Belief update in CLG Bayesian networks with lazy propagation. International Journal of Approximate Reasoning **49** (2008) 503–521

[35] Cinicioglu, E.N., Shenoy, P.P.: Arc reversals in hybrid Bayesian networks with deterministic variables. International Journal of Approximate Reasoning **50** (2009) 763–777

[36] de Campos, L.M., Gámez, J.A., Moral, S.: Partial abductive inference in Bayesian belief networks by simulated annealing. International Journal of Approximate Reasoning **27** (2001) 263–283

[37] Park, J., Darwiche, A.: Approximating MAP using local search. In: Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI–01), Morgan Kaufmann Publishers (2001) 403–410

[38] Boyen, X., Koller, D.: Tractable inference for complex stochastic processes. In: Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98). (1998) 33–42

[39] Boyen, X., Koller, D.: Exploiting the architecture of dynamic systems. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99). (1999) 313–320

[40] Murphy, K., Weiss, Y.: The factored frontier algorithm for approximate inference in DBNs. In: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI). (2001) 378–385